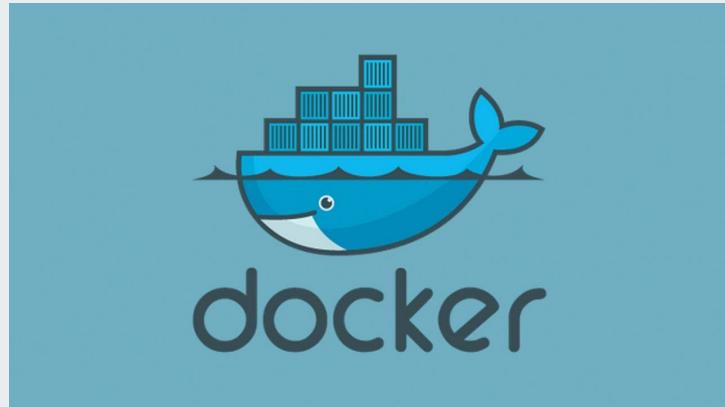
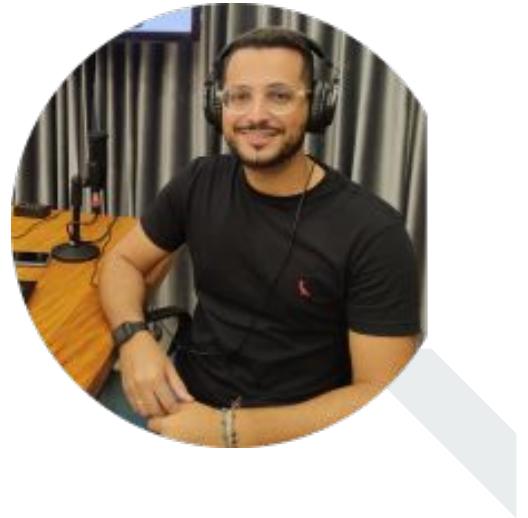

Mini Curso Docker

Entendendo um pouco sobre docker



Um pouco sobre mim...

- Weslley Fratini
- Software Engineer FullStack
- Co-host podcast CaféDebug
- Entusiasta por tecnologia
- Um bom pescador nas horas vagas



Agenda

- História do docker
- Curiosidades
- O que é docker?
- Imagens, volumes, containers, swarm
- Docker compose
- Benefícios de uso
- Hands on

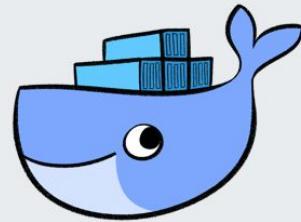
Antes de comerçarmos...

Perguntas

<https://www.menti.com/v2ahdp1uyz>

Docker Play

[Contribute](#)



Play with Docker

A simple, interactive and fun playground to learn Docker

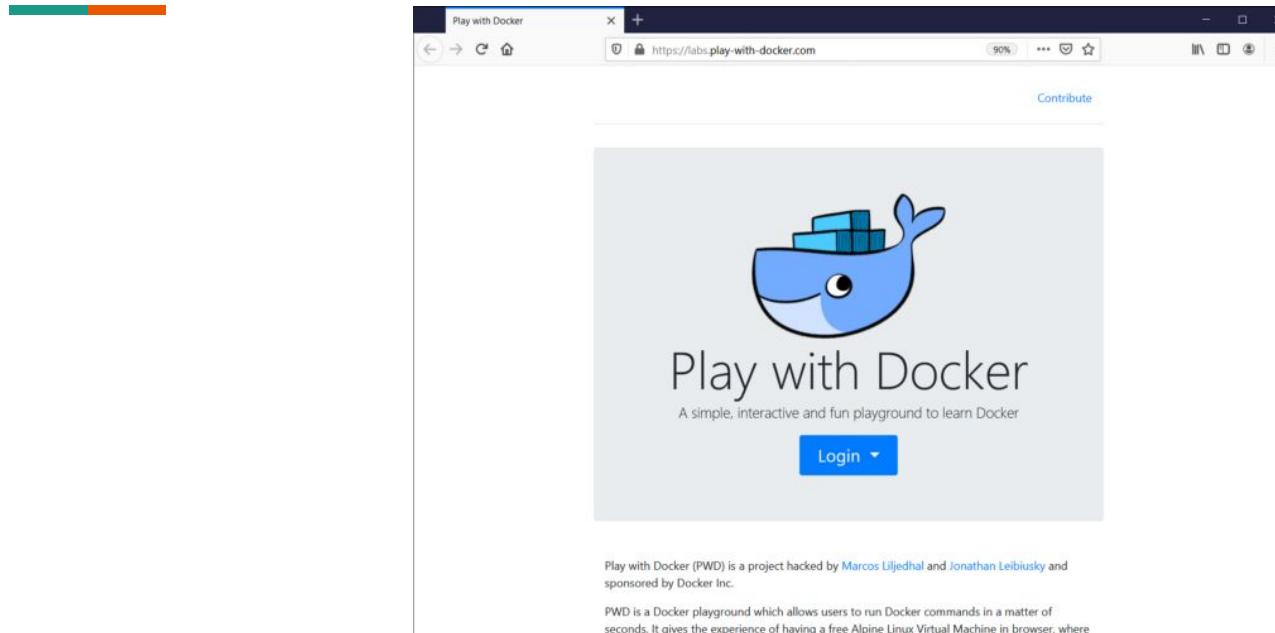
Start



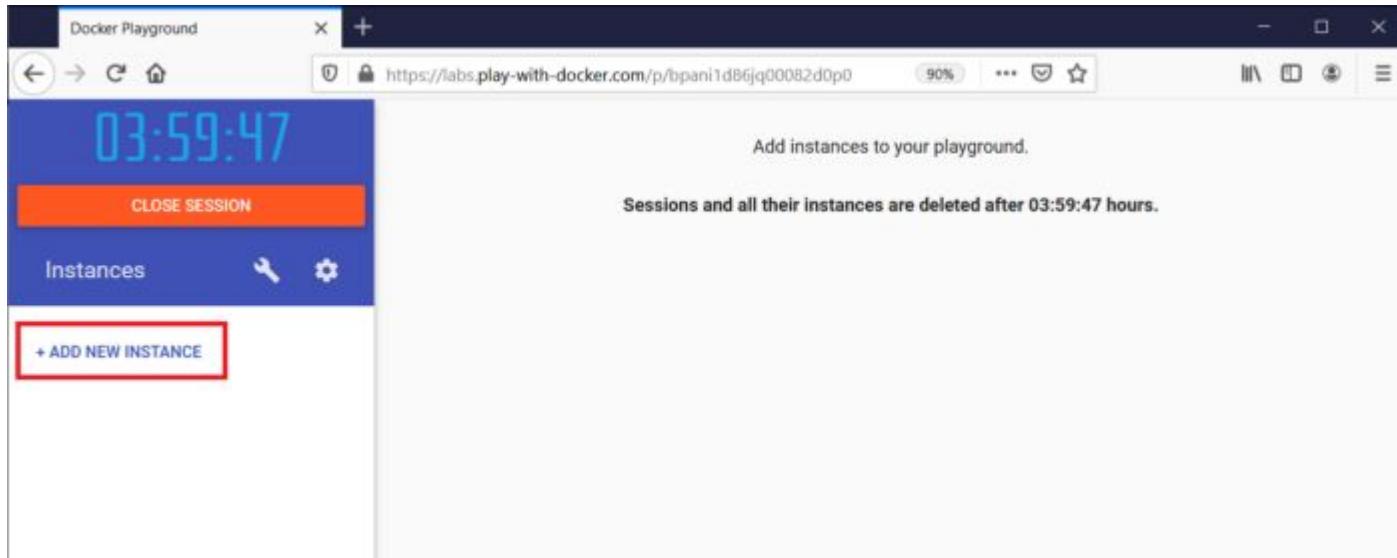
Docker play site

Disponibiliza máquinas-virtuais Linux para testes por um período de 4 horas.

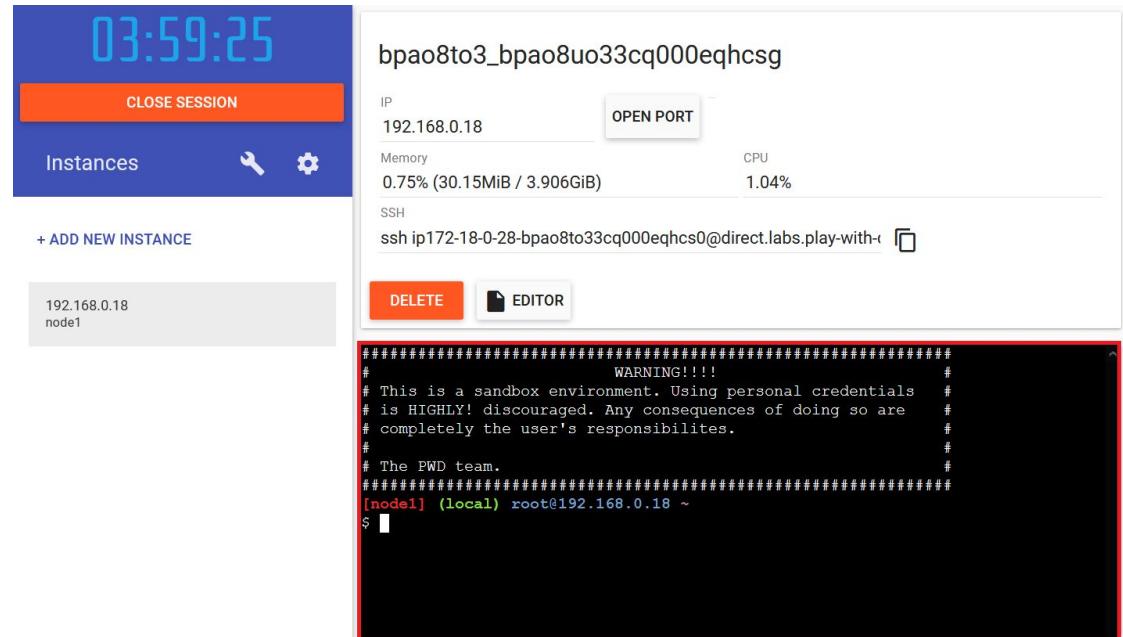
<https://labs.play-with-docker.com/>



Para criar uma máquina virtual para testes acionar a opção **+ADD NEW INSTANCE**:



Logo a VM tenha sido criada, a mesma já estará devidamente configurada para uso do **Docker** e assim disponível para testes:



Como surgiu?

- O Docker foi desenvolvido em 2013 por uma empresa chamada dotCloud
- A startup foi criada por Solomon Hykes, junto com Sebastien Pahl, fundou a empresa Docker.
- Sua sede está localizada em Palo Alto, Califórnia.



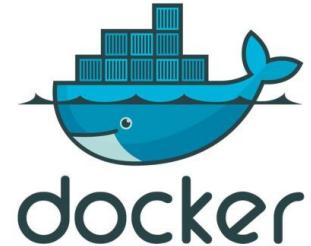
2013-2015

2015-2017



2017-PRESENTE

Evolução do logo ao longo dos tempos



2013 - 2015

- O desenvolvedor do símbolo é o estúdio freelance 99designs.
- Forneceu à empresa 84 protótipos do logotipo, dos quais Docker preferiu um desenho de Ricky Asamanis da Indonésia.

 2015 – 2017



- Nesta versão, todos os elementos do logotipo anterior são preservados.
- As mudanças afetaram apenas seu agrupamento.
- Os designers moveram a palavra “Docker” para a direita, atrás do ícone de uma baleia carregada de contêineres.

 2017 – presente



- O logotipo atual é uma reformulação esquemática da versão de estreia.
- Os autores abandonaram o detalhamento dos contêineres e da baleia, com foco nas silhuetas.
- Isso é feito para indicar a seriedade da aplicação, que se tornou muito melhor e mais conveniente com o tempo.

Sobre o docker

Docker se refere a muitas coisas:

- Um projeto da comunidade open source
- É um pacote de software de código aberto para automatizar processos de entrega de aplicativos em um ambiente isolado chamado contêineres



- É uma tecnologia de containerização para criação e uso de containers Linux.
- A empresa Docker se baseia no trabalho realizado pela comunidade, tornando-o mais seguro, e compartilha os avanços com a comunidade em geral.



Docker	
	docker
Autor	Solomon Hykes
Desenvolvedor	Docker, Inc.
Lançamento	13 de março de 2013 (8 anos)
Escrito em	Go
Sistema operacional	Linux, Windows e Mac
Gênero(s)	virtualização tipo contêiner
Licença	Licença Apache 2.0 ou Proprietária (no caso do "Docker Enterprise")
Página oficial	www.docker.com

Qual a motivação para usar?

Acelere a maneira como você cria,
compartilha e executa aplicativos
modernos.

+ 13 milhões de
desenvolvedores

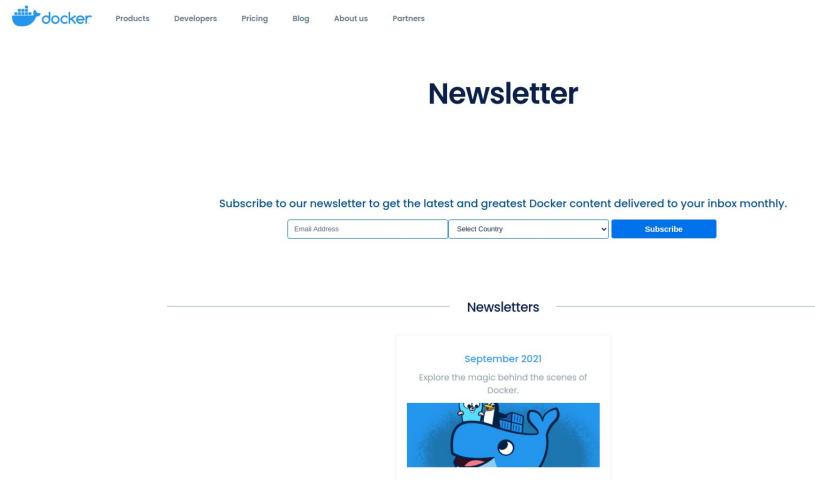
+ 7 milhões de
aplicativos

Mais de 13 bilhões de
downloads mensais de imagens

Segundo o stackOverFlow



Quer ficar informado de tudo do docker?



The screenshot shows the Docker newsletter subscription form. At the top, there's a navigation bar with links to Products, Developers, Pricing, Blog, About us, and Partners. Below that is a large "Newsletter" heading. A sub-headline says "Subscribe to our newsletter to get the latest and greatest Docker content delivered to your inbox monthly." Below this are three input fields: "Email Address", "Select Country", and a "Subscribe" button. At the bottom, there's a section titled "Newsletters" featuring a thumbnail for the "September 2021" issue, which is described as "Explore the magic behind the scenes of Docker." It features an illustration of a blue whale.



The screenshot shows the Docker Community Slack invite page. It features the Slack logo at the top. Below it is the headline "See what Docker Community is up to". A sub-headline states "Slack is a messaging app that brings your whole team together." There's a small thumbnail of four people and the text "Shannyn and 36,034 others have already joined".

We suggest using the email account you use for work.

 Continue with Google

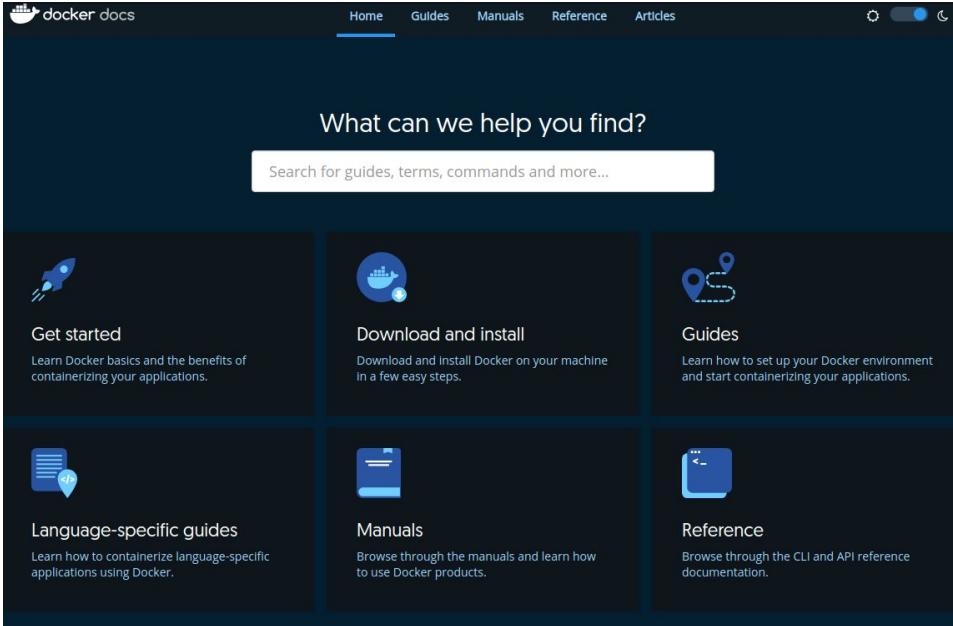
 Continue with Apple

 Continue with Email

<https://www.docker.com/newsletter-subscription>

https://dockercommunity.slack.com/join/shared_invite/zt-wk3lwpw2-0BX0AXXbdoZ6ydSd6lxTeQ#/shared-invite/email

Documentação

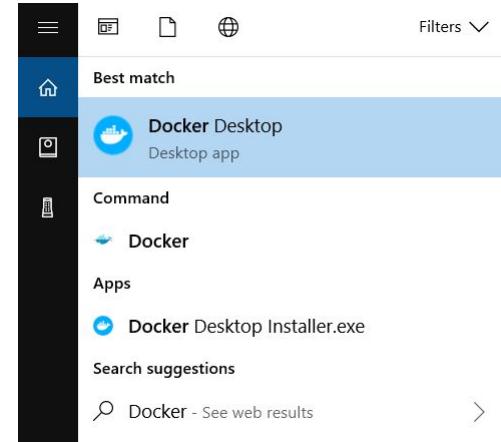


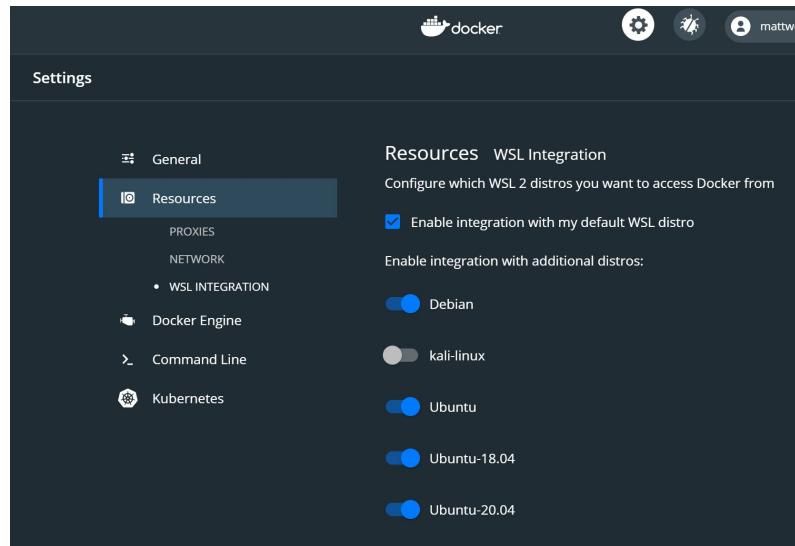
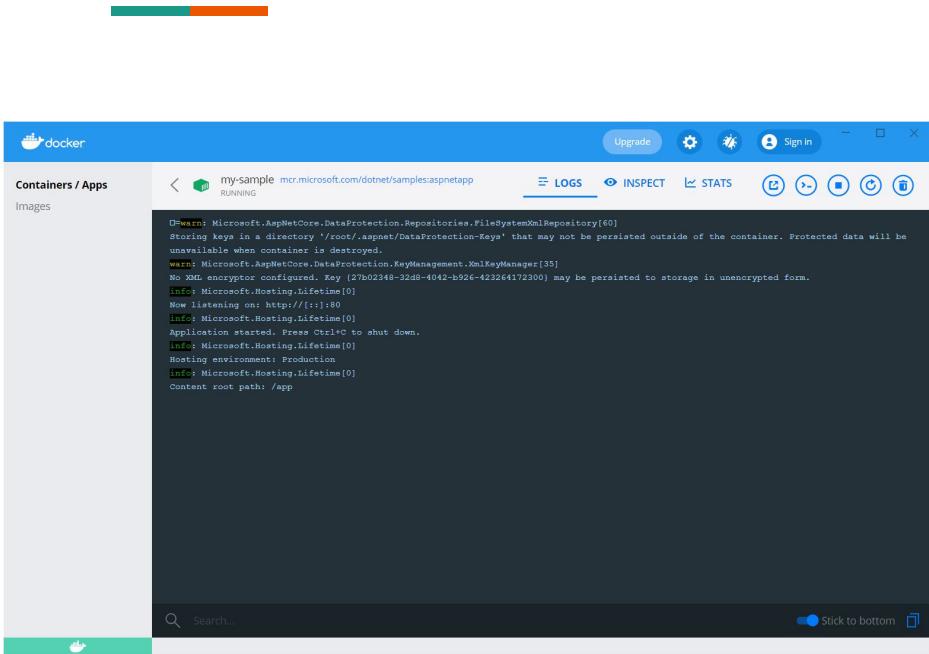
The screenshot shows the Docker documentation website (docs.docker.com) with a dark blue header. The header includes the Docker logo, navigation links for Home, Guides, Manuals, Reference, and Articles, and a toggle switch for light/dark mode. The main content area features a search bar with the placeholder "Search for guides, terms, commands and more...". Below the search bar are six cards arranged in a 2x3 grid:

- Get started**: Learn Docker basics and the benefits of containerizing your applications.
- Download and install**: Download and install Docker on your machine in a few easy steps.
- Guides**: Learn how to set up your Docker environment and start containerizing your applications.
- Language-specific guides**: Learn how to containerize language-specific applications using Docker.
- Manuals**: Browse through the manuals and learn how to use Docker products.
- Reference**: Browse through the CLI and API reference documentation.

Docker para Mac e Windows

- **Windows e Mac**, temos o Docker Desktop
- Docker Desktop está disponível apenas para as versões **Professional e Enterprise**





<https://docs.docker.com/desktop/windows/install/>



The screenshot shows the Docker Docs website with a dark theme. The navigation bar includes links for Home, Guides, Manuals (which is underlined), Reference, and Samples. The left sidebar is titled "Docker Desktop" and contains the following sections:

- Overview
- Mac
- Windows
 - Install Docker Desktop for Windows
 - User manual
 - Networking
 - Logs and troubleshooting
 - Docker Desktop WSL 2 backend
 - Release notes
 - Previous versions
- Dashboard
- Dev Environments (Preview)
- Multi-arch support
- Deploy on Kubernetes
- FAQs
- Back up and restore data
- Docker Engine
- Docker Compose
- Docker Hub

The main content area has a title "Install Docker Desktop on Windows" with an estimated reading time of 9 minutes. It features a callout box for "Update to the Docker Desktop terms". Below it, there's a section for "Download Docker Desktop for Windows" with a large blue button labeled "Docker Desktop for Windows". The "System requirements" section follows, with a note about meeting requirements for successful installation. At the bottom, tabs for "WSL 2 backend" and "Hyper-V backend and Windows containers" are shown. A detailed list of system requirements for the WSL 2 backend is provided.

Install Docker Desktop on Windows
Estimated reading time: 9 minutes

● Update to the Docker Desktop terms

Professional use of Docker Desktop in large organizations (more than 250 employees or more than \$10 million in annual revenue) requires users to have a paid Docker subscription. While the effective date of these terms is August 31, 2021, there is a grace period until January 31, 2022, for those that require a paid subscription. For more information, see the blog [Docker Is Updating and Extending Our Product Subscriptions](#) and the [Docker Desktop License Agreement](#).

Welcome to Docker Desktop for Windows. This page contains information about Docker Desktop for Windows system requirements, download URL, instructions to install and update Docker Desktop for Windows.

Download Docker Desktop for Windows

[Docker Desktop for Windows](#)

System requirements

Your Windows machine must meet the following requirements to successfully install Docker Desktop.

[WSL 2 backend](#) [Hyper-V backend and Windows containers](#)

WSL 2 backend

- Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit: Home or Pro 2004 (build 19041) or higher, or Enterprise or Education 1909 (build 18363) or higher.
- Enable the WSL 2 feature on Windows. For detailed instructions, refer to the [Microsoft documentation](#).
- The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11.

Linux

```
weslley-fratini@weslley-fratini ➔ sudo systemctl status docker
[sudo] password for weslley-fratini:
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Sun 2021-10-31 16:39:57 -03; 2 days ago
    TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 1671 (dockerd)
     Tasks: 36
    Memory: 870.8M
      CGroup: /system.slice/docker.service
              └─ 1671 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
                  ├─ 69628 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8080 -container-ip 172.17.0.0
                  ├─ 69634 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 8080 -container-ip 172.17.0.2 -c>
out 31 16:39:57 weslley-fratini dockerd[1671]: time="2021-10-31T16:39:57.018068636-03:00" level=info msg="API>
out 31 21:42:24 weslley-fratini dockerd[1671]: time="2021-10-31T21:42:24.302541406-03:00" level=info msg="Att>
out 31 21:43:17 weslley-fratini dockerd[1671]: time="2021-10-31T21:43:17.431654884-03:00" level=warning msg=">
out 31 21:43:17 weslley-fratini dockerd[1671]: time="2021-10-31T21:43:17.552046132-03:00" level=error msg="ce>
out 31 21:43:17 weslley-fratini dockerd[1671]: time="2021-10-31T21:43:17.552141866-03:00" level=error msg="Ha>
out 31 21:43:24 weslley-fratini dockerd[1671]: time="2021-10-31T21:43:24.928948602-03:00" level=warning msg=">
out 31 21:43:25 weslley-fratini dockerd[1671]: time="2021-10-31T21:43:25.028534037-03:00" level=error msg="b6>
out 31 21:43:25 weslley-fratini dockerd[1671]: time="2021-10-31T21:43:25.028597084-03:00" level=error msg="Ha>
out 31 21:44:00 weslley-fratini dockerd[1671]: time="2021-10-31T21:44:00.837720832-03:00" level=info msg="ign>
out 31 21:44:00 weslley-fratini dockerd[1671]: time="2021-10-31T21:44:00.837871535-03:00" level=error msg="at>
lines 1-23/23 (END)
```

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

CONTENTS

- Pré-requisitos
- Passo 1 — Instalando o Docker
- Passo 2 — Executando o Comando Docker Sem Sudo (Opcional)
- Passo 3 — Usando o Comando Docker
- Passo 4 — Trabalhando com Imagens do Docker
- Passo 5 — Executando um Container do Docker
- Passo 6 — Gerenciando os Containers do Docker
- Passo 7 — Enviando Alterações em um Container para uma Imagem do Docker
- Passo 8 — Empurrando Imagens do Docker para um Repositório do Docker
- Conclusão

TUTORIAL

Como Instalar e Utilizar o Docker no Ubuntu 20.04

Docker Ubuntu 20.04

By Brian Hogan Published on June 11, 2020 57.4k

XA Português

Introdução

O Docker é um aplicativo que simplifica o processo de gerenciamento de processos de aplicação em *containers*. Os containers deixam você executar suas aplicações em processos isolados de recurso. Eles são semelhantes a máquinas virtuais, mas os containers são mais portáveis, mais fáceis de usar e mais dependentes do sistema operacional do host.

Para uma introdução detalhada aos diferentes componentes de um container Docker, verifique [O Ecossistema Docker: Uma Introdução aos Componentes Comuns](#).

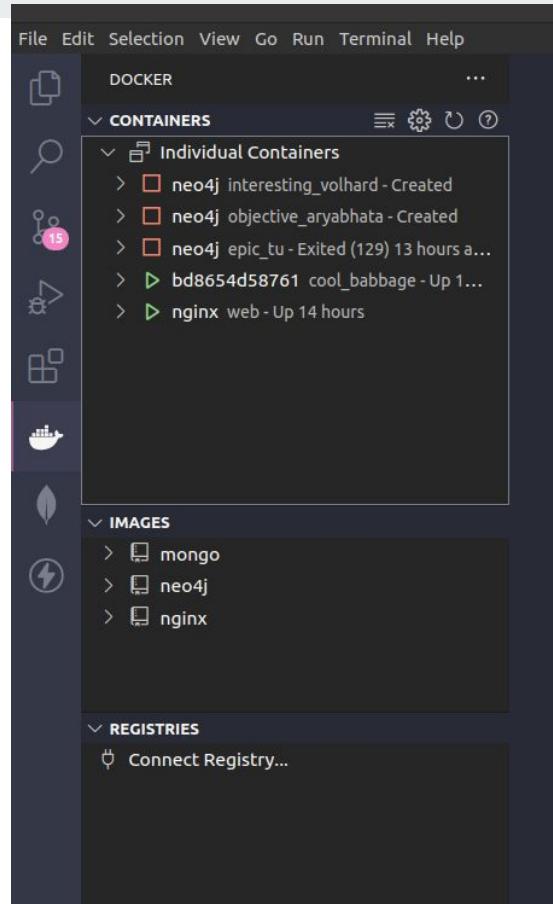
Neste tutorial, você irá instalar e usar a Edição Community (CE) do Docker no Ubuntu 20.04. Você instalará o Docker propriamente dito, trabalhará com contêineres e imagens, e enviará uma imagem para um repositório do Docker.

Pré-requisitos

Para seguir este tutorial, você precisará do seguinte:

- Um servidor Ubuntu 20.04 configurado conforme o [Guia de configuração inicial de servidor do Ubuntu 20.04](#), incluindo um usuário sudo não root e um firewall.
- Uma conta no [Docker Hub](#) se você deseja criar suas próprias imagens e enviá-las para o Docker Hub, como mostrado nos passos 7 e 8.

Extensão vsCode



Qual o objetivo do docker?



- Criar, testar e implementar aplicações em um ambiente separado da máquina original, chamado de container.
- O desenvolvedor consegue empacotar o software de maneira padronizada.

Por que usar docker?

- Torna o desenvolvimento eficiente
- Elimina as tarefas de configuração rotineiras e repetitivas
- É usado em todo o ciclo de vida de desenvolvimento para o criação de aplicações
- Inclui UIs, CLIs, APIs e segurança que são projetadas para trabalhar juntas em todo o ciclo de entrega da aplicação.



Comandos relacionados à informações

docker version: exibe a versão do docker que está instalada.

docker inspect ID_CONTAINER: retorna diversas informações sobre o container.

docker ps: exibe todos os containers em execução no momento.

docker ps -a: exibe todos os containers, independente de estarem em execução ou não

Alternativas docker



MESOS



kubernetes



Core OS

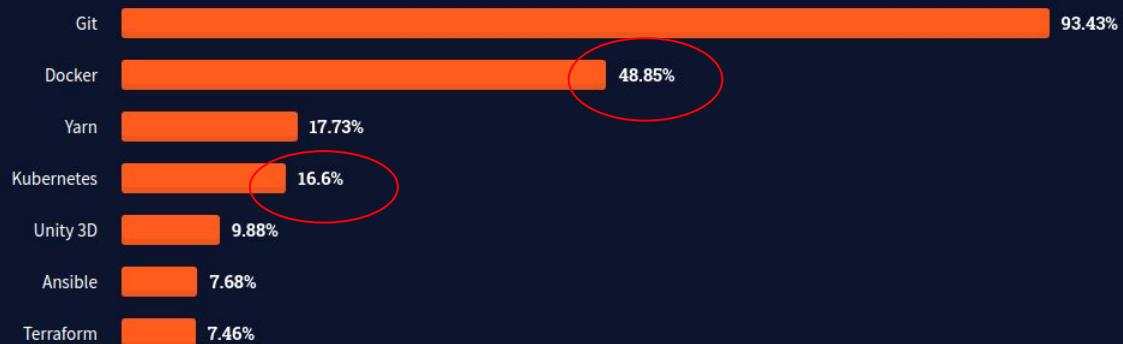
Open Source Projects for Linux Containers



All Respondents

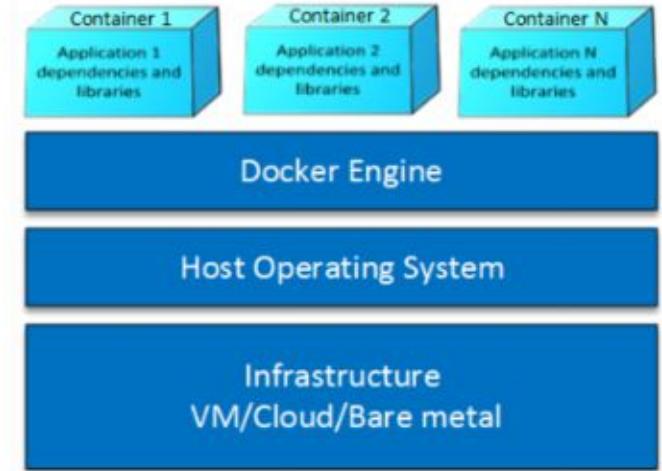
Professional Developers

76,253 responses





Arquitetura docker



Os principais componentes da arquitetura

- **Docker** para Mac, Linux e Windows – versões que permitem instalar e executar containers nos sistemas operacionais de forma isolada.
- **Docker Daemon** – Software que roda na máquina onde o Docker está instalado. Usuário não interage diretamente.
- **Docker Client** – CLI ou REST API que aceita comandos do usuário e repassa estes comandos ao Docker daemon.

- 
- **Docker Image** – É um template. Uma imagem contém todos os dados necessários para executar *containers* a partir de uma imagem.
 - **Docker Container** – Tudo que é necessário para uma aplicação ser executada.
 - **Docker Engine** – Usado para criar imagens e containers.
 - **Docker Registry** – Uma coleção de imagens hospedadas, registro pode ser público ou privado.

- 
- **Docker Hub** – Registro usado para hospedar e baixar diversas imagens.
 - **Dockerfile** – Um arquivo texto contendo uma sintaxe simples para criação de novas imagens.
 - **Docker Compose** – Usado para definir aplicações usando diversos containers.
 - **Docker Swarm** – Ferramenta que permite o agrupamento (*clustering*) de Containers Docker.

O que são containers?



- O **container** é um ambiente isolado
- Tudo pode ser instalado no servidor e é armazenado nos containers
- Os mesmos softwares e as suas versões podem ter uma execução facilitada em qualquer ambiente de desenvolvimento

Exemplo: Um navio cargueiro pode carregar diversos containers. Caso um dos recipientes seja danificado, os demais não são afetados.

Afinal, são isolados, protegidos e estão carregando seus próprios produtos.

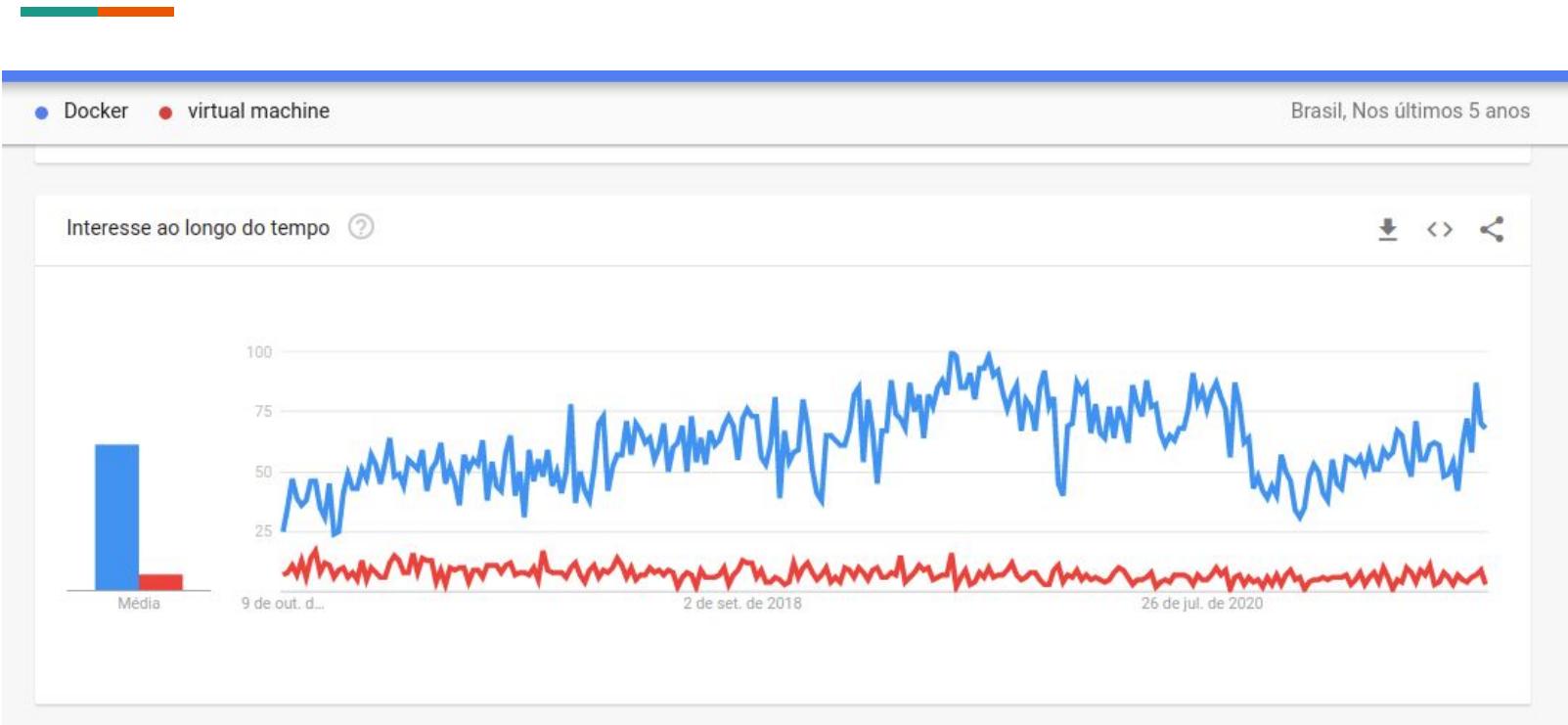
Agora no mundo do desenvolvimento, cada container possui uma função e sua responsabilidade. Caso um deles sofra um dano, o funcionamento do sistema não para e a função afetada é redirecionada para um novo container.



Alguns fatos sobre Docker

- Existem mais de **500.000.000 aplicações dockerizadas**, um crescimento de 3100% em 2 anos
- Mais de **8 bilhões de containers** já foram puxados (pull) até hoje
- Tem uma ampla comunidade de colaboradores e usuários
- Cresceu mais de 40% no último ano
- Cerca de 40% dos containers estão rodando em produção
- 35% das empresas que já ouviram falar em docker planejam usar

Comparação últimos 5 anos...



Stack OverFlow

The screenshot shows the Stack Overflow Tags page. The left sidebar includes links for Home, PUBLIC Questions, Tags (which is the active tab), Users, COLLECTIVES, FIND A JOB (Jobs and Companies), and TEAMS (Stack Overflow for Teams). The main content area has a search bar with 'docker' typed in, a 'Popular' sorting button, and a list of tags:

Tag	Description	Questions	Asked Today	This Week
docker	Docker is a tool to build and run containers. Questions concerning Dockerfiles, operations, and architecture are accepted. Questions...	103907	73	479
docker-compose	Compose is a tool for defining and running complex applications with Docker. With Compose, you define a multi-container application in a single...	22391	13	114
dockerfile	A Dockerfile is a file containing instructions to build a Docker image	11339	7	55
docker-swarm	Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual host.	2857	12	41
docker-machine	docker-machine creates Docker Engines on your computer, on cloud providers, and/or in your data center, and then configures the Docker client...	1575	10	137
docker-registry	Docker Registry is a service which you can push Docker images to for storage and sharing. It is also the tool's name.	1376	6	29
docker-volume	A docker data volume is a specially-designated directory within one or more containers that bypasses the Union File System.	1111	19	254
boot2docker	boot2docker is a lightweight Linux distribution based on Tiny Core Linux made specifically to run Docker containers. It runs completely from...	916		
docker-container	Docker Containers are the core of the docker platform in which programs and applications can be packaged and run in simulated environments.	908	7	18
docker-for-windows	Issues relating to running Windows or Linux containers on a Windows host by using Docker for Windows.	896	9	121
dockerhub	Docker Hub is a product by Docker Inc built on top of the open source Docker Registry to distribute Docker images.	790	8	130
docker-image	A Docker image is an inert, immutable, file that's essentially a snapshot of a Linux container. Images are created with the 'docker build' command or...	747	5	14

Algumas empresas que usam docker



Preço docker

monthly annual

Personal 1

Ideal for individual developers, education, open source communities, and small businesses.

\$0

- Docker Desktop ⓘ
- Unlimited public repositories
- Docker Engine + Kubernetes ⓘ
- Limited image pulls per day

[Start Now](#)

Pro 1

Includes pro tools for individual developers who want to accelerate their productivity.

\$5 /month

← Everything in Personal plus:

- Docker Desktop ⓘ
- Unlimited private repositories
- 5,000 image pulls per day
- 5 concurrent builds ⓘ
- 300 Hub vulnerability scans
- 5 scoped access tokens ⓘ

Billed annually for \$60.

[Buy Now](#)

TEAM 5+

Developer Favorite

Ideal for teams and includes capabilities for collaboration, productivity and security.

\$7 /user/month Start with minimum 5 users for \$35.

← Everything in Pro, plus:

- Docker Desktop ⓘ
- Unlimited teams
- 15 concurrent builds ⓘ
- Unlimited image scans
- Unlimited scoped tokens ⓘ
- Role-based access control
- Audit logs ⓘ

Billed annually starting at \$300.

[Buy Now](#)

Business 50+

Ideal for medium and large businesses who need centralized management and advanced security capabilities.

\$21 /user/month

← Everything in Team, plus:

- Docker Desktop ⓘ
- Centralized management
- Image Access Management
- SAML SSO *coming soon
- Purchase via invoice

Only available with an annual subscription.

[Contact Sales](#)

Docker Hub

- O Docker Hub é parecido com o nosso querido GitHub, porém somente de imagens Docker.
- Existem várias imagens prontas com serviços mais utilizados pra você baixar e usar
- Vale a pena brincar um pouco com a criação e utilização de imagens no Docker

Docker Hub

The Docker Hub homepage features a large blue header with the text "Build and Ship any Application Anywhere". Below the header is a sign-up form for new users. The form includes fields for "Docker ID", "Email", "Password", and a checkbox for "Send me occasional product updates and announcements". A reCAPTCHA verification is also present. At the bottom of the form is a prominent blue "Sign Up" button.

The Docker Hub search results page displays three popular Docker images:

- couchbase**: Official Image. Updated 11 days ago. Description: Couchbase Server is a NoSQL document database with a distributed architecture. Tags: Container, Linux, x86-64, Storage, Application Frameworks. Downloads: 50M+, Stars: 796.
- ubuntu**: Official Image. Updated 11 days ago. Description: Ubuntu is a Debian-based Linux operating system based on free software. Tags: Container, Linux, PowerPC 64 LE, ARM 64, IBM Z, x86-64, 386, riscv64, ARM, Base Images, Operating Systems. Downloads: 1B+, Stars: 10K+.
- redis**: Official Image. Updated 14 days ago. Description: Redis is an open source key-value store that functions as a data structure server. Tags: Container, Windows, Linux, mips64le, 386, PowerPC 64 LE, x86-64, ARM, IBM Z, ARM 64, Databases. Downloads: 1B+, Stars: 10K+.

Comandos relacionados ao Docker Hub

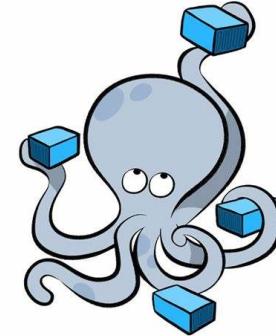
`docker login` - inicia o processo de login no Docker Hub.

`docker push NOME_USUARIO/NOME_IMAGEM` - envia a imagem criada para o Docker Hub.

`docker pull NOME_USUARIO/NOME_IMAGEM` - baixa a imagem desejada do Docker Hub.

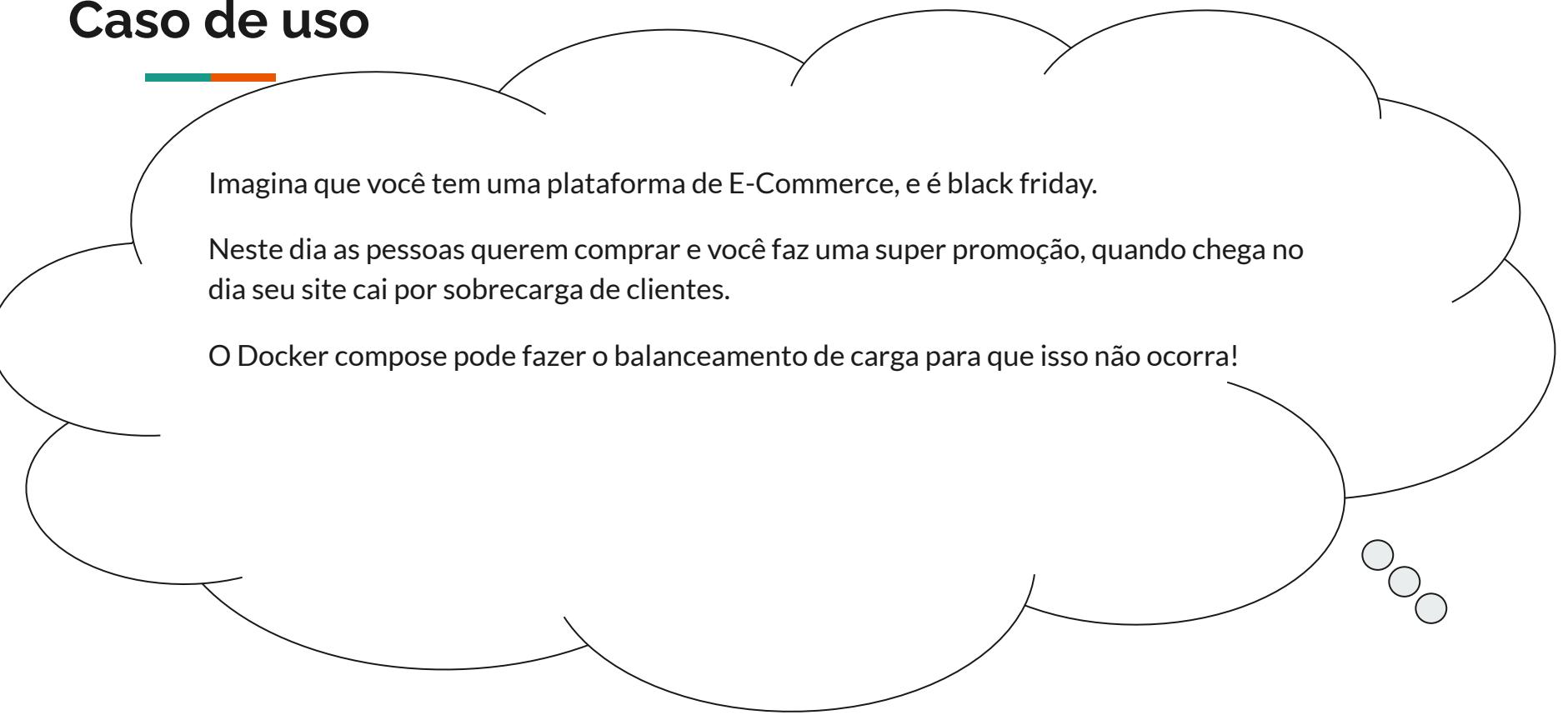
Docker compose

- É uma ferramenta para a criação e execução de múltiplos containers de aplicação.
- Arquivo do tipo **yaml**
- Com um único comando você criará e iniciará todos os serviços definidos



- O Compose (símbolo é um polvo) é uma ferramenta do Docker utilizado para orquestrar seus containers.
- Define como cada container deve se comportar na aplicação fazendo balanceamento de carga.

Caso de uso



Imagina que você tem uma plataforma de E-Commerce, e é black friday.

Neste dia as pessoas querem comprar e você faz uma super promoção, quando chega no dia seu site cai por sobrecarga de clientes.

O Docker compose pode fazer o平衡amento de carga para que isso não ocorra!

Docker compose



```
version: "3" ## especifica a versão do docker-compose file

services: ## Define um serviço
  app: ## nome do serviço
    build: . ## localização do dockerfile
    command: npm start ## comando a executar
    ports:
      - "3000:3000" ## redirecionamento de porta quando chegar alguma requisição na porta 3000 chama o container na porta 3000
    volumes:
      - .:/usr/app ## monitoro a pasta atual . e envio as alterações para /usr/app
```

```
docker run -it --rm -d -p 8080:80 --name web nginx
```

Parâmetros mais utilizados na execução do container

Parâmetro	Explicação
-d	Execução do container em background
-i	Modo interativo. Mantém o STDIN aberto mesmo sem console anexado
-t	Aloca uma pseudo TTY
--rm	Automaticamente remove o container após finalização (Não funciona com -d)
--name	Nomear o container
-v	Mapeamento de volume
-p	Mapeamento de porta
-m	Limitar o uso de memória RAM
-c	Balancear o uso de CPU

```
docker run -it --rm -d -p 8080:80 --name web nginx
```

Parâmetro	Explicação
-d	Execução do container em background
-i	Modo interativo. Mantém o STDIN aberto mesmo sem console anexado
-t	Aloca uma pseudo TTY
--rm	Automaticamente remove o container após finalização (Não funciona com -d)
--name	Nomear o container
-v	Mapeamento de volume
-p	Mapeamento de porta
-m	Limitar o uso de memória RAM
-c	Balancear o uso de CPU

Comandos relacionados à docker-compose

docker-compose build -d - Cria as imagens de acordo com o arquivo docker-compose.yaml

docker-compose up -d - Cria os serviços configurados no docker-compose.yaml

docker-compose ps - Mostras os serviços que estão rodando

docker-compose down - Para todos os serviços e remove os containers

docker-compose restart - Reinicia os containers

Docker images

- Ela é o template para rodar um container.
- Precisa baixar as imagens para rodar os containers ou pegar de algum lugar

Onde encontrar essas images

The screenshot shows the Docker Hub interface for the official Nginx image. At the top, there's a blue header bar with the Docker Hub logo, a search bar, and navigation links for Explore, Pricing, Sign In, and Sign Up. Below the header, the URL path is shown as /Explore/Official Images/nginx. The main content area features the Nginx logo and the text "nginx" followed by an "Official Image" badge and a star icon. A brief description states "Official build of Nginx." Below this, there's a download counter of "1B+" and a "Copy and paste to pull this image" button containing the command "docker pull nginx". Other options like "View Available Tags" are also present. At the bottom, there are tabs for "Description", "Reviews", and "Tags".

Baixando uma imagem docker

Para baixar uma imagem (vamos utilizar o exemplo da imagem do Ubuntu), utilize o comando:

```
docker pull nginx
```

Você pode listar as imagens baixadas em seu PC com o comando:

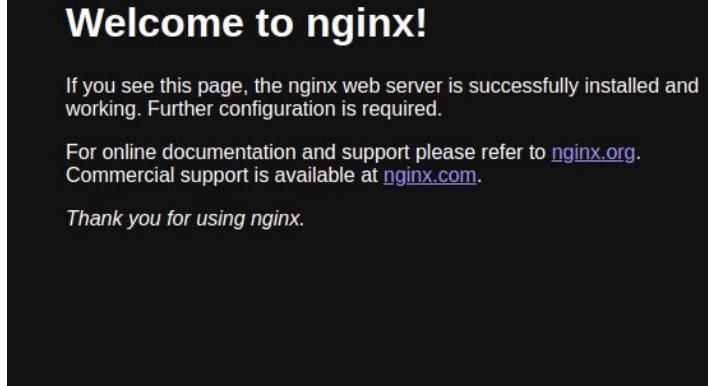
```
docker images
```

E usar a imagem para subir um container:

```
docker run -it --rm -d -p 8080:80 --name web nginx
```

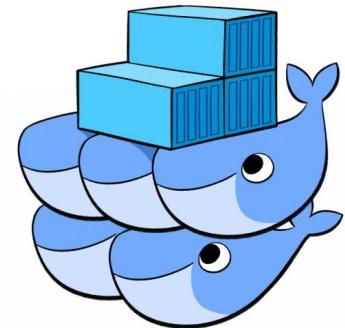
Acessando o nginx no navegador

```
http://localhost:8080/
```



Docker Swarm

- Docker swarm é uma ferramenta de orquestração de containers
- Permite ao usuário gerenciar vários containers implantados em várias máquinas host.
- Alto nível de disponibilidade oferecido para os aplicativos.



Docker Registry



- É um sistema de armazenamento, contendo imagens nomeadas do Docker, disponíveis em diferentes versões marcadas.

Exemplo: a distribuição / registro de imagens, com tags 2.0 e 2.1.

- Os usuários interagem com um registro usando comandos **docker push** e **pull**.

Docker File



- Nada mais é do que um meio que utilizamos para criar nossas próprias imagens.
- Serve como a receita para construir um container, permitindo definir um ambiente personalizado e próprio para seu projeto.

Comandos relacionados à construção de Dockerfile

`docker build -f Dockerfile` - Cria uma imagem a partir de um Dockerfile.

`docker build -f Dockerfile -t NOME_USUARIO/NOME_IMAGEM` - Constrói e nomeia uma imagem não-oficial.

`docker build -f Dockerfile -t NOME_USUARIO/NOME_IMAGEM CAMINHO_DOCKERFILE` - Constrói e nomeia uma imagem não-oficial informando o caminho para o Dockerfile.

Alguns comandos docker



Comandos relacionados à execução

docker run NOME_DA_IMAGEM - Cria um container com a respectiva imagem passada como parâmetro.

docker run -it NOME_DA_IMAGEM - Conecta o terminal que estamos utilizando com o do container.

docker run -d -P --name NOME dockersamples/static-site ao executar, dá um nome ao container.

O parâmetro **-d** utilizado juntamente com **run** significa que irá rodar em background e não vai travar o terminal até que o container morra.

Comandos relacionados à execução

docker run -d -p 12345:80 dockersamples/static-site - Define uma porta específica para ser atribuída à porta 80 do container, neste caso 12345.

docker run -v "CAMINHO_VOLUME" NOME_DA_IMAGEM - Cria um volume no respectivo caminho do container.

docker run -it --name NOME_CONTAINER --network NOME_DA_REDE NOME_IMAGEM - Cria um container especificando seu nome e qual rede deverá ser usada.

Comandos relacionados à inicialização/interrupção

`docker start ID_CONTAINER` - Inicia o container com id em questão.

`docker start -a -i ID_CONTAINER` - Inicia o container com id em questão e integra os terminais, além de permitir interação entre ambos.

`docker stop ID_CONTAINER` - Interrompe o container com id em questão.

Comandos relacionados à remoção

`docker rm ID_CONTAINER` remove o container com id.

`docker container prune` remove todos os containers que estão parados.

`docker rmi NOME_DA_IMAGEM` remove a imagem passada como parâmetro.

Comandos relacionados à rede

`docker network ls` - Lista todos as redes

`docker network create --driver <DRIVER> <NOME>` - Cria uma rede.

Exemplo: `docker network create --driver bridge alpine-net`

`docker run -d --name <NOME_DA_IMAGEM> --network <NOME DA REDE> <IMAGEM>` - Atribuindo um container a uma rede.

Exemplo: `docker run -d --name nginx03 --network alpine-net nginx:alpine`



```
docker network inspect <NOME/ID REDE> - Exibe informações de uma rede
```

```
docker network prune - Remove todas as redes não usadas
```

```
docker network rm <NOME/ID REDE> - Remove uma ou mais redes
```

Comandos relacionados volumes

`docker volume ls` - Lista todos os volumes

`docker volume rm <NOME VOLUME>` - Remove um ou mais volumes

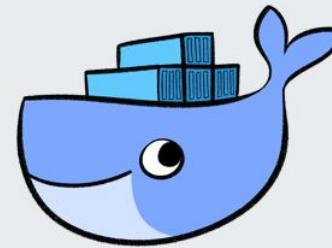
`docker volume inspect <NOME VOLUME>` - Exibe informações volume

`docker volume create <NOME VOLUME>` - Cria um volume

`docker volume prune` - Remove todos os volumes não usados

Hands-on Docker play site

[Contribute](#)



Play with Docker

A simple, interactive and fun playground to learn Docker

[Start](#)

Docker play site

The screenshot shows the Docker play site interface. On the left, there is a sidebar with a blue header containing a digital clock display showing "03:59:47". Below the clock is an orange button labeled "CLOSE SESSION". Underneath the clock are two buttons: "Instances" and a search/filter icon. At the bottom of the sidebar is a link "+ ADD NEW INSTANCE". To the right of the sidebar is a main content area. At the top of this area is a message "Add instances to your playground." Below it is another message stating "Sessions and all their instances are deleted after 03:59:47 hours."

Docker play site

```
docker swarm init --advertise-addr "ip da instância"
```

```
[node1] (local) root@192.168.0.18 ~  
$ docker swarm init --advertise-addr 192.168.0.18  
Swarm initialized: current node (zjuueeuesuu94psq9jfi2xe71) is now a manager.
```

To add a worker to this swarm, run the following command:

```
    docker swarm join --token SWMTKN-1-0mk963xf9d1hl341pwkkdbwbcwh6bb4v7akqs71iyyidqyuurh-cfr3vlajirrei5  
jxcilktxxe 192.168.0.18:2377
```

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

Docker play site

```
docker node ls
```

```
[node1] (local) root@192.168.0.18 ~
$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS  ENGINE VERSION
zjuueeuessu94psq9jfi2xe71 *  node1    Ready   Active        Leader  20.10.0
```

Docker play site

```
git clone https://github.com/WeslleyFratini/docker-repo.git
```

```
[node1] (local) root@192.168.0.18 ~
$ git clone https://github.com/WeslleyFratini/docker-repo.git
Cloning into 'docker-repo'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 14 (delta 0), reused 14 (delta 0), pack-reused 0
Receiving objects: 100% (14/14), done.
```

Docker play site

```
cd docker-repo  
ls -la
```

```
[node1] (local) root@192.168.0.18 ~  
$ cd docker-repo/  
[node1] (local) root@192.168.0.18 ~/docker-repo  
$ ls -la  
total 16  
drwxr-xr-x  5 root    root        137 Oct 27 00:55 .  
drwx-----  1 root    root        25 Oct 27 00:55 ..  
drwxr-xr-x  8 root    root       163 Oct 27 00:55 .git  
-rw-r--r--  1 root    root       226 Oct 27 00:55 Dockerfile  
-rw-r--r--  1 root    root       192 Oct 27 00:55 README.md  
-rw-r--r--  1 root    root      567 Oct 27 00:55 docker-compose-prd.yml  
-rw-r--r--  1 root    root      119 Oct 27 00:55 docker-compose.yml  
drwxr-xr-x  3 root    root        23 Oct 27 00:55 projeto  
drwxr-xr-x  2 root    root        56 Oct 27 00:55 stack
```

Docker play site

```
cd stack  
cat stack.yml
```

```
[node1] (local) root@192.168.0.18 ~/docker-repo  
$ cd stack/  
[node1] (local) root@192.168.0.18 ~/docker-repo/stack  
$ cat  
.gitkeep README.md stack.yml  
[node1] (local) root@192.168.0.18 ~/docker-repo/stack  
$ cat stack.yml  
version: '3.1'  
  
services:  
  ucp-installer:  
    image: alpine  
    command: docker run --rm --name ucp -v /var/run/docker.sock:/var/run/docker.sock docker/ucp:2.2.5 install --force-insecure-tcp --san *.${PWD_HOST_FQDN} --admin-username ad  
min --admin-password admin1234  
    volumes:  
      - /var/run/docker.sock:/var/run/docker.sock  
      - /usr/local/bin/docker:/usr/local/bin/docker  
  deploy:  
    restart_policy:  
      condition: 'none'
```

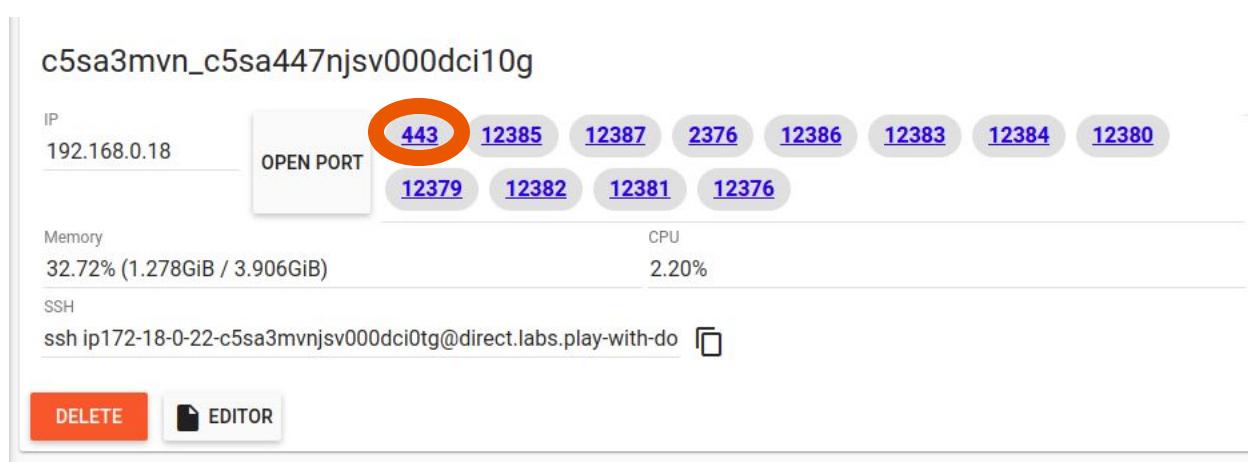
Docker play site

UCP (Universal Control Plane), também da Docker, que nada mais é que um Dashboard no qual é possível interagir com a API do Docker de forma visual, e também é um container.

```
docker stack deploy -c stack.yml ucp
```

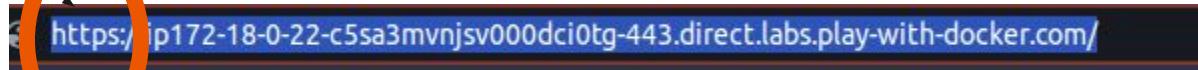
```
[node1] (local) root@192.168.0.18 ~/docker-repo/stack
$ docker stack deploy -c stack.yml ucp
Creating network ucp_default
Creating service ucp_ucp-installer
```

Docker play site



Docker play site

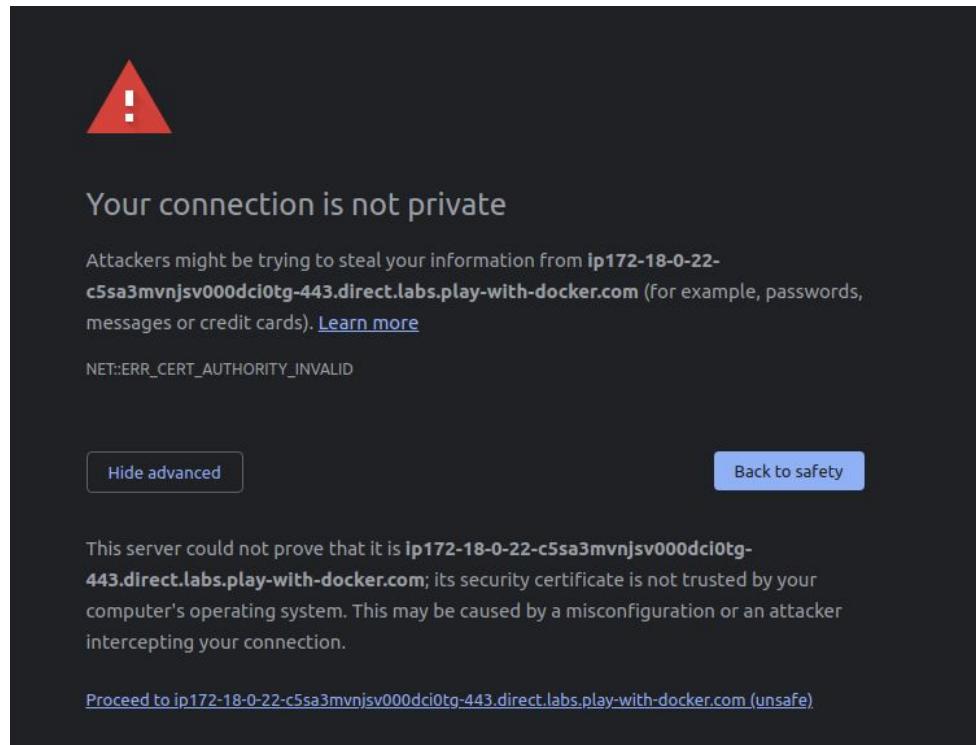
Adicionar o **HTTPS**



https://ip172-18-0-22-c5sa3mvnjsv000dc10tg-443.direct.labs.play-with-docker.com/

A screenshot of a web browser's address bar. The URL displayed is "https://ip172-18-0-22-c5sa3mvnjsv000dc10tg-443.direct.labs.play-with-docker.com/". A large orange circle highlights the "https://" prefix, which is the subject of the annotation above.

Docker play site

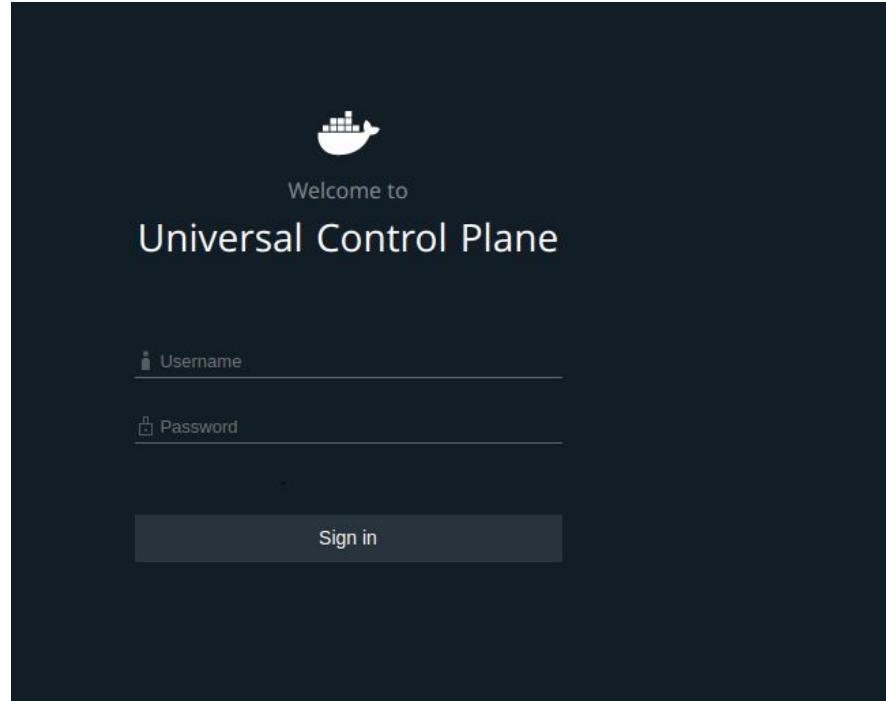


Docker play site

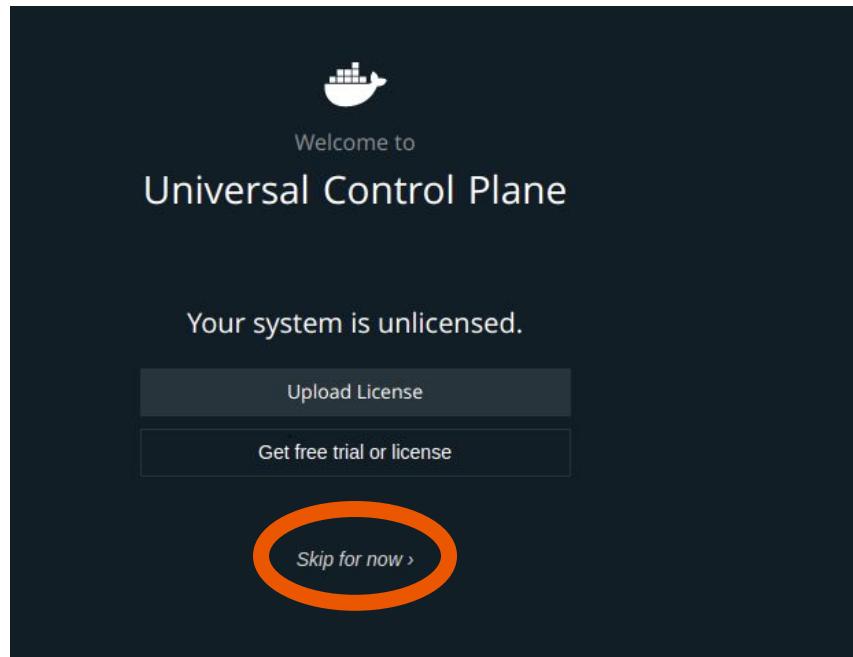
Login:

usuário: admin

senha: admin1234



Docker play site



Docker play site

New updates are available for UCP 2.2.10, 2.2.11, 2.2.12, 2.2.13, 2.2.14, 2.2.15, 2.2.16, 2.2.17, 2.2.18, 2.2.19, 2.2.20, 2.2.21, 2.2.22, 2.2.6, 2.2.7, 2.2.9, 3.0.0, 3.0.1, 3.0.10, 3.0.11, 3.0.12, 3.0.13, 3.0.14, 3.0.15, 3.0.16, 3.0.17, 3.0.18, 3.0.2, 3.0.3, 3.0.4, 3.0.5, 3.0.6, 3.0.7, 3.0.8, 3.0.9, 3.1.0, 3.1.1, 3.1.10, 3.1.11, 3.1.12, 3.1.13, 3.1.14, 3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.6, 3.1.7, 3.1.8, 3.1.9, 3.2.0, 3.2.1, 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.2.7, 3.3.0, 3.3.1: <https://ip172-18-0-22.cs3mvnjsv000dc10tg-443.direct.labs.play-with-docker.com/manage/settings/upgrade>

Unauthorized users can access the following node since it is listening on an external port: node1:2375. Learn more at <https://www.docker.com/ddc-18>

Your system is unlicensed. Please visit <https://store.docker.com/editions/enterprise/docker-ee-trial> to get a free trial or purchase a monthly license subscription.

admin

User Management

Dashboard

Collections

Stacks

Services 4

Containers 14

Images 1

Nodes 1

Networks 6

Volumes 14

Secrets 0

1 Service

Actions Create Service

Status	Name	Image	Mode	Last Updated	Last Error
0/0	ucp_ucp-installer	alpine:latest@sha256:e1c082e3d3c45cc...	Replicated	7 minutes ago	No errors

Docker play site

```
cd ..  
ls -la
```

```
[node1] (local) root@192.168.0.18 ~/docker-repo/stack  
$ cd ..  
[node1] (local) root@192.168.0.18 ~/docker-repo  
$ ls -la  
total 16  
drwxr-xr-x  5 root      root          137 Oct 27 00:55 .  
drwx-----  1 root      root          25  Oct 27 00:55 ..  
drwxr-xr-x  8 root      root         163  Oct 27 00:55 .git  
-rw-r--r--  1 root      root         226  Oct 27 00:55 Dockerfile  
-rw-r--r--  1 root      root         192  Oct 27 00:55 README.md  
-rw-r--r--  1 root      root        567  Oct 27 00:55 docker-compose-prd.yml  
-rw-r--r--  1 root      root        119  Oct 27 00:55 docker-compose.yml  
drwxr-xr-x  3 root      root         23  Oct 27 00:55 projeto  
drwxr-xr-x  2 root      root         56  Oct 27 00:55 stack
```

Docker play site

```
docker build -t bbnginx:1.0 .
```

```
[node1] (local) root@192.168.0.18 ~/docker-repo
$ docker build -t bbnginx:1.0 .
Sending build context to Docker daemon    64kB
Step 1/4 : FROM nginx:alpine
alpine: Pulling from library/nginx
a0d0a0d46f8b: Already exists
4dd4efe90939: Pull complete
c1368e94e1ec: Pull complete
3e72c40d0ff4: Pull complete
969825a5ca61: Pull complete
61074acc7dd2: Pull complete
Digest: sha256:686aac2769fd6e7bab67663fd38750c135b72d993d0bb0a942ab02ef647fc9c3
Status: Downloaded newer image for nginx:alpine
    --> 513f9a9d8748
Step 2/4 : COPY projeto/minicurso/ /usr/share/nginx/html
    --> 55488d7d6749
Step 3/4 : CMD cd /usr/share/nginx/html && sed -e s/hostname/${hostname}/ -i index.html && sed -e s/end
ip/$(hostname -i)/ -i index.html; nginx -g 'daemon off;';
    --> Running in 3683c20e3d85
Removing intermediate container 3683c20e3d85
    --> 867164b8f1fe
Step 4/4 : EXPOSE 80
    --> Running in 7b418a2fd487
Removing intermediate container 7b418a2fd487
    --> 8c4b2346c7f4
Successfully built 8c4b2346c7f4
Successfully tagged bbnginx:1.0
```

Docker play site

```
docker service create --publish 80:80 bbnginx:1.0
```

```
[node1] (local) root@192.168.0.18 ~/docker-repo
$ docker service create --publish 80:80 bbnginx:1.0
image bbnginx:1.0 could not be accessed on a registry to record
its digest. Each node will access bbnginx:1.0 independently,
possibly leading to different nodes running different
versions of the image.

m2d1gj0vsxtqjhanty4srtmg
overall progress: 0 out of 1 tasks
overall progress: 1 out of 1 tasks
1/1: running
verify: Service converged
```

Docker play site

c5sa3mvn_c5sa447njsv000dci10g

IP 192.168.0.18	OPEN PORT	443	12385	12387	2376	12386	12383	12384	12380	80
Memory 35.92% (1.403GiB / 3.906GiB)		CPU 2.05%								
SSH <code>ssh ip172-18-0-22-c5sa3mvnjsv000dci0tg@direct.labs.play-with-d</code>		<input type="button" value=""/>								
<input type="button" value="DELETE"/>	<input type="button" value="EDITOR"/>									

Docker play site



Docker Cheat Sheet



Build

Build an image from the Dockerfile in the current directory and tag the image

```
docker build -t myimage:1.0 .
```

List all images that are locally stored with the Docker Engine

```
docker image ls
```

Delete an image from the local image store

```
docker image rm alpine:3.4
```



Share

Pull an image from a registry

```
docker pull myimage:1.0
```

Tag a local image with a new image name and tag

```
docker tag myimage:1.0 myrepo/myimage:2.0
```

Push an image to a registry

```
docker push myrepo/myimage:2.0
```



Run a container from the Alpine version 3.9 image, name the running container "web" and expose port 5000 externally, mapped to port 80 inside the container.

```
docker container run --name web -p 5000:80 alpine:3.9
```

Stop a running container through SIGTERM

```
docker container stop web
```

Stop a running container through SIGKILL

```
docker container kill web
```

List the networks

```
docker network ls
```



www.docker.com

Docker Management

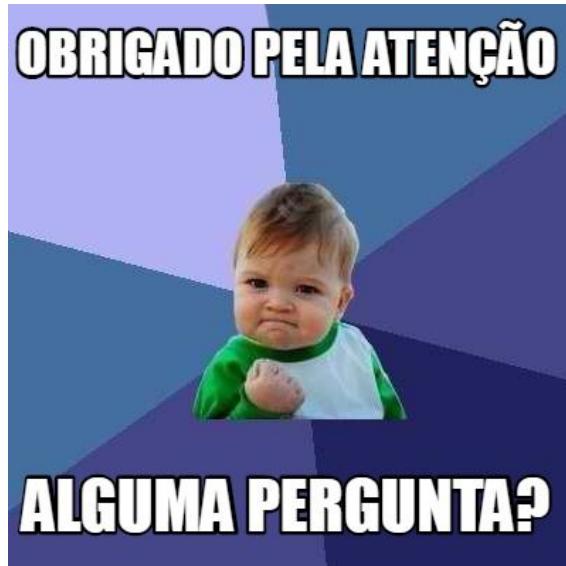
All commands below are called as options to the base `docker` command. Run `docker <command> --help` for more information on a particular command.

<code>app*</code>	<i>Docker Application</i>
<code>assemble*</code>	<i>Framework-aware builds (Docker Enterprise)</i>
<code>builder</code>	<i>Manage builds</i>
<code>cluster</code>	<i>Manage Docker clusters (Docker Enterprise)</i>
<code>config</code>	<i>Manage Docker configs</i>
<code>context</code>	<i>Manage contexts</i>
<code>engine</code>	<i>Manage the docker Engine</i>
<code>image</code>	<i>Manage images</i>
<code>network</code>	<i>Manage networks</i>
<code>node</code>	<i>Manage Swarm nodes</i>
<code>plugin</code>	<i>Manage plugins</i>
<code>registry*</code>	<i>Manage Docker registries</i>
<code>secret</code>	<i>Manage Docker secrets</i>
<code>service</code>	<i>Manage services</i>
<code>stack</code>	<i>Manage Docker stacks</i>
<code>swarm</code>	<i>Manage swarm</i>
<code>system</code>	<i>Manage Docker</i>
<code>template*</code>	<i>Quickly scaffold services (Docker Enterprise)</i>
<code>trust</code>	<i>Manage trust on Docker images</i>
<code>volume</code>	<i>Manage volumes</i>

*Experimental in Docker Enterprise 3.0.

Obrigado!

-  Linkedin: <https://www.linkedin.com/in/weslley-fratini/>
-  Email: weeslleey354@gmail.com
-  Twitter: https://twitter.com/Weslley_Fratini
-  Medium: <https://weslleyfratini.medium.com/>
-  Github: <https://github.com/weslleyfratini>



Docker play site

<https://imasters.com.br/desenvolvimento/vamos-conhecer-o-docker-swarm>

<https://www.onworks.net/programs/terminal-online>