1.Introducción. Docker para desarrolladores



# 1.Introducción. Docker para desarrolladores

# BIENVENIDOS AL CURSO.

Desde hace unos años la **adopción** de la tecnología de contenedores **Docker** por parte de los equipos de desarrollo de software ha sufrido un **incremento exponencial.**

Su uso **aumenta la productividad** de los desarrolladores y les permite **empaquetar y distribuir sus aplicaciones** y todas sus dependencias en contenedores que pueden ser desplegados con gran facilidad en todo tipo de sistemas operativos, servidores, nubes públicas o privadas etc.

Durante este curso nos introduciremos en los **fundamentos de los contenedores Docker** y buscaremos cómo **usarlos en nuestra práctica docente.**

Ejecutaremos contenedores, los configuraremos , crearemos nuestros propios contenedores y para finalizar realizaremos un flujo de despliegue continuo usando contenedores.

# Convenciones utilizadas. Uso del terminal

A lo largo del curso habrá muchas ocasiones en que los materiales muestren instrucciones para ser ejecutadas en el terminal. En esos casos:

- Se presentará el terminal con fondo negro
- Los comentarios a las instrucciones se mostrarán con letra roja y empezarán por el carácter #. Éstos no deberán copiarse en el terminal para ser ejecutados, ya que nos dará un error.
- Los comandos se mostrarán con letra verde, empezarán tras el carácter > y nos indicarán las órdenes que si debemos copiar para ser ejecutadas.

Un ejemplo:

```
# Esto es un comentario sobre el comando que se va a ejecutar a continuación. NO DEBE SER COPIADO

> echo "Esto si que es un comando válido. Debe ser copiado"
```

# 1.1 ¿Qué es Docker?

dotCloud Inc.. *Logotipo de Docker* (Dominio público)

Si tuviéramos que definir Docker de una manera rápida y poco formal diríamos lo siguiente:

***Docker es una tecnología de virtualización "ligera" cuyo elemento básico es la utilización de contenedores en vez de máquinas virtuales y cuyo objetivo principal es el despliegue de aplicaciones encapsuladas en dichos contenedores.***

Dicho de esta manera puede parecer que no es máss que otra tecnología de virtualización, pero para entender mejor cómo ha surgido esta tecnología y comprender las ventajas que aporta debemos echar un poco la vista atrás y conocer la evolución en el despliegue de aplicaciones.

En esa **evolución** nos podemos encontrar, de manera general y simplificada, con tres grandes pasos:

- **Arquitectura de un único servidor**
- **Virtualización**
- **Contenedores**

A continuación describiremos estos tres pasos haciendo especial hincapié en sus ventajas e inconvenientes.

# 1. Un único servidor

Inicialmente, allá por la prehistoria de la informática las aplicaciones se desplegaban en un único servidor siguiendo un esquema similar al que podemos ver en la imagen:

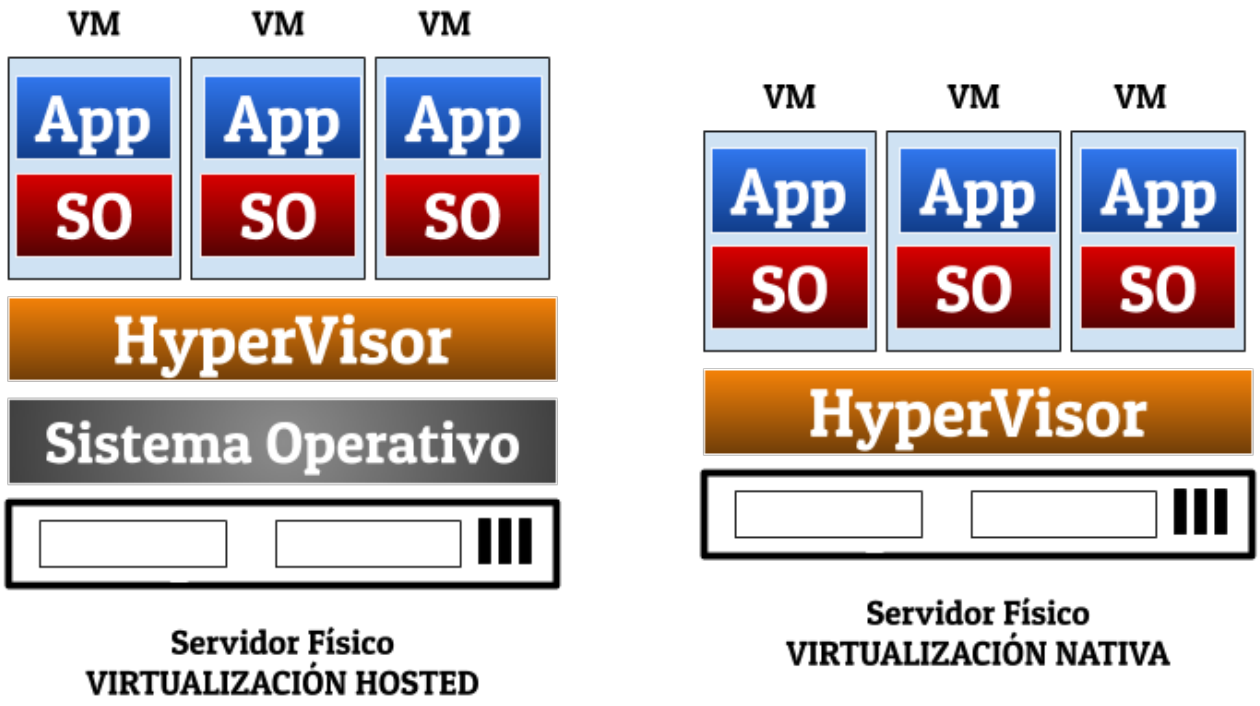

Juan Diego Pérez Jiménez. *Arquitectura de Servidor Físico* (Dominio público)

Este enfoque tuvo su momento pero tenía varias **LIMITACIONES:**

- Era un enfoque de **costes elevados** porque era necesaria un máquina de precio elevado.
- El despliegue de aplicaciones era un **proceso lento** que podía suponer en algunos casos una parada del servicio.
- El **escalado de las aplicaciones era costoso y complicado**. Si nuestra máquina llegaba un momento que se quedaba corta no había más remedio que sustituirla por otra más potente.
- La **migración a otro sistema era un proceso complicado**. La configuración del nuevo servidor, de su sistema operativo y de todas las dependencias tenía que ser compatible. Esto era algo complicado de gestionar y en algunos casos difícil de conseguir.
- En muchos momentos, aquellos en los que el servidor no se utilizaba aprovechando su potencia, se estaban **desperdiciando recursos.**
- Había mucha **dependencia del fabricante** del servidor.

# 2. Virtualización

Con el tiempo y para superar las limitaciones del modelo de un único servidor la tecnología evolucionó hacia servidores con características de virtualización. De una manera simplificada podríamos decir que para desplegar aplicaciones nos encontrábamos con arquitecturas similares a las siguientes:




Juan Diego Pérez Jiménez. *Arquitecturas basadas en virtualización.* (Dominio público)

Estos enfoques tenían una serie de **BENEFICIOS** que derivaban principalmente de superar las limitaciones del modelo de servidor único. A saber:

- Hay un **mejor aprovechamiento de los recursos**. Un servidor grande y potente se puede compartir entre distintas aplicaciones.
- Los **procesos de migración y escalado no son tan dolorosos**, simplemente le doy más recursos a la máquina virtual dentro de mi servidor o bien muevo la máquina virtual a un

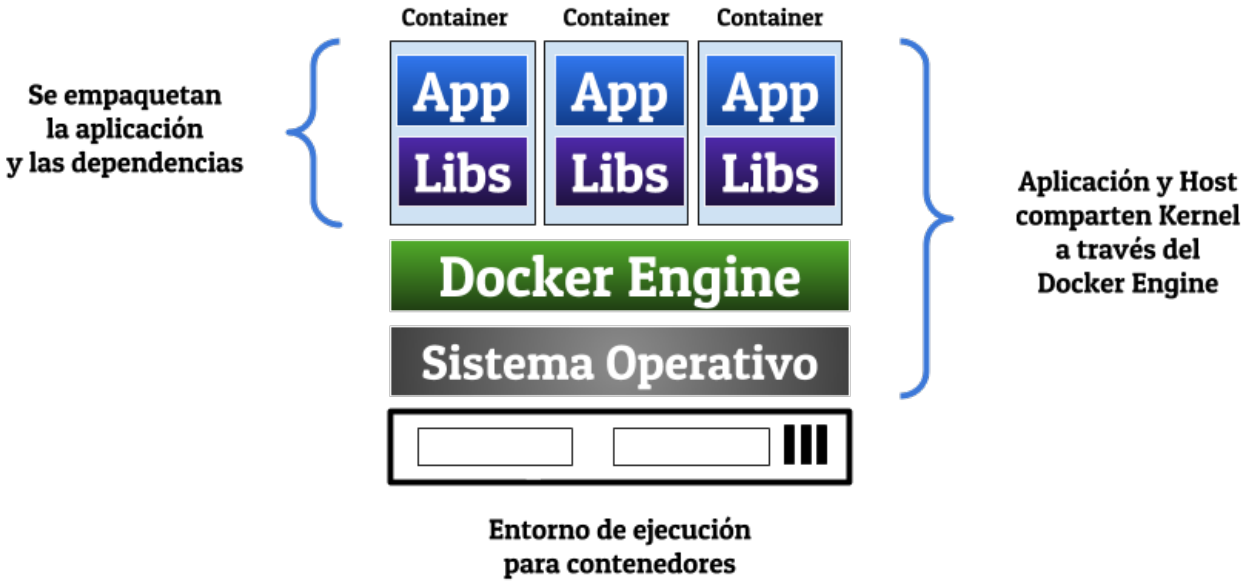
nuevo servidor, propio o en la nube, más potente y que también tenga características de virtualización.
- Esto además hizo que aparecieran **nuevos modelos de negocio en la nube** que nos permiten en cada momento tener y dimensionar las máquinas virtuales según nuestras necesidades y pagar únicamente por esas necesidades.

No obstante este enfoque también tiene algunos **INCONVENIENTES**:

- Todas las máquinas virtuales siguen teniendo su propia memoria RAM, su almacenamiento y su CPU que será aprovechada al máximo...o no.
- Para arrancar las máquinas virtuales tenemos que arrancar su sistema operativo al completo.
- La portabilidad no está garantizada al 100%.

# 3. Contenedores

El siguiente paso en la evolución, fue la aparición de los **CONTENEDORES**, eso que anteriormente hemos llamado **"máquinas virtuales ligeras"** . Su arquitectura general se puede ver en la siguiente imagen:




Juan Diego Pérez Jiménez. *Entorno de ejecución basado en contenedores* (Dominio público)

Y sus principales características son las siguientes:

- Los contenedores utilizan el **mismo Kernel Linux** que la máquina física en la que se ejecutan gracias a la estandarización de los los Kernel y a características como los Cgroups y los Namespaces. Esto **elimina la sobrecarga** que en las máquinas virtuales suponía la carga total del **sistema operativo invitado.**
- Me permiten **aislar las distintas aplicaciones** que tenga en los distintos contenedores (salvo que yo estime que deban comunicarse).
- **Facilitan la distribución de las aplicaciones** ya que éstas se empaquetan junto con sus dependencias y pueden ser ejecutadas posteriormente en cualquier sistema en el que se pueda lanzar el contenedor en cuestión.

- Se puede pensar que se añade una capa adicional el **Docker Engine**, pero esta capa **apenas añade sobrecarga** debido a que se hace uso del mismo Kernel.

Este enfoque por lo tanto aporta los siguientes **BENEFICIOS:**

- Una **mayor velocidad de arranque,** ya que prescindimos de la carga de un sistema operativo invitado. Estamos hablando de apenas segundos para arrancar un contenedor (a veces menos).
- Un **gran portabilidad,** ya que los contenedores empaquetan tanto las aplicaciones como sus dependencias de tal manera que pueden moverse a cualquier sistema en el que tengamos instalados el Docker Engine, y este se puede ser instalado en casi todos, por no decir todos.
- Una **mayor eficiencia** ya que hay un mejor aprovechamiento de los recursos. Ya no tenemos que reservar recursos, como hacemos con las máquinas virtuales, sin saber si serán aprovechados al máximo o no.

Aunque como todo en esta vida, la tecnología de contenedores tiene algún **INCONVENIENTE:**

- Los contenedores son **más frágiles que las máquinas virtuales** y en ocasiones se quedan en un estado desde el que no podemos recuperarlos. No es algo frecuente pero ocurre y para eso hay soluciones como la orquestación de contenedores que es algo que queda fuera del alcance de este curso que está más orientado a desarrolladores que a sistemas.

# GLOSARIO

Una vez hemos hecho una pequeña introducción al mundo de los contenedores, y para acabar el apartado vamos a proceder a definir una serie de términos relacionados que usaremos con frecuencia a lo largo del curso:

Imagen

De una manera simple y muy poco técnica una imagen es una plantilla (ya sea de una aplicación de un sistema) que podremos utilizar como base para la ejecución posterior de nuestras aplicaciones (contenedores). Si queremos una descripción más técnica y detallada diremos que es un archivos comprimido en el que, partiendo de un sistema base, se han ido añadiendo capas cada uno de las cuáles contiene elementos necesarios para poder ejecutar una aplicación o sistema. No tiene estado y no cambia salvo que generemos una nueva versión o una imagen derivada de la misma.

» Contenedor

Es una imagen que junto a unas instrucciones y variables de entorno determinadas se ejecuta. Tiene estado y podemos modificarlo. Estos cambios no afectan a la imagen o "plantilla" que ha servido de base.

» Repositorio

Almacén, normalmente en la nube desde el cuál podemos descargar distintas versiones de una misma imagen para poder empezar a construir nuestras aplicaciones basadas en contenedores.

» Docker

Plataforma, mayormente opensource, para el desarrollo, empaquetado y distribución de aplicaciones de la empresa Docker Inc (anteriomente Dot Cloud Inc). Es un término que se suele utilizar indistintamente al del Docker Engine.

» Docker Engine

Aplicación cliente-servidor que consta de tres componentes: un servicio dockerd para la ejecución de los contenedores, un API para que otras aplicaciones puedan comunicarse con ese servicio y una aplicación de línea de comandos docker cli que para gestionar los distintos elementos (contenedores, imágenes, redes, volúmenes etc..)

» Docker Hub

Registro de repositorios de imágenes de la empresa Docker Inc. Accesible a través de la URL https://hub.docker.com/

# 1.2 Instalación de Docker

Ocultar

Docker es una herramienta que está disponible para los sistemas operativos más comunes:

- Windows.
- Linux.
- Mac.

Las instrucciones para su instalación también están muy bien documentadas en el siguiente enlace: https://docs.docker.com/get-docker/

Dentro de estas instalaciones podemos diferenciar dos tipos de instalaciones:

- Instalación de Docker Desktop. Esto es posible para sistemas Windows y Mac. Esta aplicación lleva incluido no sólo docker, también docker-compose, Notary, Kubernetes y otras herramientas relacionadas.
- Instalación únicamente de Docker. Para sistemas Linux.

Como no podemos ver la instalación en todos los sistemas vamos a hacer una pequeña demostración de cómo se realiza esta instalación en algunos de los sistema más usados.

# Instalación de Docker en Windows

https://youtu.be/ozp84CCh0Uc?list=PL-8CyWabyNa85xowmOeBMCspbrn6qNWgl

# Instalación de Docker en Linux

Linux tiene multitud de distribuciones y en este curso nos centraremos en las distribuciones basadas en Ubuntu/Debian y en las instalaciones basadas en CentOs/RedHat. Concretamente instalaremos Docker para:

- Ubuntu 20.04
- CentOs 8

En el resto de distribuciones el proceso será igual o muy muy parecido. En todo caso siempre podemos consultar la documentación oficial.

**INSTALACIÓN DE DOCKER EN UBUNTU 20.04**

https://youtu.be/PoRA7dAhhHA?list=PL-8CyWabyNa85xowmOeBMCspbrn6qNWgl

**INSTALACIÓN DE DOCKER EN CENTOS8**

https://youtu.be/NQfnWLOlonY?list=PL-8CyWabyNa85xowmOeBMCspbrn6qNWgl

Tras realizar la instalación hay una serie de operaciones que os recomiendo:

- **Ejecutar Docker** como un usuario que no sea root, es decir, ejecutarlo **sin sudo**. Esto es fundamental porque si no trabajar con Docker llega a ser un engorro.
- **Habilitar o deshabilitar el servicio Docker** al inicio, según nos interese. Por defecto el servicio se habilita al inicio y la sobrecarga sobre el sistema es mínima, así que recomiendo dejarlo así si lo vamos a usar habitualmente.

A continuación detallaremos cada una de estas opciones:

**DOCKER SIN SUDO**

Para conseguir esto tenemos que realizar las siguientes operaciones:

```
# Crear el grupo docker si no se ha creado durante la instalación

> sudo groupadd docker

# Añadir nuestro usuario al grupo creado en el apartado anterior

> sudo usermod -aG docker $user

# Salir de sesión o reiniciar (en algunas máquinas virtuales). Puedo también activar los cambios a los grupos con la siguiente orden

> newgrp docker
```

**HABILITAR/DESHABILITAR DOCKER AL INICIO**

```
# Si quiero habilitar el servicio docker al iniciar el sistema. (recomendado para desarrollo)

> sudo systemctl enable docker

# Si quiero que el servicio docker no esté habilitado.

> sudo systemctl disable docker
```

# Instalación en Mac

Desafortunadamente no dispongo ni de máquinas virtuales ni de equipos Mac para poder realizar una demostración de la instalación de Docker Desktop para Mac. No obstante comparto con vosotros el enlace de la documentación de Docker donde que perfectamente explicado.

Instalación de Docker Desktop en Mac.

# 1.3 Hola Mundo

# HOLA MUNDO EN DOCKER

***"HOLA MUNDO"*** es una convención establecida. Muchos cursos de programación acaban su introducción con un primer programa que nos muestra *"hola mundo"* al ser ejecutado, ya sea en pantalla, en un navegador web etc...

Si conseguimos que ese mensaje se muestre significa que **todo está correctamente instalado** y que **todo funciona** (al menos de momento ;) ) como debería.

**Docker** sigue esta convención y para obtenerlo deberemos ejecutar esta orden:

```
> docker run hello-world
```

Obtendremos en nuestra terminal una salida similar a la siguiente:

```
pekechis@pop-os:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:8c5aeeb6a5f3ba4883347d3747a7249f491766ca1caa47e5da5dfcf6b9b717c0
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.
    (amd64)
 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 ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Juan Diego Pérez Jiménez. *Docker Hello World* (Dominio público)

Pero, ¿qué es lo que está sucediendo al ejecutar esa orden?:

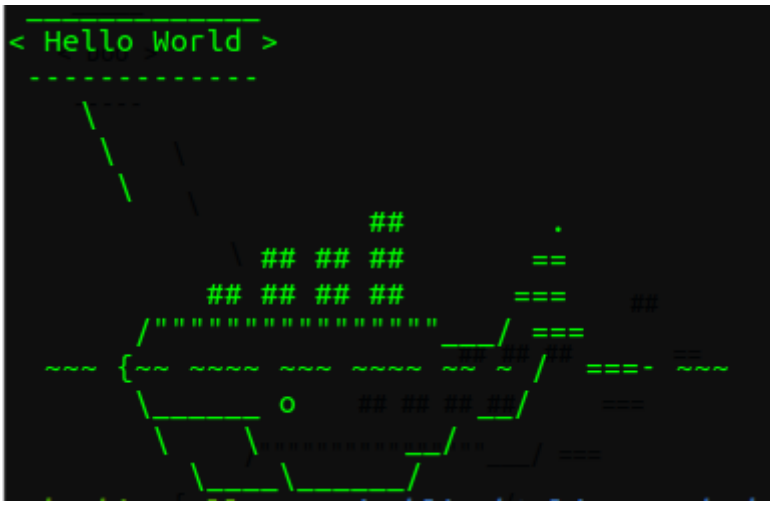
1. **Al ser la primera vez** que ejecuto un contenedor basado en esa imagen, **la imagen hello-word se descarga** desde el repositorio que se encuentra en el registro que vayamos a utilizar, en nuestro caso **DockerHub.**
2. Muestra el mensaje de bienvenida que es la consecuencia de crear y arrancar un contenedor basado en esa imagen.

Ese contenedor ya no podremos volver a arrancarlo ni podremos comunicarlos con él. Ya veremos posteriormente el porqué.

Además de este hello-world hay alguno usuarios de la comunidad que han elaborado otras imágenes cuyo objetivo es el mismo, comprobar que todo está correcto, pero que muestran una apariencia más visualmente atractiva:

- docker run docker/whalesay cowsay Hello World

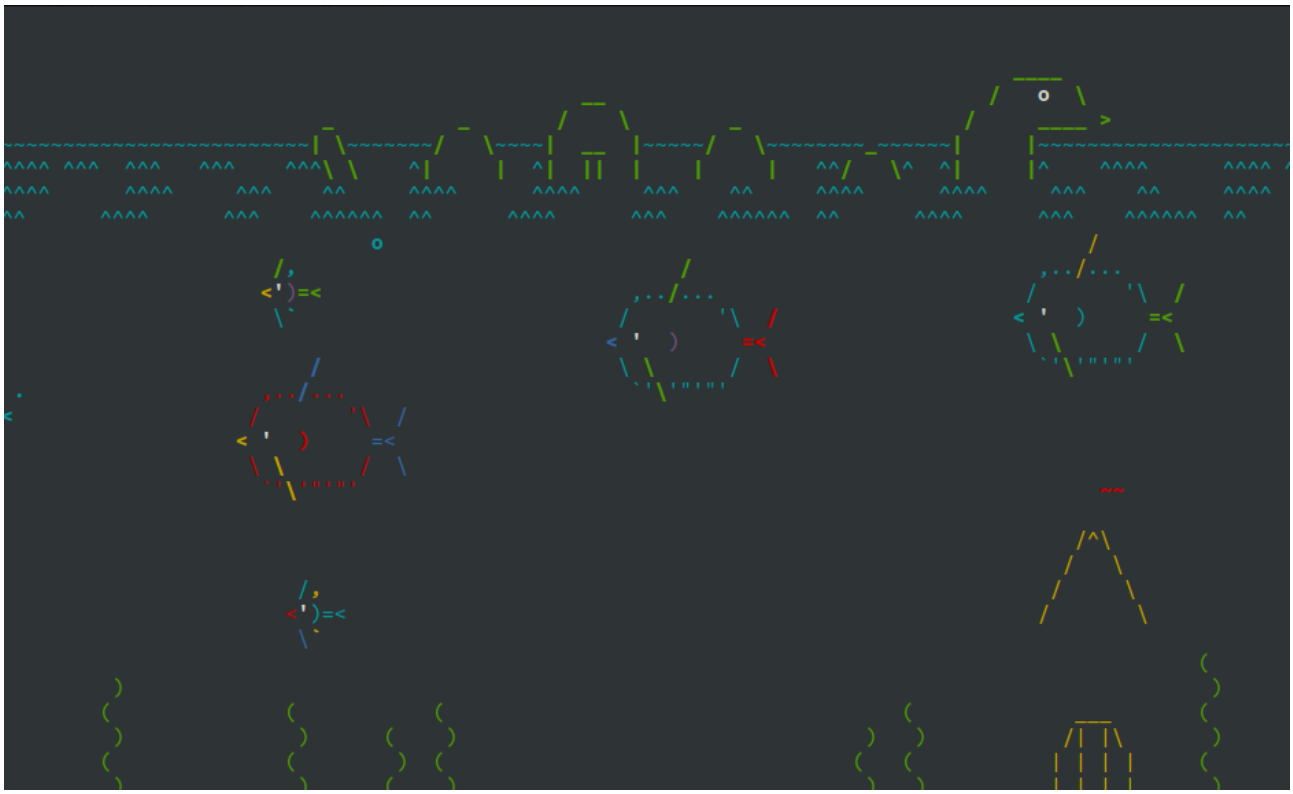
Obtendremos en nuestro terminal una salida similar a la siguiente:




Juan Diego Pérez Jiménez. *Docker WhaleSay* (Dominio público)

```
> docker run -it --rm
danielkraic/asciiquarium
```

Obtendremos en nuestro terminal una salida similar a la siguiente (Ctrl+C) para acabar.



Juan Diego Pérez. *Salida del contenedor danielkraic/asciiquarium* (Dominio público)

# 1.4 ¿Qué es Docker Hub?

Si recordáis, en el apartado anterior cuando hemos ejecutado **docker run hello-world** hemos dicho que pasaban dos cosas:

- Se **DESCARGABA** la imagen que es algo así como la "plantilla" para la creación de contenedores en ejecución.
- Se **EJECUTABA** el contenedor.

**¿PERO EXACTAMENTE, DESDE DÓNDE SE DESCARGA ESA IMAGEN?**

Las imágenes se descargan desde un **REGISTRO** de imágenes que es un "almacén en la nube" donde los usuarios pueden, entre otras cosas **crear, probar, almacenar y distribuir imágenes**. Por defecto cuando instalamos docker el registro que vamos a usar es **DockerHub** que además de todo lo anterior tiene muchas más funcionalidades.

Podríamos crear nuestro propio registro y utilizarlo pero **vamos a seguir utilizando DockerHub por varios motivos:**

- Tiene una **gran variedad de imágenes** disponibles para que usemos. La gran mayoría son públicas y gratuitas.
- Me permite **crear y distribuir imágenes de manera muy sencilla**. No olvidemos que es el repositorio por defecto para toda instalación de Docker.
- Me permite crear **organizaciones** para poder crear **equipos** y añadir posteriormente **miembros**, con sus respectivos permisos.
- Dispone de un **interfaz web** de fácil utilización.

Todas estas características son de especial interés para la docencia en los ciclos formativos de informática. Algunas de las aplicaciones que enseguida nos vienen a la mente sería las siguientes:

- Creación de una **organización para los profesores de cada IES.**
- **Distribución a los alumnos** de las imágenes creadas a medida por el profesorado. De tal manera que el entorno es el mismo tanto para profesor como para el alumno. Esto nos ahorra muchos problemas posteriormente.
- **Presentación de aplicaciones por parte de los alumnos**. Si los alumnos son capaces de crear y subir sus propias imágenes los profesores podrán utilizarlas de manera inmediata sin tener que esperar al manual de instalación o despliegue.

En el siguiente vídeo vamos a ver como:

- Crear una cuenta en DockerHub
- Crear una organización en DockerHub
- Crear un equipo en DockerHub y darle una serie de permisos.
- Hacer Login/Logout en DockerHub desde consola.

https://youtu.be/wePGnp_jeeI?list=PL-8CyWabyNa85xowmOeBMCspbrn6qNWgl