DOCKER

1. ¿Qué es Docker?

Docker es una plataforma de virtualización de aplicaciones que permite crear, administrar y ejecutar aplicaciones en contenedores. Los contenedores son una forma de empaquetar una aplicación junto con todas sus dependencias en una unidad portátil y liviana que puede ser ejecutada en cualquier entorno, lo que hace que el proceso de implementación y distribución de aplicaciones sea más fácil y rápido. Docker se ha convertido en una tecnología muy popular en el desarrollo y la implementación de aplicaciones debido a su portabilidad, escalabilidad y facilidad de uso.

2. ¿Cómo funciona Docker?

Docker funciona mediante el uso de contenedores, que son unidades aisladas y portátiles que contienen una aplicación y todas sus dependencias. Cada contenedor utiliza una imagen base que contiene el sistema operativo y el software necesario para ejecutar la aplicación, y luego se agregan las capas necesarias para personalizar el contenedor para la aplicación en cuestión.

Para crear un contenedor Docker, primero se crea una imagen que define todas las dependencias de la aplicación. Una vez que se ha creado la imagen, se puede ejecutar un contenedor a partir de ella. El contenedor utiliza la imagen como base y agrega las capas necesarias para personalizar el entorno de ejecución para la aplicación.

Además, Docker utiliza un sistema de archivos en capas para permitir la reutilización y la compartición de recursos entre contenedores. Esto significa que varias aplicaciones pueden compartir una imagen base y sólo añadir las capas personalizadas necesarias, lo que ahorra espacio de almacenamiento y reduce el tiempo de creación de los contenedores.

En resumen, Docker funciona mediante el uso de contenedores, imágenes y un motor de contenedores para proporcionar un entorno de ejecución aislado y portátil para aplicaciones.

Primeros pasos con Docker: Instalación en Centos

1. Instalación en Centos (Fedora, Red Hat y Oracle Linux).

- Las distintas tareas se harán bajo la versión 23.0.1 (Abril 2023).
- Durante las tareas se usará un usuario administrador.
- Primero borramos las versiones anteriores de Docker si existiesen:

```
sudo yum remove docker \
        docker-client \
        docker-client-latest \
        docker-common \
        docker-latest \
        docker-latest-logrotate \
        docker-logrotate \
        docker-engine
```

- Configuramos el repositorio de Docker Engine. Para descargar siempre la última versión disponible usamos:

```
$yum install -y yum-utils

$yum-config-manager \
        --add-repo \
        https://download.docker.com/linux/centos/docker-ce.repo
```

- Instalamos Docker Engine y los componentes necesarios:

```
yum install docker-ce docker-ce-cli containerd.io
```

- Comprobamos que tenemos instalado los paquetes RPM. Los resultados pueden ser diferentes dependiendo de la versión descargada:

```
rpm -qa | grep docker
```

- Arrancamos el servicio:

```
systemctl start docker
```

- Comprobamos que está funcionando:

```
systemctl status docker
```

- Tambíen podemos probar con:

```
$docker --version
```

- Configuramos docker para que se inicie junto con el sistema operativo:

```
$systemctl enable docker
```

Creando contenedores

En esta tarea conocerán como descargar imágenes y crear contenedores, ver el estado en el que se encuentra dichos contenedores y conocer comandos para ver más información de los mismos. Toda la información se puede conseguir en la página oficial de Docker y las imágenes se pueden buscar en el repositorio propio de Docker que se llama Docker Hub.

Comandos de ayuda:

$docker --help

$docker run --help

$docker ps --help

1. **Creamos nuestro primer contenedor:**

Al ejecutar el comando "docker run NOMBRE_IMAGEN:VERSION", se creará un contenedor.

En primer lugar, Docker buscará la imagen correspondiente a nivel local. Si la imagen se encuentra en el sistema, se procederá a la creación del contenedor. En caso contrario, Docker descargará la imagen del repositorio de Docker Hub y luego creará el contenedor. Si no se especifica una versión específica, Docker descargará automáticamente la última versión disponible (también conocida como "LATEST").

Si solo quisiéramos descargar la imagen sin crear el contenedor cambiaríamos el comando "docker run …" por "docker pull…"

```
$docker run hello-world
```

2. **Comprobamos que el contenedor no está activo:**

Con el siguiente comando se puede ver los contenedores que están en ejecución en nuestro equipo:

```
$docker ps
```

Como el contenedor creado anteriormente ("Hello-World") solo muestra el texto en pantalla y se cierra, vemos que ejecutando el comando anterior no muestra ningún contenedor en ejecución.

Para ver todos los contenedores usamos el siguiente comando:

```
$docker ps -a
```

Como podemos ver con el comando anterior, nos aparece que el contenedor de la imagen “hello-world” se ha salido correctamente (Exited 0) hace 56 minutos.

3. **Comprobamos que la imagen esta descargada:**

El comando “docker images” nos muestra en pantalla todas las imágenes que tenemos en nuestro equipo y del cual podemos crear contenedor sin tener que conectarnos al repositorio de Docker Hub para descargar la imagen.

```
$docker images
```

4. **Creamos otro contenedor a partir de la imagen de “hello-world” y comprobamos que existe:**

```
$docker run hello-world
$docker ps -a
```

5. **Creamos un nuevo contenedor de una imagen de Docker hub denominada busybox y comprobamos que existe dicho contenedor.**

```
$docker run busybox
$docker ps
```

- Comprobamos que existe la nueva imagen:

```
$docker images
```

- Para visualizar solo las IDs de los contenedores usaremos el comando:

```
$docker ps -aq
```

- Para visualizar solo los 2 últimos contenedores:

```
$docker ps -a -n 2
```

Contenedores interactivos

En esta actividad se trabajará con contenedores interactivos, lo que nos permitirá acceder a ellos y trabajar directamente en su interior. De esta forma, podremos trabajar sobre el contenedor creado de manera similar a como lo haríamos con una imagen de sistema operativo.

- Creamos un contenedor interactivo con una imagen de Ubuntu:
    - Con la opción -it indicamos a Docker que va a hacer un contenedor con entorno interactivo.
    - Con bash al final del comando Docker indicamos que queremos acceder por terminal, en este caso a Ubuntu.
    - Como se puede ver en la última línea de la siguiente imagen, nos encontramos dentro del contenedor de Ubuntu. Para salir del contenedor tendremos que poner el comando "exit".

```
$docker run -it ubuntu bash
```

- Podemos trabajar dentro del contenedor de Ubuntu, por ejemplo:

```
$apt-get update
```

- Desde otro terminal podemos ver que el contenedor está activo:

```
$docker ps
```

- Para salir del contenedor:

```
$exit
```

- Podemos ver que ya no está activo y con "docker ps -a" ver que ha salido:

```
$docker ps

$docker ps -a
```

- Con el comando "docker ps -a" nos aparece el ID del contenedor (CONTAINER ID) del contenedor Ubuntu que creamos anteriormente. Con los 4 primeros dígitos de la ID del contenedor o con el nombre del contenedor podemos volver a iniciar el contenedor, para ello usamos el siguiente comando:

```
$docker start -i 3c23 -> (Cuatro primeros dígitos de nuestro contenedor)
```

- Con el siguiente comando paramos el contenedor:

```
$docker stop 3c23 -> (Cuatro primeros dígitos de nuestro contenedor)
```

Contenedores en background

El término "Docker en background" se refiere a la ejecución de contenedores Docker en segundo plano, lo que significa que la aplicación o servicio en el interior del contenedor se está ejecutando sin necesidad de que el usuario interactúe directamente con él. En otras palabras, el contenedor se está ejecutando como un proceso de fondo en el sistema operativo, sin mostrar ninguna salida en la consola o en la interfaz gráfica de usuario. Esto permite que la aplicación o servicio se ejecute de forma autónoma y continúe ejecutándose incluso si se cierra la terminal o la sesión de usuario.

Para empezar esta parte de contenedores en background utilizaremos una imagen de un servidor de apache (httpd):

- Con la opción -d hacemos que el contenedor se ejecute en modo background.
- Con la opción --name podemos poner un nombre a nuestro contenedor, luego ese nombre lo podemos utilizar para parar o iniciar dicho contenedor.
- Vemos con “docker ps” que el contenedor está activo.

```
$docker run -d --name NOMBRE_CONTENEDOR httpd
$docker ps
```

Probamos ahora a realizar lo mismo, pero con una imagen de Centos:

- Veremos que se descarga bien la imagen, pero a pesar de poner la opción -d este contenedor se cierra automáticamente una vez arrancado. Esto se debe a que un contenedor de Centos no tiene nada que hacer al arrancar, pero un servidor apache sí que tiene un comando para ejecutar en el arranque. A modo resumen, no todo vale con la opción -d, sino que el servicio tiene que estar preparado para trabajar en modo background.

- Podemos hacer que nuestro contenedor Centos se ejecute en background si también le añadimos el modo interactivo, “-d -it”:

Lanzar comandos y borrar contenedores e imágenes

En esta parte de la práctica veremos como poder lanzar comandos (exec) a los contenedores y como eliminar contenedores e imágenes.

1. Creamos en modo background un contenedor de Ubuntu, y comprobamos que se está ejecutando:

```
$docker run -d -it --name NOMBRE_CONTENEDOR ubuntu
$docker ps
```

2. En el punto anterior comprobamos que el contenedor Ubuntu está ejecutándose, pero no podemos interactuar con él porque está en modo background. Para poder lanzar comandos a un contenedor en ejecución se utiliza la opción "exec".

   En el siguiente ejemplo vemos como lanzar un comando al contenedor ubuntu1 para que nos liste los archivos que hay en la raíz:

```
$docker exec ubuntu1 ls /
```

3. Podemos lanzar varios comandos a la vez si los separamos por ";":

```
$docker exec ubuntu1 ls / ; hostname -I
```

4. Si queremos entrar a la Shell (terminal) del sistema operativo podríamos ejecutar el siguiente comando:

```
$docker exec -it ubuntu1 bash
```

5. Ahora procederemos a borrar paso por paso todo lo relacionado con la imagen de Ubuntu y sus contenedores. A continuación, se explica paso por paso como hacerlo.
   a. Si estamos dentro de un contenedor hay que salir con el comando "$exit".
   b. Luego hay que parar el contenedor con el comando "$docker stop NOMBRE_CONTENEDOR".
   c. Luego procedemos a borrar el contenedor o contenedores que tengamos de Ubuntu "$docker rm NOMBRE_CONTENEDOR".
   d. Con todos los contenedores eliminados podemos borrar la imagen de Ubuntu con el comando "$docker rmi ubuntu".

En el siguiente ejemplo vemos todos los pasos que se realizan para borrar la imagen de Ubuntu y el contenedor ubuntu1:

```
$exit
 $docker stop ubuntu1
 $docker rm ubuntu1
 $docker rmi ubuntu
```

Docker inspect

En esta parte de la práctica veremos el comando “docker inspect”, que es un comando utilizado en la línea de comandos de Docker que permite obtener información detallada sobre un objeto Docker específico, como un contenedor, una imagen o un volumen. La salida del comando incluye una gran cantidad de información sobre el objeto en cuestión, como su ID, la imagen utilizada para crearlo, la configuración de red, los volúmenes montados y otros metadatos relevantes. La información proporcionada por "docker inspect" puede ser útil para realizar tareas de mantenimiento y solución de problemas en el entorno Docker, así como para obtener detalles específicos necesarios para el desarrollo y la depuración de aplicaciones.

Comando de ayuda:

```
$docker inspect --help
```

1. Descargamos una imagen, por ejemplo “node” y comprobamos que se ha descargado:

```
$docker pull node
$docker images
```

2. Lanzamos el comando “docker inspect” sobre la imagen descargada para ver la información que nos presenta o buscar algo en conceto como la versión descargada:

```
$docker inspect node
$docker inspect node | grep NODE_VERSION
```

3. Ahora creamos un contenedor a partir de la imagen de “node” y usamos el comando “docker inspect” para ver la información que se nos presenta, por ejemplo, su dirección IP asignada:

```
$docker inspect NOMBRE_CONTENEDOR
$docker inspect NOMBRE_CONTENEDOR | grep IPAaddress
```