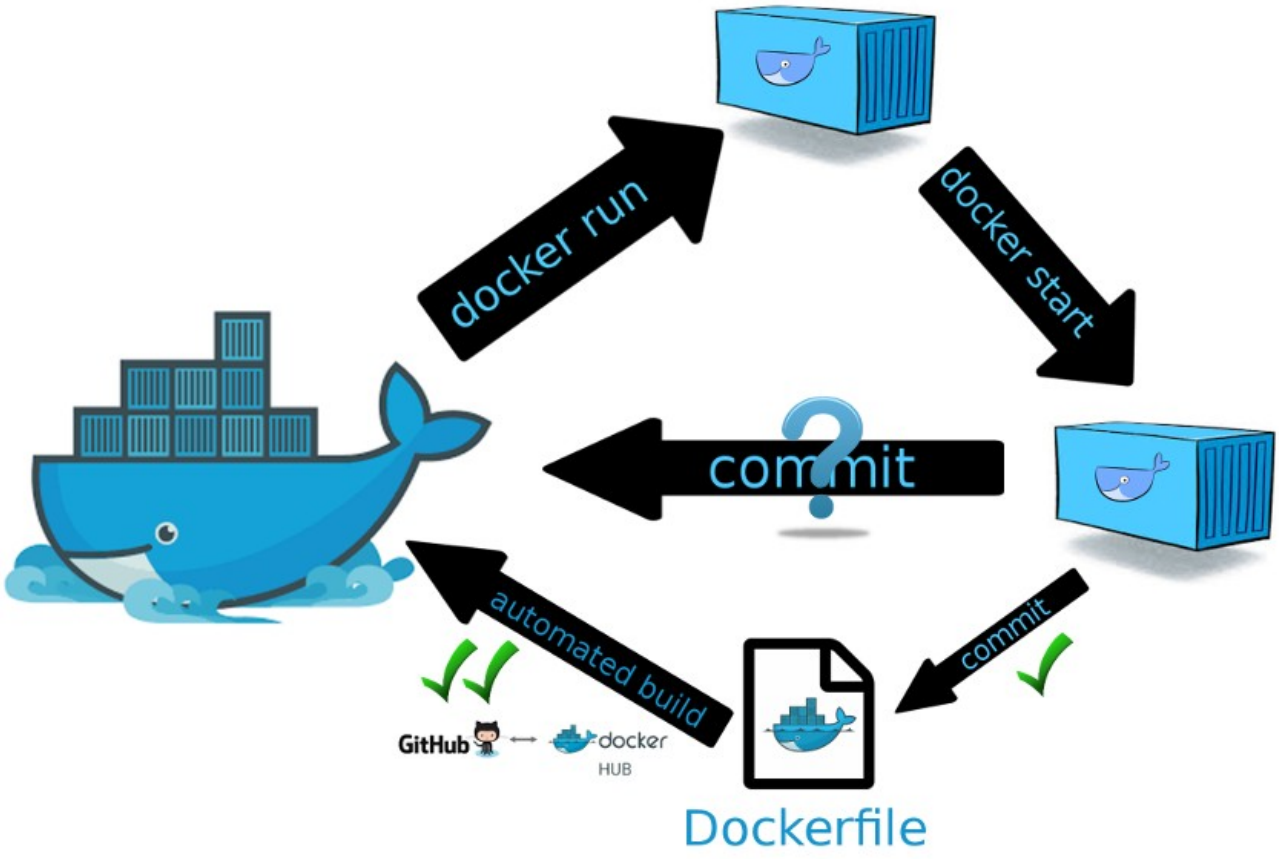
Actividad 11-05: Instalación y puesta en marcha de un contenedor



ÍNDICE

1. Introducción

En el presente documento se detalla el proceso de instalación de un contenedor Docker de Python, para su posterior ejecución, de cara a comprobar que funciona. Para ello, accederemos a Docker Hub para descargar la imagen apropiada y poder crear un script sencillo en este lenguaje de programación.

También se han incluido varios apartados 'extra', que servirán para mostrar y ejemplificar cómo se utiliza Docker desde cero, con las funcionalidades más básicas.

Tras seguir este tutorial serás capaz de:

- Comprender qué es Docker.
- Instalar Docker en Ubuntu mediante la terminal.
- Utilizar los comandos 'docker' (sin usar sudo).
- Descargar imágenes Docker desde Docker Hub.
- Diferenciar entre imagenes y contenedores, y consolidar los cambios de un contenedor en una nueva imagen.
- Poner en marcha un contenedor que permita ejecutar programas escritos en Python.
- Verificar que el contenedor funciona correctamente.
- (Cualquier otro tipo de conocimiento transversal que afortunadamente aprendas durante la lectura de este documento, o como suele decirse: etc.).

2. (Extra) Instalación de Docker en Ubuntu

(Siguiendo el tutorial: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-es)

En resumidas cuentas, y siguiendo con el enlace de arriba, haremos lo siguiente:

- Primero, actualizamos la lista de paquetes existentes de nuestra Ubuntu:

```
alberto@maq-docker-Alberto:~$ sudo apt update
[sudo] password for alberto:
Hit:1 http://es.archive.ubuntu.com/ubuntu focal InRelease
```

- Ahora, instalaremos algunos paquetes de requisitos previos, que nos permitirán usar paquetes a través del protocolo HTTPS:

sudo apt install apt-transport-https ca-certificates curl software-properties-common

```
alberto@maq-docker-Alberto:~$ sudo apt install apt-transport-https ca-certificates curl software
-properties-common
Reading package lists... Done
Building dependency tree
```

- Luego, añadimos la clave GPG (GNU Privacy Guard) para el repositorio oficial de Docker:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

```
alberto@maq-docker-Alberto:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt
-key add -
OK
```

- Ahora agregamos el repositorio de Docker a las fuentes de APT:

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

```
alberto@maq-docker-Alberto:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.
com/linux/ubuntu focal stable"
```

- Una vez hecho eso, procedemos a actualizar de nuevo la lista de paquetes, con el repositorio de Docker recién agregado:

```
alberto@maq-docker-Alberto:~$ sudo apt update
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease
```

- Tenemos que comprobar desde qué repositorio se va a instalar Docker antes de instalarlo, pues nos interesa que se instale desde el repositorio de Docker que acabamos de agregar, en lugar de desde el predeterminado de Ubuntu (porque sería probable que nos diese problemas). Para ello, ejecutamos el siguiente comando:

apt-cache policy docker-ce

```
alberto@maq-docker-Alberto:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.14~3-0~ubuntu-focal
  Version table:
     5:20.10.14~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
```

(De este modo, podemos ver que se nos detalla la lista de repositorios desde los que se va a instalar Docker, por orden de prioridad. Como podemos comprobar, el primero pertenece al repositorio de Docker recién agregado, así que todo es correcto).

- Podemos instalar Docker (la versión de la comunidad, Community Edition):

```
alberto@maq-docker-Alberto:~$ sudo apt install docker-ce
[sudo] password for alberto:
Reading package lists... Done
```

Con esto, Docker estaría instalado y, como demonio de Ubuntu, podemos manejarlo mediante 'systemctl' o 'service':

```
alberto@maq-docker-Alberto:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
```

```
alberto@maq-docker-Alberto:~$ sudo service docker status
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
```

Cómo ejecutar el comando 'docker' sin necesidad de usar 'sudo'

Por defecto, el comando docker, que nos proporciona el demonio Docker, está configurado para ser utilizado por root o por los usuarios pertenecientes al grupo docker, que se crea automáticamente durante el proceso de instalación de Docker. Los superusuarios (sudoers, pertenecientes al grupo sudo), deberían ser también capaces de utilizar estos comandos.

Para evitar tener que usar 'sudo' todo el rato, simplemente agregamos nuestro usuario al grupo 'docker' ya mencionado, mediante el siguiente comando:

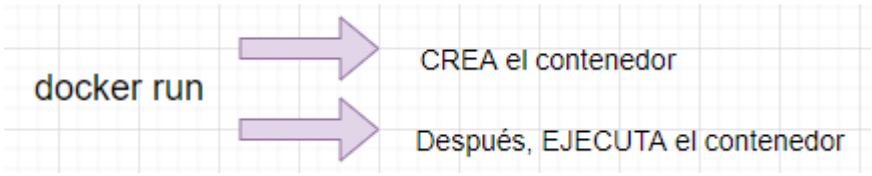
sudo usermod -aG docker ${USER}

```
alberto@maq-docker-Alberto:~$ sudo usermod -aG docker ${USER}
[sudo] password for alberto:
alberto@maq-docker-Alberto:~$ groups
alberto adm cdrom sudo dip plugdev lpadmin lxd sambashare vboxsf
alberto@maq-docker-Alberto:~$ sudo usermod -aG docker alberto
alberto@maq-docker-Alberto:~$ groups
alberto adm cdrom sudo dip plugdev lpadmin lxd sambashare vboxsf
alberto@maq-docker-Alberto:~$
alberto@maq-docker-Alberto:~$
alberto@maq-docker-Alberto:~$ sudo addgroup alberto docker
The user `alberto' is already a member of `docker'.
alberto@maq-docker-Alberto:~$
```

Como se puede ver en la imagen, cuando utilizo ‘groups’, no aparece el grupo docker como si mi usuario perteneciera a el, pero el comando ha funcionado. El motivo de que no aparezca se debe a que, para que los cambios se apliquen correctamente, hay que reiniciar la máquina (si solo reiniciamos la consola puede que no se apliquen los cambios).

¡Ya tenemos Docker funcionando en nuestra máquina! ¡Y con un usuario del grupo ‘docker’ que no necesita usar ‘sudo’ para los comandos!

3. (Extra) vista rápida del comando 'docker'



docker → permite consultar todos los subcomandos disponibles.

docker docker-subcommand --help → Para ver las opciones disponibles de un subcomando especifico.

docker run → Primero, crea un contenedor (desde una instancia de una imagen. Si no tiene ninguna, se la descarga de hub.docker.com). Despues, ejecuta dicho contenedor.

Ejemplo: docker run hello-world → Le estamos diciendo a docker que nos cree un contenedor con un programa llamado 'hello world' (classic), y que luego lo ejecute. Como podemos ver en la siguiente captura, primero lo va a buscar en local y, al no encontrarlo, lo buscará y descargará directamente de Docker Hub.

```
alberto@maq-docker-Alberto:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Un poco de teoría: los contenedores de Docker se construyen a partir de imágenes de Docker. Estas imágenes se obtienen de Docker Hub, un registro de imágenes gestionado por una empresa llamada, sorprendentemente, Docker. Cualquiera puede subir sus imágenes ahí, de modo que hay

una gran librería de imágenes.

docker search [options] → busca una imagen en Docker Hub.

Ejemplo: docker search ubuntu:

```
alberto@maq-docker-Alberto:~$ docker search ubuntu
NAME                      DESCRIPTION                                     STARS     OFFIC
IAL   AUTOMATED
ubuntu                    Ubuntu is a Debian-based Linux operating sys…   14131     [OK]
```

docker pull → una vez identificada una imagen que deseamos usar, la podemos descargar en nuestra máquina mediante el uso de este comando.

Ejemplo: docker pull ubuntu:

```
alberto@maq-docker-Alberto:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
8527c5f86ecc: Downloading  13.09MB/30.42MB
```

docker images → Nos va a decir cuántas imágenes tenemos descargadas en nuestra máquina.

```
alberto@maq-docker-Alberto:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
ubuntu        latest    3f4714ee068a   4 days ago      77.8MB
hello-world   latest    feb5d9fea6a5   7 months ago    13.3kB
```

Más teoría (lo siento): las imágenes que nos descargamos de Docker Hub pueden modificarse para crear imágenes nuevas y estas, a su vez, pueden introducirse en Docker Hub o en otros registros de Docker.

docker ps → Para ver los contenedores que están corriendo.

```
root@630abf9b17f9: /                    alberto@maq-docker-Alberto: ~
alberto@maq-docker-Alberto:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
630abf9b17f9   ubuntu    "bash"    38 seconds ago   Up 37 seconds             vigorous_clarke
alberto@maq-docker-Alberto:~$
```

(El contenedor está ejecutándose en la otra pestaña de la shell de Linux):

docker ps -a → para ver TODOS los contenedores, tanto los activos como los inactivos:

```
alberto@maq-docker-Alberto:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS
AMES
630abf9b17f9   ubuntu        "bash"     13 minutes ago   Exited (0) 8 minutes ago
igorous_clarke
28a289246b18   hello-world   "/hello"   15 minutes ago   Exited (0) 15 minutes ago
ealous_franklin
a2e0c83dcd69   hello-world   "/hello"   28 minutes ago   Exited (0) 28 minutes ago
ice_noyce
```

docker ps -l → Para ver el último contenedor que se creó.

```
alberto@maq-docker-Alberto:~$ docker ps -l
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS                      PORTS
630abf9b17f9   ubuntu    "bash"    15 minutes ago   Exited (0) 9 minutes ago
s_clarke
```

EJECUTAR CONTENEDORES DE DOCKER

docker run -it ubuntu → los modificadores -i y -t sirven para que el proceso del contenedor sea interactivo con la terminal de Linux.

```
alberto@maq-docker-Alberto:~$ docker run -it ubuntu
root@630abf9b17f9:/#
```

(En la segunda linea podemos ver que se trata de la terminal del Ubuntu que tenemos en el contenedor).

docker run -it --name dawa ubuntu → En vez de que docker te cree el contenedor con un nombre aleatorio, lo hace con la cadena que especificas después de '--name'

```
alberto@maq-docker-Alberto:~$ docker run -it --name ubuntu_dawa ubuntu
root@52c860d064f1:/# exit
exit
```

Para cerrar el contenedor de Ubuntu, excribimos ‘exit’ en la linea de comandos:

```
root@630abf9b17f9:/# exit
exit
alberto@maq-docker-Alberto:~$
```

docker run -d [options] → Para que se descargue y ejecute en segundo plano (detached).

docker start [id/nombre_contenedor] → para iniciar un contenedor detenido (no confundir con ‘docker run’).

docker rm [id/nombre_contenedor] → Para borrar un contenedor.

4. (Extra) consolidar los cambios de una imagen de Docker.

Cuando creamos un contenedor de Docker con el comando ‘docker run [imagen]’, podemos modificar y eliminar archivos, y hacer lo que queramos con el. Pero todos esos cambios sólo se guardarán en el contenedor que tenemos. De manera que, si hacemos ‘docker rm’ para eliminar el contenedor, se perderá todo.

Para prevenir esta situación, lo que podemos hacer es consolidar nuestros cambios en una imagen de Docker, de tal manera que podamos compartirla fácilmente y tener siempre una copia del contenedor en el estado que queremos.

He creado un contenedor llamado ‘ubuntu_dawa’ para este ejemplo, y he instalado apache2 en este:

```
root@630abf9b17f9:/# apt install apache2
Reading package lists... Done
Building dependency tree... Done
```

Ahora, este contenedor es diferente de la imagen que utilicé para crearlo. Consolidemos los cambios:

docker commit -m “mensaje informativo” -a “nombre del autor” id_contenedor repositorio/nombre_imagen

```
alberto@maq-docker-Alberto:~$ docker commit -m "imagen de ubuntu con apache2 instalado" -a "Albe
rto" 52c860d064f1 kadex/ubuntu_dawa
sha256:7b9f125ea547f4d5cf0c055ccf430fe8117f92a3d75d4ec6166cdea67af8094d
alberto@maq-docker-Alberto:~$
```

(Ese ‘chorizaco’ indica que todo ha salido bien).

Normalmente, el valor de ‘repositorio’ coincide con el nombre de usuario de Docker Hub (se hace así para luego subir los cambios a Docker Hub).

Cuando *consolidamos* una imagen, la nueva imagen queda guardada a nivel local en nuestra máquina. De este modo, podemos subirla a Docker Hub para compartirla con otros usuarios o simplemente tenerla almacenada.

Comprobamos ahora que el proceso ha funcionado:

```
alberto@maq-docker-Alberto:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
kadex/ubuntu_dawa   latest    7b9f125ea547   2 minutes ago    77.8MB
ubuntu              latest    3f4714ee068a   4 days ago       77.8MB
hello-world         latest    feb5d9fea6a5   7 months ago     13.3kB
```

(Aquí podemos ver que la primera imagen es la que acabamos de crear).

5. Descripción del contenedor

El enlace a la imagen que voy a utilizar es el siguiente: https://hub.docker.com/_/python

Como viene indicado en el enlace, se trata de la imagen Docker oficial de Python.

El script que voy a ejecutar es muy sencillo, simplemente voy a imprimir por pantalla una cadena de texto y una operación matemática sencilla. Si todo sale bien, eso demostrará que el contenedor funciona correctamente.

¡Comenzamos!

6. Instalación y puesta en marcha del contenedor.

Mediante el comando → docker pull python → puedo descargarme la imagen Docker para crear mi contenedor Docker de Python.

Otra posibilidad seria ejecutar directamente el comando → docker run python → que, al no encontrar ninguna imagen en mi local, se la descargará directamente de Docker Hub. Pero para este ejemplo, voy a utilizar la primera opción.

```
alberto@maq-docker-Alberto:~$ docker pull python
Using default tag: latest
latest: Pulling from library/python
6aefca2dc61d: Downloading  8.596MB/54.94MB
967757d56527: Download complete
c357e2c68cb3: Download complete
```

Comprobamos que nuestra imagen se ha descargado correctamente:

```
alberto@maq-docker-Alberto:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
kadex/ubuntu_dawa   latest    7b9f125ea547   15 hours ago    77.8MB
ubuntu              latest    3f4714ee068a   5 days ago      77.8MB
python              latest    2b7ca628da40   6 days ago      920MB
hello-world         latest    feb5d9fea6a5   7 months ago    13.3kB
```

Ahora ejecutamos el comando run, con algunos modificadores para personalizarlo un poco (quiero ponerle un nombre personalizado para no tener que ir escribiendo el ID del contenedor cada vez que quiera hacer algo):

```
alberto@maq-docker-Alberto:~$ docker run -dit --name python_test python
06a103a8fac32309af354dabdddb2ee78b26f265a0fa7a3a7756ac40e700fde3
alberto@maq-docker-Alberto:~$
```

Comprobamos que el contenedor está funcionando (con docker ps):

```
alberto@maq-docker-Alberto:~$ docker ps
CONTAINER ID   IMAGE     COMMAND     CREATED          STATUS          PORTS     NAMES
06a103a8fac3   python    "python3"   38 seconds ago   Up 37 seconds             python_test
alberto@maq-docker-Alberto:~$
```

7. Verificación de su comportamiento.

Una vez que hemos comprobado que nuestro contenedor Docker de Python está instalado e iniciado, ¡vamos a utilizarlo!

Para ello, voy a cerrarlo:

```
alberto@maq-docker-Alberto:~$ docker stop python_test
python_test
alberto@maq-docker-Alberto:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
alberto@maq-docker-Alberto:~$
```

Y abrirlo de nuevo, de forma interactiva, con el comando *docker start -i python_test*, nos deberá abrir el intérprete de Python mediante nuestro contenedor de Docker, como se ve a continuación:

```
alberto@maq-docker-Alberto:~$ docker start -i python_test
Python 3.10.4 (main, Apr 20 2022, 18:21:23) [GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Una vez aquí, podríamos escribir cualquier cosa que tenga sentido en lenguaje Python, y debería ejecutarse sin problemas.

```
>>> print("I just created my first Docker container!\n AND IT WORKS!")
I just created my first Docker container!
 AND IT WORKS!
>>>
```

```
>>> 2*4
8
>>>
```

Con esto, ya hemos demostrado que nuestro contenedor funciona. De hecho, nótese que la versión de python que tenemos instalada en nuestra máquina Ubuntu es otra distinta (la 3.8.10, en contraste con la 3.10.4 de nuestro contenedor Docker):

```
alberto@maq-docker-Alberto:~$ python3
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

¡PRUEBA SUPERADA!

8. Conclusiones

Por fin, llegamos al final. Una vez aquí, has aprendido a instalar Docker en Ubuntu, conoces los comandos principales de Docker y sabes descargar imágenes del repositorio Docker Hub, así como utilizar dichas imágenes para crear contenedores.

Habrás podido comprobar la utilidad que esto puede tener a la hora de desarrollar tus propios proyectos de software, pudiendo consolidar los contenedores en imágenes nuevas que otros compañeros podrían descargar, evitando muchos problemas con las dependencias de los proyectos (y también mucho dolor de cabeza).

Y todo esto sin necesidad de tener un Sistema Operativo completo funcionando para cada contenedor, como ocurre con las máquinas virtuales.

La prueba que se ha utilizado para este documento formativo es muy simple y poco ambiciosa, pero espero que te haya servido para comprender el funcionamiento básico de la herramienta.

Todavía queda mucho contenido por abordar (dockerfiles, subir el contenido a Docker Hub, etc. pero lo dejaremos para más adelante).

Referencias//Bibliografía

- https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-es
- El inagotable conocimiento de mi profesor; Fernando Usero Fuentes.
- https://hub.docker.com/_/python
- https://www.digitalocean.com/community/tutorials/how-to-remove-docker-images-containers-and-volumes-es
- https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/holamundo.html