

# TP2

Microsoft Windows [Versión 10.0.22621.1992]

(c) Microsoft Corporation. Todos los derechos reservados.

**C:\Users\Hp>docker version**

```
C:\Users\Hp>docker version
Client:
 Cloud integration: v1.0.35-desktop+001
 Version:          24.0.5
 API version:      1.43
 Go version:       go1.20.6
 Git commit:       ced0996
 Built:            Fri Jul 21 20:36:24 2023
 OS/Arch:          windows/amd64
 Context:          default
```

Client:

Cloud integration: v1.0.31

Version: 20.10.23

API version: 1.41

Go version: go1.18.10

Git commit: 7155243

Built: Thu Jan 19 17:43:10 2023

OS/Arch: windows/amd64

Context: default

Experimental: true

Server: Docker Desktop 4.17.1 (101757)

Engine:

Version: 20.10.23

API version: 1.41 (minimum version 1.12)

Go version: go1.18.10

Git commit: 6051f14

Built: Thu Jan 19 17:32:04 2023

OS/Arch: linux/amd64

Experimental: false

containerd:

Version: 1.6.18

GitCommit: 2456e983eb9e37e47538f59ea18f2043c9a73640

runc:

Version: 1.1.4

GitCommit: v1.1.4-0-g5fd4c4d

docker-init:

Version: 0.19.0

GitCommit: de40ad0

**C:\Users\Hp>docker pull busybox**

Using default tag: latest

latest: Pulling from library/busybox

3f4d90098f5b: Pull complete

Digest: sha256:3fbc632167424a6d997e74f52b878d7cc478225cffac6bc977eedfe51c7f4e79

Status: Downloaded newer image for busybox:latest

[docker.io/library/busybox:latest](https://docker.io/library/busybox:latest)

**C:\Users\Hp>docker images**

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

busybox	latest	a416a98b71e2	2 weeks ago	4.26MB
---------	--------	--------------	-------------	--------

```
C:\Users\Hp>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis               alpine             6c09b0364aa8       2 weeks ago        30.2MB
busybox             latest             a416a98b71e2       5 weeks ago        4.26MB
alexisfr/flask-app  latest             5f184752a58e       3 years ago        700MB
```

**C:\Users\Hp>docker run busybox**

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

Hola Mundo

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

Hola Mundo

**C:\Users\Hp>docker ps**

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

**C:\Users\Hp>docker ps -a**

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
60d6ef763a35	busybox	"echo 'Hola Mundo'"	52 seconds ago	Exited (0) 33 seconds ago
		naughty_matsumoto		

17531ab866af busybox "sh" About a minute ago Exited (0) About a minute ago flamboyant\_jennings

**C:\Users\Hp>docker run -it busybox sh**

/ #

/ # exit

**C:\Users\Hp>docker run -it busybox sh**

/ #

/ # ps

PID	USER	TIME	COMMAND
-----	------	------	---------

1	root	0:00	sh
---	------	------	----

7	root	0:00	ps
---	------	------	----

/ # uptime

18:42:24 up 7 min, 0 users, load average: 0.12, 0.12, 0.08

/ # free

	total	used	free	shared	buff/cache	available
--	-------	------	------	--------	------------	-----------

Mem:	12203320	612896	9704516	1520	1885908	11314160
------	----------	--------	---------	------	---------	----------

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	Aug 8 18:42	dev
------------	---	------	------	-----	-------------	-----

drwxr-xr-x	1	root	root	4096	Aug 8 18:42	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	502	root	root	0	Aug 8 18:42	proc
------------	-----	------	------	---	-------------	------

drwx-----	1	root	root	4096	Aug 8 18:42	root
-----------	---	------	------	------	-------------	------

dr-xr-xr-x	11	root	root	0	Aug 8 18:42	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

```

C:\Users\Hp>docker run -it busybox sh
/ # ps
PID   USER     TIME  COMMAND
    1  root         0:00  sh
    7  root         0:00  ps
/ # free
             total        used        free      shared  buff/cache   available
Mem:      12203320        700736       1110932         1576       391652       11257552
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 Aug 29 16:21 dev
drwxr-xr-x  1 root    root      4096 Aug 29 16:21 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 237 root    root         0 Aug 29 16:21 proc
drwx----- 1 root    root     4096 Aug 29 16:21 root
dr-xr-xr-x 11 root    root         0 Aug 29 16:21 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
/ #

```

**C:\Users\Hp>docker ps -a**

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
adf8a2168c2b	busybox	"sh"	59 seconds ago	Exited (0) 15 seconds ago
jovial_swanson				
05f53aa8747f	busybox	"sh"	About a minute ago	Exited (0) About a minute ago
eloquent_blackburn				
60d6ef763a35	busybox	"echo 'Hola Mundo'"	3 minutes ago	Exited (0) 2 minutes ago
naughty_matsumoto				
17531ab866af	busybox	"sh"	3 minutes ago	Exited (0) 3 minutes ago
flamboyant_jennings				

**C:\Users\Hp>docker rm elated\_lalande**

Error: No such container: elated\_lalande

**C:\Users\Hp>docker rm**

"docker rm" requires at least 1 argument.

See 'docker rm --help'.

Usage: docker rm [OPTIONS] CONTAINER [CONTAINER...]

Remove one or more containers

**C:\Users\Hp>docker rm flamboyant\_jennings**

flamboyant\_jennings

**C:\Users\Hp>docker rm \$(docker ps -a -q -f status=exited)**

unknown shorthand flag: 'a' in -a

See 'docker rm --help'.

**C:\Users\Hp>docker container prune**

WARNING! This will remove all stopped containers.

Are you sure you want to continue? [y/N] y

Deleted Containers:

adf8a2168c2be07df895c389f279d80e9731ad0d040b63cdeb79a62970f42938

05f53aa8747f437a5bb714311cbb7b2fcfd0e52bc6ed2819b78a380604596a42

60d6ef763a35756df5132467c24c6ca16d0aa215ac3ac72c32620838f5d13ebc

Total reclaimed space: 33B

```
PS D:\Santiago\Ingenieria en Sistemas\4to Año\2do Semestre\Ingenieria de Software III\Practico\ing-software-3\SimpleWebA
PI> docker buildx build -t mywebapi .
[+] Building 48.8s (18/18) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 810B                                0.0s
=> [internal] load .dockerignore                                  0.1s
=> => transferring context: 2B                                       0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0  3.9s
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0 3.8s
```

## Explicacion del DockerFile

```
FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
EXPOSE 5254

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /src
COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]
RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"
COPY . .
WORKDIR "/src/SimpleWebAPI"
RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build

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

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]
#CMD ["/bin/bash"]
```

1. `FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base`
  - Esta línea establece la imagen base para esta etapa del proceso de construcción. En este caso, se utiliza la imagen de ASP.NET 7.0 como base.
  - `AS base` le da un nombre a esta etapa, que se puede referenciar en otras partes del Dockerfile.
2. `WORKDIR /app`
  - Cambia el directorio de trabajo actual dentro del contenedor a `/app`. Esto significa que los comandos posteriores se ejecutarán en este directorio.
3. `EXPOSE 80`
  - Indica que el contenedor expone el puerto 80. Esto no realiza una publicación real del puerto, pero proporciona información a otros desarrolladores o administradores sobre qué puertos podría necesitar el contenedor.
4. `EXPOSE 443`
  - Similar al anterior, indica que el contenedor expone el puerto 443, comúnmente utilizado para conexiones HTTPS.
5. `EXPOSE 5254`
  - Este comando expone el puerto 5254. Puede ser un puerto personalizado específico para esta aplicación.
6. `FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build`
  - Establece otra imagen base para esta etapa, en este caso, la imagen SDK de .NET 7.0. Esta imagen contiene herramientas de desarrollo necesarias para compilar aplicaciones.
7. `WORKDIR /src`
  - Cambia el directorio de trabajo actual nuevamente, esta vez a `/src`.
8. `COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]`
  - Copia el archivo `SimpleWebAPI.csproj` del directorio `SimpleWebAPI/` del contexto de construcción al directorio actual en el contenedor (`/src`).
9. `RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"`
  - Ejecuta el comando `dotnet restore` para restaurar las dependencias del proyecto contenidas en el archivo `SimpleWebAPI.csproj`.
10. `COPY . .`

- Copia todo el contenido del contexto de construcción al directorio actual en el contenedor ( `/src` ).
11. `WORKDIR "/src/SimpleWebAPI"`
    - Cambia el directorio de trabajo actual al subdirectorio `SimpleWebAPI` dentro de `/src` .
  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 coloca los resultados en el directorio `/app/build` .
  13. `FROM build AS publish`
    - Define una nueva etapa 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 modo Release y almacena los archivos publicados en el directorio `/app/publish` . La opción `/p:UseAppHost=false` indica que no se utilizará un host de aplicación específico al publicar.
  15. `FROM base AS final`
    - Establece otra etapa llamada `final` basada en la etapa `base` .
  16. `WORKDIR /app`
    - Cambia el directorio de trabajo actual al directorio `/app` .
  17. `COPY --from=publish /app/publish .`
    - Copia los archivos publicados desde la etapa `publish` al directorio actual ( `/app` ) en esta etapa `final` .
  18. `ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]`
    - Establece el comando de entrada principal cuando se inicie el contenedor. En este caso, ejecuta el archivo `SimpleWebAPI.dll` utilizando el entorno de ejecución de .NET.
  19. `#CMD ["/bin/bash"]`
    - Este comando está comentado. Si se descomenta, establecería el comando por defecto que se ejecutaría al iniciar el contenedor en lugar del `ENTRYPOINT` . En este caso, ejecutaría el shell de bash dentro del contenedor.

```
PS C:\Users\Hp> docker run --name myapi -d mywebapi
4282a6679b7161fcbd4a7ceed3e43fa4458cc6224d5ac38ccbfaf76ca97ece082
```



```
Hp@DESKTOP-HLFCCT MINGW64 ~
$ docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/

Hp@DESKTOP-HLFCCT MINGW64 ~
$ docker tag mywebapi sanrivsal/mywebapi

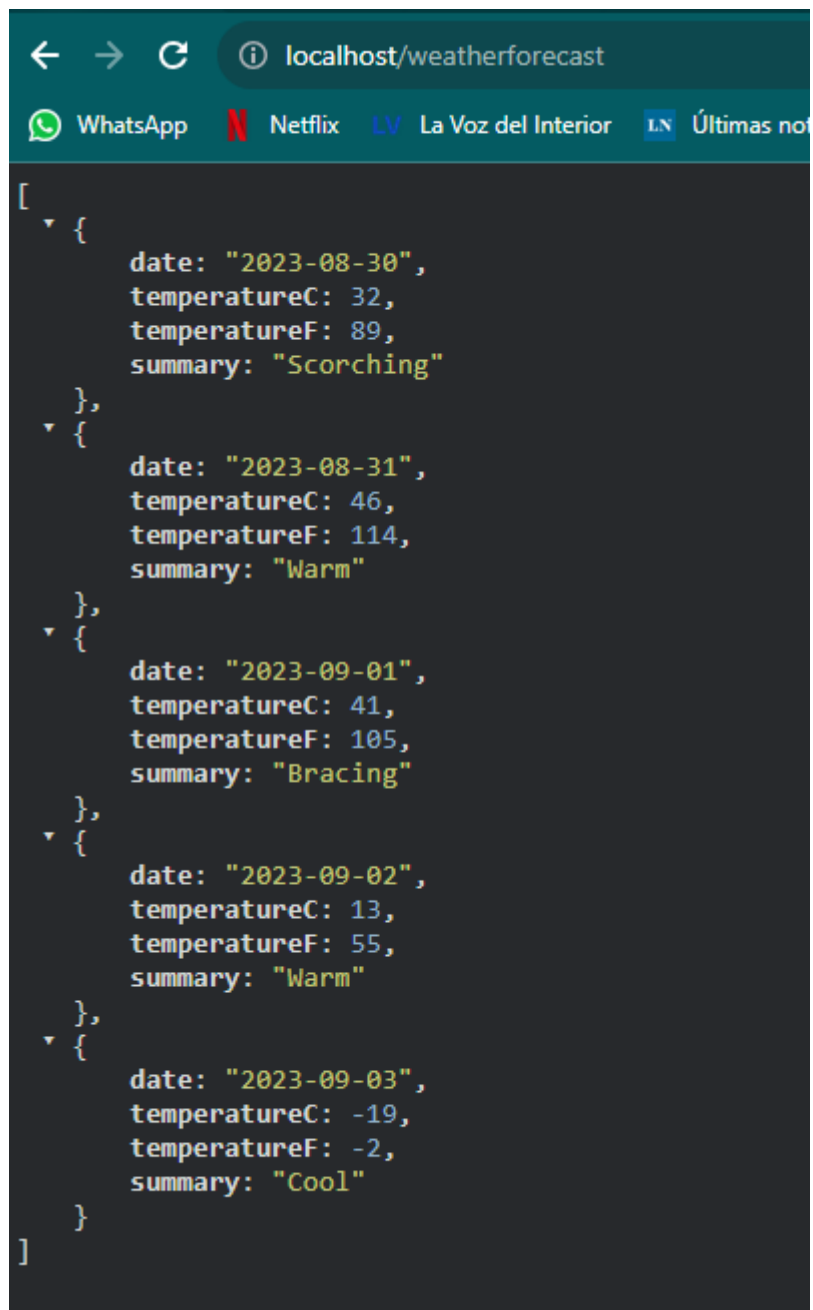
Hp@DESKTOP-HLFCCT MINGW64 ~
$ docker push sanrivsal/mywebapi
Using default tag: latest
The push refers to repository [docker.io/sanrivsal/mywebapi]
6a636481a036: Preparing
5f70bf18a086: Preparing
d8d80da8ed3d: Preparing
b260d977135d: Preparing
c65e011b79f6: Preparing
7af65d5c9e2a: Preparing
a5511d7cb706: Preparing
63290f9c9e52: Preparing
7af65d5c9e2a: Waiting
a5511d7cb706: Waiting
63290f9c9e52: Waiting
5f70bf18a086: Pushed
d8d80da8ed3d: Pushed
c65e011b79f6: Pushed
6a636481a036: Pushed
b260d977135d: Pushed
a5511d7cb706: Pushed
7af65d5c9e2a: Pushed
63290f9c9e52: Pushed
latest: digest: sha256:ae20544efcbd8897f838c1973e7f048885d8b66bfd97dee83bf7b305e1fba0c9 size: 1996
```

```
PS C:\Users\Hp> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
60553d3d346b   mywebapi  "dotnet SimpleWebAPI..." 6 minutes ago  Up 6 minutes  0.0.0.0:80->80/tcp, 0.0.0.0:5254->5254/tcp, 443/tcp
```

```
PS C:\Users\Hp> docker kill myapi
myapi
PS C:\Users\Hp> docker rm myapi
myapi
PS C:\Users\Hp> |
```

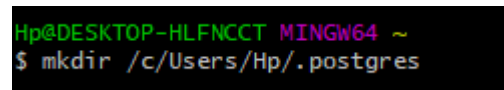
```
PS C:\Users\Hp> docker run --name myapi -d -p 80:80 -p 5254:5254 mywebapi
c73cf4b5c658b8c3c72c0afcc95d57cbcd596851739f71bfe1ee188afb102a95
```





A screenshot of a web browser window. The address bar shows 'localhost/weatherforecast'. The browser's bookmark bar includes 'WhatsApp', 'Netflix', 'La Voz del Interior', and 'Últimas not'. The main content area displays a JSON array of five weather forecast objects, each with 'date', 'temperatureC', 'temperatureF', and 'summary' fields. The data is as follows:

date	temperatureC	temperatureF	summary
2023-08-30	32	89	Scorching
2023-08-31	46	114	Warm
2023-09-01	41	105	Bracing
2023-09-02	13	55	Warm
2023-09-03	-19	-2	Cool



A screenshot of a terminal window. The prompt is 'Hp@DESKTOP-HLFNCCT MINGW64 ~'. The command entered is 'mkdir /c/Users/Hp/.postgres'.

```

$ docker run --name my-postgres -e POSTGRES_PASSWORD=mysecretpassword -v Hp/.postgres:/var/lib/postgresql/data -p 5432:5432 -d postgres:9.4
Unable to find image 'postgres:9.4' locally
9.4: Pulling from library/postgres
319014d83c02: Pulling fs layer
7ec0fe6664f6: Pulling fs layer
9ca7ba8f7764: Pulling fs layer
9e1155d037e2: Pulling fs layer
Febcfb7f8870: Pulling fs layer
9c78c79412b5: Pulling fs layer
3a35744405c5: Pulling fs layer
27717922e067: Pulling fs layer
36f0c5255550: Pulling fs layer
dbf0a396f422: Pulling fs layer

```

```

PS C:\Users\Hp> docker exec -it my-postgres /bin/bash
root@ab467154cecc:/# psql -h localhost -U postgres
psql (9.4.26)
Type "help" for help.

postgres=# \l
                                List of databases
  Name      | Owner   | Encoding | Collate  | Ctype    | Access privileges
-----+-----+-----+-----+-----+-----
 postgres   | postgres | UTF8      | en_US.utf8 | en_US.utf8 | 
 template0  | postgres | UTF8      | en_US.utf8 | en_US.utf8 | =c/postgres +
            |          |          |          |          | postgres=CTc/postgres
 template1  | postgres | UTF8      | en_US.utf8 | en_US.utf8 | =c/postgres +
            |          |          |          |          | postgres=CTc/postgres
(3 rows)

postgres=# create database test;
CREATE DATABASE
postgres=# \connect test
You are now connected to database "test" as user "postgres".
test=# create table tabla_a (mensaje varchar(50));
CREATE TABLE
test=# insert into tabla_a (mensaje) values('Hola mundo!');
INSERT 0 1
test=# select * from tabla_a;
      mensaje
-----
 Hola mundo!
(1 row)

```

Explicar que se logró con el comando *docker run* y *docker exec* ejecutados en este ejercicio

**docker run:** se crea y ejecuta un nuevo contenedor de Docker a partir de una imagen de PostgreSQL con una contraseña definida y un volumen para persistencia de datos. Además, el mapeo de puertos permite acceder al servidor PostgreSQL desde el host.

**docker exec:** Se interactúa con el contenedor de PostgreSQL en ejecución. Entramos en el entorno del contenedor y ejecutamos comandos adentro de él.