

Trabajo Practico 2

Alumno: Joaquin Vietto Herrera

Clave: 2006111

Profesor: Ariel Schwindt

Carrera: Ingeniería en Sistemas

Año: 2023

Ejercicio 1

```
C:\Users\Joaquin>docker version
Client:
 Cloud integration: v1.0.29
 Version:          20.10.22
 API version:      1.41
 Go version:       go1.18.9
 Git commit:       3a2c30b
 Built:            Thu Dec 15 22:36:18 2022
 OS/Arch:          windows/amd64
 Context:          default
 Experimental:     true

Server: Docker Desktop 4.16.3 (96739)
Engine:
 Version:          20.10.22
 API version:      1.41 (minimum version 1.12)
 Go version:       go1.18.9
 Git commit:       42c8b31
 Built:            Thu Dec 15 22:26:14 2022
 OS/Arch:          linux/amd64
 Experimental:     false
containerd:
 Version:          1.6.14
 GitCommit:        9ba4b250366a5ddde94bb7c9d1def331423aa323
runc:
 Version:          1.1.4
 GitCommit:        v1.1.4-0-g5fd4c4d
docker-init:
 Version:          0.19.0
 GitCommit:        de40ad0
```

Ejercicio 2



joaquinpk [Edit profile](#)



Community User



Joined April 19, 2022

Ejercicio 3

```
C:\Users\Joaquin>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:3fbc632167424a6d997e74f52b878d7cc478225cffac6bc977eedfe51c7f4e79
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest

C:\Users\Joaquin>docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
redis               alpine         6c09b0364aa8   2 months ago   30.2MB
busybox             latest         a416a98b71e2   2 months ago   4.26MB
```

Ejercicio 4

```
C:\Users\Joaquin>docker run busybox
```

```
C:\Users\Joaquin>
```

La imagen "busybox" es una imagen muy liviana de Linux que proporciona un entorno mínimo. Si no se le proporciona un comando específico a ejecutar, el contenedor podría iniciarse y luego finalizar inmediatamente, ya que no tiene ningún proceso en ejecución. En otras palabras, se crea y se destruye muy rápidamente.

```
C:\Users\Joaquin>docker run busybox echo "Hola Mundo"
Hola Mundo
```

```
C:\Users\Joaquin>docker ps
```

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

```
C:\Users\Joaquin>
```

```
C:\Users\Joaquin>docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
ce4a99228e82	busybox	echo 'Hola Mundo'	About a minute ago	Exited (0)	About a minute ago
e7d6c9edaf58	busybox	sh	6 minutes ago	Exited (0)	5 minutes ago
49a440aee6a7	weaveworksdemos/user-db:0.4.0	/entrypoint.sh mong...	7 weeks ago	Exited (0)	9 minutes ago
af2ead8b69b	rabbitmq:3.6.8	docker-entrypoint.s...	7 weeks ago	Exited (143)	9 minutes ago
fbf2383be355	weaveworksdemos/user:0.4.4	/user -port=80	7 weeks ago	Exited (0)	9 minutes ago
56644290bfc1	weaveworksdemos/catalogue-db:0.3.0	docker-entrypoint.s...	7 weeks ago	Exited (0)	9 minutes ago
7f3a079217b3	mongo:3.4	docker-entrypoint.s...	7 weeks ago	Exited (0)	9 minutes ago
a7e899b219c5	weaveworksdemos/front-end:0.3.12	/usr/local/bin/npm ...	7 weeks ago	Exited (137)	9 minutes ago
4139efddb577	weaveworksdemos/load-test:0.1.1	/usr/local/bin/runL...	7 weeks ago	Exited (139)	7 weeks ago
038a1fb29725	weaveworksdemos/payment:0.4.3	/app -port=80	7 weeks ago	Exited (0)	9 minutes ago

La salida de **docker ps -a** proporciona información detallada sobre todos los contenedores en el sistema, lo que puede ser útil para el monitoreo y la gestión de contenedores.

Ejercicio 5

```
C:\Users\Joaquin>docker run -it busybox sh
/ #
/ # ps
PID   USER     TIME   COMMAND
1  root    0:00   sh
7  root    0:00   ps
/ # uptime
 22:56:54 up 13 min,  0 users,  load average: 0.09, 0.49, 0.54
/ # free
              total        used        free      shared  buff/cache   available
Mem:      10068472     1117020     1381884        1528       7569568     8641072
Swap:      3145728           0       3145728
/ # ls -l /
total 40
drwxr-xr-x  2 root   root    12288 Jul 17 18:30 bin
drwxr-xr-x  5 root   root     360 Oct 16 22:56 dev
drwxr-xr-x  1 root   root    4096 Oct 16 22:56 etc
drwxr-xr-x  2 nobody nobody   4096 Jul 17 18:30 home
drwxr-xr-x  2 root   root    4096 Jul 17 18:30 lib
lrwxrwxrwx  1 root   root      3 Jul 17 18:30 lib64 -> lib
dr-xr-xr-x 750 root   root      0 Oct 16 22:56 proc
drwx----- 1 root   root    4096 Oct 16 22:56 root
dr-xr-xr-x 11 root   root      0 Oct 16 22:56 sys
drwxrwxrwt  2 root   root    4096 Jul 17 18:30 tmp
drwxr-xr-x  4 root   root    4096 Jul 17 18:30 usr
drwxr-xr-x  4 root   root    4096 Jul 17 18:30 var
/ # exit
```

Ejercicio 6

```
C:\Users\Joaquin>docker ps -a
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
ce4a99228e82	busybox	romantic_bohr	"echo 'Hola Mundo'"	About a minute ago	Exited (0) About a minute ago	
e7d6c9edaf58	busybox	affectionate_stonebraker	"sh"	6 minutes ago	Exited (0) 5 minutes ago	
49a440aee6a7	weaveworksdemos/user-db:0.4.0	docker-compose-user-db-1	"/entrypoint.sh mong..."	7 weeks ago	Exited (0) 9 minutes ago	
af2eead8b69b	rabbitmq:3.6.8	docker-compose-rabbitmq-1	"docker-entrypoint.s..."	7 weeks ago	Exited (143) 9 minutes ago	
fbf2383be355	weaveworksdemos/user:0.4.4	docker-compose-user-1	"/user -port=80"	7 weeks ago	Exited (0) 9 minutes ago	
56644290bfc1	weaveworksdemos/catalogue-db:0.3.0	docker-compose-catalogue-db-1	"docker-entrypoint.s..."	7 weeks ago	Exited (0) 9 minutes ago	
7f3a079217b3	mongo:3.4	docker-compose-orders-db-1	"docker-entrypoint.s..."	7 weeks ago	Exited (0) 9 minutes ago	
a7e899b219c5	weaveworksdemos/front-end:0.3.12	docker-compose-front-end-1	"/usr/local/bin/npm ..."	7 weeks ago	Exited (137) 9 minutes ago	
4139efddb577	weaveworksdemos/load-test:0.1.1	docker-compose-user-sim-1	"/usr/local/bin/runL..."	7 weeks ago	Exited (139) 7 weeks ago	
038a1fb29725	weaveworksdemos/payment:0.4.3		"/app -port=80"	7 weeks ago	Exited (0) 9 minutes ago	

```
C:\Users\Joaquin>docker rm e7d6c9edaf58
e7d6c9edaf58
```

(No se borrarán los contenedores ya que hay algunos que utilizo actualmente)

Ejercicio 7

Clone a repository

GitHub.com

GitHub Enterprise

URL

Repository URL or GitHub username and repository
(hubot/cool-repo)

https://github.com/ingsoft3ucc/SimpleWebAPI.git

Local path
C:\Users\Joaquin\Desktop\SimpleWebAPI

Choose...

Clone

Cancel

```
C:\Users\Joaquin\Desktop\SimpleWebAPI>docker build -t mywebapi .
Sending build context to Docker daemon 123.4kB
Step 1/18 : FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
7.0: Pulling from dotnet/aspnet
e67fdae35593: Pull complete
0ab66724116f: Pull complete
14ccddeb1bc: Pull complete
5e265b51b431: Pull complete
3bda6efdfc5b: Pull complete
Digest: sha256:77ea66eb200fa06d2929cc937b75724bb2f5cdf2bcf6dc6aaf0851bb75bb631f
Status: Downloaded newer image for mcr.microsoft.com/dotnet/aspnet:7.0
--> b06452c8ee91
Step 2/18 : WORKDIR /app
--> Running in 615dc0ba52ab
Removing intermediate container 615dc0ba52ab
--> 641c704ac31e
Step 3/18 : EXPOSE 80
--> Running in 2f2171b11c7f
Removing intermediate container 2f2171b11c7f
--> 581531bea83c
Step 4/18 : EXPOSE 443
```

1. FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
 - Establece la imagen base para la construcción del contenedor. En este caso, se está utilizando la imagen `mcr.microsoft.com/dotnet/aspnet:7.0`, que es una imagen ASP.NET para .NET 7.0.
2. WORKDIR /app
 - Establece el directorio de trabajo actual dentro del contenedor en `/app`.
3. EXPOSE 80
 - Expone el puerto 80 del contenedor. Esto permite que otros contenedores se comuniquen con este contenedor a través del puerto 80.
4. EXPOSE 443
 - Expone el puerto 443 del contenedor. Esto generalmente se usa para conexiones seguras mediante HTTPS.
5. EXPOSE 5254
 - Expone el puerto 5254 del contenedor. Esto podría ser un puerto específico para la aplicación que se está ejecutando en el contenedor.
6. FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
 - Inicia una nueva etapa de construcción del contenedor basada en la imagen `mcr.microsoft.com/dotnet/sdk:7.0`, que es una imagen SDK de .NET para .NET 7.0.
7. WORKDIR /src
 - Establece el directorio de trabajo actual en `/src` en esta nueva etapa.
8. COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]
 - Copia el archivo de proyecto `SimpleWebAPI.csproj` desde la carpeta `SimpleWebAPI` del sistema de archivos del host al directorio de trabajo `/src/SimpleWebAPI` en el contenedor.
9. RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"
 - Ejecuta el comando `dotnet restore` para restaurar las dependencias del proyecto especificado en el archivo `SimpleWebAPI.csproj`.
10. COPY . .
 - Copia todo el contenido del directorio de trabajo actual del host al directorio de trabajo actual del contenedor (`/src/SimpleWebAPI`).

11. WORKDIR "/src/SimpleWebAPI"

- Cambia el directorio de trabajo actual al subdirectorio /src/SimpleWebAPI en el contenedor.

12. RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build

- Ejecuta el comando dotnet build para compilar el proyecto SimpleWebAPI.csproj en modo Release y se almacena en la carpeta /app/build en el contenedor.

13. FROM build AS publish

- Define una nueva etapa de construcción llamada publish basada en la etapa anterior build.

14. RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false

- Ejecuta el comando dotnet publish para publicar la aplicación en la carpeta /app/publish en el contenedor. La opción /p:UseAppHost=false indica que no se debe usar un host de aplicación.

15. FROM base AS final

- Inicia una nueva etapa llamada final basada en la etapa base, que fue definida al principio del Dockerfile.

16. WORKDIR /app

- Cambia el directorio de trabajo actual a /app en esta última etapa.

17. COPY --from=publish /app/publish .

- Copia el resultado de la etapa publish la carpeta /app/publish en el contenedor a la carpeta de trabajo actual en la etapa "final".

18. ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]

- Define el punto de entrada del contenedor. Cuando se inicia el contenedor, se ejecutará la aplicación SimpleWebAPI.dll utilizando el comando dotnet.

19. #CMD ["/bin/bash"]

- Este es un comentario en el Dockerfile.

```
C:\Users\Joaquin>docker image ls
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
mywebapi             latest       9a5d15e2798c   9 minutes ago  216MB
```

```
C:\Users\Joaquin>docker run --name mywebapi-container -p 8080:80 mywebapi
info: Microsoft.Hosting.Lifetime[14]
    Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
    Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
    Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
    Content root path: /app
```

```
C:\Users\Joaquin>docker tag mywebapi joaquinpk/mywebapi

C:\Users\Joaquin>docker push joaquinpk/mywebapi
Using default tag: latest
The push refers to repository [docker.io/joaquinpk/mywebapi]
4ac0d487136a: Pushed
c7c7e67ef1d1: Pushed
60d2e88f8a9a: Pushed
94709d31a668: Pushed
5056c4c3c1c6: Pushed
9a822c647cc9: Pushed
633f5bf471f7: Pushed
latest: digest: sha256:0d797038f9a752ebe9e6d9add457561d8934760caa8ea886ae70e594e22a4e8b size: 1790
```

Ejercicio 8

```
C:\Users\Joaquin>docker run --name myapi -d mywebapi
fb06722f7e2f132fd44fd769f5ca6f9d2ebf213c0288e6909faddb3d84a71e08

C:\Users\Joaquin>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
fb06722f7e2f   mywebapi   "dotnet SimpleWebAPI..." 28 seconds ago Up 28 seconds 80/tcp, 443/tcp, 5254/tcp          myapi

C:\Users\Joaquin>docker kill myapi
myapi

C:\Users\Joaquin>docker rm myapi
myapi

C:\Users\Joaquin>docker run --name myapi -d -p 80:80 -p 5254:5254 mywebapi
26d0c9ce30bc67f318a8a03e8b3e124bac414f55c676afab322ebbe32e7cca85
```

```
{{"date":"2023-10-17","temperatureC":-13,"temperatureF":9,"summary":"Scorching"},{"date":"2023-10-18","temperatureC":-5,"temperatureF":24,"summary":"Fresquito"},{"date":"2023-10-19","temperatureC":34,"temperatureF":93,"summary":"Fresco"},{"date":"2023-10-20","temperatureC":53,"temperatureF":127,"summary":"Fresquito"},{"date":"2023-10-21","temperatureC":-7,"temperatureF":20,"summary":"Mild"}}
```

Ejercicio 9

```
22 COPY --from=publish /app/publish .
23 #ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]
24 CMD ["/bin/bash"]
25
```

```
C:\Users\Joaquin\Desktop\SimpleWebAPI>docker build -t mywebapi .
Sending build context to Docker daemon 123.4kB
Step 1/18 : FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
--> b06452c8ee91
Step 2/18 : WORKDIR /app
--> Using cache
--> 641c704ac31e
Step 3/18 : EXPOSE 80
```

```
C:\Users\Joaquin\Desktop\SimpleWebAPI>docker run -it --rm -p 80:80 mywebapi
docker: Error response from daemon: driver failed programming external connectivity on endpoint funny_poincare (dc189b69e30d7cc4e3e86829bf3f492dcae3c20fc821c8a5b15002d668ba1c7e): Bind for 0.0.0.0:80 failed: port is already allocated.

C:\Users\Joaquin\Desktop\SimpleWebAPI>docker run -it --rm -p 80:80 mywebapi
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
```

```
C:\Users\Joaquin\Desktop\SimpleWebAPI>docker run -it --rm -p 80:80 mywebapi dotnet SimpleWebAPI.dll
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
```