# Docker

Conceitos Gerais

# 1. Introdução

- Uma máquina virtual é uma ferramenta muito importante na computação, pois permite que plataformas e tecnologias diferentes possam rodar na mesma máquina.

- Entretanto, é um recurso que demanda uma fatia considerável de recursos computacionais. Neste ponto, o uso de containers pode ser uma alternativa interessante.

- Usar **containers no Docker** pode ser melhor que **máquinas virtuais (VMs)** em vários cenários, principalmente quando você quer **leveza, velocidade e eficiência**.

- O container foi criado para acabar com situações em que um determinado sistema roda em um computador, mas não em outro.

- A ideia consiste em colocar tudo o que for necessário para um sistema rodar (bibliotecas, plugins...) dentro de um container, que será portável para qualquer máquina com docker.

# 2. Máquina Virtual

- Antigamente, uma empresa com várias aplicações diferentes (plataformas diferentes) precisava ter diversos servidores. Isso gerava custo com manutenção, energia, licenças, entre outros.

# 2. Máquina Virtual

- Outro problema no uso de servidores nativos é que o hardware tem que ser mensurado sempre na demanda máxima. Na maior parte do tempo o servidor ficará com hardware ocioso.

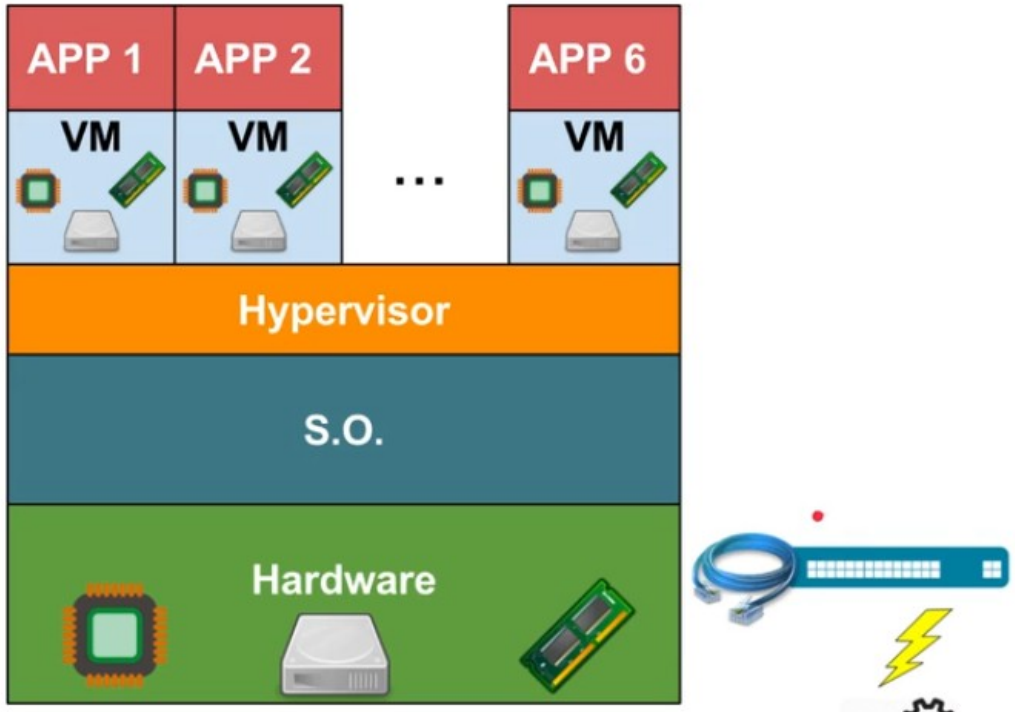

Carga


Muito tempo ocioso!

Muitos recursos desperdiçados!

# 2. Máquina Virtual

- Com as máquinas virtuais, é possível ter diversos sistemas rodando no mesmo servidor físico. Além disso, existe um melhor aproveitamento do hardware.



# 2. Máquina Virtual

- Apesar de ser uma excelente ferramenta, as máquinas virtuais também apresentam fragilidades e problemas.

- Para funcionar, além do hypervisor, cada máquina virtual precisa ter um sistema operacional rodando. A soma dos recursos necessários para rodar todos estes sistemas é alto.

- Além disso, cada máquina virtual requer manutenção, atualizações e configurações. Esse processo pode ser demorado e custoso.

# 3. Container

- Um **container** é uma forma de **empacotar e executar um software junto com tudo o que ele precisa** (código, bibliotecas, dependências e configurações) de maneira **isolada**, leve e portátil.

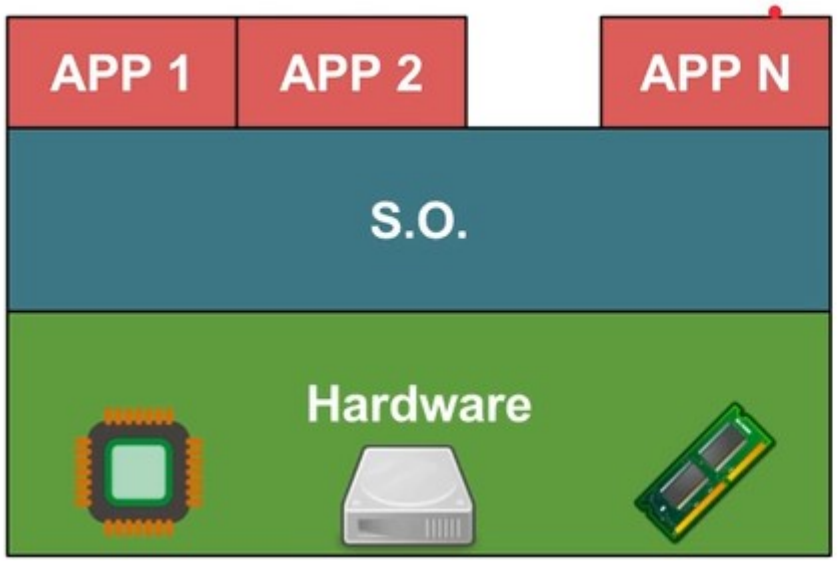
- Os containers rodam sobre o sistema operacional do host, compartilhando o mesmo sistema operacional. Por essa razão, rodar um container é muito mais leve do que rodar uma máquina virtual.

- Imagine um exemplo de um programa Python que utiliza a versão 3.1, com a biblioteca Pandas 4.5 e o banco de dados MySQL 8.1.3.

- É possível empacotar todos esses requisitos em um container e rodá-lo em qualquer máquina, sem risco de problemas ou conflito de versões.

# 4. Importância da Virtualização

- Em um ambiente corporativo, é arriscado instalar toda a estrutura em um único servidor, com um único SO.

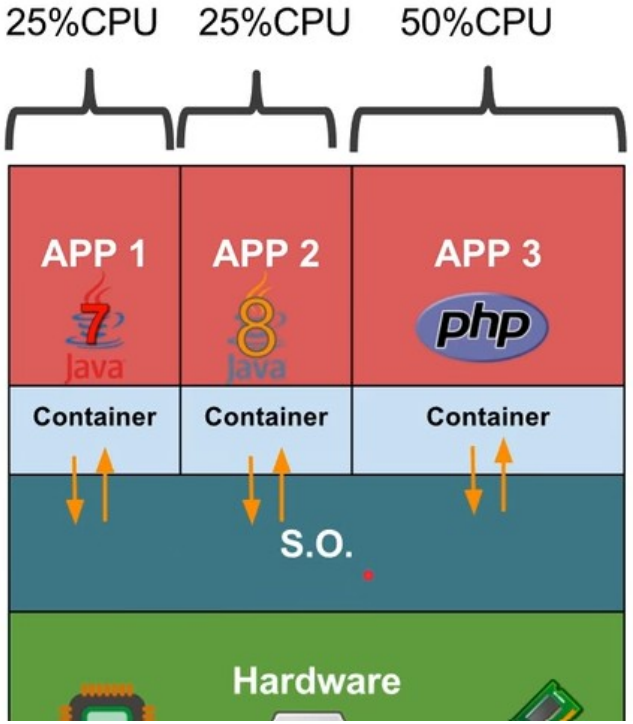



Os problemas desta abordagem:

- Dois apps utilizando a mesma porta de rede ?
- E se um app começar a consumir muito de um recurso, como a CPU?
- E se cada app precisar de uma versão específica de uma linguagem ?

# 4. Importância da Virtualização

- Um container possibilita uma flexibilidade maior, mensurando o poder computacional para cada uma das aplicações do servidor.




- Melhor controle sobre o uso de cada recurso ( CPU, Disco, Rede...etc)
- Agilidade na hora de criar ou remover um container
- Maior facilidade na hora de trabalhar com diferentes versões de linguagens e bibliotecas
- Mais leves que as VM

# 5. Docker

- O Docker é um conjunto de ferramentas desenvolvido pela dotCloud que fornece a infraestrutura para rodas os containers.

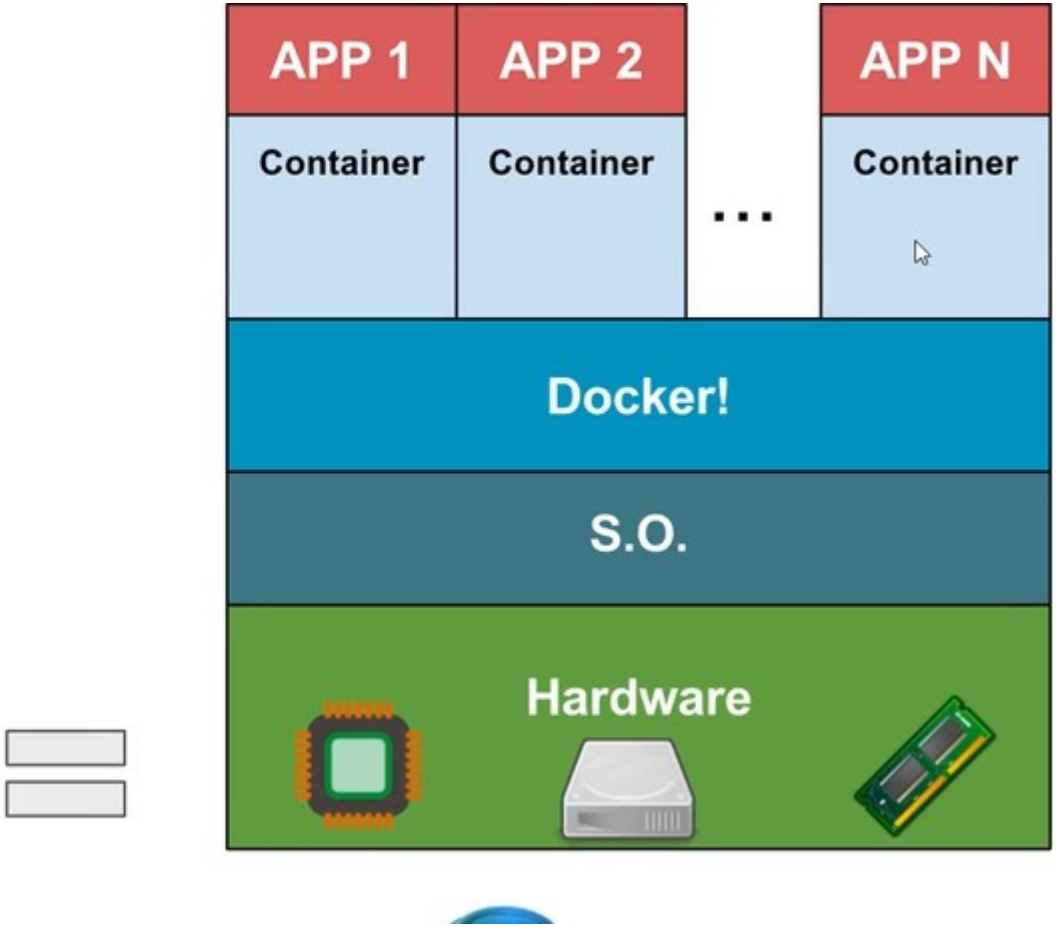
- Inicialmente a dotCloud fornecia serviços de PaaS (Plataform as a Service), onde os desenvolvedores hospedavam o seu código, e a empresa cuidada de toda a infraestrutura para executá-lo.

- A dotCloud alugava servidores de Amazon, e faziam o gerenciamento para os seus clientes. Visando economizar, ela desenvolveu o Docker, que permitia que tendo apenas um servidor, ela pudesse rodar aplicações de diversos clientes.

- Com o passar do tempo, ela percebeu que o Docker era uma ferramenta muito valiosa. Com isso, a empresa mudou seu nome para Docker Inc, e atualmente vende serviços relacionados ao Docker.

# 5. Docker

- A imagem abaixo mostra como a dotClould usava o Docker para economizar na Amazon:

# 5. Docker

- O Docker é um conjunto de ferramentas com a tecnologia para gerenciar os containers. Ele faz a intermediação entre o sistema operacional e os containers. O software que faz a execução dos containers é o **Docker Engine**.

- Existem instaladores do Docker Engine para a maioria dos sistemas operacionais. Basta fazer a instalação, para que o computador esteja apto a rodar qualquer container.

- Essa característica que fornece grande portabilidade a tecnologia de containers. Além disso, o Docker Engine é muito mais leve que um Hypervisor.

- Existem outros software que fazem parte do ecossistema, como o **Docker Compose** (orquestrar múltiplos containers), **Docker Swarm** (containers rodando em clusters), entre outros.

- Também existe o **Docker Store**, que oferece milhares de imagens de containers prontos para uso.

# 5. Docker

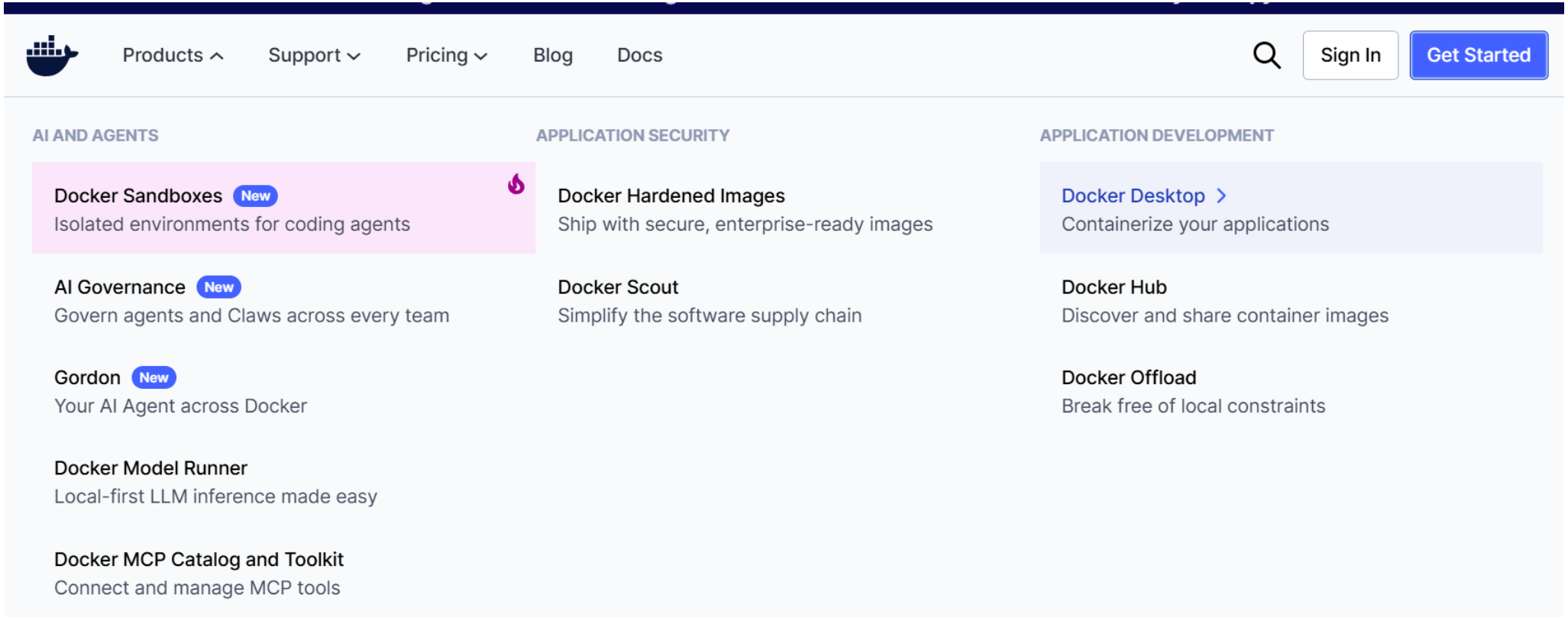
- O **Docker Engine,** que é o “motor” que executa os containers, pode ser usado gratuitamente.

- Mas também existe o **Docker Pro**, que é uma versão com recursos corporativos, e possui segurança avançada, gerenciamento de usuários, controle de acesso e suporte técnico.

- Também é possível pagar uma assinatura para utilizar o **Docker Store** com recursos avançados, mais armazenamento e a possibilidade de ter imagens de containers privadas.

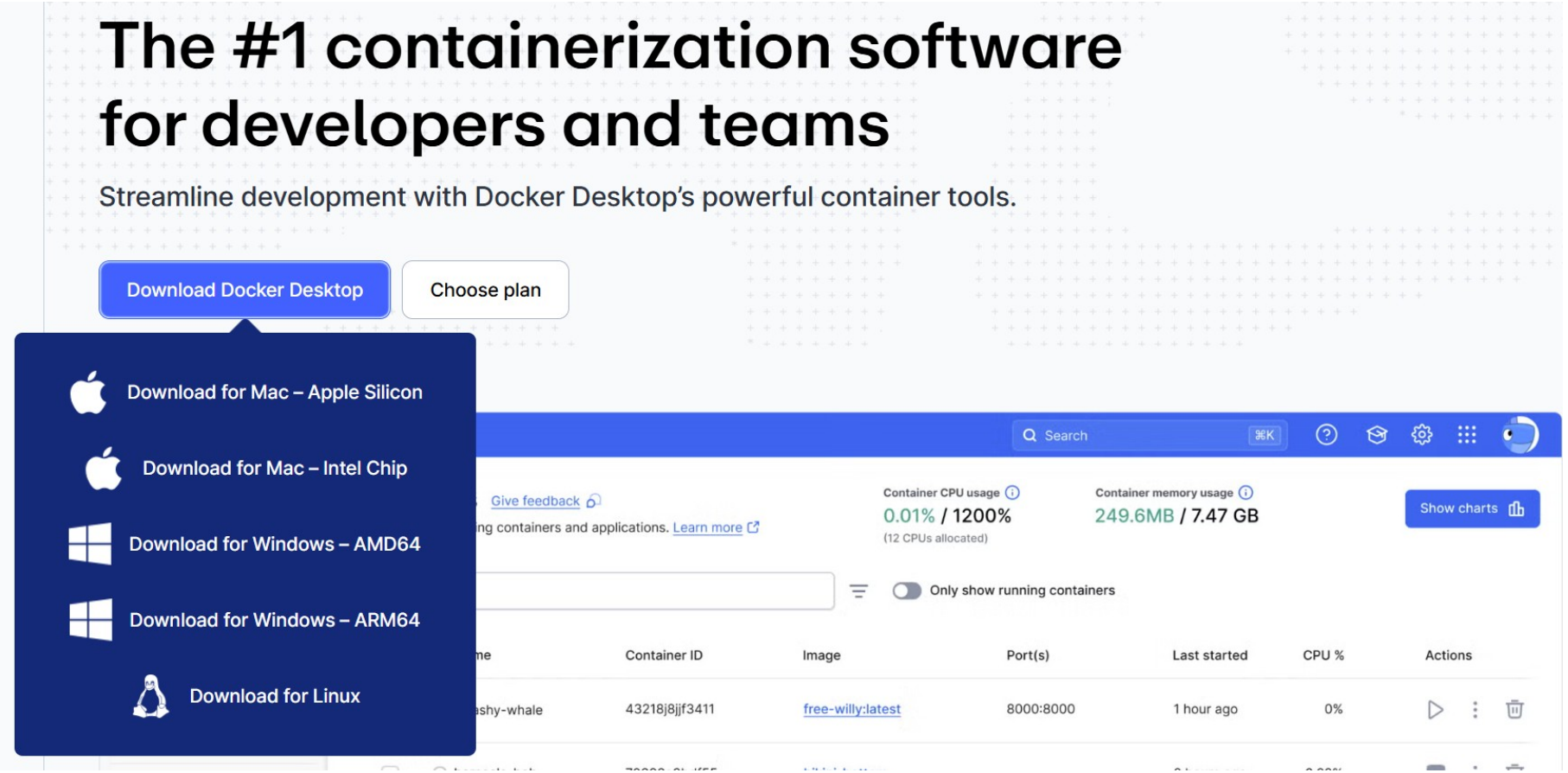
# 6. Instalação do Docker

- Entrando no site do Docker, no menu "Products", opção "Docker Desktop", é possível fazer o download do "Docker Engine":



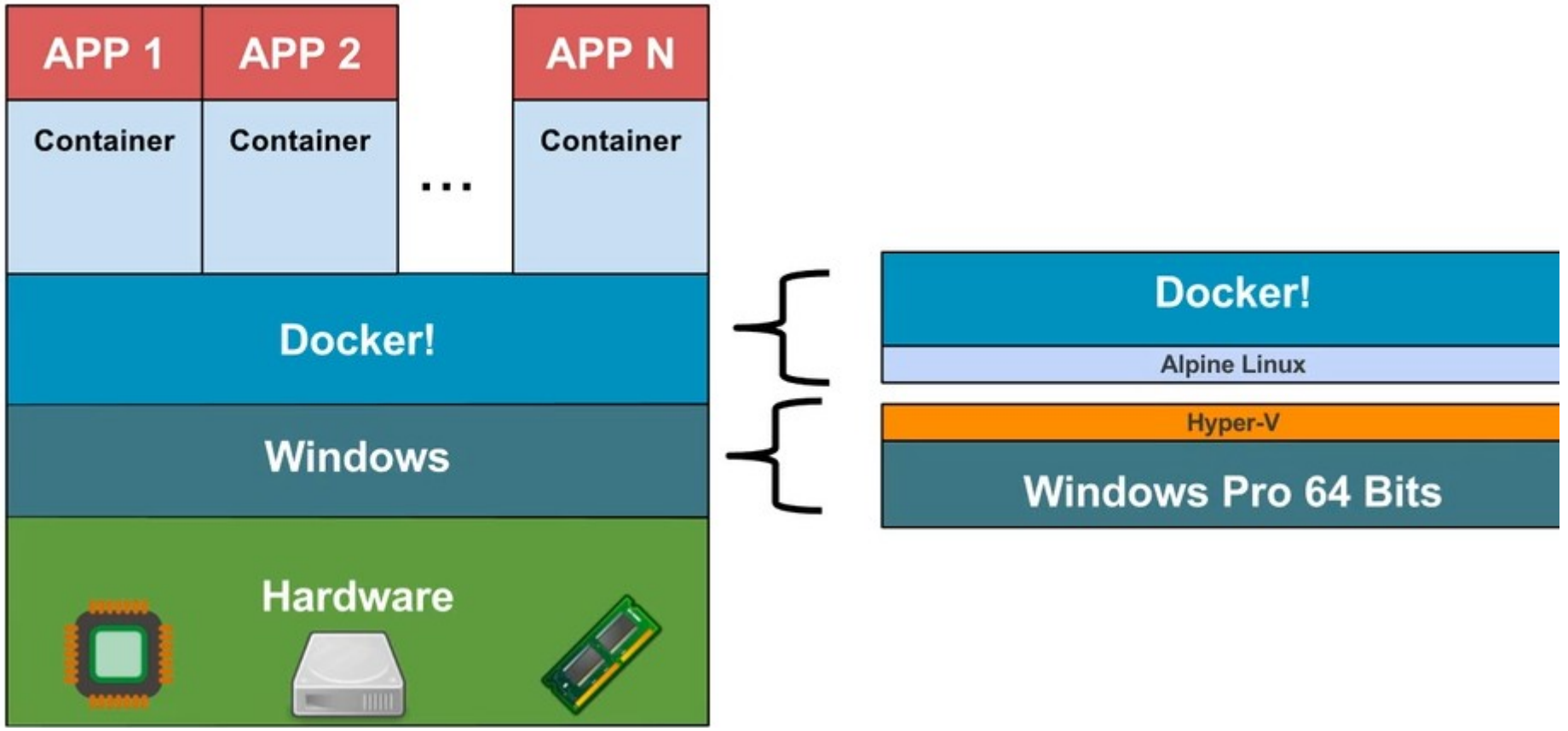
# 6. Instalação do Docker

- Basta selecionar o "Docker Engine" para o sistema operacional desejado:



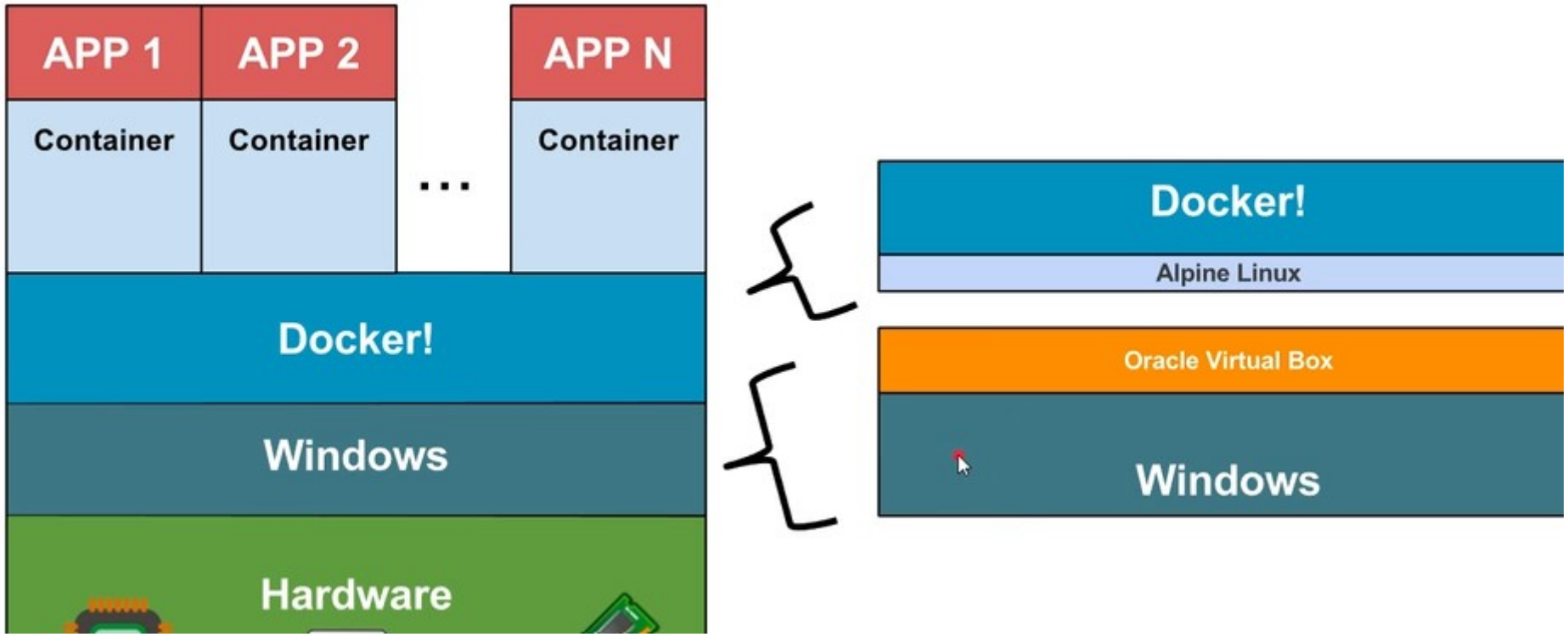
# 6. Instalação do Docker

- O Docker utiliza uma pequena máquina virtual chamada "Alpine Linux" para funcionar. Por isso é necessário um hypervisor. No caso do Windows 11, ele já vem com o Hyper-V, então a instalação ocorre sem problemas:



# 6. Instalação do Docker

- Para quem usa Windows 10, é preciso instalar o "Docker Toolbox", que vai utilizar o virtualbox da Oracle para rodar. Esta opção não é recomendada, apenas se não houver alternativa:



# 6. Instalação do Docker

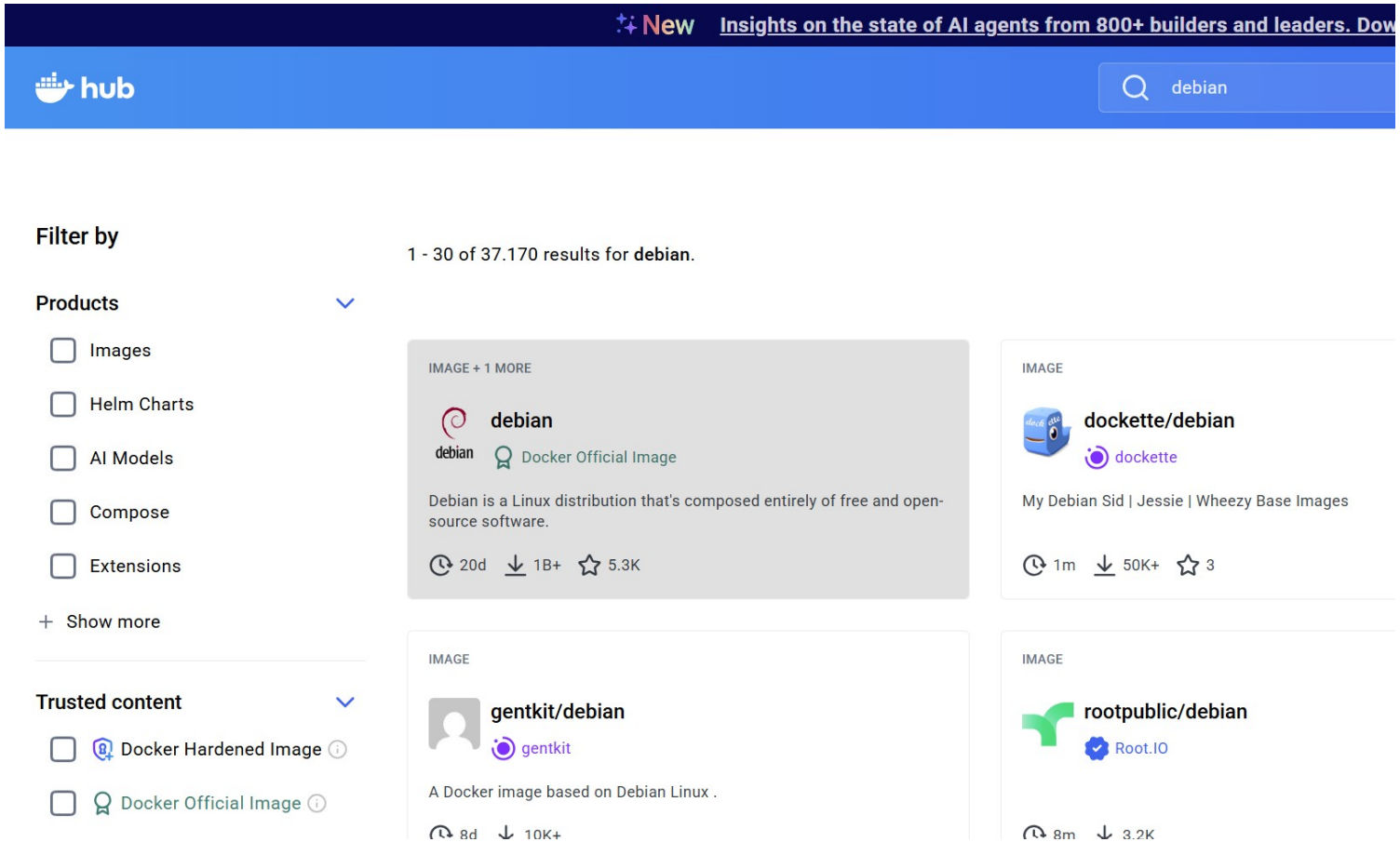
- Para verificar se a instalação está funcionando, podemos utilizar os seguintes comandos:

```
#mostra a versão do docker
sudo docker version

#executa um container chamado hello-world
sudo docker run hello-world
```

# 7. Docker Store

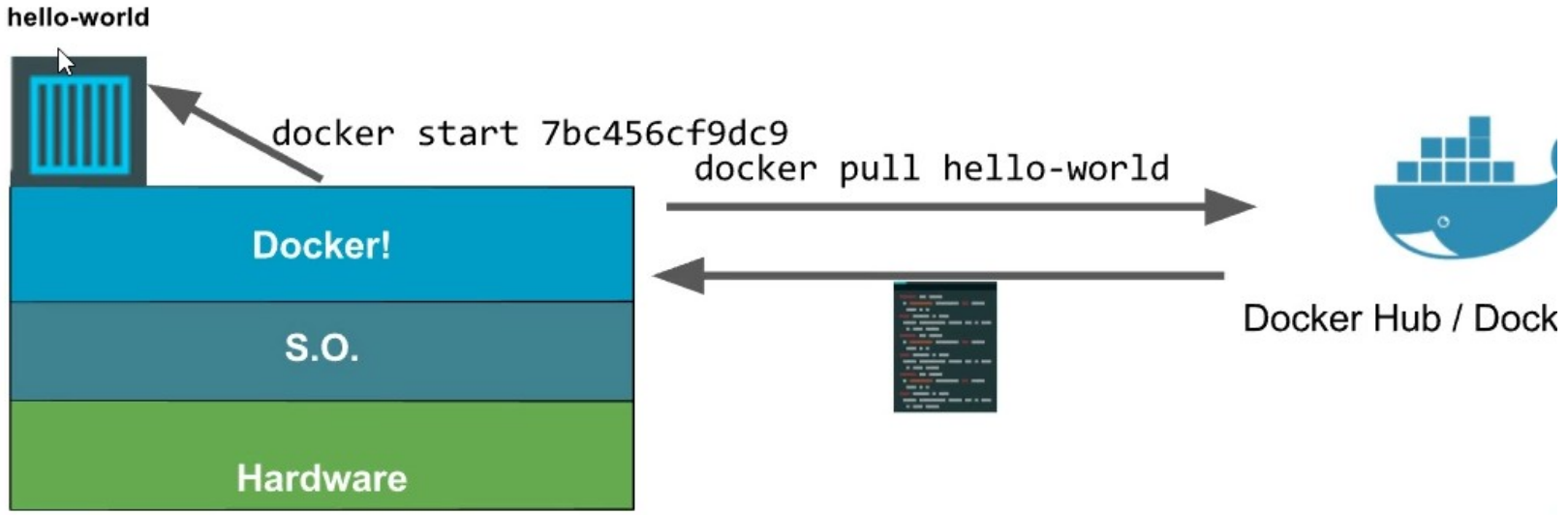
- A Docker Store é um repositório com milhares de imagens para construção de containers. A figura abaixo mostra uma imagem oficial do Debian:



# 8. Comandos Básicos do Docker

- Para entender o fluxo de funcionamento do Docker, o comando "docker run hello-world" faz a busca local da imagem "hello-world" para iniciar um container. Caso não tenha local, a busca é feita no Docker Hub.

- Uma imagem é como se fosse uma receita para criar o container. Assim que a imagem é encontrada, o container é formado e inicializá-lo:



# 8. Comandos Básicos do Docker

- Vendo o resultado do "docker run" na prática:

```
#baixar imagem do debian
sudo docker run debian

#mostra os containers ativos no momento (no caso nenhum, todos estão parados)
sudo docker ps

#mostra todos os containers parados
sudo docker ps –a

#cria um novo container do debian e executa o comando ls –l /
sudo docker run debian ls –l /

#cria um container mas atrela o terminal ao debian (como se tivesse no debian)
sudo docker run –it debian
```

# 8. Comandos Básicos do Docker

- Vendo o resultado do "docker run" na prática:

```
#inicia o container pelo id (não cria, apenas inicia) e atrela ao terminal
#precisa ver o id do container com ps -a
sudo docker start –a –i 4589786529

#parar o container
sudo docker stop 4589786529

#remover um container
sudo docker rm 4589786529

#remove todos os containers inativos
sudo docker container prune
```

# 8. Comandos Básicos do Docker

- Vendo o resultado do "docker run" na prática:

```
#ver as imagens baixadas
sudo docker images

#apagar uma imagem
sudo docker rmi hello-world

#baixa uma imagem de uma versão especifica do ubuntu
sudo docker run ubuntu:14.04
```

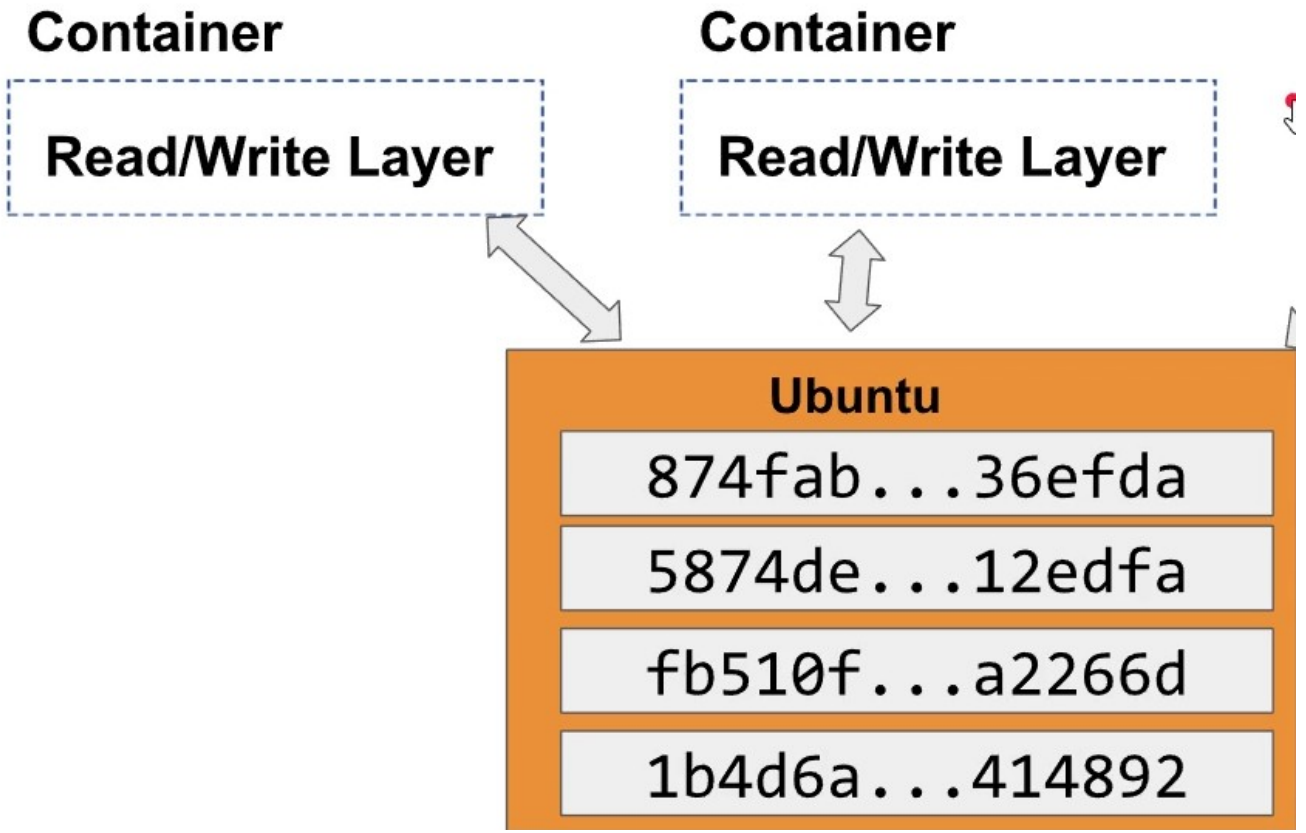
# 9. Layered Filesystem

- Quando uma imagem é mais complexa, ela vem em várias camadas. A vantagem é que outra imagem pode aproveitar as mesmas camadas.

- Na figura abaixo, as camadas da imagem do Ubuntu que são iguais aos do CentOS são baixadas apenas uma vez, e aproveitada por ambas, economizando recursos computacionais:

# 9. Layered Filesystem

- As camadas das imagens são "ready only", logo não podem ser modificadas. Quando um container é criado, é como se uma nova camada de cópia fosse criada, essa pode ser modificada. A vantagem é que uma imagem pode ser aproveitada por vários containers:



# 10. Imagens Não-Oficiais

- As imagens não oficiais precisam ser especificadas com o nome do usuário, como mostra o quadro:

```
#cria container com imagem que instala um servidor web e possui um site estático
sudo docker run dockersamples/static-site

#ao instalar o terminal fica inacessível, com um servidor web rodando
#precisa fechar o terminal e depois utilizar um run com parametro "d"
#com "d" o servidor roda e permite o uso do terminal
#o "P" é para alocar uma porta para o servidor web
sudo docker run –d –P dockersamples/static-site

#com esse comando é possível ver o IP e a porta
#basta inserir o ip e a porta no navegador para ver o site
sudo docker port dedrdygugu
```

# 10. Imagens Não-Oficiais

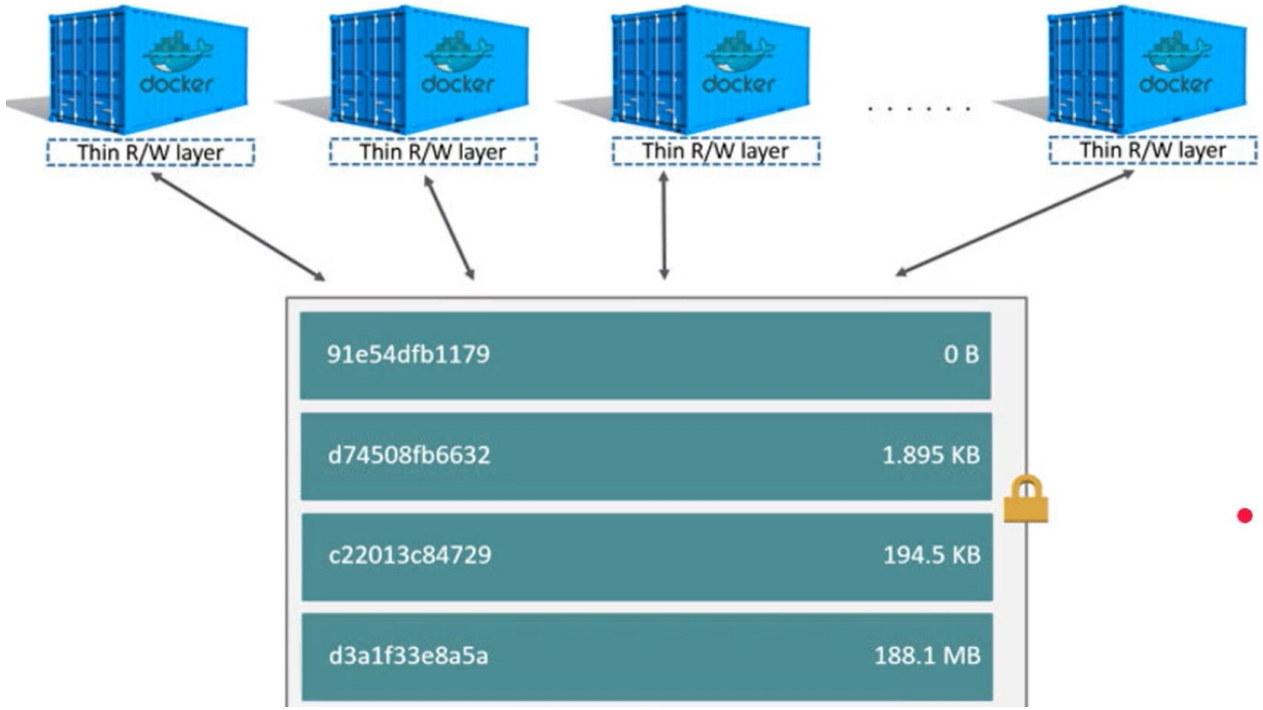
- Mais alguns comandos básicos do docker run:

```
#cria o container com uma alias, para não precisar usar id
#parametro "p" permite especificar a porta mapeada com a porta 80
#"P" cria uma porta aleatória
sudo docker run –d –p 12345:80 –name meu-site dockersamples/static-site

#para ver a porta usa o alias meu-site
sudo docker port meu-site
```

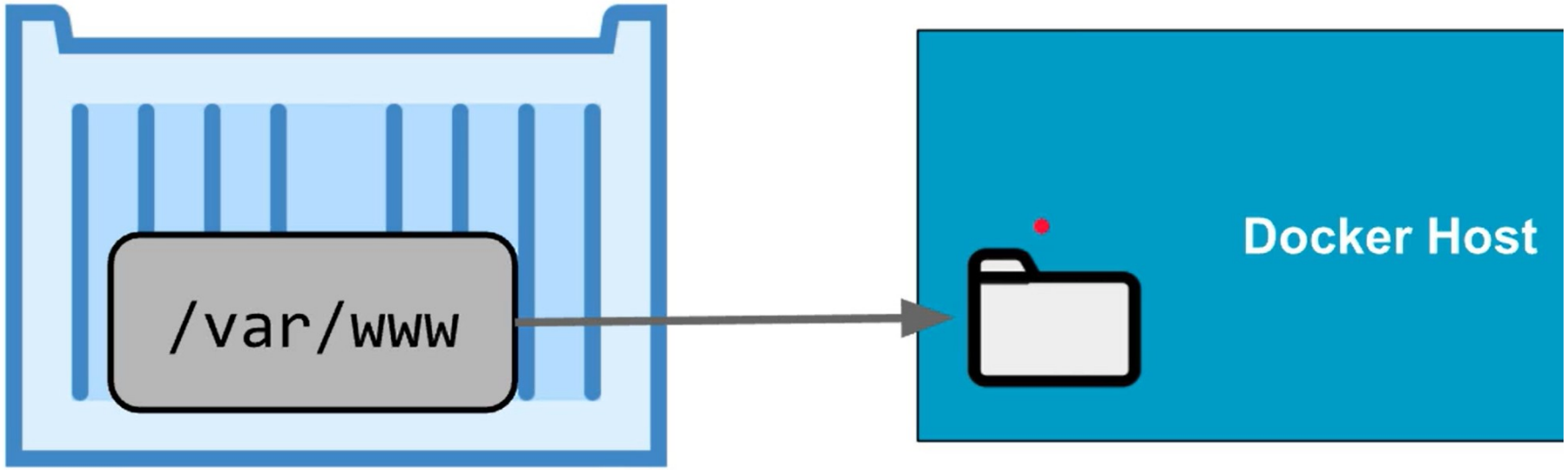
# 11. Volumes

- As camadas da imagem que forma o container são "ready only". Quando o container é criado, existe uma camada que permite alteração e inserção de dados. Entretanto, o container é volátil, ou seja, quando ele é excluído todos os seus dados automaticamente são perdidos.



# 11. Volumes

- Pode ser necessário persistir os dados de um container, pois ele pode conter serviços de banco de dados, log, etc. Para isso, podemos criar um volume, que é basicamente criar uma conexão de um diretório do container com um diretório do sistema operacional.



# 11. Volumes

- O parâmetro para criar o volume é o "v". O quadro abaixo mostra como criar um volume em um container:

```
#cria o container com a imagem do ubuntu
#com "v" cria o volume
#dados do container em /var/www vão para /home/paulo/Documents/container
# o parametro "it" é para vincular o terminal ao container
sudo docker run –it –v "/home/paulo/Documents/container:/var/www" ubuntu

#criando um arquivo em /var/www ele ficará persistido no volume
touch /var/www/log.txt
```

# 11. Volumes

- Uma outra opção muito interessante é executar um código em um container. No exemplo abaixo, vamos executar um código PHP em um container que já vem com o servidor Apache e o PHP instalados:

```
#cria o container com a imagem do apache e php
#com "v" cria o volume onde ficará o programa PHP para ser executado
sudo docker run –d –p 8181:80 –v "/home/paulo/Documents/container:/var/www/html
php:8.2-apache

#basta colocar um arquivo php na pasta do volume, e ele será executado no
navegador
```