

1. Introdução

Criar um contêiner Docker Hello World e colocá-lo para funcionar verifica se você tem a infraestrutura Docker configurada corretamente em seu sistema de desenvolvimento. Faremos isso de 4 maneiras diferentes:

1. Execute a imagem hello world do Docker fornecida pelo Docker
2. Obtenha um "Hello, world" impresso de outra imagem básica do Docker
3. Escreva um programa simples "Hello, World" em Java e execute-o em um contêiner Docker
4. Execute um programa "Hello, World" em Java usando Gradle e Docker

2. Execute a imagem Docker Hello World fornecida pelo Docker

Esta é a maneira mais simples possível de verificar se você instalou o Docker corretamente. Basta executar a imagem do Docker [hello-world](#) em um terminal.

\$ docker run hello-world

Este comando fará o download da `hello-world` imagem Docker do Docker Hub, se ainda não estiver presente, e a executará. Como resultado, você deve ver a saída abaixo se tudo correr bem.

```
File Edit View Search Terminal Help
sh-4.2$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c04b14da8d14: Pull complete
Digest: sha256:8256e8a36e2070f7bf2d0b0763dbabdd67798512411de4cdc9f9431a1feb60fd9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker Hub account:
https://hub.docker.com

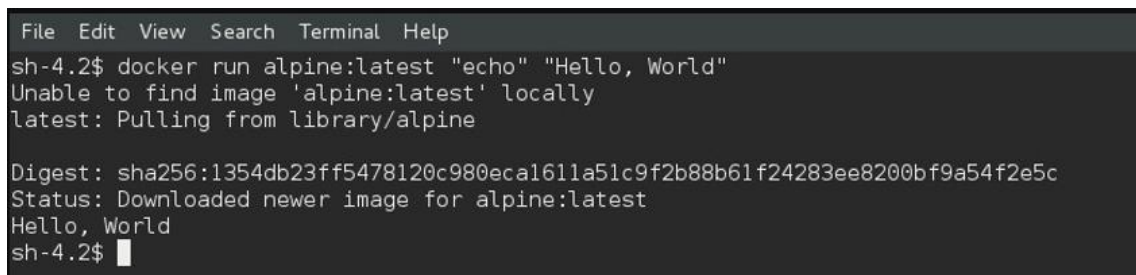
For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/
```

3. Obtenha um “Hello, world” impresso de outra imagem básica do Docker

Isso também é simples. O Docker fornece algumas [imagens base](#) que podem ser usadas diretamente para imprimir um “Hello, World”. Isso pode ser feito conforme mostrado abaixo e verifica se a instalação do Docker foi bem-sucedida.

```
$ docker run alpine:latest "echo" "Hello, World"
```

Este comando baixa a imagem base [Alpine](#) pela primeira vez e cria um contêiner Docker. Em seguida, ele executa o contêiner e executa o comando echo. O comando echo ecoa a string “Hello, World”. Como resultado, você deve ver a saída abaixo.

A terminal window with a dark background and light text. The menu bar at the top shows 'File Edit View Search Terminal Help'. The prompt is 'sh-4.2\$'. The command entered is 'docker run alpine:latest "echo" "Hello, World"'. The output shows the Docker process pulling the 'alpine:latest' image from the library, displaying its digest and status, and finally printing 'Hello, World' before returning to the prompt 'sh-4.2\$'.

```
File Edit View Search Terminal Help
sh-4.2$ docker run alpine:latest "echo" "Hello, World"
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine

Digest: sha256:1354db23ff5478120c980eca1611a51c9f2b88b61f24283ee8200bf9a54f2e5c
Status: Downloaded newer image for alpine:latest
Hello, World
sh-4.2$
```

4. Escreva um programa simples “Hello, World” em Java e execute-o em um contêiner Docker

Vamos subir um pouco agora. Vamos escrever um programa simples hello world em Java e executá-lo em um contêiner do Docker.

4.1. Crie HelloWorld.java

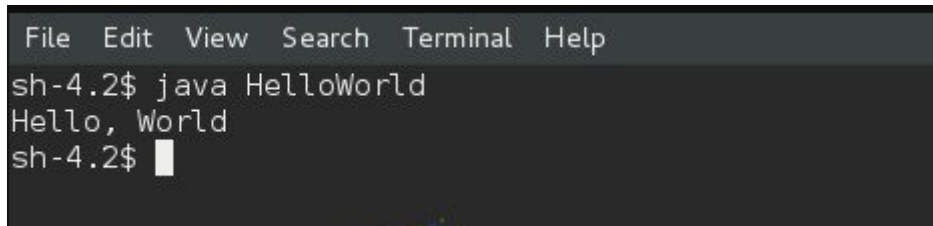
Em primeiro lugar, vamos criar um programa Java simples que imprima “Hello, World”. Abra seu editor de texto favorito e digite o seguinte código:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

Este é um programa HelloWorld padrão em Java simples. Salve o arquivo como `HelloWorld.java`. Agora, compile este arquivo usando o compilador Java.

```
$ javac HelloWorld.java
```

Isso deve criar o arquivo de classe. `HelloWorld.class`. Normalmente, usaríamos o comando `java` para executar `HelloWorld` como abaixo:

A terminal window with a dark background and light text. The menu bar at the top shows 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal content shows a shell prompt 'sh-4.2\$' followed by the command 'java HelloWorld'. The output is 'Hello, World' on the next line. The prompt 'sh-4.2\$' is followed by a cursor on the third line.

```
File Edit View Search Terminal Help
sh-4.2$ java HelloWorld
Hello, World
sh-4.2$
```

Mas queremos executá-lo em um contêiner do Docker. Para fazer isso, precisamos criar uma imagem Docker. Vamos fazer isso agora.

4.2. Crie um Dockerfile

Para criar uma nova imagem Docker, precisamos criar um **Dockerfile**. Um Dockerfile define uma imagem do docker. Vamos criar isso agora. Crie um novo arquivo denominado `Dockerfile` e digite o seguinte nele e salve-o:

```
FROM alpine:latest
```

```
ADD HelloWorld.class HelloWorld.class
```

```
RUN apk --update add openjdk8-jre
```

```
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "HelloWorld"]
```

A sintaxe completa do Dockerfile pode ser encontrada nos documentos do Docker, mas aqui está um resumo do que fizemos. Estendemos nossa imagem da imagem base Alpine.

Em seguida, adicionamos `HelloWorld.class` na imagem com o mesmo nome. Posteriormente, instalamos um ambiente JRE usando OpenJDK . Finalmente, demos o comando para executar quando esta imagem for executada - isto é, para executar nosso `HelloWorld` na JVM.

4.3. Crie uma imagem do Docker Hello World e execute-a

Agora, crie uma imagem a partir deste Dockerfile executando o comando abaixo:

```
$ docker build --tag "docker-hello-world:latest" .
```

Como resultado disso, você deve ver a seguinte saída

```
File Edit View Search Terminal Help
sh-4.2$ docker build --tag "docker-hello-world:latest" .
Sending build context to Docker daemon 4.096 kB
Step 1 : FROM alpine:latest
--> baa5d63471ea
Step 2 : ADD HelloWorld.class HelloWorld.class
--> Using cache
--> 44a4cc613c43
Step 3 : RUN apk --update add openjdk8-jre
--> Using cache
--> 392eefc0aad3
Step 4 : ENTRYPOINT java -Djava.security.egd=file:/dev/./urandom HelloWorld
--> Running in 051e53de6d98
--> 7d363cc7ff0b
Removing intermediate container 051e53de6d98
Successfully built 7d363cc7ff0b
sh-4.2$
```

Por fim, execute a imagem Docker para ver o "Hello, World" impresso.

\$ docker run docker-hello-world:latest

Como resultado disso, você deve ver a saída abaixo:

```
File Edit View Search Terminal Help
sh-4.2$ docker run docker-hello-world:latest
Hello, World
sh-4.2$
```