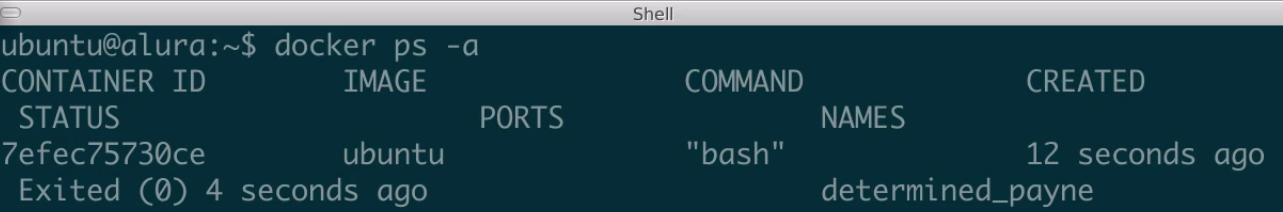
# **Capítulo 3 - Interatividade com containers conhecendo os principais comandos**

Comecemos esta aula criando um container com as opções já vistas:

docker run -it ubuntu bash

Será feito o *pull* da imagem do ubuntu e o container será criado. Automaticamente entramos dentro do container, saímos usando o comando **exit**. Podemos observá-lo fazendo **docker ps -a**:

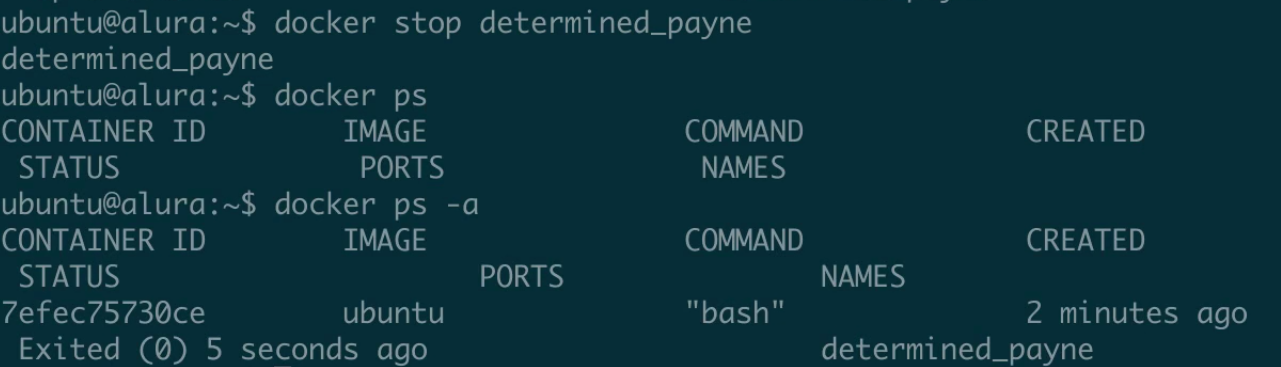


O container possui estados, ele é persistente. Perceba que saímos dele 4 segundos atrás o "(0)" é seu estado, ou seja, "Exited". O Docker nos dá o controle do estado desses containers. Vamos reiniciar o container usando o comando **docker start**:

docker start determined\_payne

Para não precisarmos ficar usando o **ps -a** todo momento que quisermos buscar informações sobre os containers, basta escrevermos o comando que queremos processar e apertar a tecla Tab duas vezes seguidas, autocompletanto as informações.

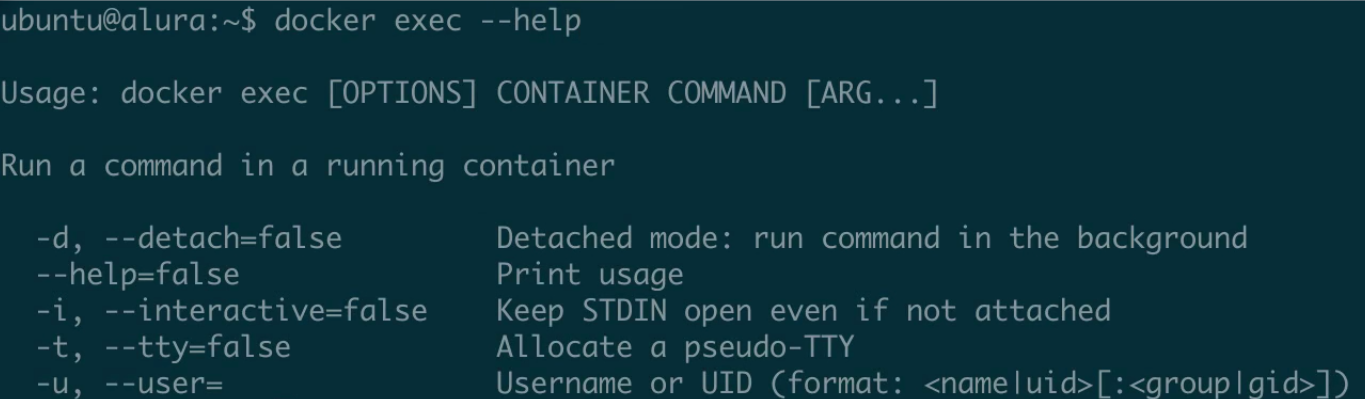
Para o container parar de funcionar usamos o comando **docker stop**. Assim ele ficará visível ao acessarmos com o **-a**:



Continuando com os controles de estados de containers, vamos reiniciá-lo novamente e ver algumas instruções mais profundamente:

### **- docker exec**

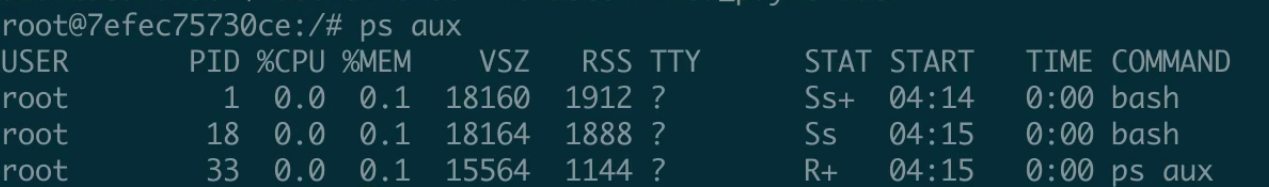
Executamos comandos em tempo de execução para o container. Vamos conferir todas suas instruções possíveis:



Perceba que todas as opções são desabilitadas por padrão. Vamos executar a imagem ubuntu com o **-i** e o **-t**, lembrando que passamos o nome do container:

docker exec -it determined\_payne bash

Assim entramos dentro do container e, se fizermos **ps aux** percebemos que criou-se um novo bash:



O primeiro bash foi quando rodamos o **run** no começo da aula, o qual reiniciamos várias vezes. O segundo é o que acabamos de pedir para executar.

Uma outra forma de uso interessante para o **exec** é enviar comandos para dentro desse container que está em execução e a saída desses comandos vai retornar para o Shell da máquina local. Um comando é o **ps aux**:

docker exec -it determined\_payne ps aux

Retorna a mesma saída de antes, só que dessa vez com apenas um processo bash. Isso acontence porque o processo que passamos na instrução **exec** não foi um **bash**, mas um **ps aux**.

### **- docker kill**

Digamos que o Apache travou e nosso container está rodando nele. A instrução **stop**não está obedecendo, sendo ela a melhor para solucionar o problema até agora. Temos que achar outra solução.

No Linux, para matar um processo, pegamos seu PID e passamos o comando **kill** com a opção **-9**:

kill -9 [PID do processo]

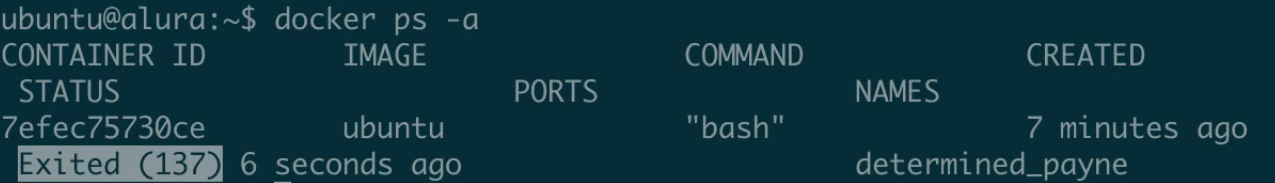
O Docker possui a mesma funcionalidade, sendo que podemos passar tanto o ID quanto o nome do container:

docker kill 7efec75730ce

ou

docker kill determined\_payne

O status de pausa dele será diferente:



## **Containers *não persistentes* (ou *descartáveis*)**

Toda vez que criamos esses containers, os mesmos aparecem na listagem do docker (**ps**). Para tais containers persistentes temos total controle utilizando as instruções vistas até agora. Mas e se quisermos criar um que não seja persistente, no qual não precisaremos utilizar as instruções?

Digamos que queremos criar um container para verificar a instalação de um pacote do Apache. Queremos testar tal pacote com o **apt-get install** na nossa distribuição do Ubuntu. Então teríamos o seguinte comando:

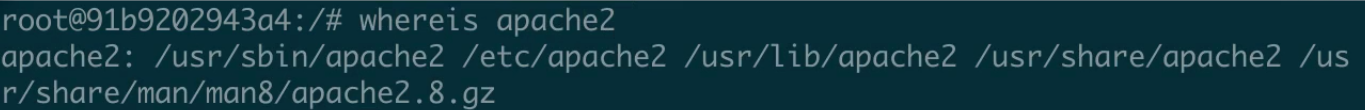
docker run --rm -it ubuntu bash

O **--rm** é uma instrução que irá funcionar no final da execução do container e irá removê-lo. Ou seja, quando sairmos dele, não constará no background dos processos, ou seja, é um container não persistente.

Façamos o teste:

apt-get update && apt-get install -y apache2

E verificamos que a instalação ocorreu como o previsto:



A tarefa com esse container está concluída. Ele foi criado somente para testar a instalação do Apache. Quando sairmos do container e verificarmos com **docker ps -a**, observamos que ele não existe mais. O **--rm** foi aplicado assim que a tarefa terminou.

Na próxima aula daremos os primeiros passos para construirmos nossa própria imagem.