

## DOCUMENTACIÓN PRÁCTICA 5

- **Execute docker for a first time, to verify installation**

Una vez instalado el *docker* con el comando que se nos indicaba en el enunciado de la práctica, hemos ejecutado el hello world. En esta ejecución, hemos obtenido el siguiente resultado que se muestra en la imagen:

```
mininet@mininet:~$ sudo docker run hello-world

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/
```

Como podemos observar, realmente lo que se nos muestra es una pequeña descripción del *docker*, y se nos da la bienvenida.

- **Compile the image with the command.**

Después de haber creado el Dockerfile tal y como se nos indicaba en la teoría, hemos creado una imagen *docker*, tal y como se nos indicaba en el enunciado. Esta imagen se encarga de leer las instrucciones del Dockerfile. A partir de la siguiente imagen, vemos que se ha creado satisfactoriamente con el nombre *restapi*.

```
Successfully built 5637c6a5ec21
Successfully tagged restapi:latest
```

- **Create a network for your containers:**

A partir del comando propuesto en la práctica, el que se muestra en la imagen, hemos creado una subred 172.18.0.0/16 con nombre *dockerNet*. El resultado es el que se muestra a continuación:

```
mininet@mininet:~/Docker$ sudo docker network create --subnet=172.18.0.0/16 dockerNet
bea88c2882c5b5351199b262589f397ead46b80d46b6fa050791e8b9bddcf091
```

Sara Soriano - 240007

Rubén Vera - 241456

Eneko Treviño - 241679

- **Start a container for your image. Parameter detach executes it in background mode (will not stop at execution), rm removes the image after finishing, net uses the network named in previous point, ip associates the address to the image, and name is the alias associated to the container.**

El docker ejecuta la imagen restapi como un Docker container en la red seleccionada, la que hemos creado, es decir, dockerNet. Cualquier solicitud hecha al puerto 5200 será pasada al container a través del puerto 5200(gracias al publish 5200:5200). A causa de la flag `--rm` imprimirá su nombre cuando se pare el container. Después del parámetro `--name` ponemos el nombre del container, restapiTest en nuestro caso. Finalmente, lo dejamos en el background con `--detach` y el container pertenecerá a Dockernet(`--net`) usando la ip 172.18.0.2(`--ip`).

```
mininet@mininet:~/Docker$ sudo docker run --detach --rm --publish 5200:5200 --name restapiTest --net dockerNet --ip 172.18.0.2 restapiTest
```

- **The started container offers a functional virtual host running the flask web server. Verify with the usual wget test. Notice that the same web server can not be run directly on the native host, try `$ sudo python3 Flask1.py`, because the python packages flask and jsonify are not available in the host.**

A partir del comando `wget` vemos que el *container* se está ejecutando de manera correcta ya que vemos en la primera imagen que se ha establecido conexión. En cambio, si ejecutamos el comando que se muestra en la segunda imagen vemos que ocurre un error. Tal como vemos esto ocurre porque *Flask* y *jsonify* no están disponibles.

```
mininet@mininet:~/Docker$ wget -O - --progress=dot http://172.18.0.2:5200/system
--2022-05-12 18:54:06-- http://172.18.0.2:5200/system
Connecting to 172.18.0.2:5200... connected.
HTTP request sent, awaiting response... 200 OK
Length: 49 [application/json]
Saving to: 'STDOUT'
{"email":"my_email@gmail.com","system":"Docker"}

OK 100% 6,95M=0s
2022-05-12 18:54:06 (6,95 MB/s) - written to stdout [49/49]
```

```
mininet@mininet:~/Docker$ sudo python3 Flask.py
Traceback (most recent call last):
  File "Flask.py", line 1, in <module>
    from flask import Flask, jsonify
ModuleNotFoundError: No module named 'flask'
```

- **You can list the running containers:**

A partir del comando realizado, hemos podido visualizar los *containers* que se están ejecutando. En nuestro caso, la información obtenida es la que se muestra en la siguiente imagen, que corresponde a nuestro container restapiTest con su correspondiente imagen restapi:

Sara Soriano - 240007  
Rubén Vera - 241456  
Eneko Treviño - 241679

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f7d430a95b94	restapi	"python3 Flask.py"	3 seconds ago	Up 2 seconds	0.0.0.0:5200->5200/tcp, :::5200->5200/tcp	restapiTest

- **Attach from a second terminal to your container, and execute again the wget test in a third terminal. What do you see in the second terminal?**

A partir de un segundo terminal, como se nos pedía, hemos realizado el *attach* al *containerID*, es decir nos hemos adjuntado al container para ver todo el tráfico que recibía y enviaba. Esto ha sido posible añadiendo al final la ID que sabíamos del apartado anterior.

```
mininet@mininet:~/Docker$ sudo docker attach f7d430a95b94
```

Una vez realizado, el *attach* en el segundo terminal, hemos realizado de nuevo el *wget* en un tercer terminal, obteniendo que la operación ha sido exitosa. Si nos fijamos en la última imagen de este apartado, vemos la información que le llega al terminal 2. La información recibida es que se ha enviado una request de HTTP a través del puerto 5200 a la dirección ip de nuestro container.

```
mininet@mininet:~/Docker$ wget -O - --progress=dot http://172.18.0.2:5200/system
--2022-05-12 18:57:27-- http://172.18.0.2:5200/system
Connecting to 172.18.0.2:5200... connected.
HTTP request sent, awaiting response... 200 OK
Length: 49 [application/json]
Saving to: 'STDOUT'
{"email":"my_email@gmail.com","system":"Docker"}

      OK                                     100% 5,16M=0s
2022-05-12 18:57:27 (5,16 MB/s) - written to stdout [49/49]
```

```
mininet@mininet:~/Docker$ sudo docker attach f7d430a95b94
172.18.0.1 - - [12/May/2022 16:57:27] "GET /system HTTP/1.1" 200
```

- **Stop your container using the name you assigned:**

```
mininet@mininet:~/Docker$ wget -O - --progress=dot http://172.18.0.2:5200/system
--2022-05-12 19:02:38-- http://172.18.0.2:5200/system
Connecting to 172.18.0.2:5200... failed: No route to host.
```

En la anterior imagen, vemos cómo una vez hemos parado el *container* y hemos intentado realizar de nuevo el *wget* dice que no encuentra una ruta para llegar al host ya que este ya no existe. Este es el resultado esperado y nos confirma que el proceso ha sido realizado correctamente, ya que con el comando correspondiente hemos parado nuestro *container*.

- **You can clean up the whole docker environment, remember that after that your network, containers and images will no longer be available:**

La siguiente imagen muestra la información que se muestra por pantalla después de limpiar nuestro *docker* donde se ve como se ha eliminado nuestra red "dockerNet".

Sara Soriano - 240007  
Rubén Vera - 241456  
Eneko Treviño - 241679

```
mininet@mininet:~/Docker$ sudo docker system prune
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all dangling images
- all dangling build cache

Are you sure you want to continue? [y/N] y
Deleted Networks:
dockerNet

Total reclaimed space: 0B
```

- Now we are going to save the Flask output you probably saw in step 10 in a file of the container, and make it readable from the server. Now make one or more requests to the server as in step 8, and verify in /tmp directory the existence of a new file “flask.log”. What does it contain, why is it there? You can see in real time how the content is added to the file with this command, run it in another terminal: `$ tail -F /tmp/flask.log`; and then make more requests to the server.

Una vez realizados todos los comandos que se nos pedía en la práctica, hemos comprobado que el archivo “flask.log” estuviese en el directorio /tmp, tal y como debía. Lo que contiene el archivo son las siguientes líneas:

```
1 * Serving Flask app 'Flask' (lazy loading)
2 * Environment: production
3 WARNING: This is a development server. Do not use it in a production deployment.
4 Use a production WSGI server instead.
5 * Debug mode: off
6 * Running on all addresses (0.0.0.0)
7 WARNING: This is a development server. Do not use it in a production deployment.
8 * Running on http://127.0.0.1:5200
9 * Running on http://172.18.0.2:5200 (Press CTRL+C to quit)
```

El archivo se encuentra en ese directorio porque lo hemos establecido allí a través de añadir `-v /tmp:/app/logs`, que creará el archivo flask.log, al comando correspondiente e indicar en el Dockerfile que el output se enviase a /tmp. En las líneas de código se ve que el Flask se está ejecutando en 127.18.0.1 y .2 a través del puerto 5200.

Además hemos utilizado el comando correspondiente *tail* para poder ver en tiempo real lo que se añade en el archivo. Después de hacer varias veces *wget* en un terminal distinto obtenemos el siguiente resultado:

```
mininet@mininet:~$ tail -F /tmp/flask.log
* Serving Flask app 'Flask' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses (0.0.0.0)
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:5200
* Running on http://172.18.0.2:5200 (Press CTRL+C to quit)
172.18.0.1 - - [12/May/2022 17:55:18] "GET /system HTTP/1.1" 200 -
172.18.0.1 - - [12/May/2022 17:58:59] "GET /system HTTP/1.1" 200 -
172.18.0.1 - - [12/May/2022 17:58:59] "GET /system HTTP/1.1" 200 -
```

Sara Soriano - 240007  
Rubén Vera - 241456  
Eneko Treviño - 241679

- Create an account in <https://hub.docker.com>, we will use it to upload the docker image to the repository and make it available to anyone – also to the teacher of the session.


Una vez creada la cuenta de Docker Hub e iniciada la sesión en el terminal, hemos podido subir la *docker image* a un repositorio creado, tal y como en la siguiente imagen se muestra.

```
root@mininet:/home/mininet/Docker# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: saritty
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
root@mininet:/home/mininet/Docker# docker tag restapi saritty/restapi:restapi
root@mininet:/home/mininet/Docker# docker push saritty/restapi:restapi
The push refers to repository [docker.io/saritty/restapi]
7cca26203d17: Pushed
3030c0e2187b: Pushed
435a8ae9df18: Pushed
b5c89fd1fb0e: Pushed
0e7eceb89a22: Mounted from library/python
86ccba8f6653: Mounted from library/python
d321ddd833b8: Mounted from library/python
3759be374189: Mounted from library/python
fd95118ead9: Mounted from library/python
restapi: digest: sha256:be664b69d67bd3709350ae2ade9955cdfda55c7deea9a1617b487e34be1a2af1 size: 2202
root@mininet:/home/mininet/Docker#
```


Si nos dirigimos a la página web de Docker Hub vemos que dicho repositorio ha sido creado y el fichero esperado también está subido. Dicho repositorio está de manera pública de tal manera que cualquier persona que intente acceder lo podrá hacer.

 saritty / restapi


This repository does not have a description 

 Last pushed: 3 hours ago

Tags and Scans

 VULNERABILITY SCANNING - DISABLED  
[Enable](#)

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
 restapi		---	3 hours ago

<https://hub.docker.com/repository/docker/saritty/restapi>