# **Capítulo 1 - Rodando um WordPress e MySQL com Docker**

# **Sejam bem-vindos ao curso de Docker! Aqui veremos como construir, testar e rodar nossos *containers*. E, para esta primeira aula, nada melhor do que começarmos com um teste básico para criarmos nosso primeiro *container*.**

# **Com o Docker instalado, no Terminal fazemos:**

# **docker run ubuntu /bin/echo "Hello Docker!"**

# **A opção run vai servir para dar a inicialização no nosso container;**

# **Informamos o nome da imagem que queremos utilizar, aqui "ubuntu";**

# **O nome do comando que queremos executar dentro do container, para esse exemplo faremos /bin/echo;**

# **Tal co ntainer vai apenas replicar (echo) a string "Hello Docker!"**

# **Ao rodarmos, o Docker fará o download dessa imagem para criar o nosso container:**

# 

# **E, de fato, após esse processo a mensagem é impressa no Terminal:**

# 

# **Porém, de onde veio tal imagem? Ela veio do Docker Hub, o qual é o repositório oficial e público de imagens:**

# 

# **O Docker cria containers de forma a executar aplicações isoladas dentro da máquina. Esta ideia ficará muito mais clara no momento em que a colocarmos em prática, utilizando um exemplo no mundo real.**

# **Imaginemos que o pessoal do Alura nos contratou para criarmos um blog. Utilizaremos o *WordPress* e, junto dele, o banco de dados MySQL. Para fazermos isso rapidamente com o Docker, abrimos seu repositórtio e procuramos a opção pull (não se preocupe, não entraremos em detalhes sobre cada opção agora, mas sim no decorrer das aulas e com exemplos práticos):**

# **docker pull mysql**

# **O Docker fará o *pull* (puxar, download) dessa imagem para o nosso sistema:**

# **... Status: Downloaded newer image for mysql:latest**

# **Para vermos tais imagens das quais fizemos download, fazemos no Terminal:**

# **docker images**

# **O que nos retorna:**

# 

# **Para criarmos um container com a imagem, primeiro precisamos dar um nome para ele usando --name. Precisamos disso pois, quando o Docker cria um container, este vem com um nome aleatório. Se fizermos docker ps -a conseguimos visualizá-lo:**

# 

# **Perceba o "elegant\_ptolemy". Este foi um nome aleatório dado pelo Docker para nossa imagem "ubuntu". Então façamos diferentemente para a imagem do MySQL passando um nome próprio, um que seja mais fácil de memorizar:**

# **docker run --name database -e MYSQL\_ROOT\_PASSWORD=teste123 -d mysql**

# **O -e é a opção de *enviroment*, ou seja, setamos um valor para a variável de ambiente para o MySQL. O valor que definirmos na variável "MYSQL\_ROOT\_PASSWORD" será a senha de root para conseguirmos acessar o banco de dados do MySQL. Veremos mais detalhes sobre a opção -e em aulas futuras.**

# **Pedimos para o container rodar em background, para isso usamos a opção -d.**

# **Com o comando docker ps podemos observar o container que acabamos de criar:**

# 

# **STATUS: vemos que o container está rodando há mais de 13 segundos;**

# **PORTS: o container está com a porta 3306, o padrão do MySQL;**

# **NAMES: vemos que o nome que definimos, "database", realmente foi dado ao container.**

# **Precisamos fazer o mesmo para o WordPress, mas faremos de uma forma direta, sem usar o pull. Simplesmente se a imagem não existir no sistema, o Docker fará automaticamente o download:**

# **docker run --name blog-alura --link database:mysql -e WORDPRESS\_DB\_PASSWORD=teste123 -p 80:80 -d wordpress**

# **A opção --link serve para estabelecer uma conexão do nosso container, o qual vai criar o blog no WordPress, com o banco de dados. Passamos o nome do banco de dados (database):o nome fantasia (mysql)**

# **Da mesma forma que passamos uma senha para o container do MySQL, pasaremos a mesma aqui para -e WORDPRESS\_DB\_PASSWORD**

# **Queremos que o container rode na porta 80 do servidor, então fazemos -p 80:80. A primeira é a porta da máquina local e a segunda a do container. Dentro do container do WordPress vai existir um Web Server do Apache.**

# **Ao executar o comando, o Docker faz o download da imagem primeiro para depois criar de fato o container. Podemos verificar isso:**

# 

# **E aqui estão os dois containers que criamos em execução.**

# **O blog já foi criado e já está funcionando! Vamos verificar isso pelo navegador, fugindo um pouco do Docker. Escrevemos o *ip* da máquina local e a opção de lingua do WordPress aparece para nós:**

# 

# **Selecionaremos "Português do Brasil" e vamos para a página de configuração da instalação.**

# **Título do site: *Alura Blog***

# **Nome de usuário: *infoslack***

# **Senha: (uma qualquer)**

# **Endereço de e-mail: *infoslack@gmail.com***

# **E o WordPress está instalado. Podemos fazer o login normalmente e começar a trabalhar com o Blog. Perceba como foi rápido e fácil fazer isso usando o Docker**

# **Vimos, então, nesta aula alguns comandos:**

# **docker images: lista toda as imagens, como seus nomes criados aleatoriamente ou não,**

# **docker run: cria os containers,**

# **docker pull: faz o download de imagens,**

# **docker ps: acompanha os containers em execução.**

# **Na próxima aula veremos com mais detalhes e com calma o que cada um faz. O intuito dessa aula era para vermos um uso prático e rápido para captarmos a ideia do Docker.**

# 