Docker for dummies

Leandro Kollenberger

redbee studios

30 de Marzo de 2018

¿Qué es Docker?

Docker es una herramienta que nos facilita enpaquetar una aplicación junto con su *runtime* dentro de una **imagen**, que luego se ejecutará dentro de un **contenedor**.

Gracias a la gran cantidad de **imágenes base** que hay disponibles en el **Docker Hub** y la simple sintaxis del **Dockerfile**, crear imágenes propias con nuestra aplicación es muy simple.

Dockerfile

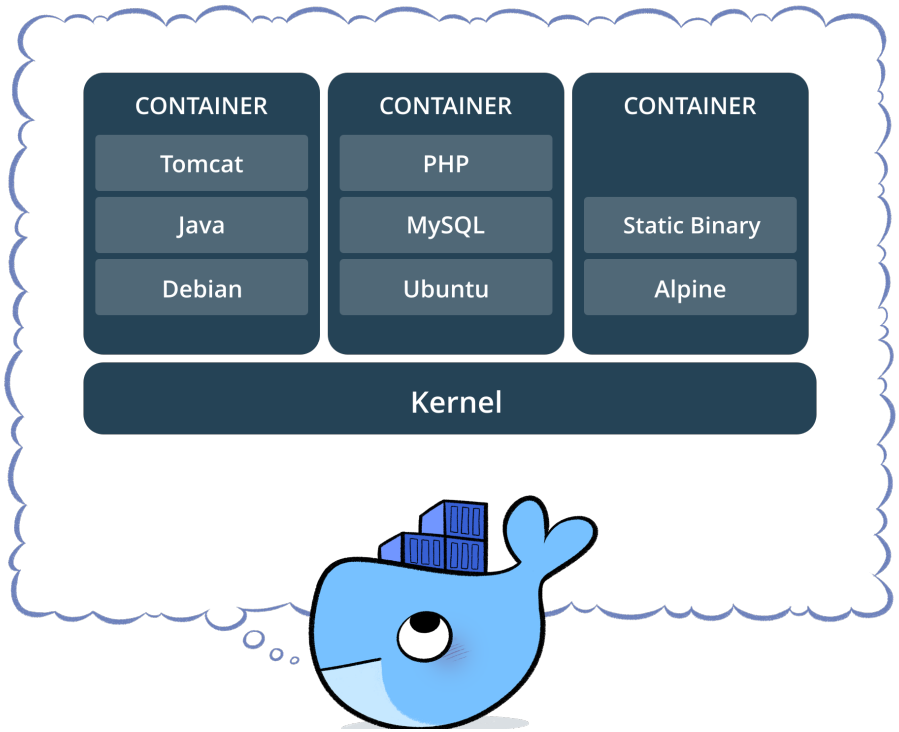
```
FROM debian:9
RUN echo "echo 'Hola mundo!'"\
                    > /hi.sh && \
                    chmod +x /hi.sh
ENTRYPOINT hi.sh
```

¿Qué es un container?

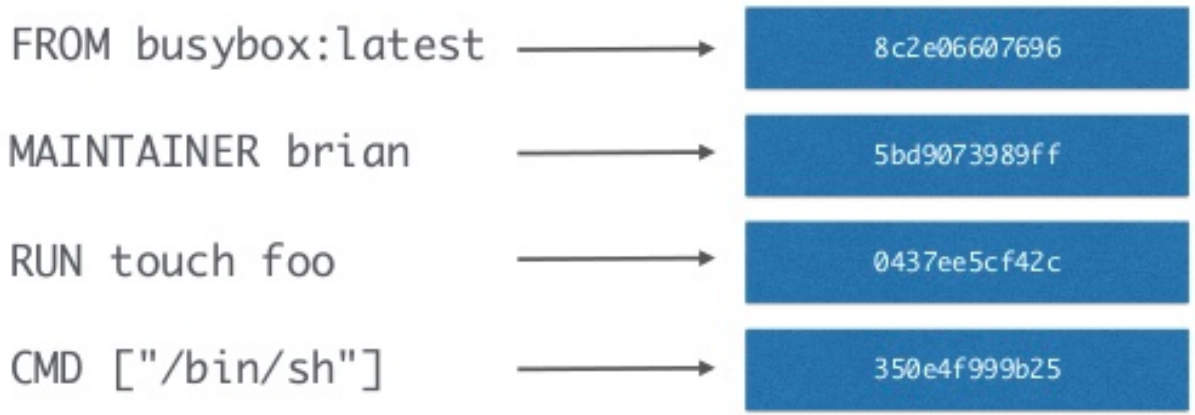
Un **container** es una funcionalidad del sistema operativo que nos permite ejecutar ciertas aplicaciones dentro de un **namespace** aparte.

Un **namespace** es una separación lógica de los recursos del sistema, como por ejemplo CPU, memoria RAM, puntos de montaje (disco), árbol de procesos, network, etc.

De esta forma un container nos provee seguridad y aislamiento de las aplicaciones que corren en él, similar a lo que nos permite una VM, sin el overhead de simular hardware y correr un sistema operativo entero, ya que todos los containers corriendo en un mismo host comparten su kernel.



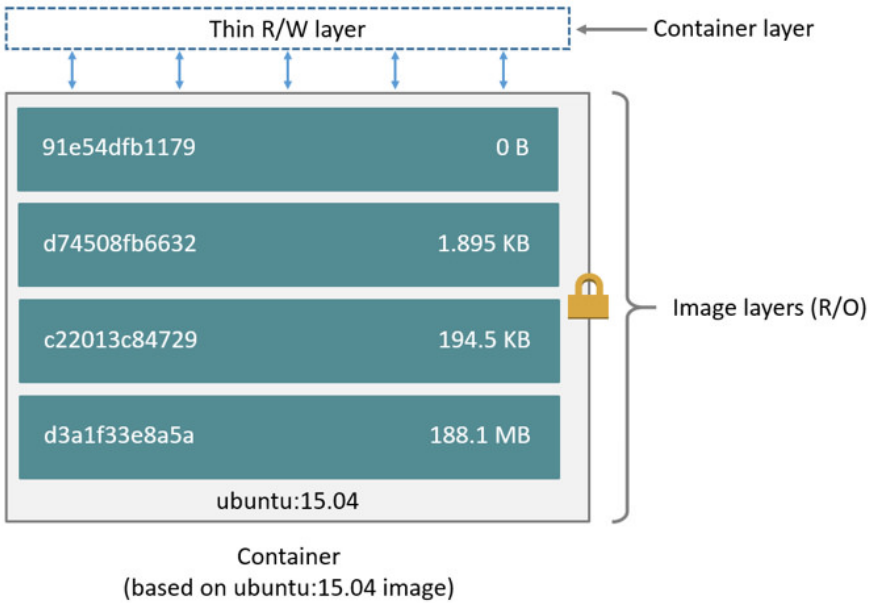
El concepto de "layer"



Las imágenes de Docker que podemos usar a diario están compuestas por una o más **layers**. Éstas imágenes contienen un sistema de archivos correspondiente al sistema operativo o la aplicación que queremos ejecutar. Esta imagen está identificada por un hash y opcionalmente uno o más **tags**, como por ejemplo *redis:4.0.6* o *debian:latest*.

Cada layer contiene únicamente los cambios (ya sean archivos, comandos o metadatos) con respecto a la layer anterior, se identifican con un hash y corresponden cada una a un comando del Dockerfile. Todas las layers que componen una imagen son **read only**.

El concepto de "layer"



Cada layer contiene únicamente los cambios (ya sean archivos, comandos o metadatos) con respecto a la layer anterior, se identifican con un hash y corresponden cada una a un comando del Dockerfile. Todas las layers que componen una imagen son **read only**.

Al momento de crear un nuevo container a base de una imagen, se crea una pequeña layer **read-write**, que contiene los cambios creados por el container en ejecución (por ejemplo, por la aplicación).

Esta última layer se **pierde al eliminar el container**.

Gracias!

Preguntas?

redbee®