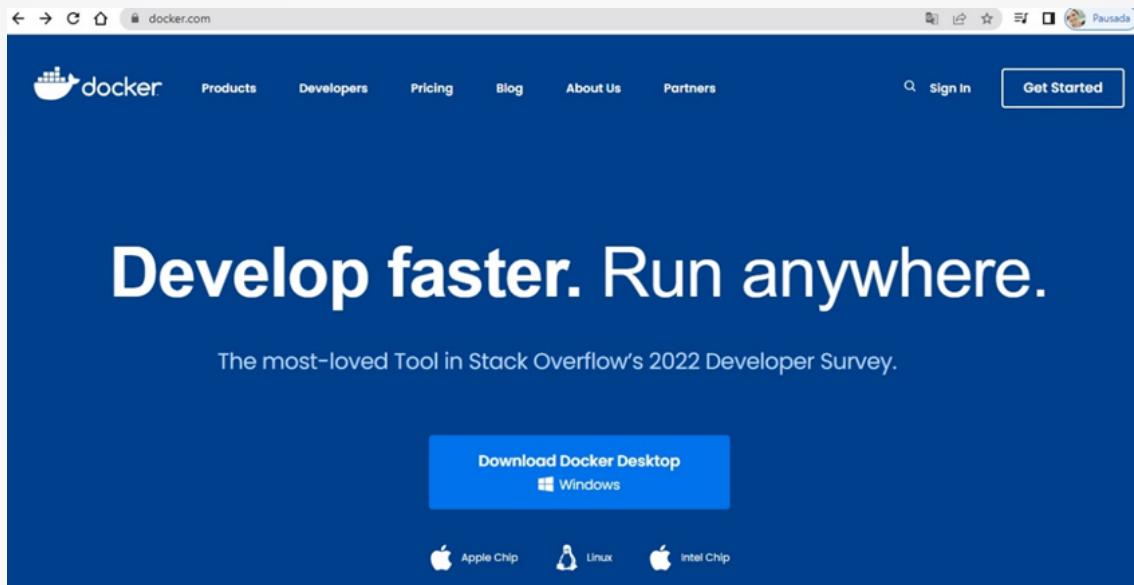


TUTORIAL: DOCKER HUB

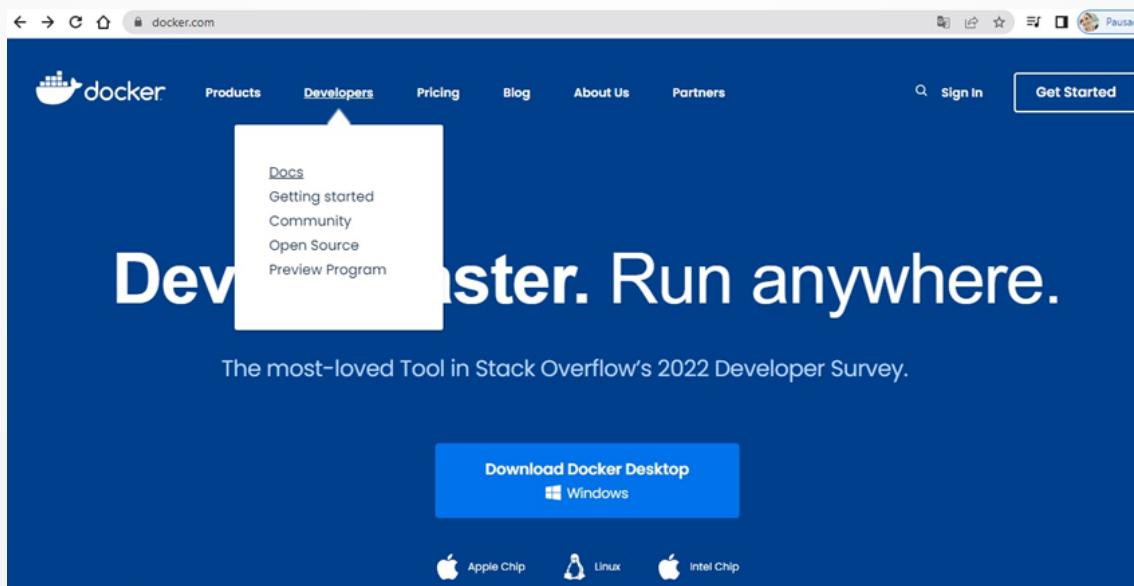
DevOps básico

Criação de contas/containers

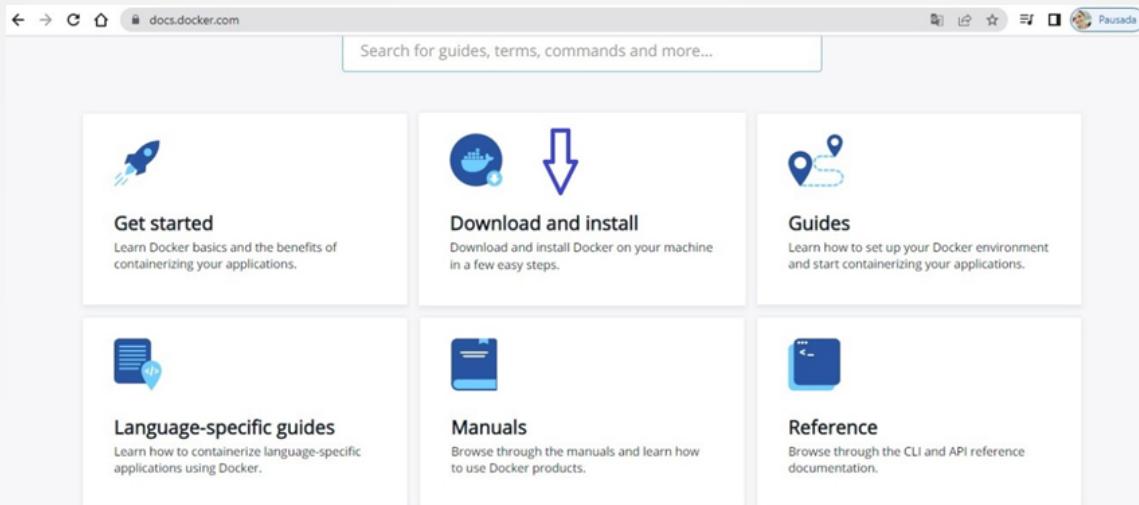
Para instalar o Docker, entre em www.docker.com:



Clique em Developers -> Docs:



Clique em Download Install:



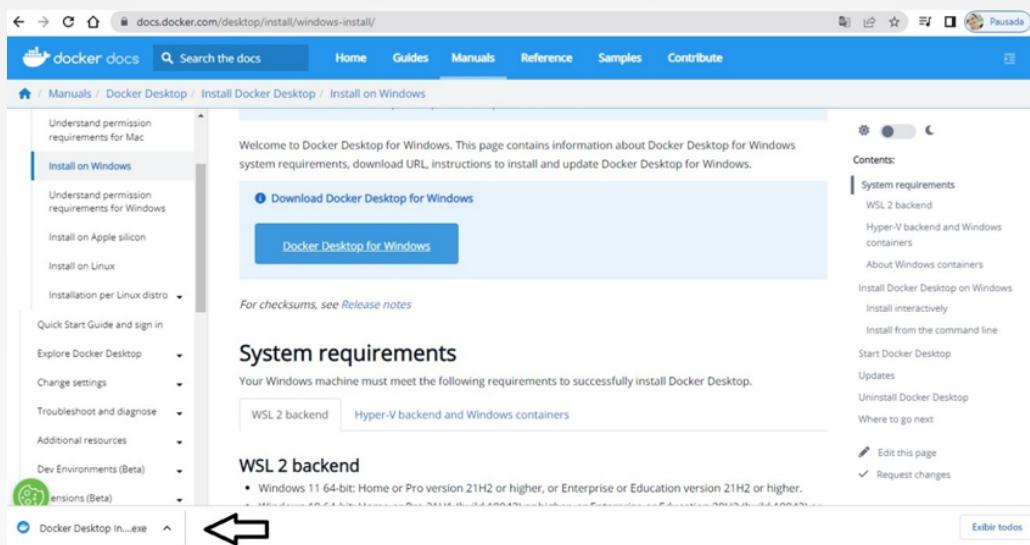
Clique no Docker para o S.O. selecionado para instalar:

A screenshot of the 'Get Docker' page at docs.docker.com/get-docker/. The left sidebar shows a dropdown menu under 'Get Docker' with options like 'Get started', 'Language-specific guides', and 'Docker Desktop for Windows'. The main content area has a heading 'Docker is an open platform for developing, shipping, and running applications.' followed by a paragraph about Docker's methodology. Below this are three sections: 'Docker Desktop for Mac' (Mac icon), 'Docker Desktop for Windows' (Windows icon with a downward arrow), and 'Docker Desktop for Linux' (Linux icon). The 'Docker Desktop for Windows' section includes a brief description and links to download URLs.

Neste caso, o Docker selecionado foi o Desktop para Windows:

A screenshot of the 'Install Docker Desktop / Install on Windows' page at docs.docker.com/desktop/install/windows-install/. The left sidebar shows a dropdown menu under 'Install on Windows' with options like 'Understand permission requirements for Mac', 'Understand permission requirements for Windows', and 'Download Docker Desktop for Windows'. The main content area has a heading 'Docker Desktop terms' with a note about paid subscriptions, a section 'Download Docker Desktop for Windows' with a prominent blue button labeled 'Docker Desktop for Windows', and a 'System requirements' section. The sidebar on the right lists various Docker-related topics such as 'System requirements', 'WSL 2 backend', 'Hyper-V backend and Windows containers', and 'About Windows containers'.

Faça o download do Docker:



Importante: No S.O. Windows, é necessário instalar o WSL2 na sua máquina:

- WSL 2 back-end Windows 11 de 64 bits: Home ou Pro versão 21H2 ou superior, ou Enterprise ou Education versão 21H2 ou superior;
- Windows 10 de 64 bits: Home ou Pro 21H1 (build 19043) ou superior, ou Enterprise ou Education 20H2 (build 19042) ou superior;
- Habilite o recurso WSL 2 no Windows. Para obter instruções detalhadas, consulte a documentação da Microsoft;
- Os seguintes pré-requisitos de hardware são necessários para executar com êxito o WSL 2 no Windows 10 ou Windows 11:
 - Processador de 64 bits com tradução de endereço de segundo nível (SLAT);
 - 4GB de RAM do sistema;
 - O suporte à virtualização de hardware no nível do BIOS deve ser ativado nas configurações do BIOS. Para obter mais informações, consulte Virtualização;
 - Baixe e instale o pacote de atualização do kernel Linux.

Obs.: O Docker só oferece suporte ao Docker Desktop no Windows para as versões do Windows 10 que ainda estão dentro do cronograma de manutenção da Microsoft.

Instalação do WSL:

Para verificar a versão do Windows e o número da compilação, selecione a tecla do logotipo do Windows + R, digite winver e selecione OK. Você pode atualizar para a versão mais recente do Windows selecionando Iniciar > Configurações > Windows Update > Verificar atualizações.

Se você estiver executando uma compilação mais antiga ou apenas preferir não usar o comando de instalação e quiser instruções passo a passo, consulte as etapas de instalação manual do WSL para versões mais antigas.

Instalar o comando WSL:

Instale tudo o que precisa para executar o Windows Subsystem for Linux (WSL)

digitando este comando em um administrador do PowerShell ou do prompt de comando do Windows e, em seguida, reiniciando sua máquina.

wsl -install

Este comando habilitará os recursos necessários para executar o WSL e instalar a distribuição Ubuntu do Linux. (Esta distribuição padrão pode ser alterada).

Na primeira vez que você iniciar uma distribuição Linux recém-instalada, uma janela do console será aberta e você será solicitado a aguardar que os arquivos sejam descompactados e armazenados em sua máquina. Todos os lançamentos futuros devem levar menos de um segundo.

Obs.: O comando acima só funciona se o WSL não estiver instalado, se você executar **wsl --install** e ver o texto de ajuda do WSL, tente executar **wsl --list --online** para ver uma lista de distros disponíveis e execute **wsl --install -d Ubuntu** para instalar uma distribuição.

Altere a distribuição padrão do Linux instalada:

Por padrão, a distribuição Linux instalada será o Ubuntu. Isso pode ser alterado usando o sinalizador **-d**.

Para alterar a distribuição instalada, digite: **wsl --install -d Debian**. Substitua <Nome da Distribuição> pelo nome da distribuição que você deseja instalar.

Para ver uma lista de distribuições Linux disponíveis para download na loja online, digite: **wsl --list --online ou wsl -l -o**.

Para instalar distribuições Linux adicionais após a instalação inicial, você também pode usar o comando: **wsl --install -d Debian**.

Dica: Se você deseja instalar distribuições adicionais de dentro de uma linha de comando Linux/Bash (em vez do PowerShell ou Prompt de Comando), você deve usar .exe no comando: **wsl.exe --install -d Debian** ou para listar disponível distribuições: **wsl.exe -l -o**.

Se você tiver um problema durante o processo de instalação, verifique a seção de instalação do guia de solução de problemas.

Para instalar uma distribuição Linux que não esteja listada como disponível, você pode importar qualquer distribuição Linux usando um arquivo TAR. Ou, em alguns casos, como no Arch Linux, você pode instalar usando um arquivo .appx. Você também pode criar sua própria distribuição Linux personalizada para usar com o WSL.

Configure suas informações de usuário do Linux:

Depois de instalar o WSL, você precisará criar uma conta de usuário e senha para sua distribuição Linux recém-instalada. Consulte as práticas recomendadas para configurar um guia de ambiente de desenvolvimento WSL para saber mais.

Configuração e práticas recomendadas:

Recomendamos seguir nossas práticas recomendadas para configurar um guia de ambiente de desenvolvimento WSL para obter um passo a passo de como configurar um nome de usuário e senha para suas distribuições Linux instaladas, usando comandos WSL básicos, instalando e personalização do Windows Terminal,

configuração para controle de versão do Git, edição e depuração de código usando o servidor remoto do VS Code, boas práticas para armazenamento de arquivos, configuração de um banco de dados, montagem de uma unidade externa, configuração de aceleração de GPU e muito mais.

Verifique qual versão do WSL você está executando:

Você pode listar suas distribuições Linux instaladas e verificar a versão do WSL para a qual cada uma está definida digitando o comando: **wsl -l -v** no **PowerShell** ou no prompt de comando do Windows.

Para definir a versão padrão para WSL 1 ou WSL 2 quando uma nova distribuição Linux for instalada, use o comando: **wsl --set-default-version <Version#>**, substituindo <Version#> por 1 ou 2.

Para definir a distribuição Linux padrão usada com o comando wsl, digite: **wsl -s <DistributionName>** ou **wsl --setdefault <DistributionName>**, substituindo <DistributionName> pelo nome da distribuição Linux que você gostaria de usar. Por exemplo, no PowerShell/CMD, digite: **wsl -s Debian** para definir a distribuição padrão como Debian. Agora, executar **wsl npm init** do Powershell executará o comando **npm init no Debian**.

Para executar uma distribuição wsl específica de dentro do PowerShell ou do prompt de comando do Windows sem alterar sua distribuição padrão, use o comando: **wsl -d <DistributionName>**, substituindo <DistributionName> pelo nome da distribuição que você deseja usar.

Atualize a versão do WSL 1 para o WSL 2:

Novas instalações do Linux, instaladas usando o comando **wsl --install**, serão definidas como WSL 2 por padrão.

O comando **wsl --set-version** pode ser usado para fazer downgrade de WSL 2 para WSL 1 ou para atualizar distribuições Linux instaladas anteriormente de WSL 1 para WSL 2.

Para ver se sua distribuição Linux está configurada para WSL 1 ou WSL 2, use o comando: **wsl -l -v**.

Para alterar as versões, use o comando: **wsl --set-version <distro name> 2** substituindo <distro name> pelo nome da distribuição Linux que você deseja atualizar. Por exemplo, **wsl --set-version Ubuntu-20.04 2** configurará sua distribuição Ubuntu 20.04 para usar WSL 2.

Se você instalou manualmente o WSL antes de o comando **wsl --install** estar disponível, talvez seja necessário habilitar o componente opcional da máquina virtual usado pelo WSL 2 e instalar o pacote do kernel, caso ainda não tenha feito isso.

Maneiras de executar várias distribuições Linux com WSL:

O WSL suporta a execução de tantas distribuições Linux diferentes quanto você deseja instalar. Isso pode incluir escolher distribuições da Microsoft Store, importar uma distribuição personalizada ou criar sua própria distribuição personalizada.

Existem várias maneiras de executar suas distribuições Linux depois de instaladas:

Instalar o Windows Terminal (recomendado). O uso do Windows Terminal oferece suporte a quantas linhas de comando você deseja instalar e permite abri-las em várias guias ou painéis de janela e alternar rapidamente entre várias distribuições Linux ou outras linhas de comando (PowerShell, Prompt de Comando, PowerShell , CLI do Azure etc.). Você pode personalizar totalmente seu terminal com esquemas de cores exclusivos, estilos de fonte, tamanhos, imagens de fundo e atalhos de teclado personalizados.

Saiba mais:

Você pode abrir diretamente sua distribuição Linux visitando o menu Iniciar do Windows e digitando o nome de suas distribuições instaladas. Por exemplo: “Ubuntu”. Isso abrirá o Ubuntu em sua própria janela de console.

No prompt de comando do Windows ou no PowerShell, você pode inserir o nome da sua distribuição instalada. Por exemplo: ubuntu.

No prompt de comando do Windows ou no PowerShell, você pode abrir sua distribuição padrão do Linux dentro da linha de comando atual, digitando: wsl.exe.

No prompt de comando do Windows ou no PowerShell, você pode usar sua distribuição padrão do Linux dentro de sua linha de comando atual, sem inserir uma nova, digitando: **wsl [command]**. Substituindo [command] por um comando WSL, como: **wsl -l -v** para listar as distribuições instaladas ou **wsl pwd** para ver onde o caminho do diretório atual está montado em wsl. No PowerShell, o comando **get-date** fornecerá a data do sistema de arquivos do Windows e wsl date fornecerá a data do sistema de arquivos do Linux.

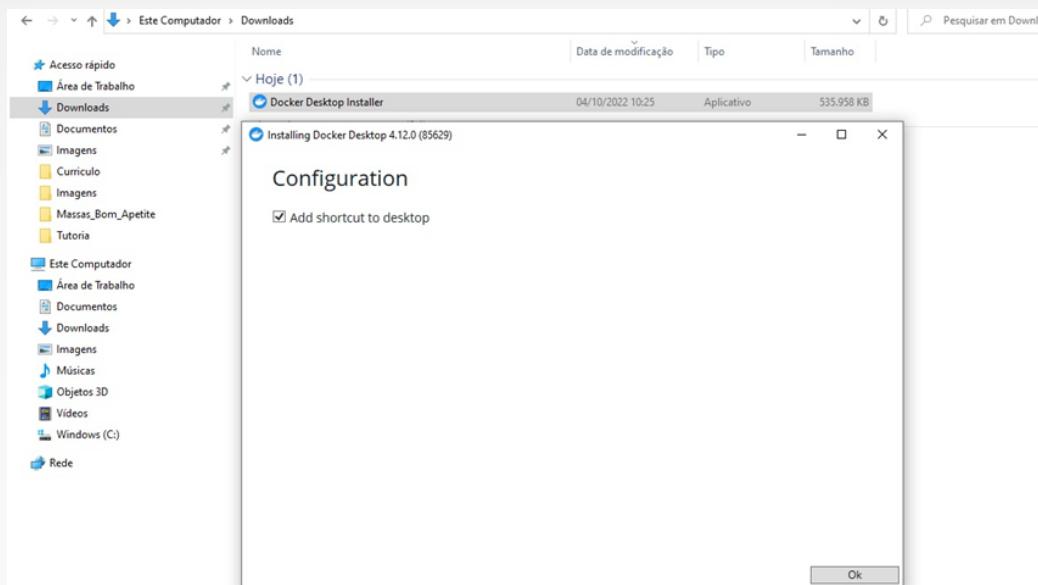
O método selecionado deve depender do que você está fazendo. Se você abriu uma linha de comando WSL em uma janela do Windows Prompt ou PowerShell e deseja sair, digite o comando: exit.

Experimente os recursos de visualização mais recentes da WSL:

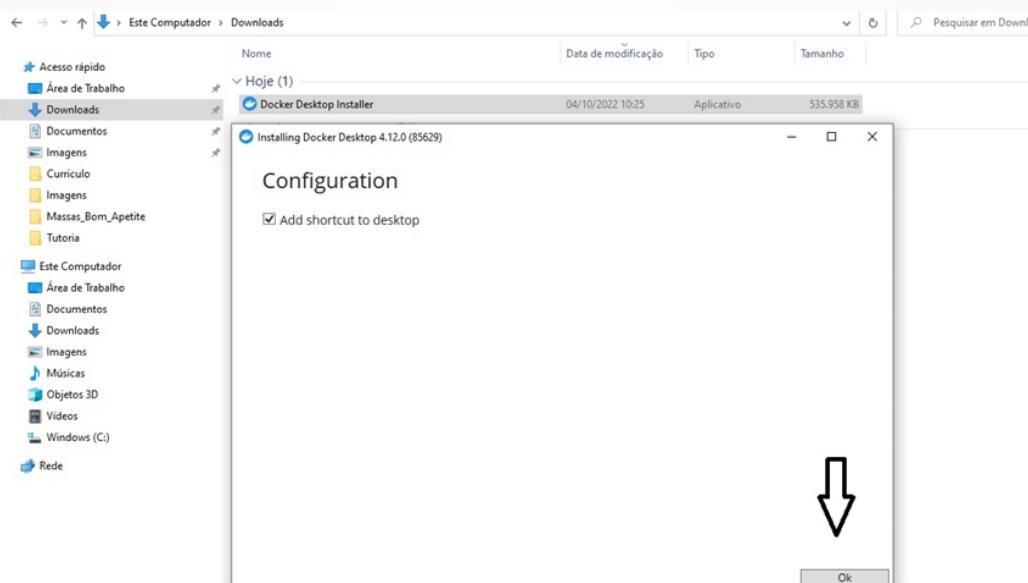
Experimente os recursos ou atualizações mais recentes do WSL ingressando no Programa Windows Insiders. Depois de ingressar no Windows Insiders, você pode escolher o canal que gostaria de receber as compilações de visualização no menu de configurações do Windows para receber automaticamente todas as atualizações do WSL ou recursos de visualização associados a essa compilação. Você pode escolher entre:

- Canal dev: atualizações mais recentes, mas baixa estabilidade;
- Canal beta: ideal para early adopters, compilações mais confiáveis do que o canal Dev;
- Canal Release Preview: pré-visualize as correções e os principais recursos da próxima versão do Windows, pouco antes de estar disponível para o público em geral.

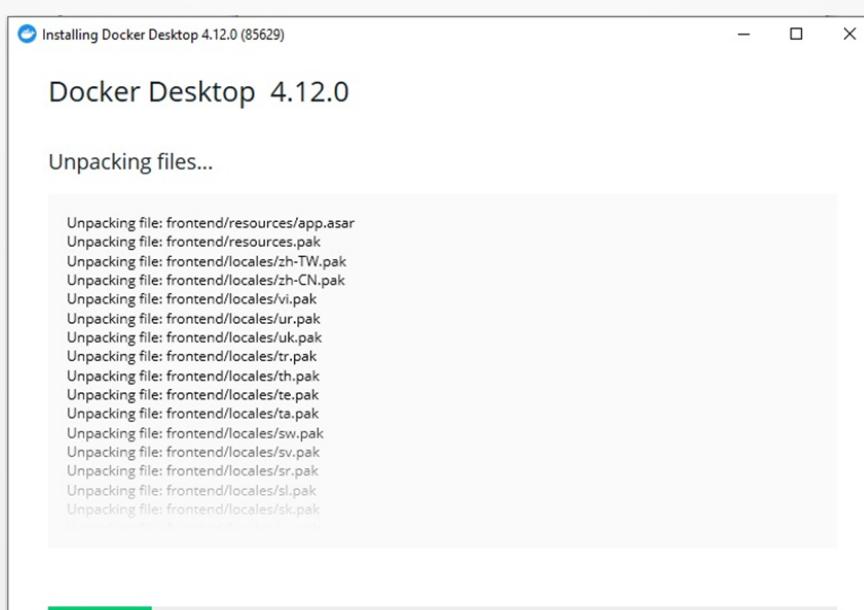
Clique em Docker Desktop Install salvo na sua máquina:



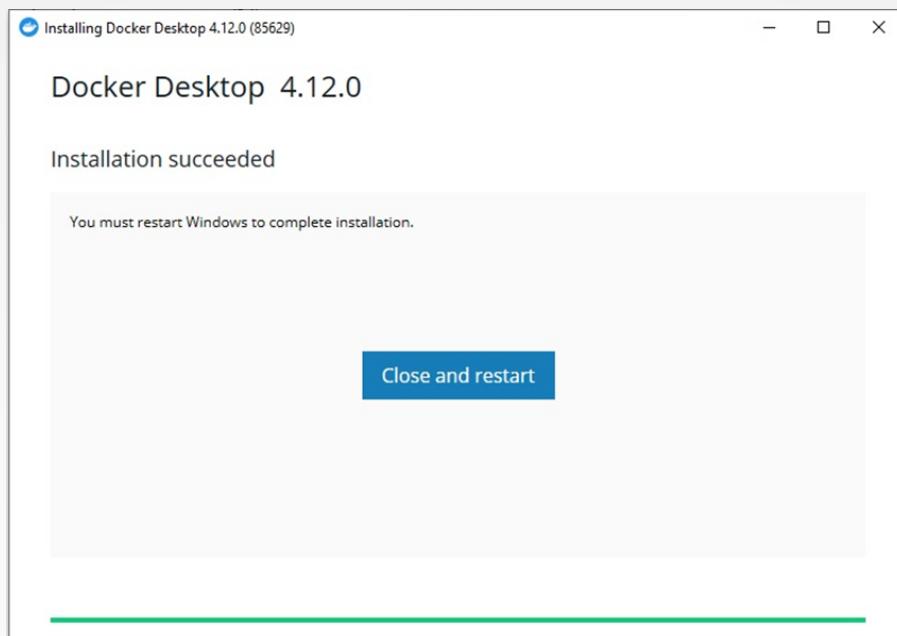
Clique em OK:



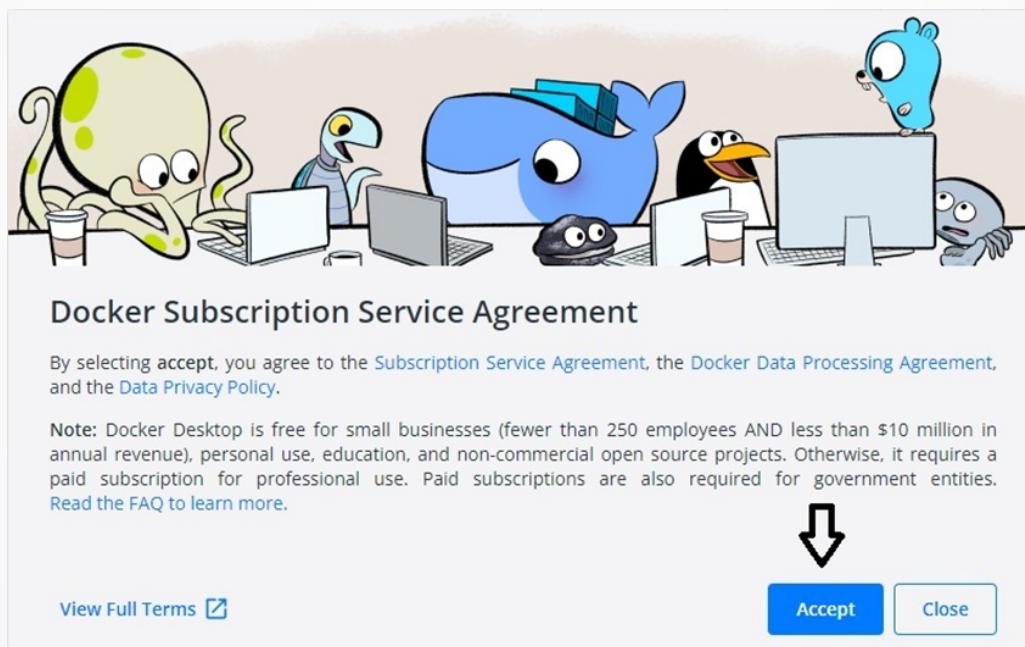
O Docker Desktop será instalado:



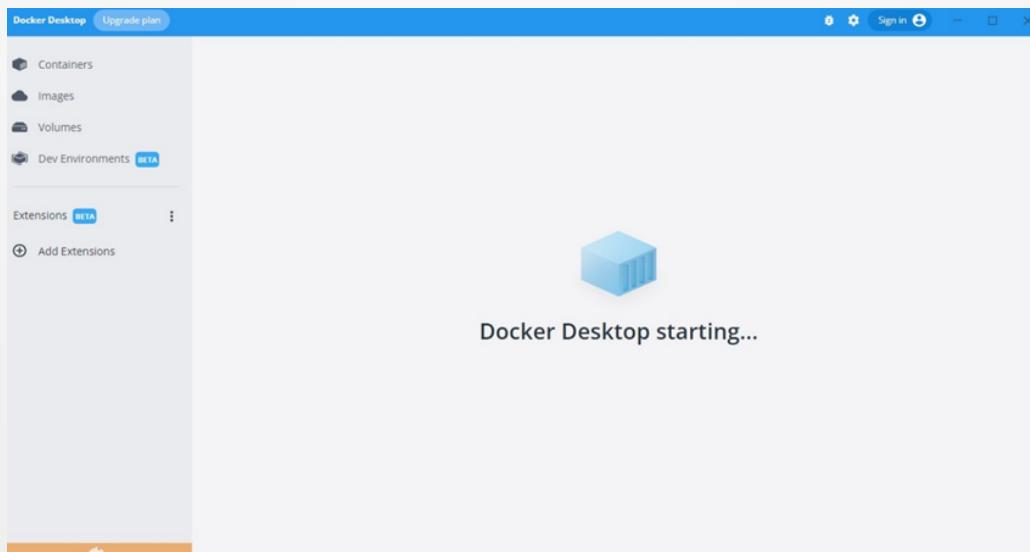
Feche e reinicie sua máquina:



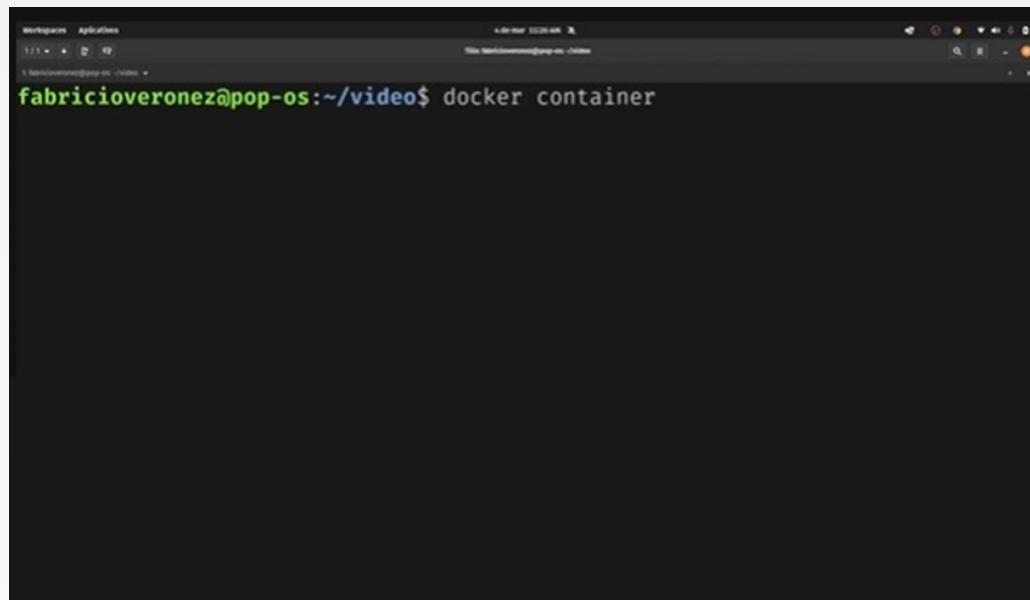
Feche e reinicie sua máquina:



O Docker está instalado e sendo inicializado:

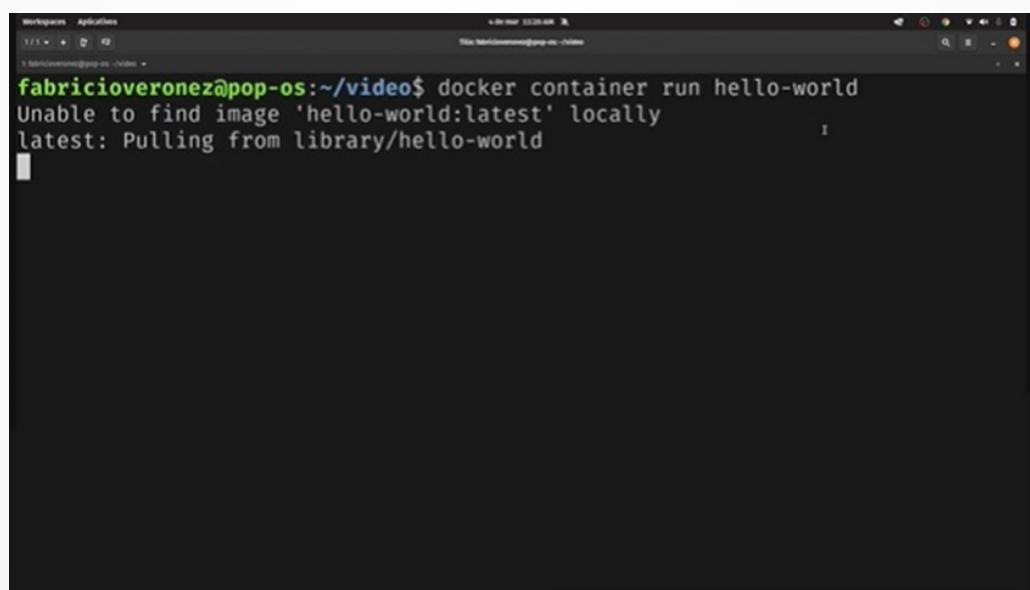


Para criar um Container:



```
fabricioveronez@pop-os:~/video$ docker container
```

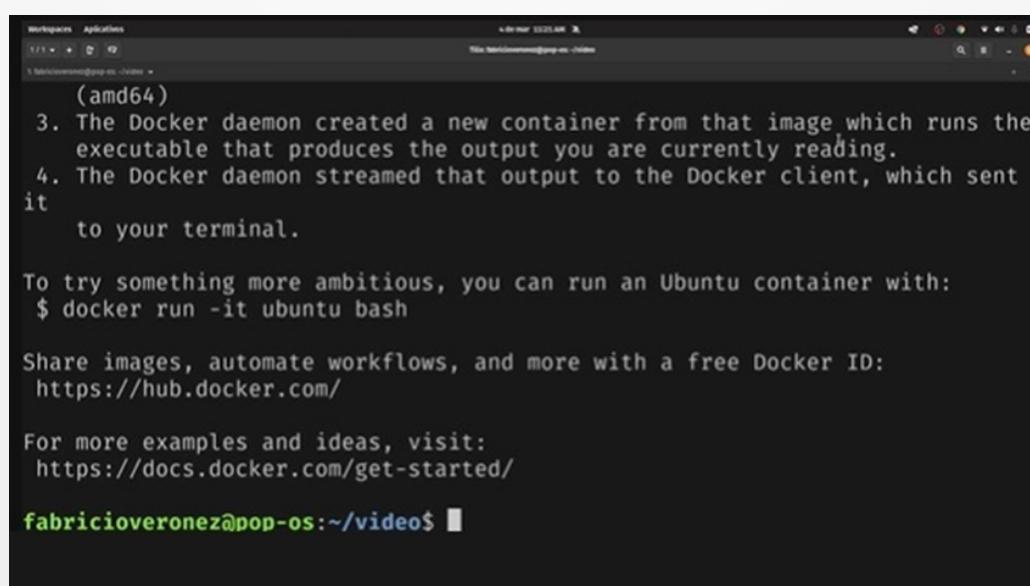
Execute o Container com o comando Run, já com a imagem que será executada:



```
fabricioveronez@pop-os:~/video$ docker container run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world

```

Executa o comando de forma rápida:



```
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

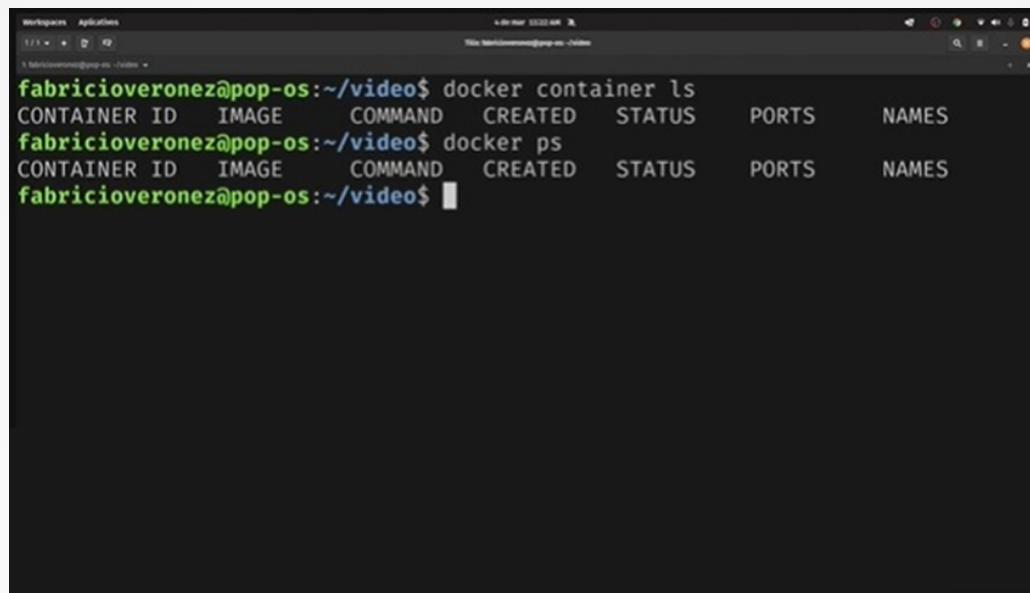
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

```

Obs.: O comando docker run nome_da_imagem, sem a palavra Container também funciona.

Para ver os Containers existem em seu Docker, execute o comando docker container ls ou docker container ps:



```
fabricioveronez@pop-os:~/video$ docker container ls
CONTAINER ID   IMAGE      COMMAND   CREATED    STATUS     PORTS      NAMES
fabricioveronez@pop-os:~/video$ docker ps
CONTAINER ID   IMAGE      COMMAND   CREATED    STATUS     PORTS      NAMES
fabricioveronez@pop-os:~/video$
```

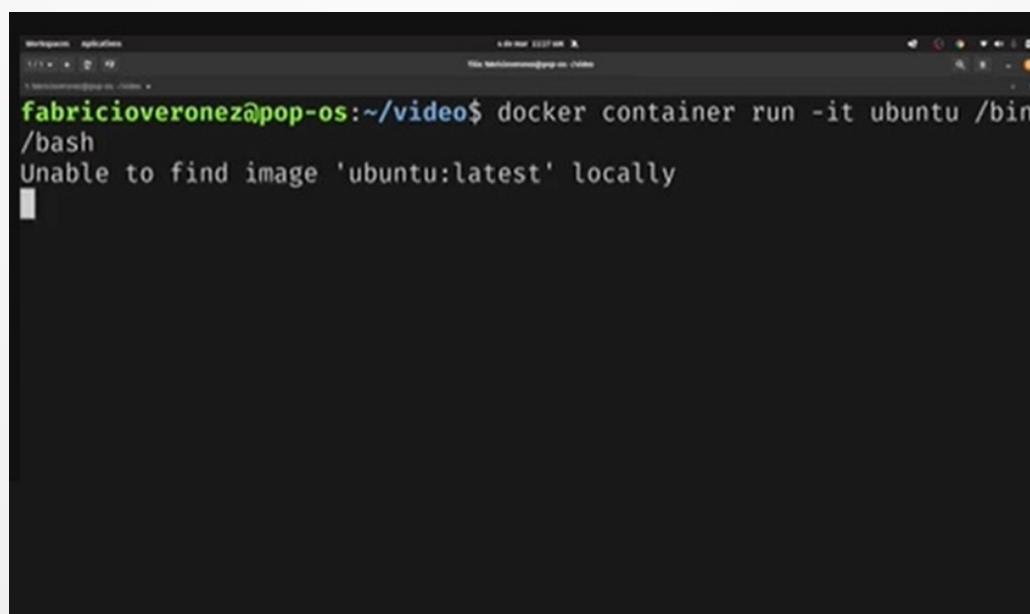
Para ver o Containers inativos utilize o comando **docker container ls -a**.

Obs.: Para especificar o seu name, utilize o comando **docker container run --name meucontainer hello-world**.

Para excluir um container, **docker container run --rm --name meucontainer hello-world**.

Para entrar em modo iterativo com o TTY, utilize o comando **docker container run -it**. Este comando permite a entrada no container para habilitar o console, o terminal para poder interagir com o container, ou seja, executar instruções dentro do container.

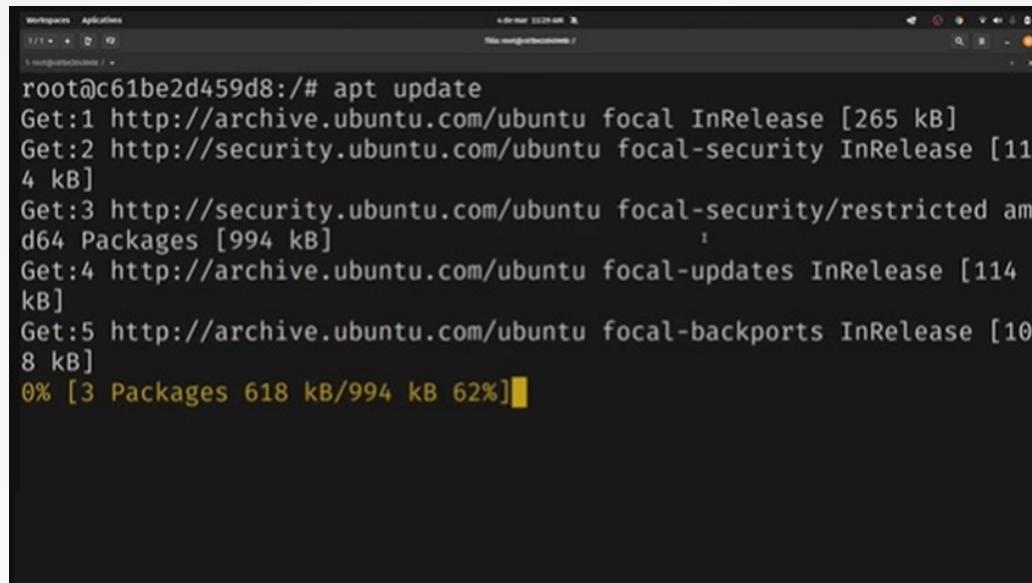
Para que o comando docker container run -it seja utilizado com sucesso digite, ainda na mesma linha de comando o nome da imagem a ser utilizada e o processo a ser executado ou /bin/bash:



```
fabricioveronez@pop-os:~/video$ docker container run -it ubuntu /bin
/bash
Unable to find image 'ubuntu:latest' locally
```

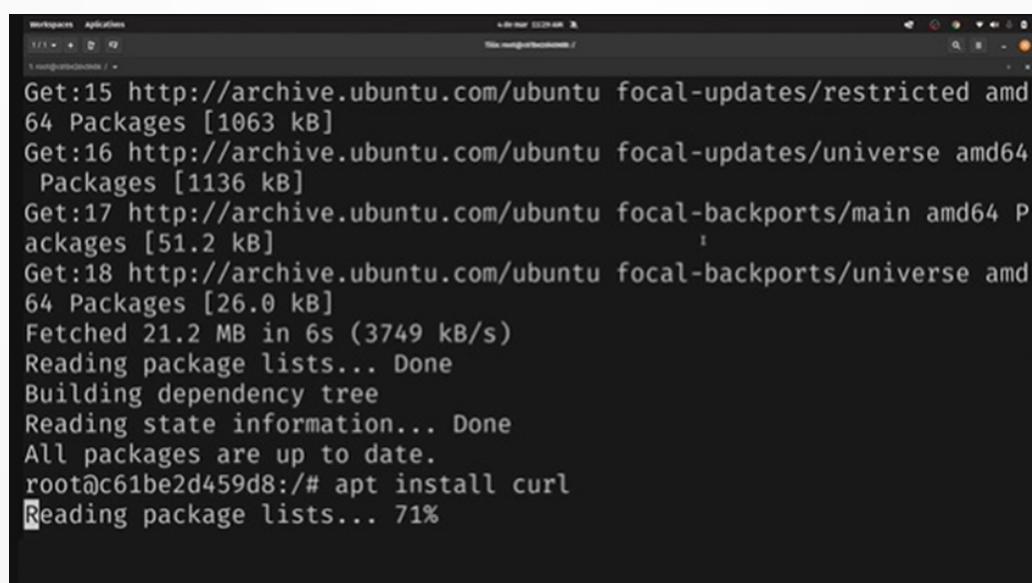
Obs.: Ao executar, a imagem do ubuntu é baixada.

Dentro do root, já com o id do container criado execute o comando **apt update** para instalar aplicações no ubuntu:



```
root@c61be2d459d8:/# apt update
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [994 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
0% [3 Packages 618 kB/994 kB 62%]
```

Agora, execute o **apt install curl** para executá-lo:



```
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1063 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1136 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [51.2 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [26.0 kB]
Fetched 21.2 MB in 6s (3749 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
root@c61be2d459d8:/# apt install curl
Reading package lists... 71%
```

Digite yes para instalar:

```
root@1c61be2d459d8:/# apt-get update  
root@1c61be2d459d8:/# apt-get upgrade  
Reading package lists... Done  
The following NEW packages will be installed:  
  ca-certificates curl krb5-locales libasn1-8-heimdal libbrotlii  
  libcurl4 libgssapi-krb5-2 libgssapi3-heimdal libhcrypto4-heimdal  
  libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal  
  libk5crypto3 libkeyutils1 libkrb5-26-heimdal libkrb5-3  
  libkrb5support0 libldap-2.4-2 libldap-common libnhttp2-14  
  libpssl5 libroken18-heimdal librtmp1 libsasl2-2 libsasl2-modules  
  libsasl2-modules-db libssqlite3-0 libssh-4 libssl1.1  
  libwind0-heimdal openssl publicsuffix  
0 upgraded, 32 newly installed, 0 to remove and 0 not upgraded.  
Need to get 5447 kB of archives.  
After this operation, 16.7 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
0% [Working]
```

E finalize a instalação.

Para executar o curl digite o comando **curl https://www.google.com.br**:

```
root@c61be2d459d8:/# curl https://www.google.com.br
```

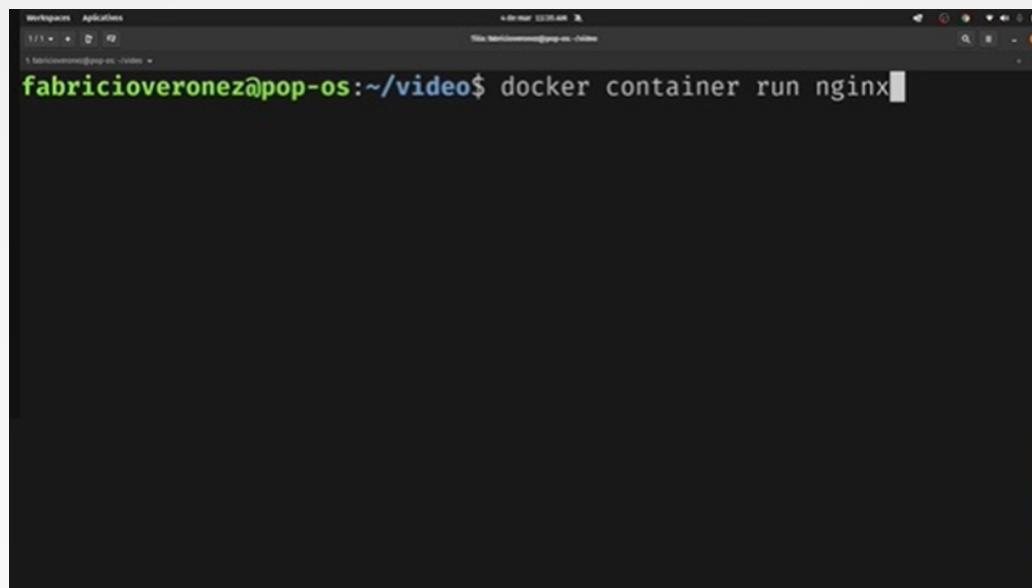
Obs.: Com esse comando, é possível executar o curl. Ele faz uma requisição http e faz um output, como se estivesse entrado no navegador e aberto o Google.

Ao finalizar o uso, digite o comando exit. Ao executar o comando **docker container ls**, este não está mais ativo. Ao executar o comando **docker container ls -a**, serão listados os containers.

Cada vez que um container é criado, é necessário instalar o curl.

Se houver a vontade de excluir algum container criado, inicialmente lista esses com o comando docker container ls. É possível excluir um container pelo nome com o comando **docker rm nomeDoContainer** ou pelo ID com o comando **docker rm idDoContainer**. Para verificar se o container realmente foi excluído, use o comando docker container ls. Pode-se também excluir container apenas com o início do ID, assim: **docker container rm 169**. O container com este ID será excluído com sucesso.

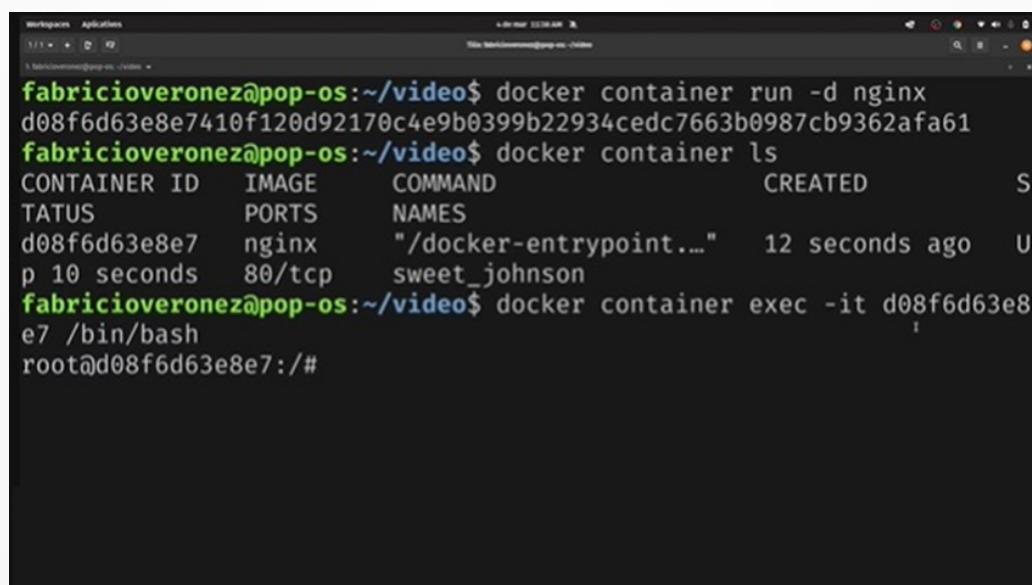
Para rodar um container com o nginx, use o comando **docker container run nginx**:



```
fabricioveronez@pop-os:~/video$ docker container run nginx
```

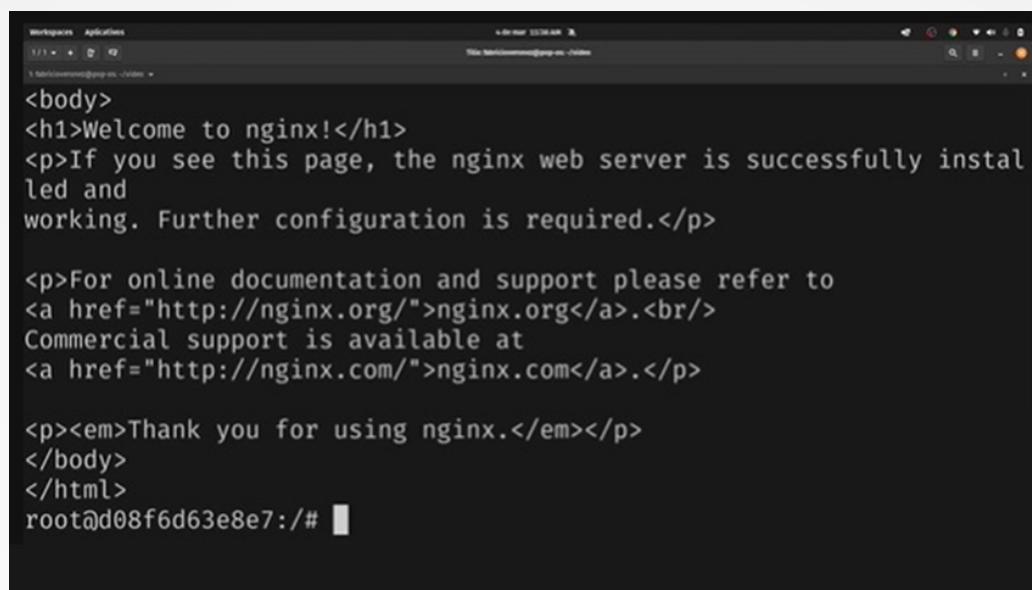
Obs.: Seu cursor pode ficar preso, pois o nginx é um processo que roda continuamente. Logo, para rodar os processos que são contínuos né necessário passar o parâmetro: **docker container run -d nginx**. Com esse comando será mostrado o ID do container e com o docker container ls serão listados os containeres, incluindo o que utiliza nginx.

Para acessar a aplicação rodando, use o comando **docker container exec -it** insira o ID do seu container/bin/bash e execute:



```
fabricioveronez@pop-os:~/video$ docker container run -d nginx
d08f6d63e8e7410f120d92170c4e9b0399b22934cedc7663b0987cb9362afa61
fabricioveronez@pop-os:~/video$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
TATUS              NAMES
d08f6d63e8e7        nginx              "/docker-entrypoint..."   12 seconds ago    Up 10 seconds      80/tcp
                sweet_johnson
fabricioveronez@pop-os:~/video$ docker container exec -it d08f6d63e8e7 /bin/bash
root@d08f6d63e8e7:/#
```

Obs.: Ao executar este último comando, digite **curl http://localhost**, será apresentado, no terminal, será apresentada o HTML da página do nginx rodando:



```
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

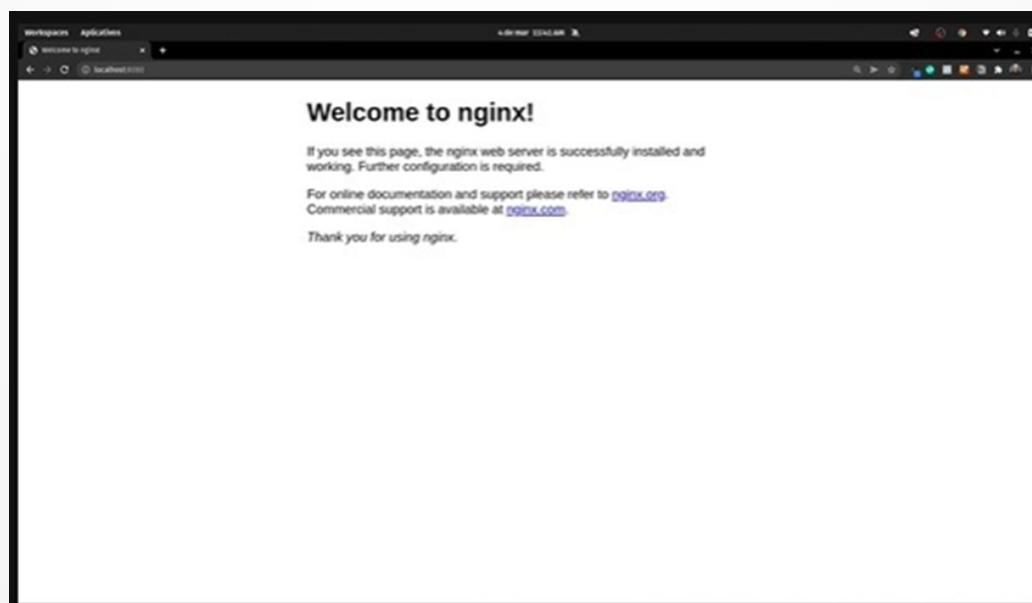
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@d08f6d63e8e7:/#
```

Saia da aplicação com o comando **exit**.

Para acessar na máquina local o nginx, faça um port building (se vincula uma porta da sua máquina com a porta do container). Para isso, execute o comando **docker container run -d -p 8080:80 nginx**. A porta 8080 equivale a sua máquina e a 80 ao container. Liste os containers com o comando **docker container ls**.

Abra o navegador e digite localhost:8080:



Assim o nginx roda no container em sua máquina.

Obs.: Para que o terminal liste apenas os containers pelo ID, utilize o comando **docker container ls -q** (este comando exibe apenas os containers ativos). Para listar todos os containers, inclusive os inativos, utilize o comando **docker container -pa**. Para remover todos os containers, utilize o comando **docker container rm -f** (**docker container -pa**), assim todos os containers serão removidos. Para verificar a remoção, utilize o comando **docker container ls**.

Cuidados com versionamento

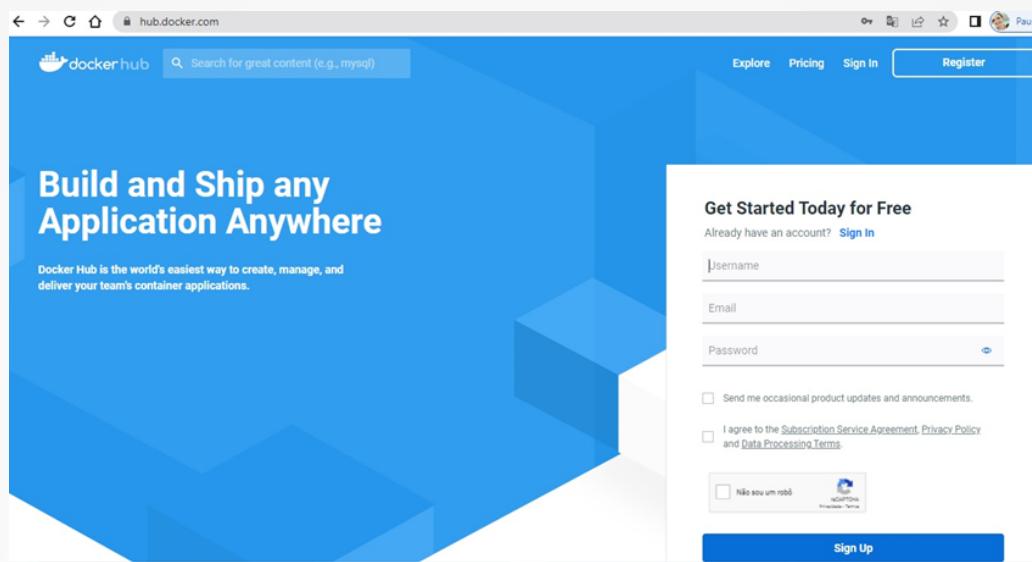
Para salvar alterações do container em uma imagem, tenho os seguintes cuidados:

- Após criar um container e instalar programas ou mudar configurações, para replicar estas alterações em outros containers, é necessário salvar estas alterações como uma imagem;
- Para isso, utilizamos o comando **docker commit** e passamos como parâmetros o <container já criado> e depois o <novo repositório>, com o comando **docker commit webapp nginx-webapp**.

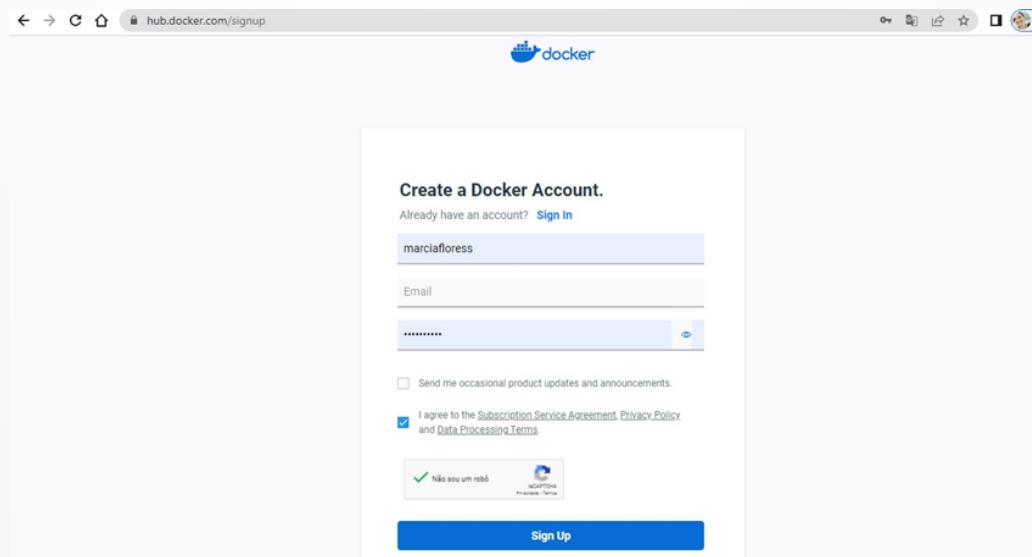
Importante: Os repositórios são criados no Docker Hub. Para isso, veja o próximo capítulo deste tutorial.

Criação de conta no Docker Hub

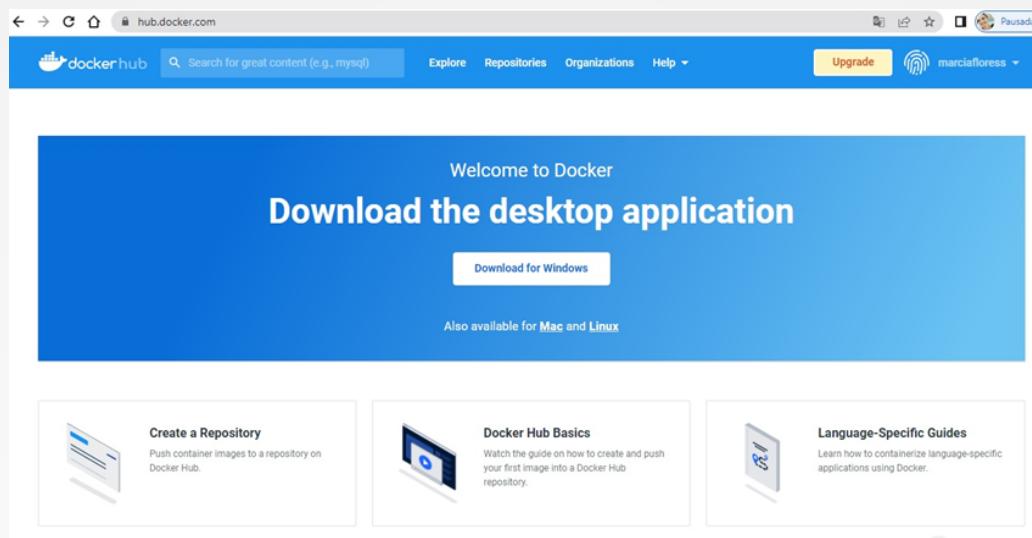
Para criar sua conta no Docker Hub, clique em hub.docker.com.



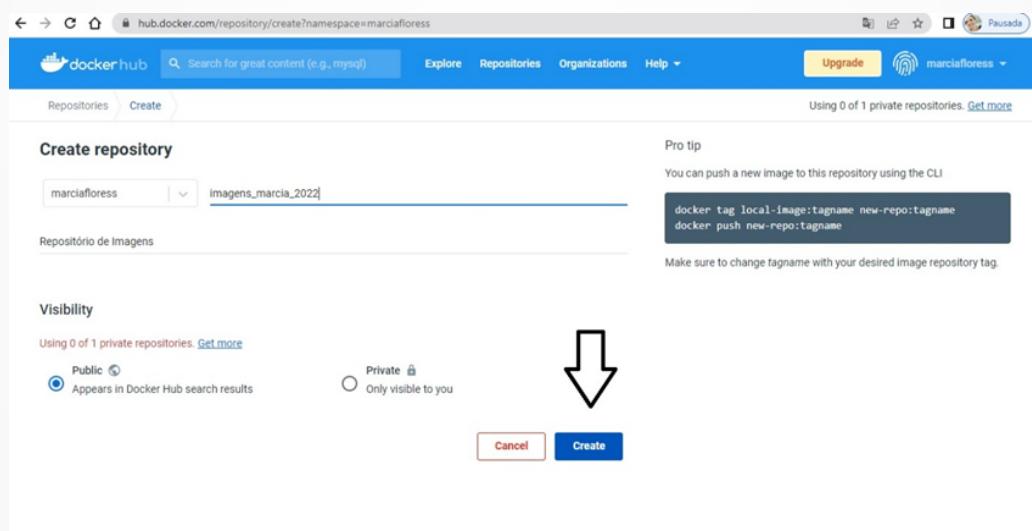
Preencha o formulário, selecione os Checkbox de clique em Sign Up:



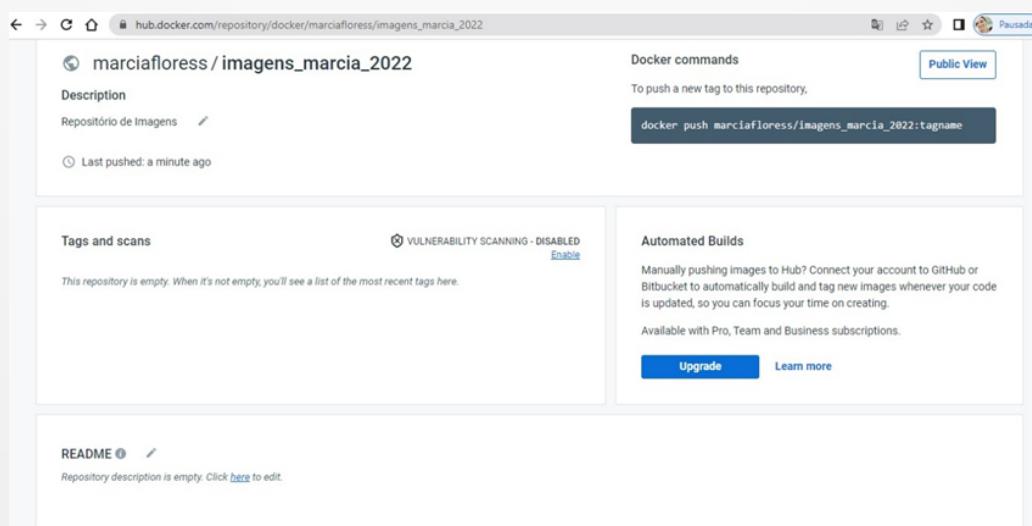
Sua conta será criada:



Para criar um Repositório, clique em Create a Repository, preencha os dados solicitados. Seu repositório poderá ser público ou privado. Lembre-se de usar letras minúsculas na criação do repositório. Após o preenchimento clique em create:

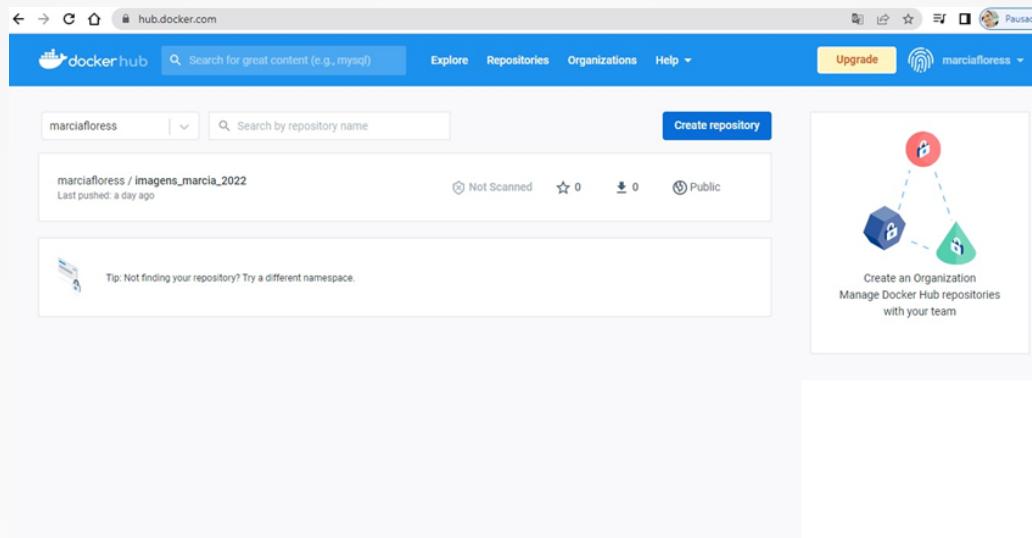


Seu repositório está criado:



Subir imagens no Docker Hub

Para criar a imagem é necessário estar logado no Docker Hub, ou seja, entre com seu usuário e senha:



Veja que o repositório de imagens já está criado. É nele que a imagem será salva.

Abra o cmd ou Prompt de Comando, pode-se utilizar também o Power Shell no caso do Windows:

A screenshot of a Windows PowerShell window titled 'Selecionar Windows PowerShell'. The command 'docker login' is being run, and the output shows 'Authenticating with existing credentials...' followed by 'Login Succeeded'. A note at the bottom of the terminal window reads: 'Logging in with your password grants your terminal complete access to your account. For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/'.

Obs.: Caso seja seu primeiro acesso, será necessário entrar com usuário e senha no Power Shell, caso contrário, o login já foi realizado com sucesso.

Selecione uma imagem no Docker Hub, em Explore para enviar para o Docker:

The screenshot shows the Docker Hub search results for 'nginx'. It lists several official Docker images, with 'nginx' being the most prominent. The 'nginx' entry includes a red icon, the text 'DOCKER OFFICIAL IMAGE', and a download count of '1B+'. Below the image details, there is a brief description: 'Official build of Nginx.' and a list of supported architectures: Linux, IBM Z, x86-64, ARM, ARM 64, 386, mips64le, PowerPC 64 LE, IBM Z, x86-64, ARM, ARM 64, 386, mips64le, PowerPC 64 LE.

Selecione a imagem, abra o terminal do Windows ou Power Shell, digite o comando **docker pull nome_da_imagem**, aqui, como exemplo será baixado para o Docker a imagem nginx:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

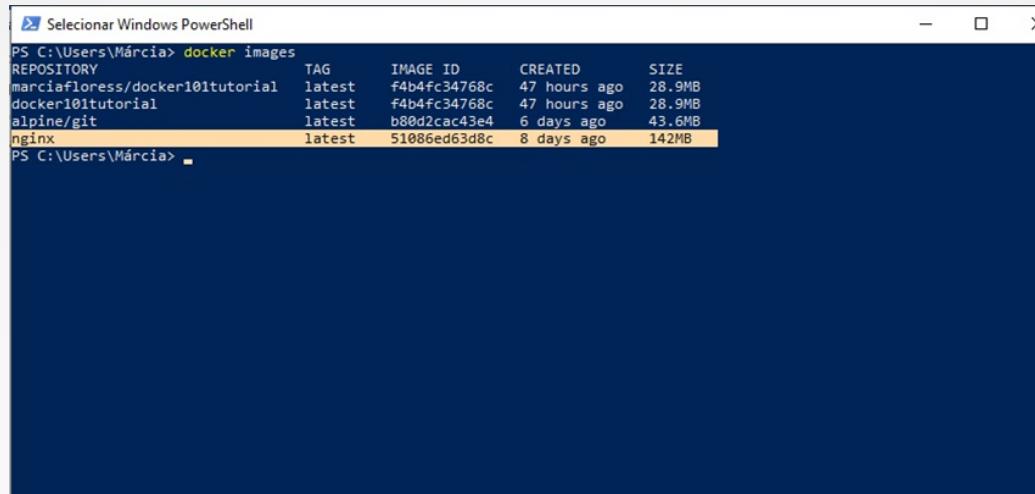
PS C:\Users\Márcia> docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
PS C:\Users\Márcia> docker images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
docke...tutorial    latest   f4b4fc34768c  47 hours ago  28.9MB
marciafloress/docke...tutorial    latest   f4b4fc34768c  47 hours ago  28.9MB
alpine/git          latest   b80d2cac43e4  6 days ago   43.6MB
PS C:\Users\Márcia> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
bd159e379b3b: Downloading [=====] 26.04MB/31.42MB
8d634ce99fb9: Download complete
98b00bcc0ec6: Download complete
6ab6a6301bde: Download complete
f5d8edcd47b1: Download complete
fe24ce36f968: Download complete
```

Entre no Docker e verifique se a imagem foi baixada:

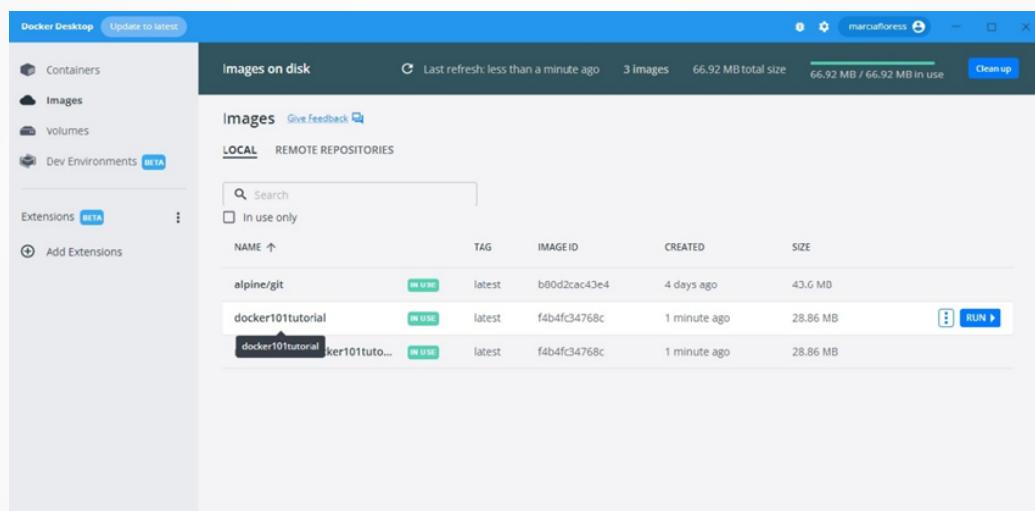
The screenshot shows the Docker Desktop interface with the 'Images' tab selected. It displays a list of images on disk, including 'alpine/git', 'docke...tutorial', 'marciafloress/docke...tutorial', and 'nginx'. The 'nginx' image is highlighted with a blue border. The Docker Desktop sidebar on the left shows 'Containers', 'Images', 'Volumes', and 'Dev Environments'. The bottom status bar indicates 'RAM 1.87GB', 'CPU 0.40%', 'Connected to Hub', and 'v4.12.0'.

Ou utilize o comando docker images no terminal do Windows ou PowerShell:

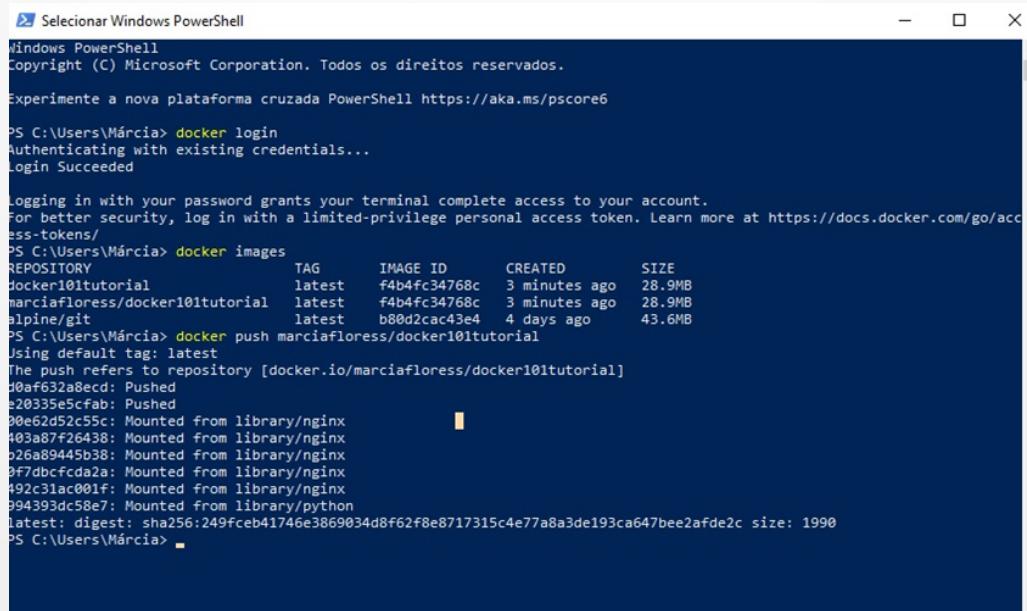


```
PS C:\Users\Márcia> docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
marciafloress/docker101tutorial  latest    f4b4fc34768c  47 hours ago  28.9MB
docker101tutorial  latest    f4b4fc34768c  47 hours ago  28.9MB
alpine/git      latest    b80d2cac43e4  6 days ago   43.6MB
nginx           latest    51086ed63d8c  8 days ago   142MB
PS C:\Users\Márcia>
```

Selecione uma imagem qualquer para subir para o Docker Hub:



No Power Shell digite os comandos:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Márcia> docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
PS C:\Users\Márcia> docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
docker101tutorial  latest    f4b4fc34768c  3 minutes ago  28.9MB
marciafloress/docker101tutorial  latest    f4b4fc34768c  3 minutes ago  28.9MB
alpine/git      latest    b80d2cac43e4  4 days ago   43.6MB
PS C:\Users\Márcia> docker push marciafloress/docker101tutorial
Using default tag: latest
The push refers to repository [docker.io/marciafloress/docker101tutorial]
d0af632a8ecd: Pushed
e20335e5cfab: Pushed
00e62d52c55c: Mounted from library/nginx
403a87f26438: Mounted from library/nginx
b26a89445b38: Mounted from library/nginx
9f7dbcfcda2a: Mounted from library/nginx
492c31ac001f: Mounted from library/nginx
994393dc58e7: Mounted from library/python
latest: digest: sha256:249fce41746e3869034d8f62f8e8717315c4e77a8a3de193ca647bee2afde2c size: 1990
PS C:\Users\Márcia>
```

O comando **docker login** autenticará seu usuário e senha no Docker Hub. O comando **docker images**, listará as imagens existentes no Docker e o comando **docker push marciafloress/docker101tutorial** enviará a imagem para o Docker Hub, já dentro do repositório criado anteriormente.

Dentro do repositório criado, verificaremos que a imagem subiu para o Docker Hub:

The screenshot shows the Docker Hub homepage with a blue header bar. The search bar contains 'hub.docker.com'. Below the header, there are navigation links for 'Explore', 'Repositories', 'Organizations', 'Help', and a user dropdown for 'marciafloress'. A 'Create repository' button is visible in the top right. The main content area displays two repositories under the namespace 'marciafloress': 'marciafloress / docker101tutorial' (Last pushed: 3 days ago) and 'marciafloress / imagens_marca_2022' (Last pushed: 9 days ago). Both repositories are marked as 'Not Scanned' with 0 stars, 1 commit, and are public. To the right of the repositories is a promotional section for organizations, featuring icons for a red circle, a blue hexagon, and a green triangle connected by dashed lines, with the text 'Create an Organization' and 'Manage Docker Hub repositories with your team'. At the bottom left, there is a tip message: 'Tip: Not finding your repository? Try a different namespace.' The bottom right corner features a colorful illustration of a stage with the word 'community' and 'ALL-HANDS'.

Versionamento de imagens

Alguns cuidados ao versionar uma imagem:

- Tome cuidado com as camadas de cache;
- Verifique as versões das imagens a partir das TAGs;
- Crie o Docker Ignore no seu projeto para não copiar arquivos desnecessários.

Para versionar uma imagem é necessário, em seu projeto, criar o arquivo Dockerfile, pois ele é responsável pela criação de imagens. Através deste arquivo será gerado o build para criação do container.

Para gerar a imagem a partir do Dockerfile, executamos o comando abaixo no mesmo local que o arquivo se encontra:

docker build -t nome_da_imagem

Para criar o container a partir dessa imagem construída, podemos executar o comando:

docker run nome_da_imagem

FROM

Esta Instrução é obrigatória que indica qual imagem base será utilizada. Podemos criar uma imagem a partir de uma imagem do Node.js ou um sistema Linux usando uma imagem do Ubuntu, por exemplo. É possível criar sua própria imagem usando a instrução FROM scratch.

FROM node:12-alpine

O comando acima indica que será criado uma imagem com Node.js.

RUN

Utilizado para executar comandos no processo de montagem da imagem que estamos construindo no Dockerfile, ele é executado durante o build (construção da imagem), e não durante a construção do container. Em um Dockerfile, é possível ter mais de um comando RUN.

FROM node:alpine

WORKDIR /usr/app

RUN npm init -y

RUN npm install cowsay

RUN npx cowsay Hello Docker

RUN cat package.json

Quando rodamos o comando abaixo, podemos ver o log de cada RUN sendo executado no processo de criação da imagem:

docker build -t nodej/nodej .

Resultado

```
docker build -t nodej/nodej .

 Sending build context to Docker daemon 3.072kB
Step 1/6 : FROM node:alpine
 ---> 0f2c18cef5d3
Step 2/6 : WORKDIR /usr/app
 ---> Using cache
 ---> 0399f14efe3b
Step 3/6 : **RUN npm init -y**
 ---> Using cache
 ---> da2345d90786
Step 4/6 : **RUN npm install cowsay**
 ---> Using cache
 ---> bf6c1a19a89a
Step 5/6 : **RUN npx cowsay Hello Rocketseat**
 ---> Running in 78f5486fa633
Removing intermediate container 78f5486fa633
 ---> 02d9013dbf0b
Step 6/6 : **RUN cat package.json**
 ---> Running in 675e768b4b16
{
  "name": "app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" &&
exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cowsay": "^1.4.0"
  }
}
Removing intermediate container 675e768b4b16
```

```
--> 260e87152b92
Successfully built 260e87152b92
Successfully tagged nodej/nodej:latest
```

CMD

Diferente do RUN, o CMD executa apenas a criação do container, e não do build da imagem. Ele deve ser único no Dockerfile.

```
FROM node:alpine
WORKDIR /usr/app
RUN npm init -y
RUN npm install cowsay
RUN npx cowsay Hello Rocketseat
RUN cat package.json
CMD npm start
```

Execute o comando abaixo para recriar a imagem:

```
docker build -t nodej/nodej .
```

Depois rode o comando abaixo para criar o container:

```
docker run -t nodej/nodej
```

O comando vai criar a imagem e dentro do container será executado o npm start, porém no package.json não foi definido no script start, mas perceba que o CMD executou apenas no container e não na construção da imagem. E o log do erro só apareceu quando criamos o container.

```
docker run -t nodej/nodej
npm ERR! missing script: start
npm ERR! A complete log of this run can be found in:
npm ERR! /root/.npm/_logs/2020-08-20T17_40_32_268Z-
debug.log
```

EXPOSE

A Instrução que informa qual porta deverá ser liberada. EXPOSE não é um comando que libera a porta, ele serve para ficar documentado no Dockerfile quais portas deverão ser liberadas ao criar o container.

Para efetivamente liberar a porta é preciso passar o parâmetro -p no comando docker run, vamos ver a seguir:

```
FROM node:alpine
WORKDIR /usr/app
RUN npm init -y
RUN npm install cowsay
RUN npx cowsay Hello Rocketseat
```

RUN cat package.json

EXPOSE 3000 informa que deverá ser liberado a porta 3000 na criação do container. Execute:

CMD npm start

Execute o comando abaixo para criar o container e liberar a porta 3000:

docker run -p 3000:3000 -d nodejs/nodejs

Para liberar a porta, usamos o parâmetro -p 3000:3000. Está sendo realizado também o redirecionamento de portas, em que o primeiro parâmetro 3000 é a porta do host e o segundo :3000 é a porta do container. Isto é, toda vez que o host receber uma requisição na porta 3000, o container irá ouvir a requisição na porta 3000 também.

Liberando portas na aplicação: Porta 3000 Host e :3000 Container.

Podemos liberar portas TCP e UDP, por padrão, quando não é especificado /tcp ou /udp, é liberado a porta TCP.

docker run -p 80:80/tcp -p 80:80/udp ...

Por padrão, a porta tcp é liberada quando não é especificado o protocolo, exemplo:

docker run -p 8080:80

COPY E ADD

Comandos para copiar arquivos e pastas de um lugar específico na máquina local para uma pasta no container.

Diferença é que ADD copia arquivos remotos para alguma pasta na imagem e também pode copiar arquivos compactados (tar.gz) que serão descompactados na imagem automaticamente.

Por questões de semântica, sempre utilize COPY para copiar arquivos e pastas locais e ADD para arquivos remotos ou compactados. Utilize o ADD se realmente for necessário.

Se quiser acompanhar em sua máquina, faça os seguintes procedimentos abaixo:

COPY

Digite no terminal dentro de um diretório para criar um projeto com Node.js:

yarn init -y

Altere o package.json:

```
{  
  "name": "docker-node",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {
```

```
    "start": "node index.js"  
},  
  "keywords": [],  
  "author": "",  
  "license": "MIT"  
}
```

Crie o arquivo index.js na raiz do projeto com o seguinte conteúdo:

```
console.log("Qual é a soma de 2 + 2: ", 4);
```

Crie o arquivo Dockerfile na raiz do projeto:

```
FROM node:alpine  
WORKDIR /usr/app  
COPY package.json ./  
COPY index.js ./  
RUN npm install  
RUN cat package.json  
RUN npm start  
EXPOSE 3000  
# CMD npm start
```

Execute o comando abaixo no terminal e observe o log:

```
docker build -t nodej/nodej .
```

No final será impresso:

```
Qual é a soma de 2 + 2 = 4
```

ADD

Exemplo com o comando ADD:

```
FROM ubuntu:18.04  
RUN mkdir /myfolder  
WORKDIR /myfolder  
ADD /sample.tar.gz ./  
RUN ls ## listando arquivos  
RUN ls project  
RUN ls -la  
RUN echo "FIM"
```

Execute o comando abaixo e observe o log:

```
docker build -t nodej/nodej .
```

Será adicionado um arquivo compactado da máquina do host para o container recém criado dentro da pasta myfolder, em seguida o arquivo será descompactado automaticamente.

VOLUME

Quando adicionamos VOLUME no Dockerfile, estamos informando um ponto de montagem, criando uma pasta que ficará disponível entre o container e o host.

```
FROM ubuntu:18.04
RUN mkdir /minha_pasta
RUN echo "hello world" > /minha_pasta/hello.txt
VOLUME /minha_pasta
```

Dockerfile acima define a utilização da imagem do ubuntu:18.04, cria uma pasta chamada minha_pasta e depois cria um arquivo hello.txt com o conteúdo "hello world" dentro da pasta, por fim é disponibilizado o VOLUME /minha_pasta para que o host tenha acesso.

WORKDIR

Define uma pasta dentro do container onde os comandos serão executados:

```
FROM node:alpine
WORKDIR /usr/app
COPY package.json .
RUN npm install
CMD npm start
```

Estamos definindo que na pasta usr/app serão executados os comandos seguintes, o arquivo package.json será copiado do host para user/app, inclusive o comando RUN npm install será executado dentro do contexto definido no WORKDIR.

USER

Podemos alterar os usuários para executar os comandos. No exemplo abaixo, defini o usuário node para executar os comandos e, como não precisa de privilégios de super usuário (root), podemos usar usuário sem tantos privilégios.

```
FROM node:alpine
USER node #definindo o usuário que realizar os comandos.
RUN whoami #qual usuário está usando o terminal, será impresso node,
WORKDIR /usr/app
COPY package.json .
RUN npm install
CMD npm start
```

Para enviar a nova versão da imagem para o Docker Hub, utilize o comando:

```
docker container run -d -p 8080:80 nome do container/nome da imagem:
versão
```

