Título	2. Docker.
Destinatario	1º DAW - Semipresencial
Autor	Pascual Martínez
Correo	pmartinez@florida-uni.es

2. Docker.

1. **¿Qué es Docker? 2**
2. **Ejemplo....3**
3. **Elementos básicos de Docker....4**
4. **Virtualización VS contenedores....5**
5. **CaaS....7**
6. **Conclusiones. 7**

Título	2. Docker.
Destinatario	1º DAW - Semipresencial
Autor	Pascual Martínez
Correo	pmartinez@florida-uni.es

1. ¿Qué es Docker?

En castellano, se traduce como estibador o portuario.

Según la RAE, un estibador es un trabajador que se ocupa de la carga y descarga de un buque u otro medio de transporte, generalmente mediante contenedores, y distribuye convenientemente los pesos en él.

Puede que tenga mucho ver… la etimología es importante.

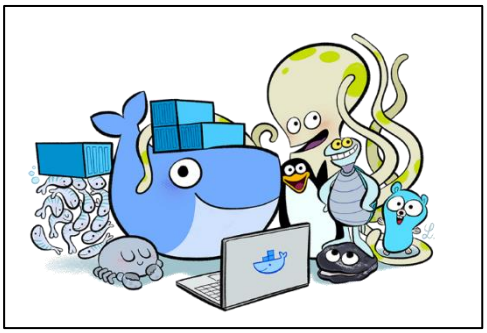
Docker es un **producto tecnológico** de tipo **Open Source**, basado en un conjunto de **herramientas** que permiten crear **contenedores ligeros y portables** para aplicaciones o servicios software. Dichos contenedores **pueden ejecutarse en cualquier máquina que tenga el software Docker instalado**, independientemente del S.O. residente en dicha máquina.

La empresa que desarrolló la tecnología y sacó al mercado las herramientas, también se denomina **Docker**.

Docker está creado por desarrolladores y para desarrolladores.

Solomon Hykes, uno de los creadores de Docker, lo explicaba del siguiente modo:

"*Utilizando contenedores para ejecutar tu código solventamos el típico quebradero de cabeza de que estés usando una versión de Python 2.7 diferente en local o en el entorno de pruebas pero en producción la versión sea totalmente distinta como Python 3 u otras dependencias propias del entorno de ejecución, incluso el sistema operativo. Todo lo que necesitas está dentro del propio contenedor y es invariable*".



2. Ejemplo.

Antes de Docker…

Un posible escenario podría ser el siguiente: Jorge y Carlos trabajan en una empresa de servicios informáticos. Ambos pertenecen al departamento de desarrollo web, concretamente al área de tecnología Java. Cada uno está asignado a un proyecto diferente.

Jorge está asignado a un proyecto de desarrollo a medida, tiene en su ordenador instalada la última versión para desarrollar en Java y está programando una funcionalidad específica de una aplicación, usando algo que solo está disponible en esa versión de Java.

Carlos realiza el mantenimiento y modificaciones en una aplicación que se implantó hace ya un tiempo y que se desarrolló con una versión anterior de Java.

Jorge y Carlos son compañeros desde hace tiempo, se conocen, se entienden bien y se ayudan mutua y habitualmente. Jorge quiere que Carlos ejecute el código de su aplicación en su máquina. Pues bien, o Carlos se instala la versión actual de Java, o la aplicación en su máquina fallará.

Despúes de Docker…

El escenario descrito anteriorente desaparece con Docker. Para ejecutar la aplicación, Jorge se crea un contenedor Docker con la aplicación que está desarrollando, la última versión de Java y el resto de recursos necesarios. Lo publica y se lo pasa a Carlos, que se lo descarga y teniendo Docker instalado en su ordenador, puede ejecutar la aplicación a través del contenedor, sin tener que instalar nada más y manteniendo en su máquina la verdión anterior de Java, para seguir haciendo el mantenimiento y modificaciones en la aplicación a la que está asignado.

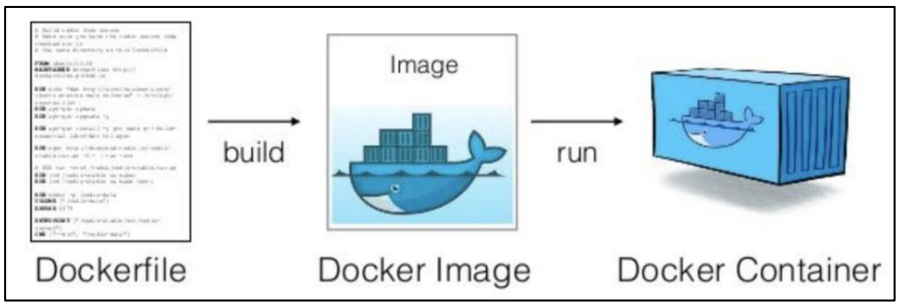
3. Elementos básicos de Docker.

- **Fichero Dockerfile:** es un fichero de configuración utilizado para generar imágenes. En este archivo se detalla lo que debe contener la imagen para adecuar el entorno de ejecución.

- **Imagen:** una imagen es una especie de plantilla, una captura del estado de un contenedor. Las imágenes se utilizan para crear contenedores y son estáticas.

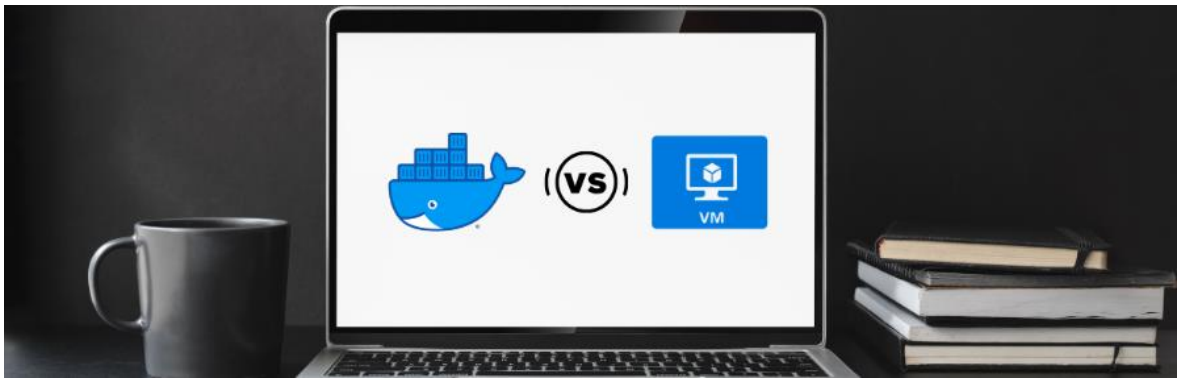
  Por ejemplo, una imagen podría contener un sistema operativo Ubuntu con un servidor Apache y tu aplicación web instalada (o enlazada). Hay muchas imágenes públicas oficiales con elementos básicos como Java, Ubuntu, Apache…, etc., que se pueden descargar y utilizar. Normalmente cuando creas imágenes, partimos de una imagen padre a la que le vamos añadiendo cosas (por ejemplo: una imagen padre con Ubuntu y Apache, que hemos modificado para instalar nuestra aplicación).

- **Contenedor:** son instancias en ejecución de una imagen. A partir de una única imagen, podemos ejecutar varios contenedores, los que necesitemos.



4. Virtualización VS contenedores.

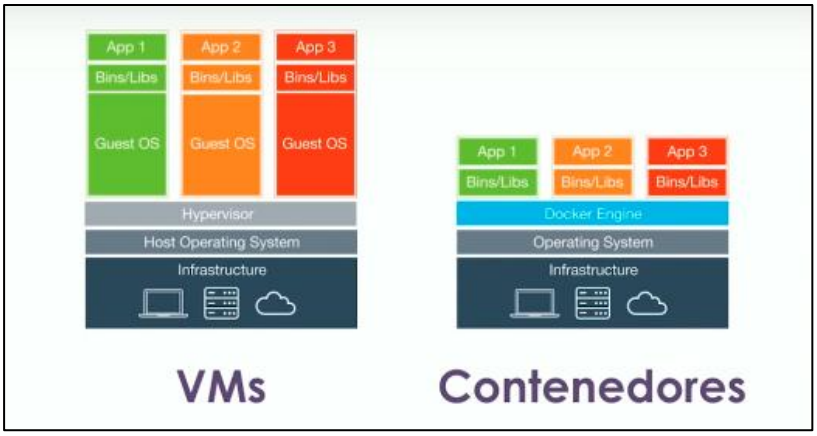
Gracias a la **virtualización** podemos tener **diferentes sistemas operativos ejecutándose en paralelo** sobre la misma máquina física, cada uno con sus **recursos de infraestructura** y completamente **aislados** unos de otros. Gracias al avance de los hipervisores en los últimos años, y a las tecnologías orientadas a la virtualización que ofrecen los procesadores modernos, la pérdida de rendimiento es mínima siendo un **eficiente mecanismo de compartir el hardware para sacarle más partido**.



La filosofía de los contenedores es parecida a la de las MVs (o VMs) , si bien tratan también de aislar las aplicaciones y generar un entorno replicable y estable para que funcionen. Pero **en el caso de los contenedores (Docker**), vamos un paso más allá y **en lugar de albergar un sistema operativo completo, cuenta con una parte reducida** (una especie de Kernel) y **comparte los recursos del propio sistema operativo anfitrión,** "host", sobre el que se ejecutan.

A simple vista puede parecer que no hemos ganado mucho. Al fin y al cabo, solo **desaparece la capa del sistema operativo huésped**, y **se sustituye el hipervisor por lo que se ha denominado "Docker Engine"**, es decir, Docker. Sin embargo, las diferencias son notables.

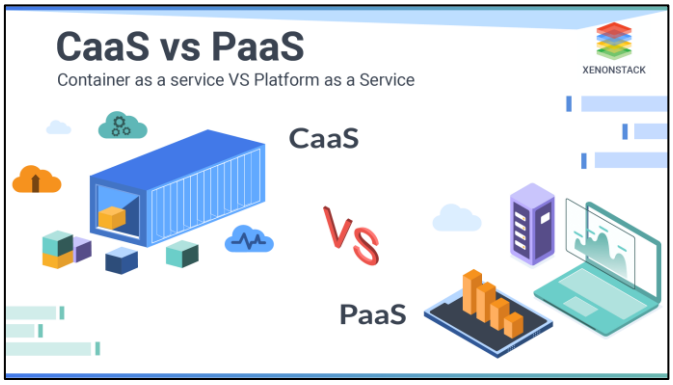
En primer lugar, debemos tener en cuenta que, en el caso de los contenedores, el hecho de que no necesiten un sistema operativo completo, sino que reutilicen el subyacente, **reduce mucho la carga que debe soportar la máquina física, el espacio de almacenamiento utilizado y el tiempo necesario para lanzar las aplicaciones**. Un sistema operativo puede ocupar desde poco menos de 1GB para algunas distribuciones de Linux con lo mínimo necesario, hasta más de 10GB en el caso de un sistema Windows. Además, estos sistemas operativos, para funcionar requieren un mínimo de memoria RAM reservada, que puede ir desde 1 hasta varios GB, dependiendo de nuestras necesidades. Por lo tanto, **los contenedores son más ligeros, ágiles y portables que las máquinas virtuales.**



Cuando se ponen en funcionamiento uno o varios contenedores a partir de una imagen, con una aplicación contenida en cada uno de ellos, es como si estuviese ejecutándose en su propio sistema operativo, aislado de cualquier otra aplicación que hubiese en la máquina física en ese momento. Pero la realidad es que están compartiendo el sistema operativo anfitrión o "host" que hay por debajo. Los recursos hardware se van asignando dinámicamente en función de las necesidades, de lo cual se ocupa Docker (en nuestro caso, porque cabe recalcar que existen otras herramientas similares a Docker, aunque menos difundidas y utilizadas ya que Docker fue la pionera). Por lo tanto, podemos concluir con que **Docker aísla aplicaciones, no sistemas operativos completos**.

5. CaaS.

Los **contenedores como servicio** (Container as a Service), consisten en un modelo de servicio, de uso relativamente reciente de **servicios en la nube**. Permite a las personas desarrolladoras **implementar y gestionar aplicaciones bajo demanda**, a través de la **abstracción** y el **aislamiento**, basada en contenedores. Las personas desarrolladoras escriben el código y gestionar sus datos y aplicaciones. Sin embargo, el entorno para crear e implementar aplicaciones en contenedores lo gestiona y mantiene el proveedor de servicios en la nube.



6. Conclusiones.

La tendencia actual apunta a que el **uso de contenedores va a continuar creciendo**. Estamos asistiendo a una **estandarización progresiva de ciertas tecnologías y herramientas**, como **Docker**. **Tienen como objetivo simplificar, focalizar y aislar el trabajo de los desarrolladores**.

AWS, Google y otras compañías con mucho peso específico en el mercado de la nube, han apostado abiertamente por esta tecnología. Es más, Google es probablemente la primera compañía que se dio cuenta de que necesitaba, una forma más eficiente de implementar y administrar sus componentes software para poder escalar a nivel mundial.

Aunque, si el número de aplicaciones y usuarios crece en nuestro sistema y se convierte en algo complejo de gestionar, Docker no será suficiente….