2.1 Ejecutando contenedores.

Tras instalar docker y hacer la comprobación de que todo está correcto lo que nos pide el cuerpo es empezar a ejecutar contenedores de todo tipo. Querremos probar todo tipo de sistemas operativos y todo tipo de servicios. Para ello necesitaremos las imágenes de todo aquello que queremos probar. Eso lo podemos conseguir de la siguiente manera:

- **docker pull nombre_imagen:version** que descargará desde el repositorio una imagen con la versión indicada o la última versión **(latest)** si no indicamos versión.
- Y la orden fundamental para ejecutar contenedores que es **docker run** cuya función principal es poner en ejecución contenedores en base a una imagen de referencia que le indicaremos.Una **CUESTIÓN IMPORTANTE** que debemos de tener en cuenta al usar **docker run** es que si vamos a **ejecutar un contenedor** que usa como base una **imagen que no tenemos ésta se descargará de manera automática**. Para buscar las imágenes que queremos la opción que os recomiendo es usar el buscador de Docker Hub.

Esta orden docker run tiene una sintaxis sencilla pero multitud de opciones de las que explicaremos algunas. Iremos introduciendo apartado a apartado las que yo considero que son más importantes para nuestro trabajo diario:

- **-d o --detach** para ejecutar un contenedor (normalmente porque tenga un servicio) en background.
- **-e o --env** para establecer variables de entorno en la ejecución del contenedor.
- **-h o --hostname** para establecer el nombre de red parar el contenedor.
- **--help** para obtener ayuda de las opciones de docker.
- **--interactive o -i** para mantener la STDIN abierta en el contenedor.
- **--ip** si quiero darle una ip concreta al contenedor.
- **--name** para darle nombre al contenedor.
- **--net o --network** para conectar el contenedor a una red determinada.
- **-p o --publish** para conectar puertos del contenedor con los de nuestro host.
- **--restart** que permite reiniciar un contenedor si este se "cae" por cualquier motivo.
- **--rm** que destruye el contenedor al pararlo.
- **--tty o -t** para que el contenedor que vamos a ejecutar nos permita un acceso a un terminal para poder ejecutar órdenes en él.
- **--user o -u** para establecer el usuario con el que vamos a ejecutar el contenedor.
- **--volume o -v** para montar un bind mount o un volumen en nuestro contenedor.
- **--wordirk o -w** para establecer el directorio de trabajo en un contenedor.

A continuación vamos a ver algunos ejemplos básicos.

EJEMPLO 1:

```
# Descargar  una imagen de manera previa
> docker pull ubuntu:18.04
# Crear un contenedor de ubuntu:18.04 y tener acceso a un shell en él. Si no hemos descargado la imagen de manera previa se descargará.
> docker run -it ubuntu:18.04 /bin/bash
root@ef2bea1d6cb1:/#
```

Al crear el contenedor se nos da un acceso a un shell del mismo. Es importante destacar que

estamos **accediendo como root** que tiene unas implicaciones de seguridad . Al salir del terminal el contenedor se para.

EJEMPLO 2:

```
# Crear un contenedor de centOs:18.04 y listar el contendido de la carpeta /
> docker run centOs:18.04 ls /
bin etc lib lost+found mnt proc run srv tmp var

dev home lib64 media opt root sbin sys usr
```

Al crear el contenedor se ejecuta la orden ls / y posteriormente el contenedor pasa a estar parado. Y ya no podremos acceder a él. Explicaremos en el próximo apartado el porqué.

EJEMPLO 3:

```
# Crear un contenedor de httpd (Servidor Apache)
> docker run httpd
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.6. Set the 'ServerName' directive globally to suppress this message

AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.6. Set the 'ServerName' directive globally to suppress this message

[Mon Dec 07 10:01:52.670809 2020] [mpm_event:notice] [pid 1:tid 140412541457536] AH00489: Apache/2.4.46 (Unix) configured -- resuming normal operations

[Mon Dec 07 10:01:52.670973 2020] [core:notice] [pid 1:tid 140412541457536] AH00094: Command line: 'httpd -D FOREGROUND'
```

Al ejecutar esa orden se crea un servidor Web Apache 2.4 en la ip mostrada y se nos muestra por pantalla el log de dicho servicio.

EJEMPLO 4:

```
# Crear un contenedor de  debian 9 y mostrar el contenido de una carpeta establecida con el parámetro -w
> docker run -it -w /etc debian:9 ls
```

Al crear el contenedor se ejecuta la orden ls desde el directorio /etc, posteriormente el contenedor pasa a estar parado. Y ya no podremos acceder a él. Explicaremos en el próximo apartado el porqué.

Conforme vayamos creando contenedores hay dos órdenes que nos van a interesar para hacer un seguimiento de qué tenemos en nuestro sistema:

```
# Mostrar los contenedores en ejecución (Estado Up)
> docker ps
# Mostrar todos los contenedores creados ya estén en ejecución (Estado Up) o parados (Estado Exited)
```

```
> docker ps -a
```

REFERENCIA COMPLETA DE DOCKER RUN

Para aquellos que quieran tener un acceso directo a la **referencia completa de *docker run*** os dejo aquí el enlace:
https://docs.docker.com/engine/reference/commandline/run/