

ENTORNOS DE DESARROLLO

UD3.2 Principales acciones con
Docker

Contenidos

- ❖ ¿Qué es Docker y cómo funciona?
- ❖ Introducción a los contenedores y a Docker
- ❖ Instalación de Docker
- ❖ **Primeros pasos con Docker**
- ❖ Creación de imágenes personalizadas con Docker.
- ❖ Redes y volúmenes en Docker.
- ❖ Docker compose.
- ❖ Utilidades para gestionar Docker fácilmente.

Índice

- ❖ Docker e interfaz gráfica
- ❖ Imágenes y contenedores
 - Imagen vs Contenedor
 - ¿Dónde se almacenan imágenes, contenedores y datos?
- ❖ Docker Hub
- ❖ Crear e iniciar contenedores con “`docker run`”
 - `docker run`
 - Crear contenedor sin iniciarlo
 - Caso práctico “Hello World”

Índice

- ❖ Listar contenedores en el sistema con `"docker ps"`
- ❖ Parar y arrancar contenedores con `"docker run/start/stop"`
- ❖ Ejecutar comandos en un contenedor con `"docker exec"`
- ❖ Copiar ficheros entre anfitrión y contenedores con `"docker cp"`
- ❖ Renombrar contenedores con `"docker rename"`
- ❖ Parámetros para la ejecución de `"docker run"`

Docker con interfaz gráfica

Existen distintas herramientas para gestionar Docker desde una interfaz gráfica, haciendo la tarea más visual e intuitiva.

Son muy útiles pero para aprender a trabajar con Docker, pueden hacer que se nos escape la comprensión de determinados mecanismos del funcionamiento de Docker.

Por tanto, no usaremos interfaz gráfica.



Imágenes y Contenedores

Imágenes:

- La imagen es una plantilla de solo lectura que se utiliza para crear contenedores. **A partir de una imagen pueden crearse múltiples contenedores.**
- Las imágenes, además de tener su **sistema de ficheros predefinido**, tienen una serie de parámetros predefinidos (comandos, de variables de entorno, etc.) con valores por defecto y que se pueden personalizar en el momento de crear el contenedor.
- Al crear una nueva imagen, simplemente estamos añadiendo una capa a la imagen anterior, la que actúa como base.

Imágenes y Contenedores

Contenedores:

- Son instancias de una imagen.
- Pueden ser arrancados, parados y ejecutados.
- Cada contenedor Docker posee un identificador único de 64 caracteres, pero habitualmente se utiliza una versión corta con los primeros 12 caracteres.
 - Los comandos Docker habitualmente soportan ambas versiones.

Almacenamiento en Docker

El lugar donde se almacenan contenedores e imágenes puede variar según distribución/sistema operativo, driver de almacenamiento y versión de Docker.

Ejecutamos “docker info”

```
Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true

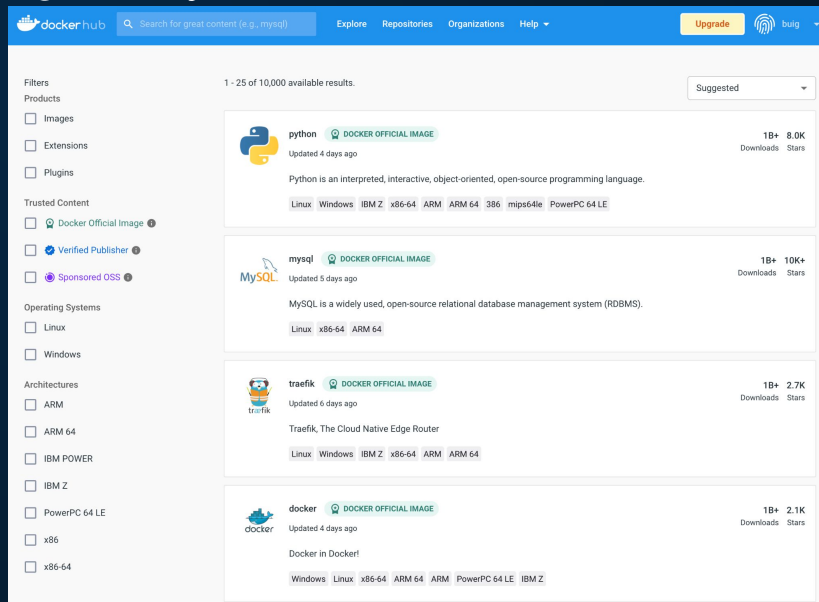
Docker Root Dir: /var/lib/docker
```

- La información de las imágenes se encuentra en **“/var/lib/docker/overlay2”**
- La configuración de los contenedores se almacena en **“/var/lib/docker/containers”**

Docker Hub

Docker Hub es una “plataforma de registro” de Docker. Los servicios básicos son gratuitos y nos permite registrar imágenes Docker, haciéndolas públicas o privadas.

Contiene un gran ecosistema de imágenes ya creadas, usualmente con instrucciones de instalación y uso.



docker run

Es el comando más utilizado.

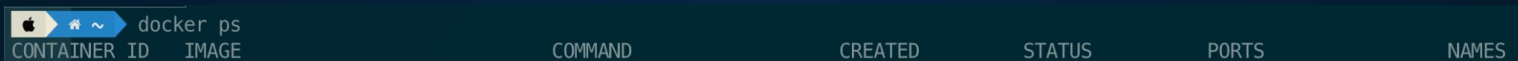
“Podríamos decir que este comando **crea** un contenedor a partir de una imagen y lo **arranca**”

Para **crear un contenedor sin arrancarlo** (recordamos, “**docker run**” **crea y arranca**), existe el comando “**docker create**”.

Vamos a ver el caso práctico: **docker run hello-world**

Listar contenedores con “docker ps”

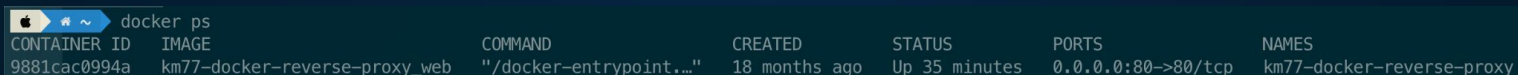
Mediante el comando “**docker ps**” podemos listar los contenedores en ejecución en el sistema. Si ejecutamos el siguiente comando, nos aparecerá un listado como este si no tenemos ningún contenedor en ejecución:



```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

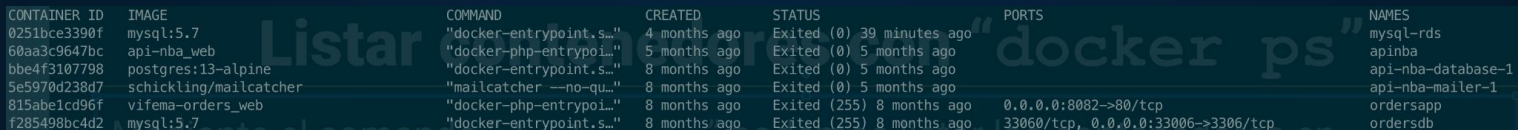
O si tenemos contenedores en ejecución, podemos obtener algo similar a esto:



```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9881cac0994a	km77-docker-reverse-proxy_web	"/docker-entrypoint..."	18 months ago	Up 35 minutes	0.0.0.0:80->80/tcp	km77-docker-reverse-proxy

Si lanzamos el comando **docker ps -a**



```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0251bce3390f	mysql:5.7	"docker-entrypoint.s..."	4 months ago	Exited (0) 39 minutes ago		mysql-rds
60aa3c9647bc	api-nba_web	"docker-php-entrypoi..."	5 months ago	Exited (0) 5 months ago		apinba
bbe4f3107798	postgres:13-alpine	"docker-entrypoint.s..."	8 months ago	Exited (0) 5 months ago		api-nba-database-1
5e5970d238d7	schickling/mailcatcher	"mailcatcher --no-qu..."	8 months ago	Exited (0) 5 months ago		api-nba-mailer-1
815abe1cd96f	vifema-orders_web	"docker-php-entrypoi..."	8 months ago	Exited (255) 8 months ago	0.0.0.0:8082->80/tcp	ordersapp
f285498bc4d2	mysql:5.7	"docker-entrypoint.s..."	8 months ago	Exited (255) 8 months ago	33060/tcp, 0.0.0.0:33006->3306/tcp	ordersdb

Obtendremos un listado de todos los contenedores, tanto aquellos en funcionamiento como aquellos que están parados.

Listar contenedores con “docker ps”

La información que obtenemos de los contenedores es la siguiente:

- **CONTAINER_ID**: identificador único del contenedor (versión 12 primeros caracteres).
- **IMAGE**: imagen utilizada para crear el contenedor.
- **COMMAND**: comando que se lanza al arrancar el contenedor.
- **CREATED**: cuando se creó el contenedor.
- **STATUS**: si el contenedor está en marcha o no
- **PORTS**: redirección de puertos del contenedor (👁👁 más adelante).
- **NAMES**: nombre del contenedor. Se puede generar como parámetro al crear el contenedor, sino, Docker genera uno por defecto.

Parar y arrancar contenedores con `docker run/start/stop`

Para arrancar/parar un contenedor ya creado (recordamos, “`docker run`” crea y arranca), existen los comandos “`docker start`”, “`docker stop`” y “`docker restart`”.

La forma más habitual de usar estos comandos, es usar el nombre del comando, seguido del **identificador único** o **nombre asignado al contenedor**.

Por ejemplo con identificador:

```
docker start 434d318b3771
```

o con nombre del contenedor

```
docker start stupefied_colden
```

Ejecutar comandos en un contenedor con docker exec

El comando “**docker exec**” nos permite ejecutar un comando dentro de un contenedor que esté en ese momento en ejecución. La forma sintaxis habitual para utilizar este comando es la siguiente

```
docker exec [OPCIONES] IDENTIFICADOR/NOMBRE COMANDO [ARGUMENTOS]
```

Algunos ejemplos de uso, suponiendo un contenedor en marcha llamando “SoyUnContenedorEnEjecucion”:

```
docker exec -d SoyUnContenedorEnEjecucion touch /tmp/prueba
```

Se ejecuta en “background”, gracias al parámetro “-d”. Este ejemplo simplemente crea mediante el comando “touch” un fichero “prueba” en “/tmp”.

Ejecutar comandos en un contenedor con docker exec

```
docker exec -it contenedor bash
```

Orden que ejecutará la “shell” bash en nuestra consola (gracias al parámetro “-it” se enlaza la entrada y salida estándar a nuestra terminal y es interactiva).

A efectos prácticos, con esta orden accederemos a una “shell” bash dentro del contenedor.

Copiar ficheros entre anfitrión y contenedores

docker cp

El comando “`docker cp`” es un comando que nos permite copiar ficheros y directorios del anfitrión a un contenedor o viceversa.

Algunos ejemplos de uso:

```
docker cp idcontainer:/tmp/prueba ./
```

Copia el fichero “/tmp/prueba” del contenedor con identificador o nombre “idcontainer” al directorio actual de la máquina que ejerce como anfitrión.

```
docker cp ./miFichero idcontainer:/tmp
```

Copia el fichero “miFichero” del directorio actual al directorio “/tmp” del contenedor.

Renombrar contenedores con docker rename

El comando “`docker rename`” nos permite cambiar el nombre asociado a un contenedor.

```
docker rename contenedor1 contenedor2
```

Cambia el nombre de “`contenedor1`” a “`contenedor2`”.

Parámetros para la ejecución de `docker run`

Vamos a trabajar con algunos ejemplos para coger práctica con el comando.

- ❖ **Ejemplo 1:** Lanzar un contenedor a partir de una imagen de ubuntu y acceder a la terminal

```
docker run -it --name=nuestroUbuntu1 ubuntu /bin/bash
```

Parámetro “-i”: indica que el proceso lanzado en el contenedor docker estará en modo **interactivo**, es decir, **enlaza la entrada estándar cuando se asigna un proceso a una terminal**.

- Parámetro “-t”: asigna al proceso lanzado al arrancar el contenedor una pseudo terminal, facilitando el acceso al mismo desde nuestra terminal.
- Parámetro “--name”: nos permite establecer un nombre a nuestro contenedor.

Parámetros para la ejecución de `docker run`

Vamos a trabajar con algunos ejemplos para coger práctica con el comando.

❖ **Ejemplo 2:** Accedemos a la terminal desde el contenedor parado

En el ejemplo anterior, tras finalizar la sesión con el contenedor (`exit`), observamos que el contenedor se para. Para volver a conectarnos a la terminal del contenedor necesitamos arrancarlo de nuevo.

Esta vez no usaremos `run`, porque no queremos crear de nuevo el contenedor, éste ya existe. Vamos a iniciarlo:

```
docker start [PARAMETROS] IDENTIFICADOR/NOMBRE
```

```
docker ps -a
```

```
docker start -ai IDENTIFICADOR
```

Parámetros para la ejecución de `docker run`

Vamos a trabajar con algunos ejemplos para coger práctica con el comando.

- ❖ **Ejemplo 3:** Ejecutando un versión de una imagen y autoeliminando el contenedor

```
docker run -it --rm ubuntu:14.04 /bin/bash
```

Estamos creando un contenedor con la versión de la imagen “ubuntu” etiquetada como “14.04” en Docker Hub y arrancándolo de forma similar al ejemplo anterior.

- Parámetro “--rm”: este parámetro hará que nada más el contenedor se pare, se borre el contenedor del sistema.

Parámetros para la ejecución de `docker run`

Vamos a trabajar con algunos ejemplos para coger práctica con el comando.

- ❖ **Ejemplo 4:** Creamos un servidor web y asociamos puertos entre máquina anfitrión y el contenedor (huésped)

```
docker run -d -p 1200:80 nginx
```

- Parámetro “-d”: parámetro “deattached”, que indica que lanza el contenedor en segundo plano. La única información que se nos muestra es el ID del contenedor lanzado.
- Parámetro “-p”: siguiendo el estilo “**pAnf:pCont**” nos indica que en el puerto de la máquina anfitrión “**pAnf**” está enlazado con el puerto interno del contenedor “**pCont**”.

Parámetros para la ejecución de `docker run`

Vamos a trabajar con algunos ejemplos para coger práctica con el comando.

- ❖ **Tarea 1:** Cambiamos el `index.html` del contenedor que hemos creado con `nginx` y consultamos los logs

Según la documentación del servidor web *nginx* la ruta que sirve la página principal (`index.html`), se encuentra en la ruta absoluta:

`"/usr/share/nginx/html"`

Con los comandos que ya conocemos:

- Accede con una shell.
- Instala un editor de texto (`apt update; apt install vim`) y ya podemos editar el fichero `index.html`.
- Muestra los logs del contenedor.