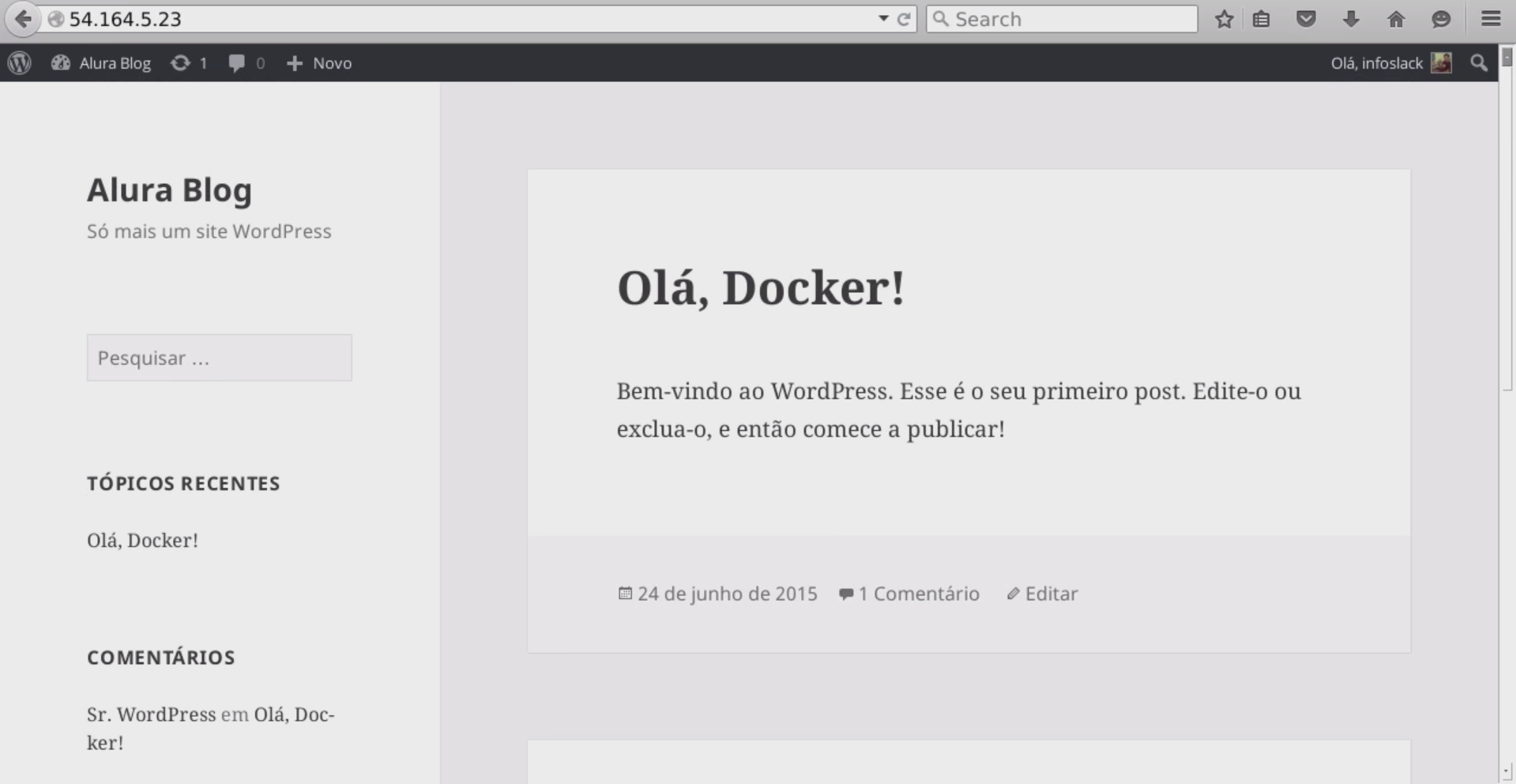
# **Capítulo 2 - O que são os *containers***

Na aula anterior criamos um blog rapidamente utilizando o WordPress. Na tela de edição já é possível criamos posts para o blog:



Fizemos isso usando o Docker. Ele possui uma série de comandos que podemos visualizá-los como um todo usando o comando

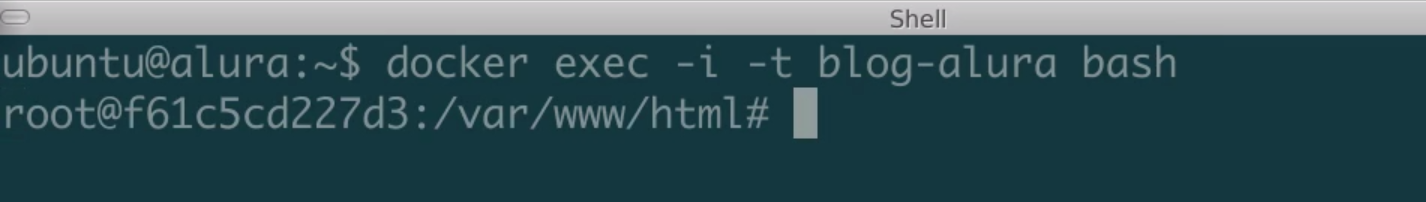
docker help

Eles deixam bem fácil a interatividade com os containers. Veremos mais alguns nesta aula. Por exemplo, como sabemos que um container está de fato isolando as imagens? Utilizando o comando **ps aux** no Terminal do Ubuntu conseguimos visualizar todos os processos correndo em nossa máquina, muito provavelmente serão muitos. Vamos contabilizá-los usando **ps aux | wc -l**.

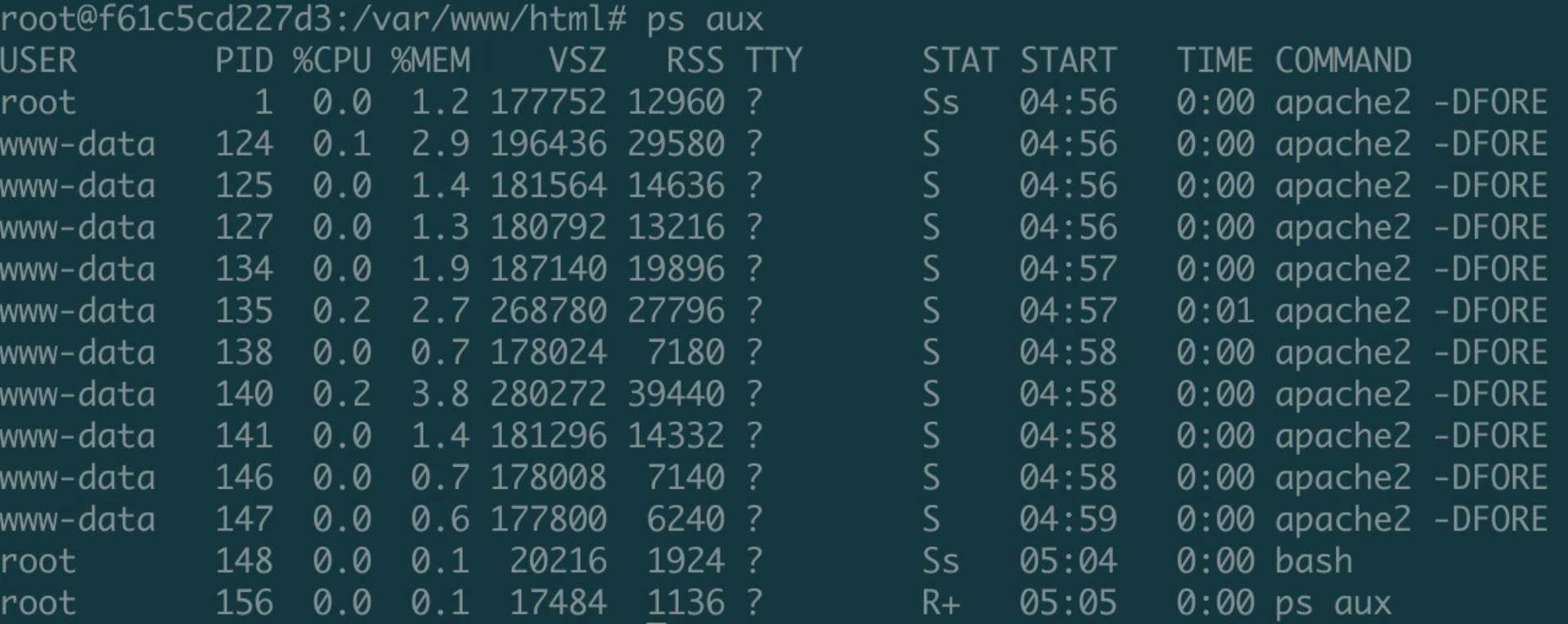
Para entrarmos em um container em execução usamos a opção **exec** + **-i**, o que garante uma interatividade com o Shell ativo de dentro desse container, e o **-t**, que irá simular o TTY. Além disso informamos o nome do container que queremos ter acesso, e o que queremos executar dentro dele, no caso, o *bash*:

docker exec -i -t blog-alura bash

Ao executarmos, perceba que o prompt irá mudar:



Fomos parar dentro do diretório onde está aplicação do WordPress. E agora podemos listar os processos dentro desse diretório, que serão em número bem menor

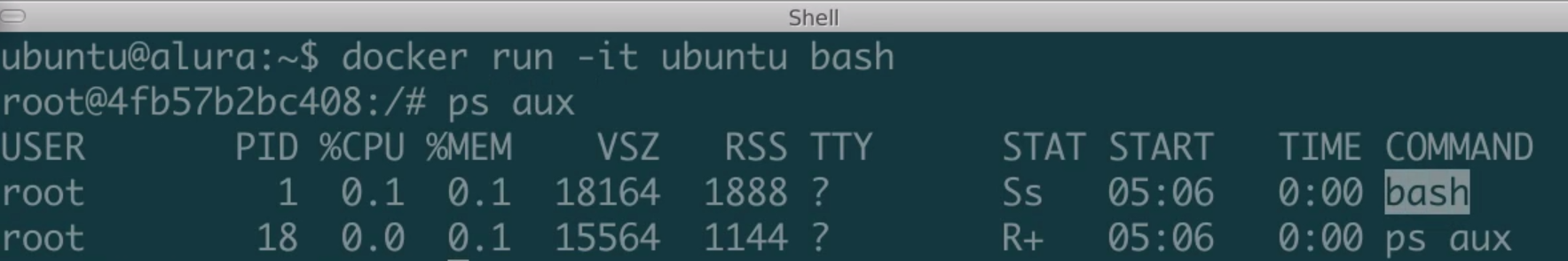


Todos os processos são do Apache rodando com PHP a aplicação no WordPress. Houve um isolamento desses processos. Saímos usando **exit** para voltar ao prompt original.

Da mesma forma que utilizamos **-i -t** com o **exec**, também podemos fazê-lo com o **run**. Toda vez que criarmos um container interativo, teremos obrigatoriamente que utilizar tais opções. Por exemplo:

docker run -it ubuntu bash

Perceba que podemos juntar as duas opções em uma só. Rodando esse comando, entramos no novo container. Buscando os processos, só irá retornar o bash:



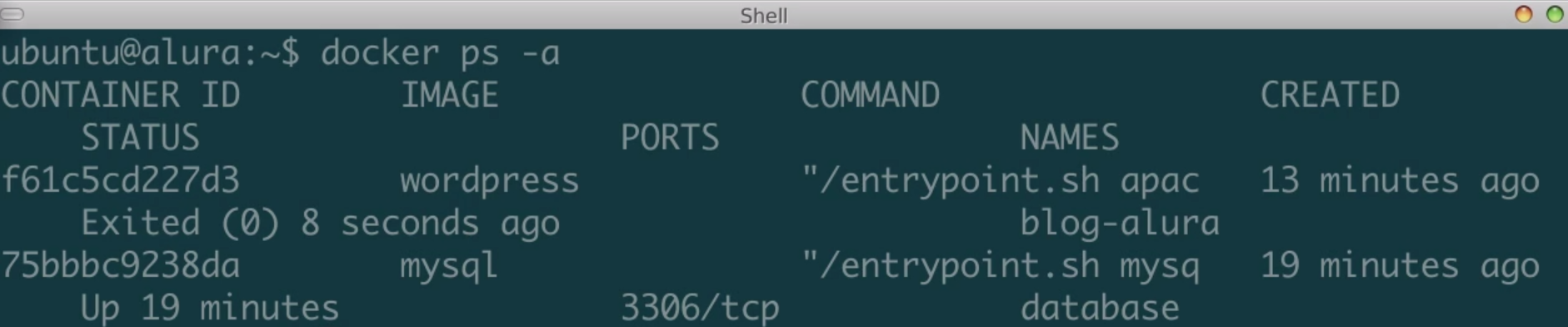
Como visto antes, o comando **docker ps** lista os containers em execução, enquanto que o **docker ps -a** lista todos os containers criados. Alguns deles não queremos mais que apareçam pois estão terminados. Para nos livrarmos deles usamos o comando **rm**, que serve para destruir um container, passando seu nome ou seu *id*:

docker rm [id]

ou

docker rm [nome]

Podemos também parar a execução de um container fazendo **docker stop [nome]**. Usando esse comando para parar o "blog-alura" e utilizando o que lista todos os containers:



Perceba que o "blog-alura" parou de rodar 8 segundos atrás.

Podemos removermos dois ou mais containers ao mesmo tempo, o que facilita muito nosso trabalho. Fazemos

docker rm $(docker ps -qa)

O **-q** serve para passar apenas os *ids* dos containers como parâmetro. Se usarmos o comando **docker ps -qa**, por exemplo, o Terminal retorna a listagem dos *ids*.

Com **rm** removemos containers. Para remover imagens usamos a opção **rmi** e passamos seu nome ou *id*:

docker rmi [nome]

ou

docker rmi [id]

E, de fato, ao excluirmos a imagem do ubuntu:

