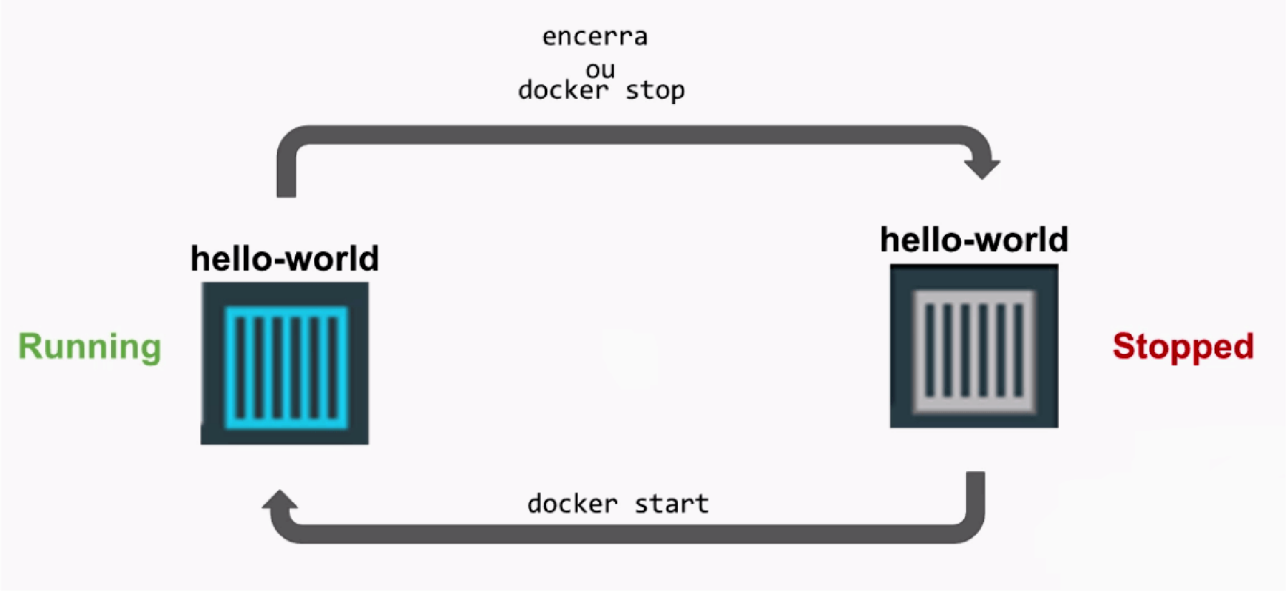
02

Layered File System

Transcrição

Vimos no vídeo anterior os dois principais estados de um *container*, quando criamos um ou iniciamos, ele fica no estado de ***running***, e quanto a sua execução encerra ou paramos, ele fica no estado de ***stopped***:



Removendo containers

Só que com os testes que fizemos até agora, acabamos criando vários *containers* (lembrando que podemos ver todos os *containers* criados executando o comando `docker ps -a`) e nunca removemos algum deles, já que os comandos acima só mudam os seus estados. Para **remover** um *container*, executamos o comando `docker rm` , passando para ele o **id** do *container* a ser removido, por exemplo:

```
docker rm c9f83bfb82a8
```

COPIAR CÓDIGO

Mas para limpar todos os *containers* inativos, devemos remover um por um? Não, pois há um novo comando do Docker, o `prune`, que serve para limparmos algo específico do Docker. Como queremos remover os *containers* parados, executamos o seguinte comando:

```
docker container prune
```

COPIAR CÓDIGO

O comando é tão poderoso que ele pede para confirmarmos se é isso mesmo que queremos fazer.

Listando e removendo imagens

E do mesmo jeito que temos o comando `docker container` para mexermos com o *container*, temos o comando `docker images`, que nos exibe as imagens que temos na nossa máquina. Para remover uma imagem, utilizarmos o comando `docker rmi`, passando para ele o nome da imagem a ser removida, por exemplo:

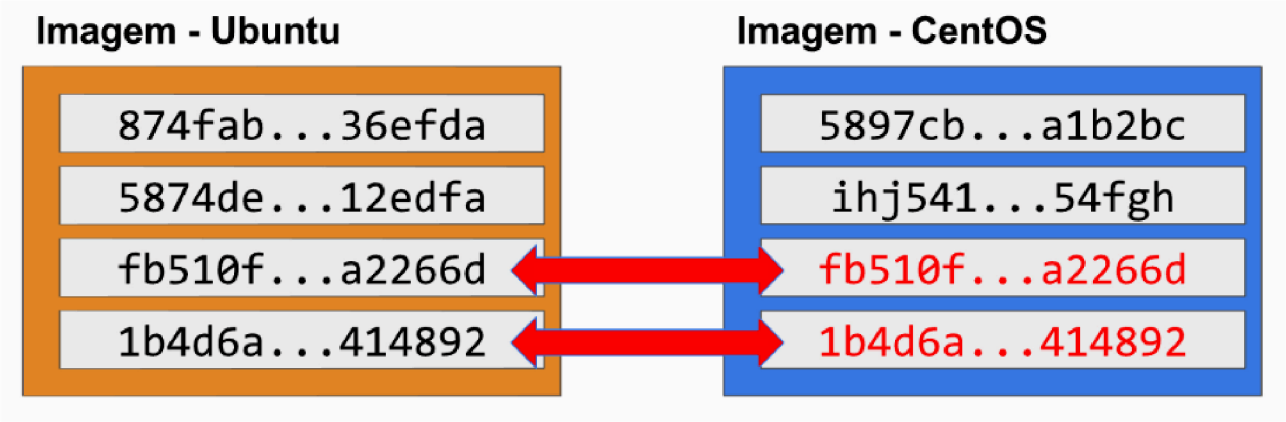
```
docker rmi hello-world
```

COPIAR CÓDIGO

Camadas de uma imagem

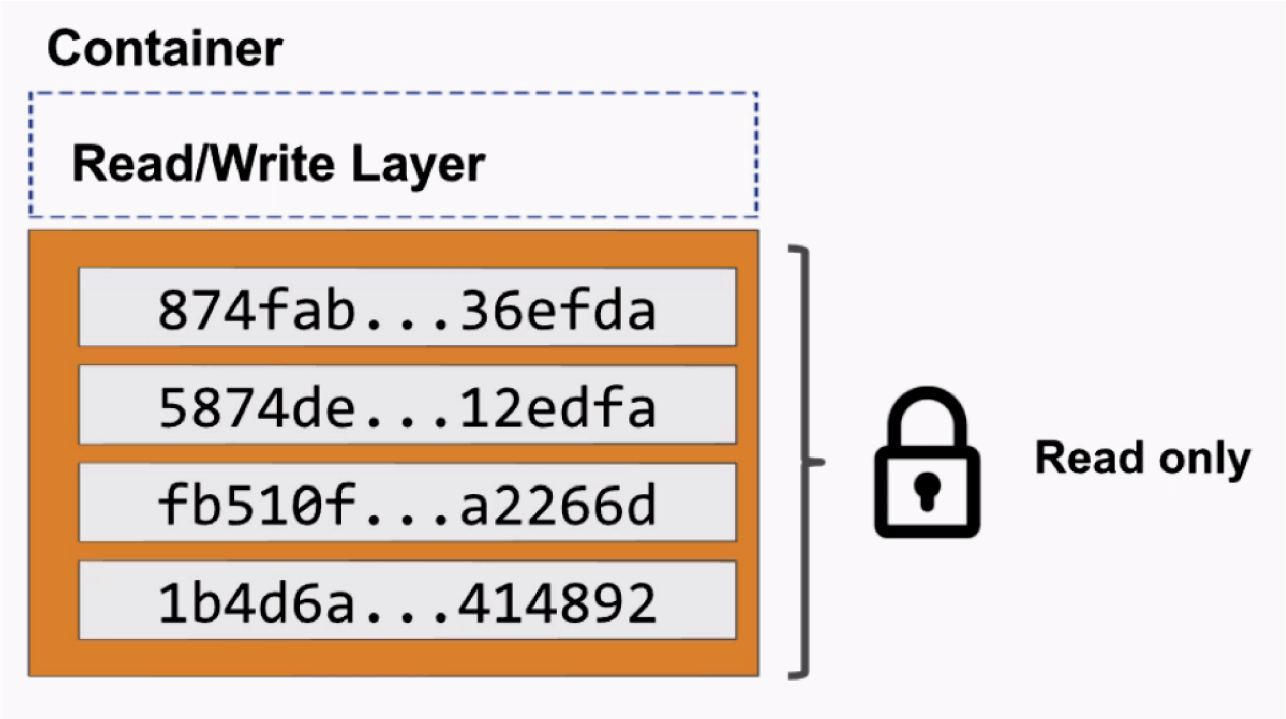
Na aula anterior, quando baixamos a imagem do Ubuntu, reparamos que ela possui **camadas**, mas como elas funcionam? Toda imagem que baixamos é composta de uma ou mais camadas, e esse sistema tem o nome de ***Layered File System***.

Essas camadas podem ser **reaproveitadas** em outras imagens. Por exemplo, já temos a imagem do Ubuntu, isso inclui as suas camadas, e agora queremos baixar a imagem do CentOS. Se o CentOS compartilha alguma camada que já tem na imagem do Ubuntu, o Docker é inteligente e só baixará as camadas diferentes, e não baixará novamente as camadas que já temos no nosso computador:

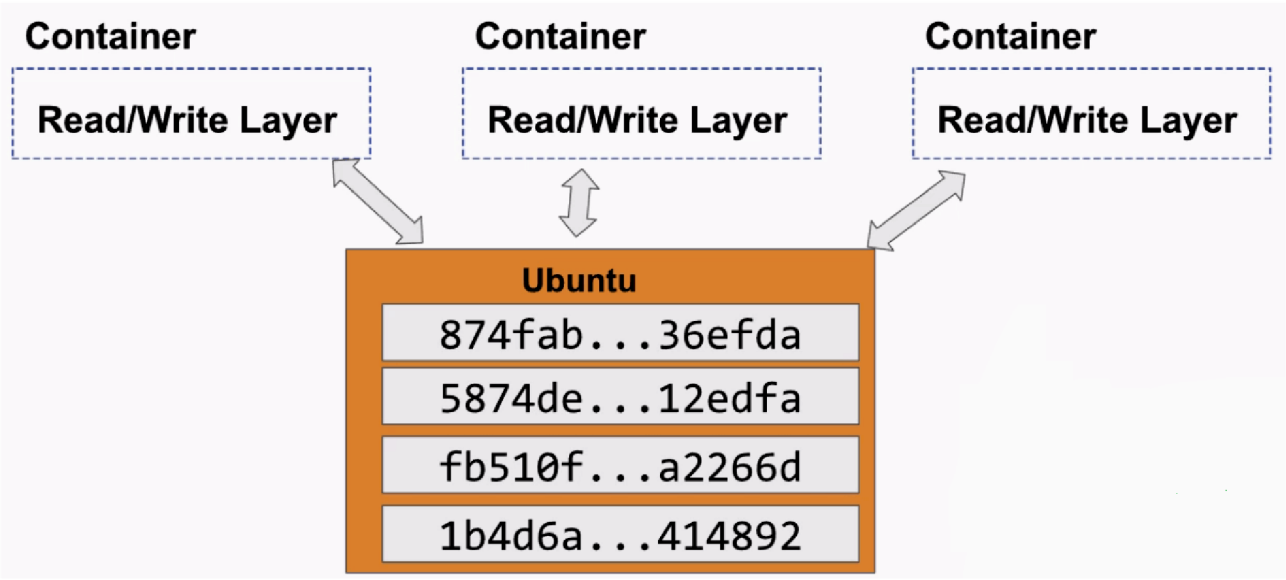


No caso da imagem acima, o Docker só baixará as duas primeiras camadas da imagem do CentOS, já que as duas últimas são as mesmas da imagem do Ubuntu, que já temos na nossa máquina. Assim poupamos tempo, já que precisamos de menos tempo para baixar uma imagem.

Uma outra vantagem é que as camadas de uma imagem são **somente para leitura**. Mas como então conseguimos criar arquivos na aula anterior? O que acontece é que não escrevemos na imagem, já que quando criamos um *container*, ele cria uma nova camada acima da imagem, e nessa camada podemos ler e escrever:



Então, quando criamos um *container*, ele é criado em cima de uma imagem já existente e nele nós conseguimos escrever. E com uma imagem base, podemos reaproveitá-la para diversos *containers*:



Isso nos traz economia de espaço, já que não precisamos ter uma imagem por *container*.