

FULL STACK DEVELOPMENT

FASE 2 | AULA 02 -

GERENCIAMENTO DE CONTÊINERES

SUMÁRIO

O QUE VEM POR AÍ?	3
HANDS ON	4
SAIBA MAIS	5
O QUE VOCÊ VIU NESTA AULA?	12
REFERÊNCIAS	13
PALAVRAS-CHAVE	14

EMSE

O QUE VEM POR AÍ?

Nesta aula, você aprenderá como gerenciar containers, criar imagens e usar comandos de forma aprofundada. Veremos também como criar o seu primeiro Dockerfile, como vê-lo rodando e visualizar os recursos usados pelo container, além de entender como a rede de um container funciona, criar um volume persistente e montar no container os logs que ele gera. Depois, você verá mais sobre registros de containers na cloud.



HANDS ON

Nesta aula, executaremos diversos comandos Linux e comandos Docker para criar nossa imagem, container e volumes. Também veremos na prática como os logs de uma aplicação funcionam e, por fim, faremos nossa primeira aplicação em NodeJS funcionar via container e a acessaremos via Web.

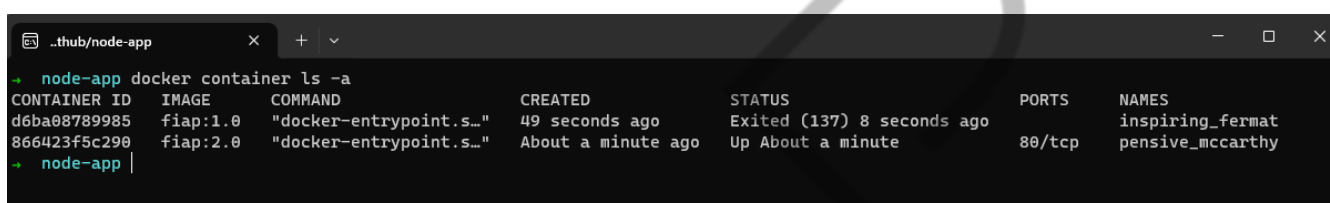
EXEMPLO

SAIBA MAIS

O Docker possui diversos comandos e vamos listar e explicar detalhadamente cada um deles.

O primeiro comando é o “docker container”. Com ele, podemos obter informações sobre os containers que estão rodando e até mesmo os que não estão rodando mais.

Executando o comando docker container ls -a:

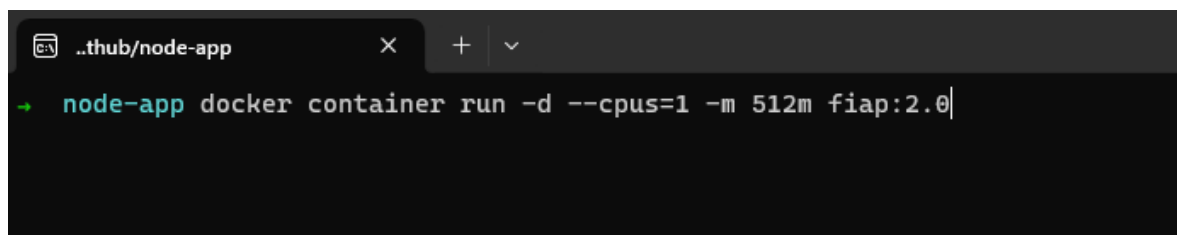


```
node-app docker container ls -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
d6ba08789985   fiap:1.0  "docker-entrypoint.s..." 49 seconds ago Exited (137) 8 seconds ago      inspiring_fermat
866423f5c290   fiap:2.0  "docker-entrypoint.s..." About a minute ago Up About a minute      80/tcp      pensive_mccarthy
```

Figura 1 – docker container ls
Fonte: Elaborado pelo autor (2023)

- CONTAINER ID: ID do container. Esse valor é único para cada container.
- IMAGE: a imagem que foi utilizada para a execução do container.
- COMMAND: o comando em execução.
- CREATED: quando foi criado.
- STATUS: o seu status atual.
- PORTS: a porta do container e do host que esse container utiliza.
- NAMES: o nome do container.

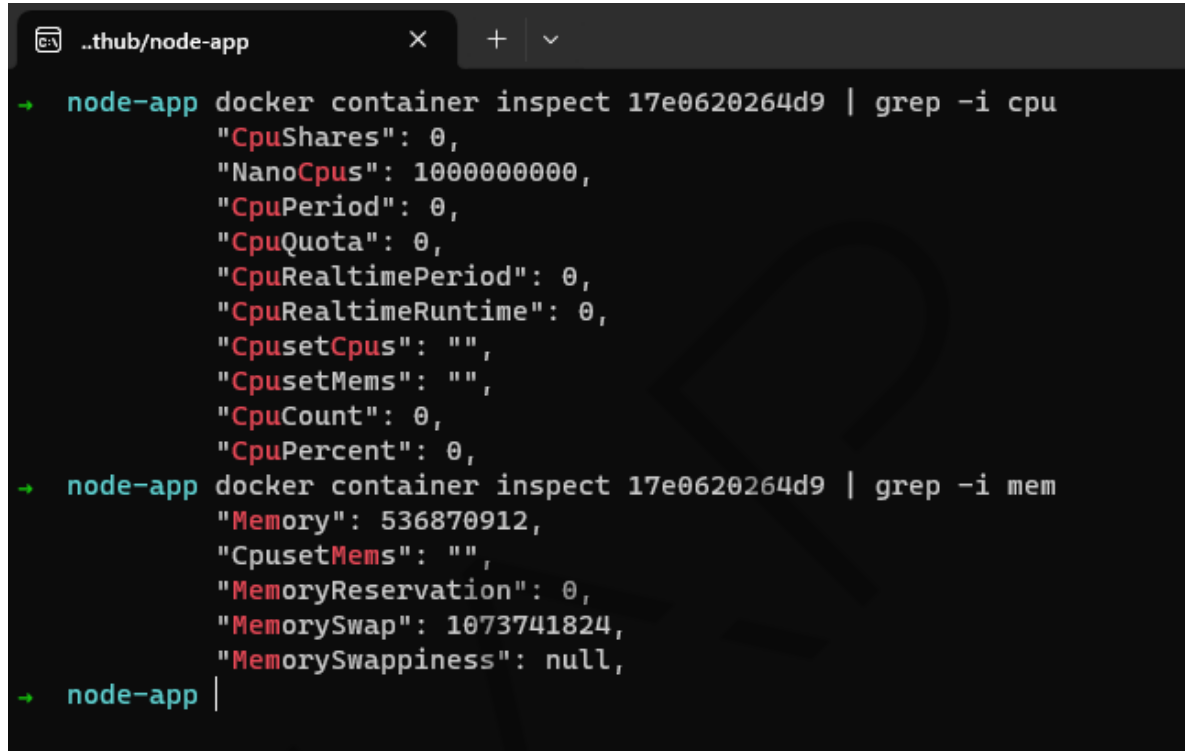
Temos a opção de configurar a CPU e a memória do container no momento da sua criação. Para rodar um container com 1 cpu e 512mb de memória, basta adicionar as flags “--cpus=1” para cpu e “-m 512m” para RAM:



```
node-app docker container run -d --cpus=1 -m 512m fiap:2.0
```

Figura 2 – docker container run
Fonte: Elaborado pelo autor (2023)

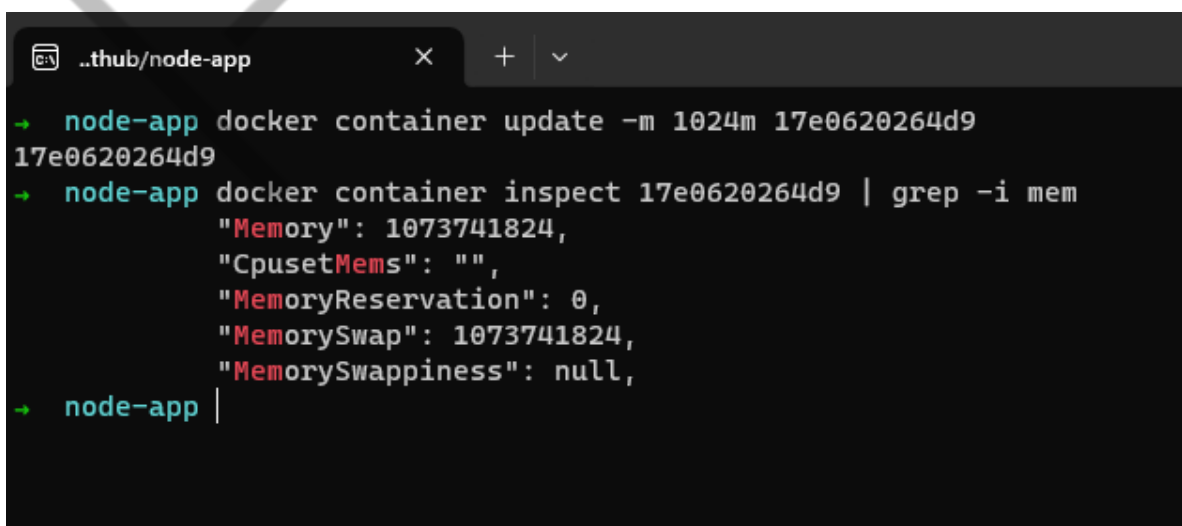
Para checar se a configuração foi aplicada, pegue o ID do container com docker container ls e dê um “inspect” com grep para pegar a informação que desejar. Use “cpu” para cpu e “mem” para memória:



```
node-app docker container inspect 17e0620264d9 | grep -i cpu
  "CpuShares": 0,
  "NanoCpus": 1000000000,
  "CpuPeriod": 0,
  "CpuQuota": 0,
  "CpuRealtimePeriod": 0,
  "CpuRealtimeRuntime": 0,
  "CpusetCpus": "",
  "CpusetMems": "",
  "CpuCount": 0,
  "CpuPercent": 0,
node-app docker container inspect 17e0620264d9 | grep -i mem
  "Memory": 536870912,
  "CpusetMems": "",
  "MemoryReservation": 0,
  "MemorySwap": 1073741824,
  "MemorySwappiness": null,
node-app |
```

Figura 3 – docker container inspect
Fonte: Elaborado pelo autor (2023)

Para aumentar ou reduzir os recursos de um container em execução, utilize a flag update:



```
node-app docker container update -m 1024m 17e0620264d9
17e0620264d9
node-app docker container inspect 17e0620264d9 | grep -i mem
  "Memory": 1073741824,
  "CpusetMems": "",
  "MemoryReservation": 0,
  "MemorySwap": 1073741824,
  "MemorySwappiness": null,
node-app |
```

Figura 4 – docker container update
Fonte: Elaborado pelo autor (2023)

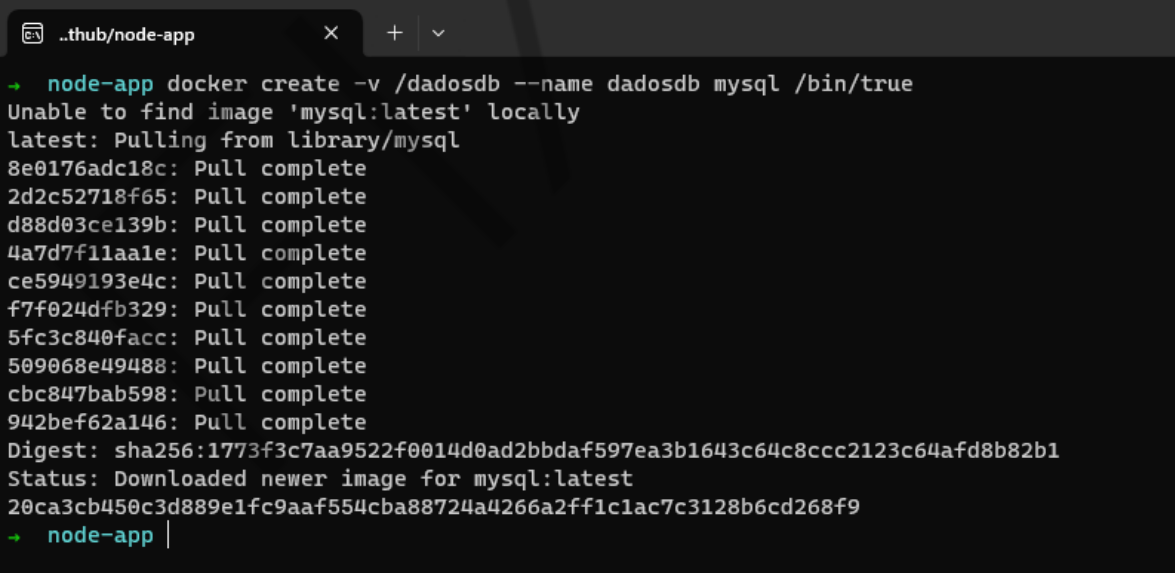
Dessa forma, o container não precisa morrer para que os recursos aumentem. O update vale para memória RAM e CPU.

Outro ponto relevante sobre o Docker são os backups dos volumes. Ele é indicado para garantir uma cópia segura de um volume existente e, caso ocorra uma falha ou exclusão por engano, é possível utilizar esse backup para criar um volume com os dados cópia. Isso garante segurança e praticidade quando for preciso um volume para certos arquivos e dados.

Containers de dados são usados para separar dados e permitir a comunicação entre vários containers, de forma simples, com os mesmos dados.

Para isso, é nomeado um volume dentro desse container, que será consumido por outros containers. Nesse caso, não é necessário criar um diretório específico persistente, um volume padrão, pois essa pasta é criada automaticamente dentro da raiz do Docker. A vantagem é não precisar se preocupar com essa pasta, já que toda conexão será feita para o container de dados e não para o diretório.

Veja um exemplo de como criar o container de dados:



```
node-app docker create -v /dadosdb --name dadosdb mysql /bin/true
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
8e0176adc18c: Pull complete
2d2c52718f65: Pull complete
d88d03ce139b: Pull complete
4a7d7f11aa1e: Pull complete
ce5949193e4c: Pull complete
f7f024dfb329: Pull complete
5fc3c840facc: Pull complete
509068e49488: Pull complete
cbc847bab598: Pull complete
942bef62a146: Pull complete
Digest: sha256:1773f3c7aa9522f0014d0ad2bbdaf597ea3b1643c64c8ccc2123c64afd8b82b1
Status: Downloaded newer image for mysql:latest
20ca3cb450c3d889e1fc9aaf554cba88724a4266a2ff1c1ac7c3128b6cd268f9
node-app |
```

Figura 5 – Container de Dados
Fonte: Elaborado pelo autor (2023)

Nesse exemplo, criamos o container de dados com uma imagem mysql e o volume dadosdb, que será usado por outros containers. A pasta dadosdb poderá ser visualizada e/ou editada por outros containers.

Para consumir esse volume de dados, basta executar o comando:

```
node-app x + v
node-app docker container run -d --volumes-from dadosdb --name fiapdadosdb cb026def805f
6772b7cbfcc3e4382cc1a9d7c65703d82b38d46ce2f8d2c549ea21ee2e7cd67e
node-app docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
6772b7cbfcc3   cb026def805f   "docker-entrypoint.s..." 6 seconds ago  Up 5 seconds  80/tcp       fiapdadosdb
node-app |
```

Figura 6 – Container de Dados
Fonte: Elaborado pelo autor (2023)

Dessa forma, o container está consumindo o volume criado no container de dados. Isso pode ser aplicado a quantos containers de dados for preciso.

Agora, seguem informações importantes sobre o Docker Hub e como encontrar imagens e versões diferentes para a aplicação que se pretende rodar.

Para isso, basta navegar até o Docker Hub ou pesquisar a imagem base no Google junto do nome “Docker Hub”.

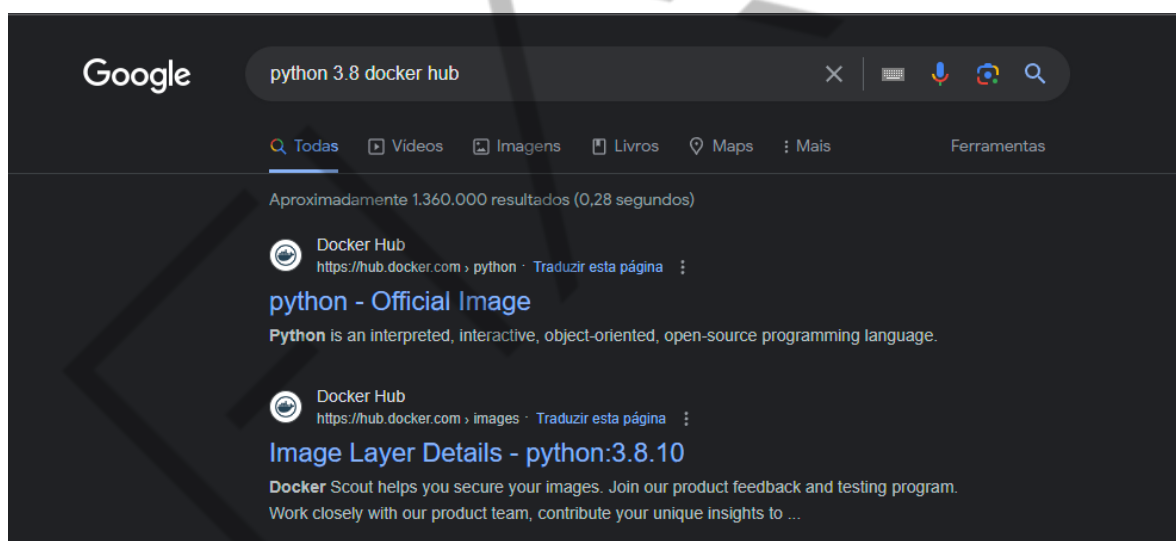
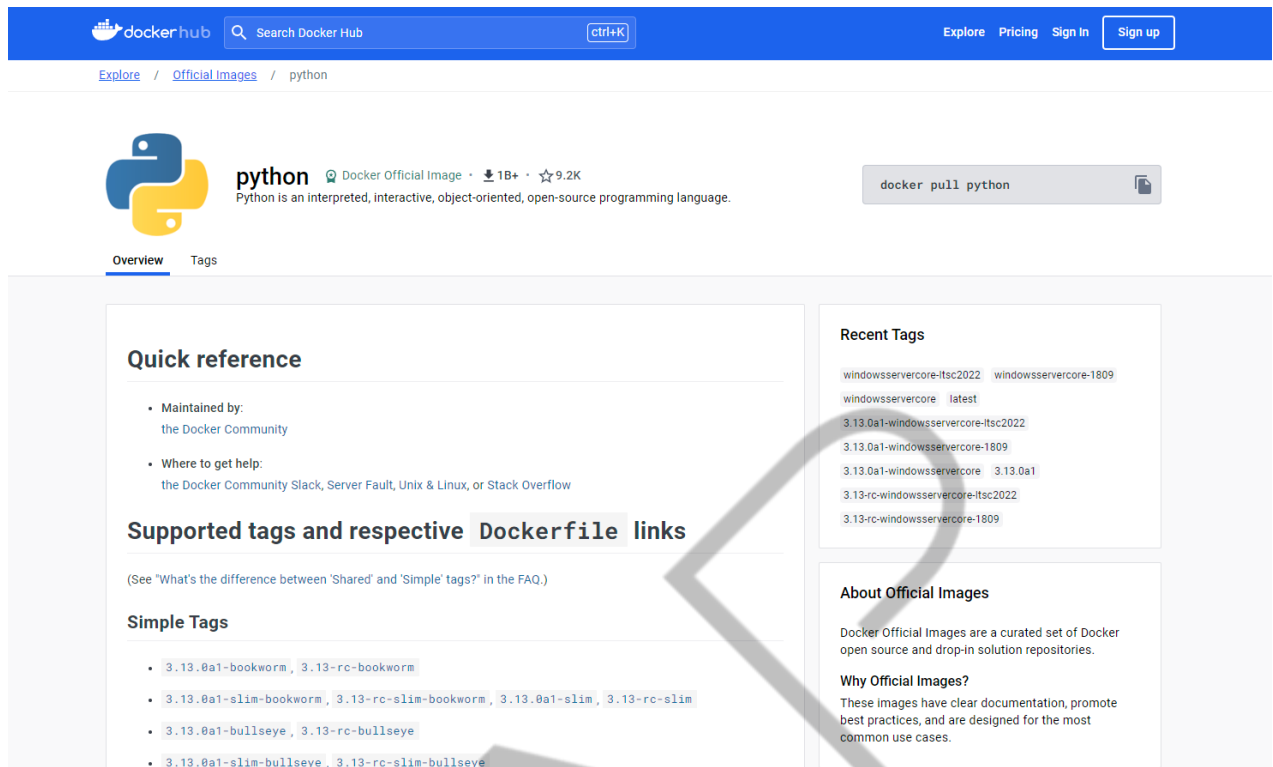


Figura 7 – Pesquisa Google Python Docker Hub
Fonte: <https://www.google.com> (2023)

Clicando no primeiro link, você chegará na página oficial de imagens do Python (conforme mostra a figura “Imagem Python”). Na aba “Tags”, teremos todas as versões Python disponíveis de forma oficial:



The screenshot shows the Docker Hub page for the official Python image. The header includes the Docker Hub logo, a search bar, and navigation links like 'Explore', 'Pricing', 'Sign in', and 'Sign up'. The main content area features the Python logo, the text 'python Docker Official Image', and a 'docker pull python' button. Below this, there are tabs for 'Overview' and 'Tags'. The 'Overview' tab is active, showing a 'Quick reference' section with links to the Docker Community and help resources. It also lists 'Supported tags and respective Dockerfile links' with a list of tags like '3.13.0a1-bookworm', '3.13-rc-bookworm', etc. A 'Recent Tags' section on the right lists tags like 'windowsservercore-ltsc2022', 'windowsservercore-latest', etc. An 'About Official Images' section explains that these are curated sets of Docker open source and drop-in solution repositories.

Quick reference

- Maintained by: the Docker Community
- Where to get help: the Docker Community Slack, Server Fault, Unix & Linux, or Stack Overflow

Supported tags and respective Dockerfile links

(See "What's the difference between 'Shared' and 'Simple' tags?" in the FAQ.)

Simple Tags

- 3.13.0a1-bookworm, 3.13-rc-bookworm
- 3.13.0a1-slim-bookworm, 3.13-rc-slim-bookworm, 3.13.0a1-slim, 3.13-rc-slim
- 3.13.0a1-bullseye, 3.13-rc-bullseye
- 3.13.0a1-slim-bullseye, 3.13-rc-slim-bullseye

Recent Tags

- windowsservercore-ltsc2022, windowsservercore-1809
- windowsservercore-latest
- 3.13.0a1-windowsservercore-ltsc2022
- 3.13.0a1-windowsservercore-1809
- 3.13.0a1-windowsservercore, 3.13.0a1
- 3.13-rc-windowsservercore-ltsc2022
- 3.13-rc-windowsservercore-1809

About Official Images

Docker Official Images are a curated set of Docker open source and drop-in solution repositories.

Why Official Images?

These images have clear documentation, promote best practices, and are designed for the most common use cases.

Figura 8 – Imagem Python
Fonte: https://hub.docker.com/_/python (2023)

A última versão sempre é denominada “latest” e, além das versões padrão, como 3.8 e 3.7, aparecerão variações, como mostra a figura “Tags Imagem Python”. Esta é uma versão “latest” com Windows Server Core como imagem base:

The screenshot shows the Docker Hub page for the 'python' image. At the top, there's a header with the Python logo, the name 'python', and a 'Docker Official Image' badge. Below this, there are tabs for 'Overview' and 'Tags', with 'Tags' being the active tab. The main content area displays a list of tags, including 'latest', 'windowsservercore-ltsc2022', and 'windowsservercore-1809'. Each tag entry shows the digest, OS/ARCH, vulnerabilities (C, H, M, L), and compressed size. A large, semi-transparent watermark 'DRAFT' is overlaid on the image.

TAG	DIGEST	OS/ARCH	VULNERABILITIES	COMPRESSED SIZE
latest	183787b02f04	linux/386	1 C, 1 H, 8 M, 93 L	364.95 MB
	50ebe3811db9	windows/amd64	0 H, 1 M, 0 L	1.97 GB
	4501c479acd7	windows/amd64	0 H, 1 M, 0 L	1.81 GB
windowsservercore-ltsc2022	4501c479acd7	windows/amd64	0 H, 1 M, 0 L	1.81 GB
windowsservercore-1809	50ebe3811db9	windows/amd64	0 H, 1 M, 0 L	1.97 GB

Figura 9 – Tags Imagem Python
Fonte: https://hub.docker.com/_/python/tags (2023)

Agora, ao procurar por “Python” na barra de pesquisa do Docker Hub, ele trará todo tipo de imagem: imagens oficiais, imagens verificadas pela Publisher e patrocinadas.

Vale ressaltar que neste ponto é preciso ter cuidado, pois podem existir imagens suspeitas no meio dos resultados. Escolha com sabedoria!

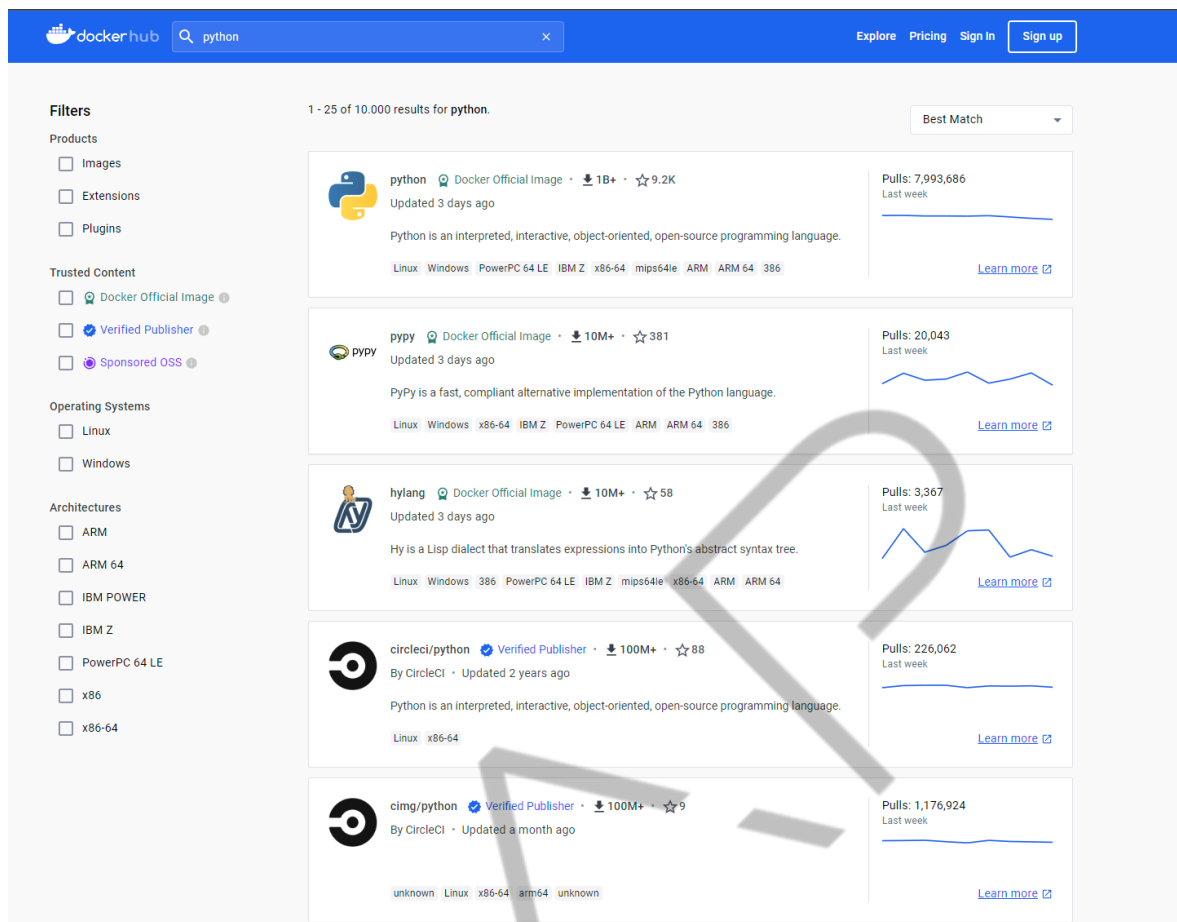


Figura 10 – Imagens diversas Python
Fonte: <https://hub.docker.com/search?q=python> (2023)

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, você viu como gerenciar containers e criar uma imagem personalizada, além de entender como e onde as imagens podem ser salvas e como a rede de um container funciona e se integra com o nosso computador.

Depois, criamos nosso primeiro volume e montamos no container. Por fim, vimos mais sobre os logs de um container, tudo executado via WSL Ubuntu. Se você ficou com alguma dúvida, assista aos vídeos e acesse a comunidade no Discord. Até mais!

REFERÊNCIAS

O que é Kernel Linux. Red Hat. 2019. Disponível em: <<https://www.redhat.com/pt-br/topics/linux/what-is-the-linux-kernel>>. Acesso em: 08 jan 2024.

Entendendo a rede no Docker. Stack.desenvolvedor.expert. [s.d]. Disponível em: <<https://stack.desenvolvedor.expert/appendix/docker/rede.html>>. Acesso em: 08 jan 2024.

Entendendo o armazenamento no Docker. Stack.desenvolvedor.expert. [s.d]. Disponível em: <<https://stack.desenvolvedor.expert/appendix/docker/armazenamento.html?q=>>>. Acesso em: 08 jan 2024.

PALAVRAS-CHAVE

Container. Docker. Docker Hub. ECR. Linux. Rede. Volumes. WSL.

EXEMPLO

POS TECH