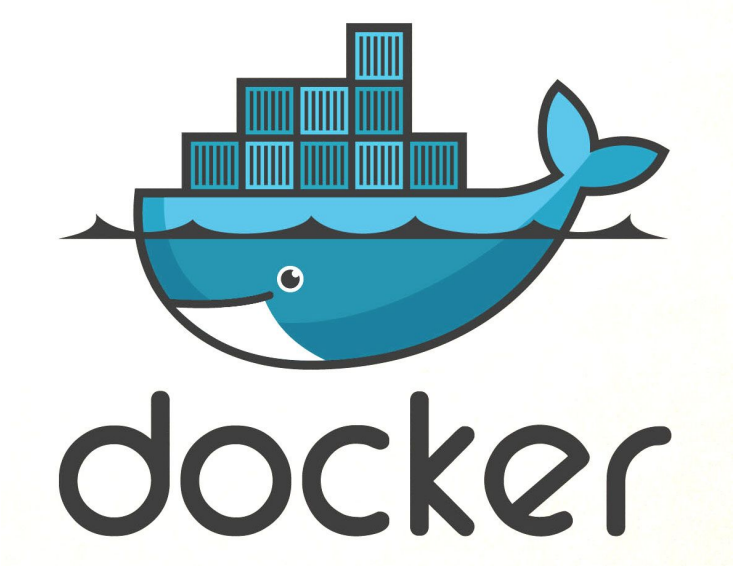
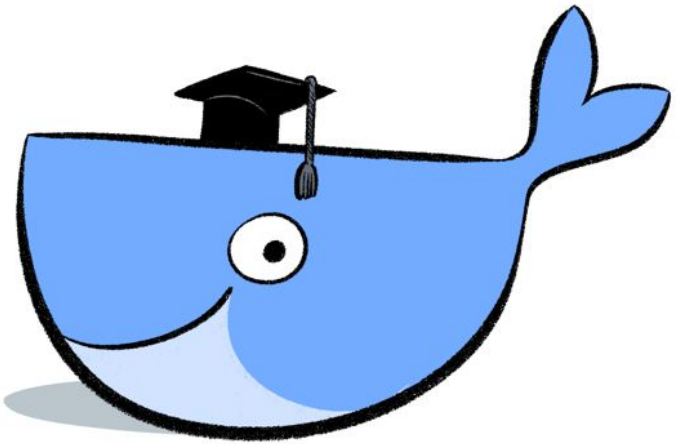
Introducción



Índice

- ¿Qué es Docker?
- ¿Qué es un contenedor?
- En resumen, ¿Qué es Docker?
- Y entonces, ¿Qué es un contenedor?
- ¿Para qué? Casos de uso.
  - Empaquetamiento de aplicaciones para su distribución.
  - Replicación de entornos de desarrollo. *It works on my machine!!*
  - Replicación de entorno de producción en desarrollo.
  - Despliegue de aplicaciones.
  - Gestión de aplicaciones como un todo. (Redundancia, tolerancia a fallos, etc.)

¿Qué es Docker?



¿Qué es Docker?

*“Docker es un proyecto de código abierto que **automatiza el despliegue de aplicaciones dentro de contenedores** de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.”*

-- Wikipedia (es) --

¿Qué es Docker?

*“Docker es un proyecto de código abierto para **automatizar la implementación de aplicaciones como contenedores portátiles y autosuficientes que se pueden ejecutar en la nube o localmente**. Docker es también una empresa que promueve e impulsa esta tecnología, en colaboración con proveedores de la nube, Linux y Windows, incluido Microsoft.”*

-- Microsoft --

¿Qué es Docker?

“Docker Engine is an open source containerization technology ***for building and containerizing your applications****. Docker Engine acts as a client-server application with:*

- *A server with a long-running daemon process* ***dockerd****.*
- ***APIs*** *which specify interfaces that programs can use to talk to and instruct the Docker daemon.*
- *A command line interface (****CLI****) client docker.“*

-- Docker --

¿Qué es un contenedor?

¿Qué es un contenedor?

*“La **virtualización a nivel de sistema operativo**, también llamada virtualización basada en **contenedores**, (...) es un método de virtualización en el que, sobre el núcleo del sistema operativo se ejecuta una capa de virtualización que permite que existan **múltiples instancias aisladas** de espacios de usuario, en lugar de solo uno…”*

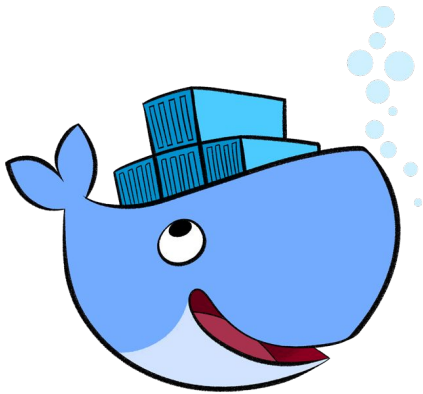
-- Wikipedia --

¿Qué es un contenedor?

*"...Tales instancias, las cuales son llamadas **contenedores**, (...), pueden verse (...) como un servidor real (...). Además de mecanismos de **aislamiento**, el kernel a menudo proporciona mecanismos de **administración de recursos** para limitar el impacto de las actividades de un contenedor sobre otros contenedores."*

-- Wikipedia --

Resumen



En resumen:

¿Qué es Docker?

- Es un software de virtualización ligera que permite crear y gestionar contenedores.
- Asigna y reparte el hardware del sistema entre los contenedores.

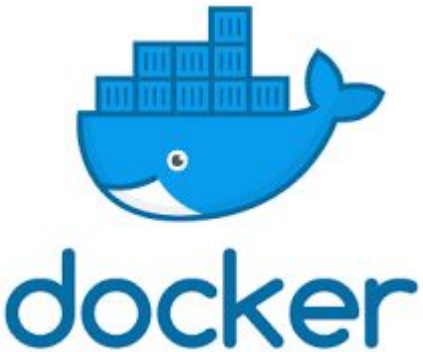
Docker no es una virtualización completa, sino que los contenedores usan el sistema subyacente. Por ejemplo, como Docker funciona sobre el SO Linux, en los contenedores solo puede instalarse Linux.

En resumen:

¿Qué es un contenedor?

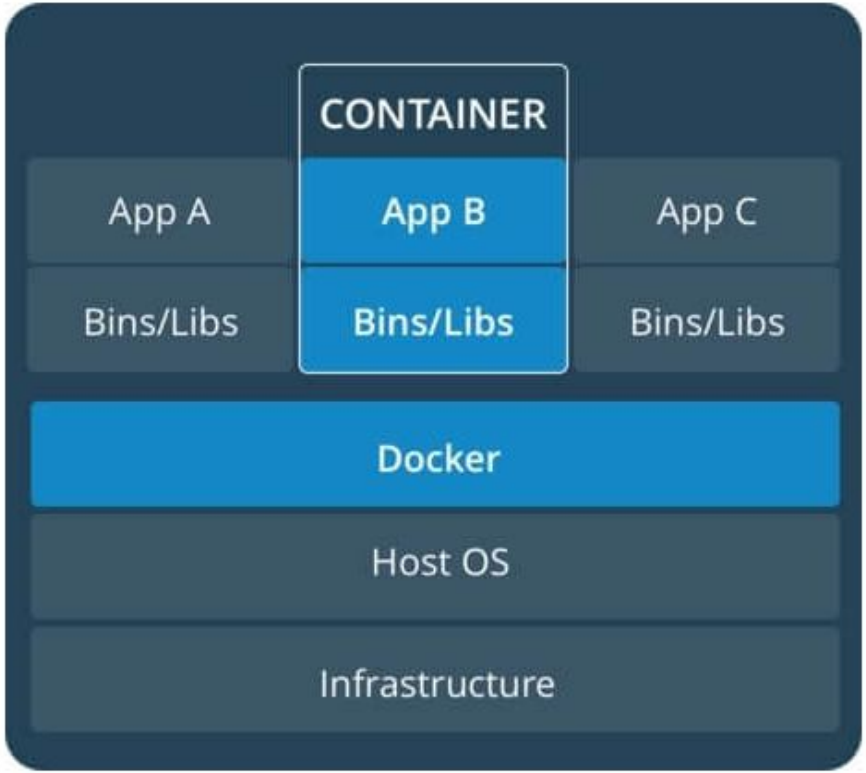
- Es una máquina virtual ligera.
- Están aislados del resto del sistema en el que se ejecutan.

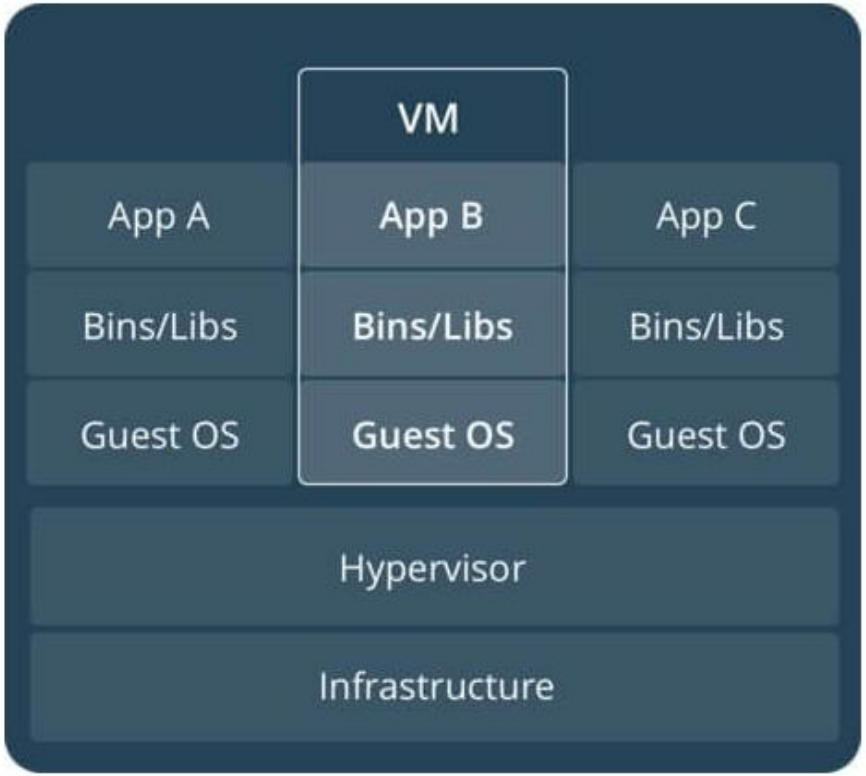
Docker vs Hipervisor



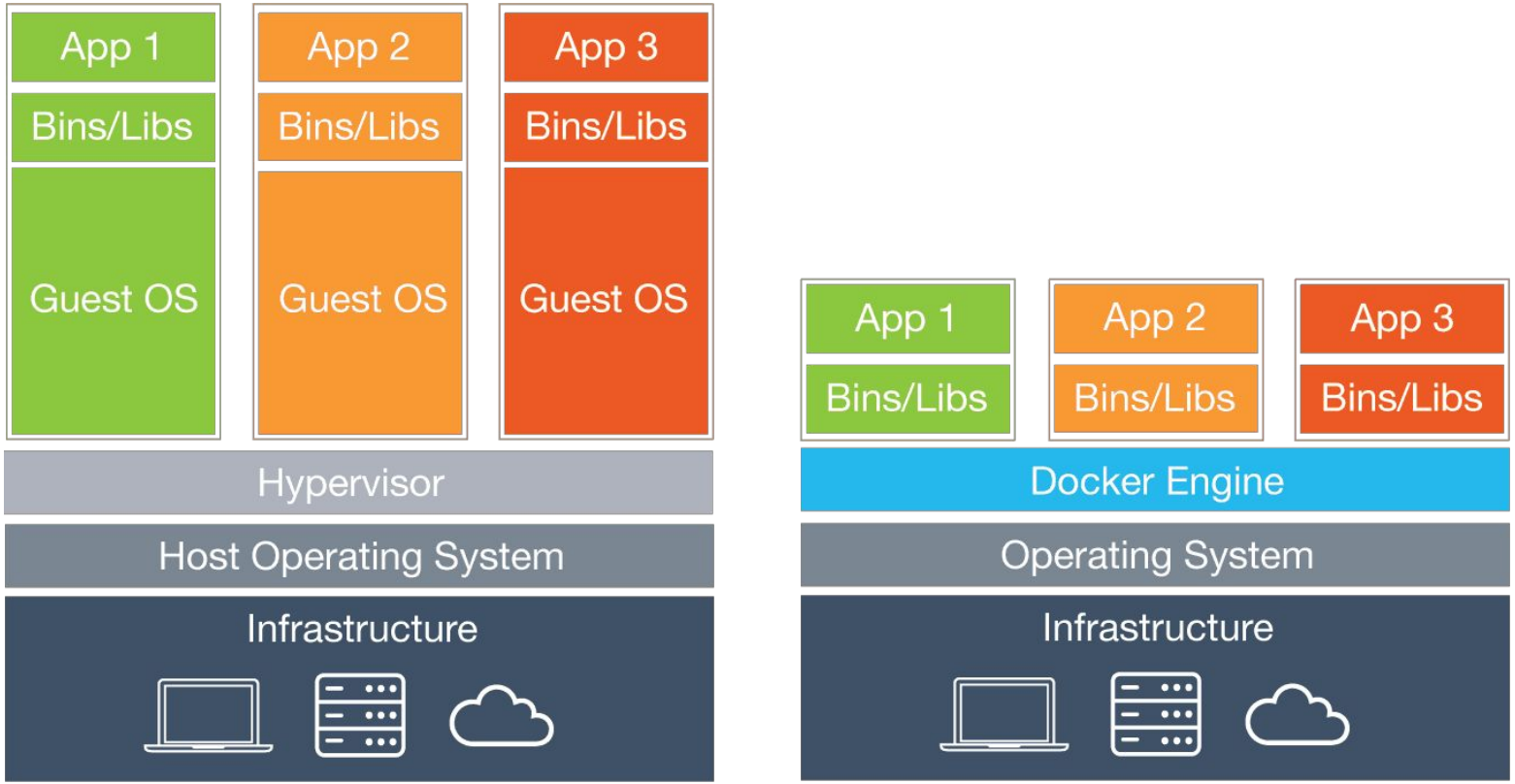
ORACLE
VM
VirtualBox

Docker vs Hipervisor





Docker vs Hipervisor



Ventajas de los contenedores frente a las máquinas virtuales

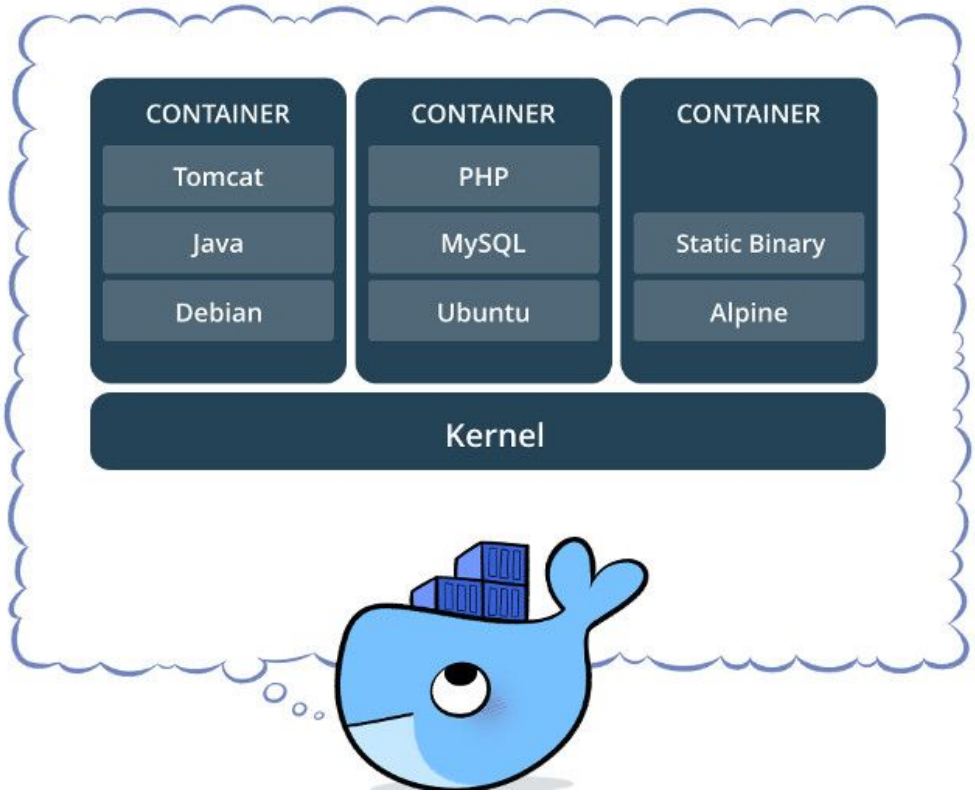
- Ocupan menos espacio.
- Su ejecución es más rápida.
- Son un estándar ampliamente soportado por la industria.

¿Para qué sirve Docker?

Casos de uso

¿Para qué sirve Docker?

Empaquetar aplicaciones



Casos de uso

Vamos a ver 5 ejemplos de casos de uso típicos de Docker:

1. Empaquetamiento de aplicaciones para su distribución.
2. Replicación de entornos de desarrollo.
3. Replicación de entorno de producción en desarrollo.
4. Despliegue de aplicaciones.
5. Gestión de aplicaciones como un todo. (Redundancia, tolerancia a fallos, escalado horizontal, etc.)

Caso 1

Empaquetamiento de aplicaciones para su distribución

1. Empaquetamiento de aplicaciones para su distribución

Las aplicaciones actuales son **complejas**.

Supongamos que un desarrollador hace **una aplicación web** que quiere distribuir para que sus clientes las puedan utilizar.

La aplicación consta de las siguientes partes/**servicios**:

- ❏ Servidor web → Nginx
- ❏ Backend → Python + Django
- ❏ Base de datos → MySQL
- ❏ Caché → Redis
- ❏ Cola de tareas asíncrona → Celery
- ❏ Broker de mensajes → RabbitMQ
- ❏ Frontend → HTML + CSS + Javascript

1. Empaquetamiento de aplicaciones para su distribución

Si un **cliente** quiere utilizar la aplicación, tendrá que:

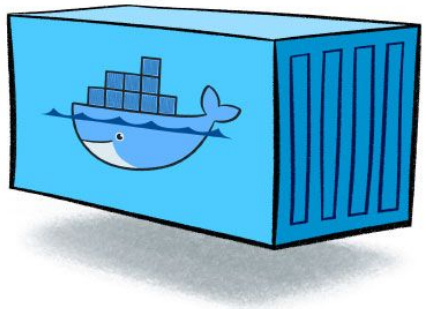
- **Instalar** cada una de las partes por separado, **respetando las versiones** que indica el desarrollador.
- **Configurar** cada una de las partes para que trabajen juntas.

Distribuir esta aplicación es, por tanto, **complicado y requiere unos conocimientos** que el cliente no tiene porqué tener.

1. Empaquetamiento de aplicaciones para su distribución

Utilizando **docker**, el **desarrollador** puede **empaquetar todos los servicios en un contenedor**, de modo que **el cliente solo necesita**:

- Tener **docker instalado**, lo que es muy sencillo, o usar un proveedor en la nube que lo lleve instalado por defecto (todos lo llevan).
- Usar un solo comando para **ejecutar el contenedor**.



Caso 2

Replicación de entornos de desarrollo

2. Replicación de entornos de desarrollo

En la actualidad una aplicación rara vez es desarrollada por 1 único desarrollador, sino por un **grupo**.

Tomando la aplicación web anterior como ejemplo, **cada desarrollador necesitaría instalarse todos los servicios en su máquina local** para poder programar.

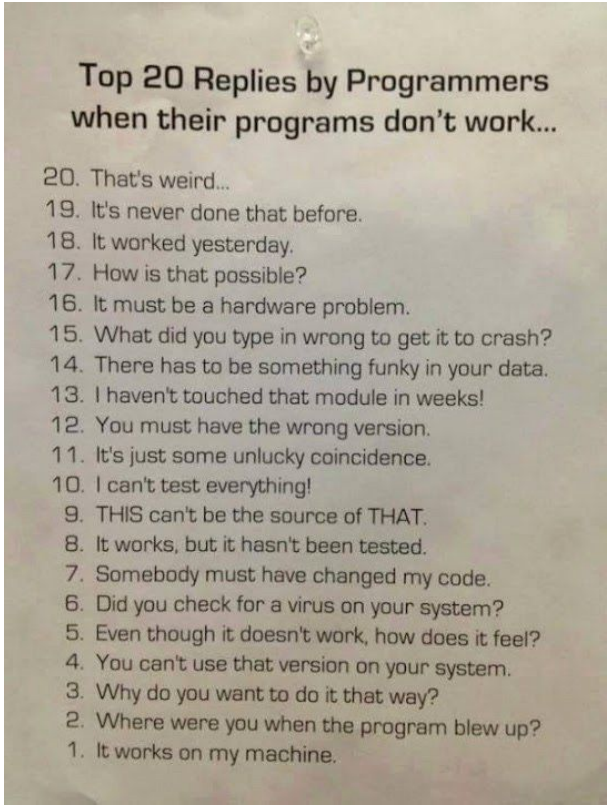
2. Replicación de entornos de desarrollo

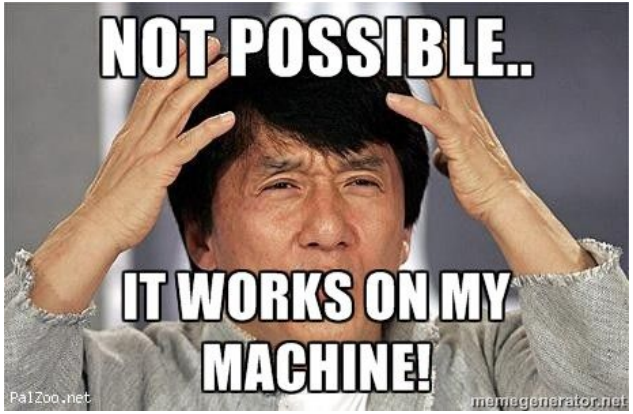
A la complejidad de la instalación, se une aquí la **dificultad de mantener los servicios y el código fuente actualizados** a las mismas versiones, dando lugar a un problema llamado:

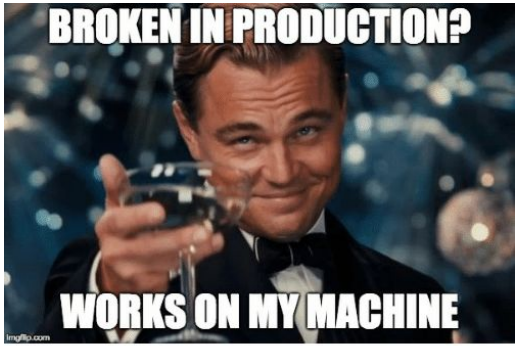
“It works on my machine”

que consiste en que algo que funciona en el ordenador de un desarrollador, no le funciona a otro cuando lo instala y configura, siendo muy complicado detectar donde está el fallo.

2. Replicación de entornos de desarrollo

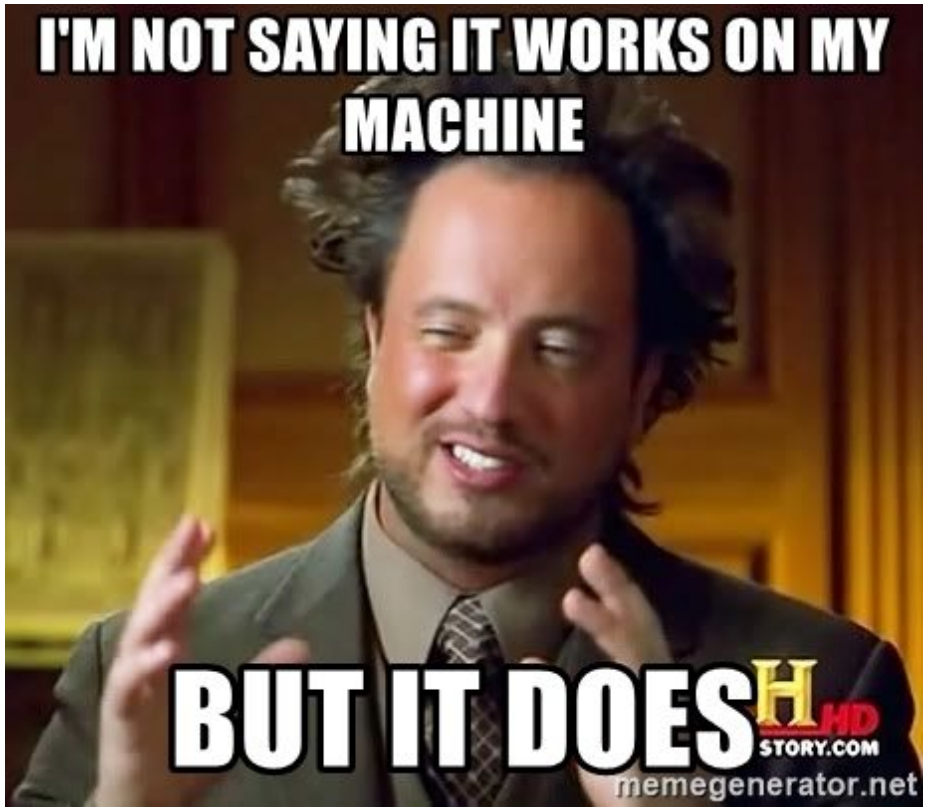






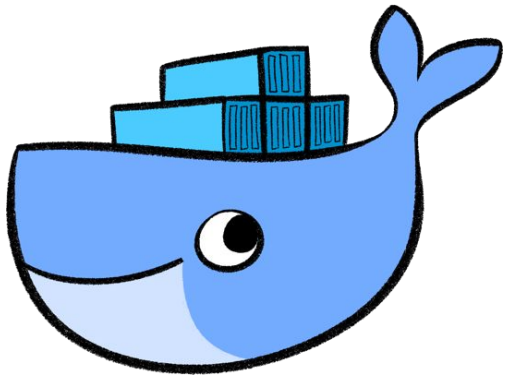
Problem? What Problem?

2. Replicación de entornos de desarrollo



2. Replicación de entornos de desarrollo

Si la aplicación está total o parcialmente **empaquetada** en contenedores, la instalación y configuración se hace **solo 1 vez**, al crear el contenedor (o después al modificarlo), de forma que si al ejecutar el contenedor funciona, lo va a hacer de igual forma **en todos los equipos**.



Caso 3

Replicación de entorno de producción en desarrollo

3. Replicación de entorno de producción en desarrollo

Otro problema con el que se encuentran comúnmente los desarrolladores es que el entorno de **producción**; es decir, donde se ejecuta la aplicación **no es igual que el entorno de desarrollo**, que es donde se programa.

Esto normalmente se debe a la dificultad o a la imposibilidad de instalar o configurar los servicios que la aplicación usa en producción, en un ordenador personal.

3. Replicación de entorno de producción en desarrollo

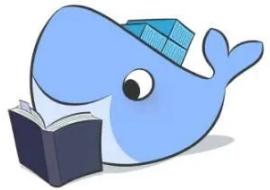
Como consecuencia, algo que funcionaba en desarrollo, puede que no funcione en producción.

Lo que se busca siempre es **que el entorno de desarrollo se asemeje lo máximo posible al de producción**, pero éste último no lo configuran programadores, sino administradores de sistemas o **devops**.

3. Replicación de entorno de producción en desarrollo

Docker proporciona un medio de simplificar el problema: se pueden **crear contenedores** con una configuración similar a la de producción **y dárselos a los programadores** para que los utilicen.

Si los programadores saben, podrían hacerlos ellos, pero en el caso peor; es decir, en el que no supieran, los propios **devops y administradores que instalan y configuran el entorno** de producción podrían crear los contenedores para que los desarrolladores los usen como una **caja negra**.



Caso 4

Despliegue de aplicaciones

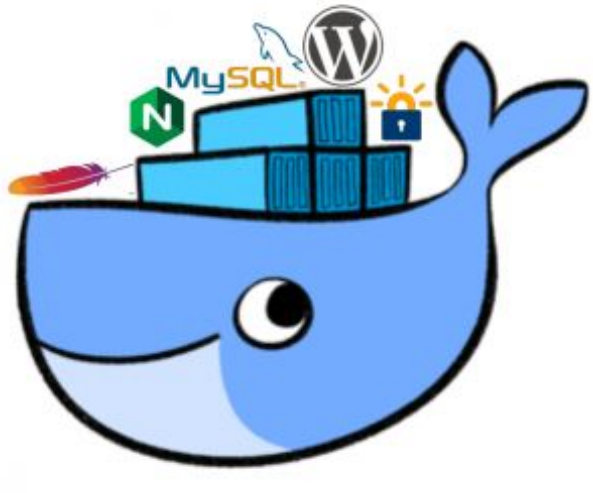
4. Despliegue de aplicaciones

De forma análoga a los casos anteriores, cuando la aplicación se quiere desplegar, debe hacerse la **instalación y configuración** de todos los servicios **en cada uno de los equipos** en los que se quiera hacer el despliegue.

En aplicaciones web, por ejemplo, es muy común que una aplicación se despliegue en varios servidores por temas de redundancia y para poder dar respuesta a más usuarios.

4. Despliegue de aplicaciones

El proceso de **despliegue es laborioso**, y si se crea **un contenedor, solo debe hacerse 1 vez**, ya que el contenedor se puede copiar en cada máquina y ejecutar como si fuera una única aplicación.



Caso 5

Gestión de aplicaciones como un todo (Redundancia, tolerancia a fallos, escalado horizontal, etc.)

5. Gestión de aplicaciones como un todo

Finalmente, y sobretodo en sistemas **en la nube**, el tener las aplicaciones en contenedores permite instaurar políticas de **redundancia, tolerancia a fallos escalado horizontal o ejecución bajo demanda**; ya que los contenedores pueden replicarse, encenderse y apagarse fácilmente y de forma **automática**.

5. Gestión de aplicaciones como un todo

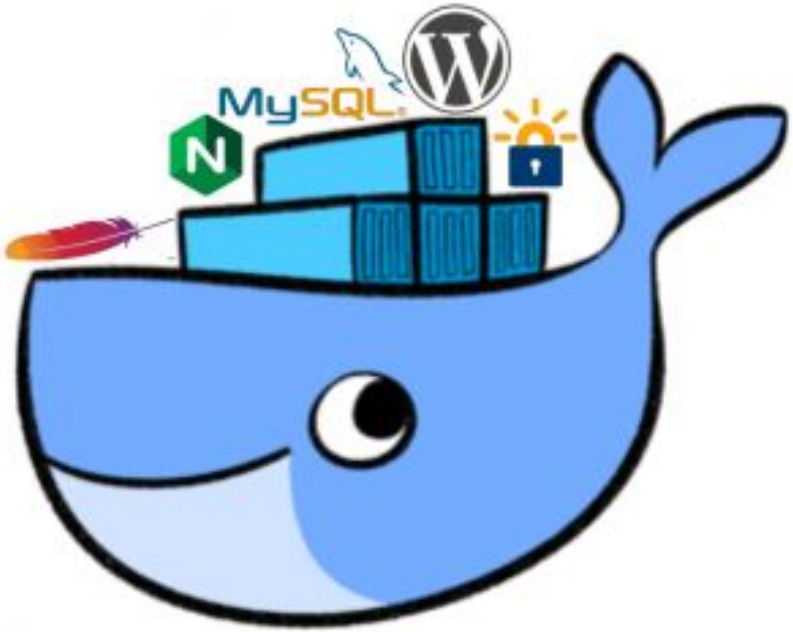
Por ejemplo, se puede tener un contenedor con la aplicación web anterior funcionando, y **si hay más usuarios** de los que puede soportar, **“levantar” automáticamente otro contenedor** para que den servicio los dos. Y posteriormente, cuando bajara la demanda, el segundo contenedor se podría **apagar**, también automáticamente.

Otro ejemplo sería que un contenedor **fallara y dejara de funcionar**, y entonces automáticamente se levantara otro contenedor.

5. Gestión de aplicaciones como un todo

Si **el despliegue de los contenedores se puede automatizar**, eso significa que existe un **ecosistema** que los contiene, y unos **parámetros de configuración** para esta automatización, o alguien que programe esa automatización.

Si esta no es una tarea puramente de programación, pero tampoco de configuración. ¿**Quién** se encarga de realizarla? Los **DevOps**. Normalmente se dice que los devops **programan la infraestructura** de las aplicaciones.


MySQL