¿Qué es Docker?

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
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.

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"**

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..)

1.2 Instalación de Docker

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

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

INSTALACIÓN DE DOCKER EN CENTOS8

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
```