Trabajo de Docker

Sistemas Operativos 2018

Nombre:	**Matías Pierobón Marcos**
Legajo:	13714/0

2) Usando las herramientas (comandos) provistas por Docker realizar las siguientes tareas:

a) Instalar la versión más reciente de la imagen de Ubuntu ¿Cuál es el tamaño de la imagen obtenida? ¿Ya puede ser considerado un container?

```
$ docker pull ubuntu:latest
```

Notar que el tag latest es opcional ya que en este caso es el por defecto

```
$ docker images
REPOSITORY      TAG     IMAGE ID        CREATED         SIZE
ubuntu          latest  113a43faa138    5 hours ago     81.2MB
```

Actualmente la imagen esta en disco y no hay un contenedor de ubuntu, unicamente cuando se inicialice la imagen en memoria podrá ser llamado contenedor a la instancia de esa imagen corriendo.

b) Generar un container a partir de la imagen obtenida en el punto anterior que simplemente ejecute el comando `ls -l`.

```
docker run ubuntu ls -l
total 64
drwxr-xr-x   2 root root 4096 May 26 00:45 bin
drwxr-xr-x   2 root root 4096 Apr 24 08:34 boot
drwxr-xr-x   5 root root  360 Jun  6 02:15 dev
drwxr-xr-x   1 root root 4096 Jun  6 02:15 etc
drwxr-xr-x   2 root root 4096 Apr 24 08:34 home
drwxr-xr-x   8 root root 4096 May 26 00:44 lib
drwxr-xr-x   2 root root 4096 May 26 00:44 lib64
drwxr-xr-x   2 root root 4096 May 26 00:44 media
drwxr-xr-x   2 root root 4096 May 26 00:44 mnt
drwxr-xr-x   2 root root 4096 May 26 00:44 opt
```

```
dr-xr-xr-x 179 root root    0 Jun  6 02:15 proc
drwx------   2 root root 4096 May 26 00:45 root
drwxr-xr-x   1 root root 4096 Jun  5 21:20 run
drwxr-xr-x   1 root root 4096 Jun  5 21:20 sbin
drwxr-xr-x   2 root root 4096 May 26 00:44 srv
dr-xr-xr-x  13 root root    0 Jun  6 02:15 sys
drwxrwxrwt   2 root root 4096 May 26 00:45 tmp
drwxr-xr-x   1 root root 4096 May 26 00:44 usr
drwxr-xr-x   1 root root 4096 May 26 00:45 var
```

c) ¿Qué sucede si ejecuta el comando docker [container] run ubuntu /bin/bash (la opción container podría no indicarse)?

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

Notar que la opcion it se agrega para poder interactuar con bash y que no termine

i) El container se ejecuta "attacheando" al proceso bash en la terminal hasta que se cierre

ii) Se pierden el archivo creado ya que no se genera una imagen a partir del contenedor con el archivo y cuando se levanta de nuevo, otro contenedor distinto es creado, a partir de la imagen original

d) Ejecutar nuevamente un container con un bash en forma interactiva. ¿Existe el archivo creado en el punto anterior? ¿Por qué?

Respuesta en el item anterior.

e) Iniciar con el comando `docker start -ia id_container` el contenedor en el cual se creó el archivo

```
$ docker ps -a
CONTAINER ID    IMAGE   COMMAND     CREATED                 STATUS
56dfd3e5d54c    ubuntu  "/bin/bash" Less than a second ago  Exited (0)
```

```
$ docker start -ia 56dfd3e5d54c
root@56dfd3e5d54c:/# ls /
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run
sbin  so2018  srv  sys  tmp  usr  var
```

f) ¿Cuántos containers se están ejecutando actualmente? ¿En qué estado se encuentran cada uno de los containers ejecutados hasta el momento?

```
$ docker ps
CONTAINER ID    IMAGE   COMMAND     CREATED         STATUS
56dfd3e5d54c    ubuntu  "/bin/bash" 5 minutes ago   Up 3 seconds
```

Actualmente se encuentran ejecuntando 1 contenedor (sin cerrar el del punto e), en estado de Up

g) Eliminar todos los contenedor creados hasta el momento

```
$ docker ps --all
CONTAINER ID    IMAGE   COMMAND     CREATED          STATUS
56dfd3e5d54c    ubuntu  "/bin/bash" 8 minutes ago    Up 3 minutes
ce60f43ef611    ubuntu  "/bin/bash" 13 minutes ago   Exited (0) 13 minutes
ago
6f17c437952a    ubuntu  "ls -l"     14 minutes ago   Exited (0) 15 minutes
ago

$ docker kill 56dfd3e5d54c
56dfd3e5d54c

$ docker rm 56dfd3e5d54c ce60f43ef611 6f17c437952a
56dfd3e5d54c
ce60f43ef611
6f17c437952a
```

3) En los siguientes puntos se generará una nueva imagen a partir de un container:

```
$ docker run -it ubuntu /bin/bash
root@d46970a2a9bc:/# apt update -qq
root@d46970a2a9bc:/# apt install apache2 -qqy
^D
$ docker ps -alq
d46970a2a9bc
$ docker commit d46970a2a9bc
sha256:5653f784d68dcd813eef05f1de66aa7f785829174890718a9c1601edb81dd94a
$ docker images
REPOSITORY  TAG     IMAGE ID        CREATED                 SIZE
<none>      <none>  5653f784d68d    Less than a second ago  218MB
ubuntu      latest  113a43faa138    5 hours ago             81.2MB
$ docker tag 5653f784d68d apache2:v1
$ docker images
REPOSITORY  TAG     IMAGE ID        CREATED                 SIZE
apache2     v1      5653f784d68d    Less than a second ago  218MB
ubuntu      latest  113a43faa138    5 hours ago             81.2MB
$ docker run -d -v /apachedata:/var/www/html -p 8080:80 apache2:v1
/usr/sbin/apache2ctl -D FOREGROUND
b50d59b62d35006d1d3758835544aaa088a5889606b59373a654256fb381f5f0
```

g) No es necesario reiniciar el contenedor ya que el directorio esta montado

h) Porque sino el proceso termina y no queda esperando en modo background y por lo tanto el contenedor tambíen termina

4) En el siguiente punto se hará uso de un archivo Dockerfile para crear una imagen con similares características a la creada en el punto anterior:

```
# /apachedata/Dockerfile
FROM ubuntu:latest
RUN apt update -qq && apt install apache2 -qqy
ADD index.html /var/www/html/
EXPOSE 80
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

```
$ docker build -t apache2:so2018 /apachedata
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM ubuntu:latest
 ---> 113a43faa138
Step 2/5 : RUN apt update -qq && apt install apache2 -qqy
 ---> Running in 968a4e6b5812
 Removing intermediate container 968a4e6b5812
 ---> cda61e2bd50d
Step 3/5 : ADD index.html /var/www/html/
 ---> 20d8a6eb38b4
Step 4/5 : EXPOSE 80
 ---> Running in 657128a7ab3e
Removing intermediate container 657128a7ab3e
 ---> 37cc977fa4c3
Step 5/5 : CMD ["apache2ctl", "-D", "FOREGROUND"]
 ---> Running in 016a7b77f69a
Removing intermediate container 016a7b77f69a
 ---> 093c0b19cc3a
Successfully built 093c0b19cc3a
Successfully tagged apache2:so2018
$ docker run -d -p 8080:80 apache2:so2018
225111b0c9fcf8fb91bd04791e73903876b5eeb1c237bacc46875dfeee5053a2
```

5) ¿Qué es un union-filesystem? ¿Cuál/es y cómo los utiliza Docker?

UnionFS es un servicio de filesystem para linux que permite montar distintos filesystem en un esquema de capas superpuestas. Docker utiliza un sistema de archivos propio llamado Docker Storage para poder crear las imagenes basandose en la arquitectura por capas e implementar asi distintos tipos de persistencia, como también la reutilizacion de capas inferiores. Por defecto todo contenedor que se inicia, monta una ultima capa de ectura y escritura aparte a las de la imagen para usarla durante ejecución.

6) ¿Qué dirección IP tienen los containers? ¿De dónde la obtienen?

Los contenedores tienen una direccion ip interna mediante una red virtual orquestada por docker (por medio de configuraciones de usuario, o fallback a docker mismo), su interfaz puede ser de muchos tipos, aunque por defecto sera un bridge bajo la red 172.17.0.0/16

7) Describa brevemente los tipos de redes que se pueden utilizar en Docker. ¿Cuál piensa que Docker utiliza por defecto?

En parte contestada en la anterior, otros adaptadores posibles son **none** (sin configuracion de red) **host** (mismo networking que la maquina host), **container** (Usa el networking configurado por otro contenedor) y **NETWORK** (conecta el contenedor a una red creada por medio de docker)