

TP2 – Docker

Docker es una herramienta que permite el despliegue de aplicaciones en contenedores. Además, provee una solución integrada tanto para la ejecución como para la creación de contenedores entre otras muchas funcionalidades.

```
MINGW64/c:/Users/BELU/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 ...
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker version
Client:
  Cloud integration: v1.0.28
  Version:          20.10.17
  API version:      1.41
  Go version:       go1.17.11
  Git commit:       100c701
  Built:            Mon Jun  6 23:09:02 2022
  OS/Arch:          windows/amd64
  Context:          default
  Experimental:     true

Server: Docker Desktop 4.11.1 (84025)
Engine:
  Version:          20.10.17
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.17.11
  Git commit:       a89b842
  Built:            Mon Jun  6 23:01:23 2022
  OS/Arch:          linux/amd64
  Experimental:     false
containerd:
  Version:          1.6.6
  GitCommit:        10c12954828e7c7c9b6e0ea9b0c02b01407d3ae1
runc:
  Version:          1.1.2
  GitCommit:        v1.1.2-0-ga916309
docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Obtener la imagen BusyBox

```






BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ 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

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
example-voting-app_result    latest             1c926a633706       8 days ago         266MB
example-voting-app_worker    latest             de453cb0460e       8 days ago         194MB
example-voting-app_vote      latest             6b0a1432b5da       8 days ago         164MB
postgres             15-alpine          ab8fb914369e       12 days ago        237MB
redis                 alpine             6c09b0364aa8       2 weeks ago        30.2MB
mywebapi              latest             0152193216f5       2 weeks ago        216MB
busybox               latest             a416a98b71e2       5 weeks ago        4.26MB
mongo                 latest             7e32c3979b02       2 months ago       653MB
solr                   latest             ee92e4df3117       2 months ago       590MB
dockersamples/examplevotingapp_result    latest             6ce23a8ce243       8 months ago       256MB
dockersamples/examplevotingapp_vote      latest             04e406d349f5       8 months ago       142MB
dockersamples/examplevotingapp_worker    latest             03edceb3a0f9       8 months ago       194MB
mysql                 latest             3842e9cdfd2        9 months ago       538MB
rabbitmq              3-management       6b94498c1b2f       11 months ago      262MB
mongo                 5.0                 ae98e14b339d       12 months ago      697MB
memcached             1.6.16             fdff4547b1b7       12 months ago      89.2MB
docker/getting-started latest             cb90f98fd791       16 months ago      28.8MB
alexisfr/flask-app    latest             5f184752a58e       3 years ago         700MB
mongo                 3.4                 f76f959b2a49       3 years ago         431MB
weaveworksdemos/load-test    0.1.1             3ab3fd5cd04e       6 years ago         570MB
weaveworksdemos/catalogue    0.3.5             0bd359b6d6e8       6 years ago         41.2MB
rabbitmq              3.6.8             8cdcbee37f62       6 years ago         179MB
weaveworksdemos/user        0.4.4             ab8af7050996       6 years ago         35.3MB
weaveworksdemos/payment     0.4.3             4f2c23055dcd       6 years ago         32.5MB
weaveworksdemos/front-end    0.3.12            b54402ef78a5       6 years ago         120MB
weaveworksdemos/shipping    0.4.8             4fc533e8180a       6 years ago         199MB
weaveworksdemos/carts       0.4.8             c00473736118       6 years ago         198MB
weaveworksdemos/orders      0.4.7             8275c5b9181b       6 years ago         198MB
weaveworksdemos/queue-master 0.3.1             76f0de7a12ac       6 years ago         179MB
weaveworksdemos/edge-router 0.1.1             584dac9095de       6 years ago         21.9MB
weaveworksdemos/catalogue-db 0.3.0             9d0c5eb88949       6 years ago         400MB
weaveworksdemos/user-db     0.4.0             196601f91030       6 years ago         717MB
```

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker run busybox

```

	NAME	IMAGE ↓	STATUS	PORT(S)	STARTED
	 gracious_rhodes 91305096b22e 	busybox:latest	Exited	-	 

Especificamos algún comando a correr dentro del contenedor:

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker run busybox echo "Hola Mundo"
Hola Mundo

```

Ver los contenedores ejecutados utilizando el comando *ps*. La opción *-a* en el comando *docker ps* significa "all" (todos), y se utiliza para mostrar una lista completa de todos los contenedores en lugar de solo los que están en estado "Up" (en ejecución).

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker ps -a

```

CONTAINER ID	IMAGE	PORTS	NAMES	COMMAND	CREATED	STATUS
3e9eb4402743	busybox			"echo 'Hola Mundo'"	About a minute ago	Exited (0)
About a minute ago			awesome_hopper			
91305096b22e	busybox			"sh"	9 minutes ago	Exited (0)
9 minutes ago			gracious_rhodes			

Ejecutamos en modo interactivo y asignamos un pseudoterminal para interactuar con el contenedor

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker run -it busybox sh
/ #
/ # ps
PID   USER     TIME   COMMAND
    1  root      0:00   sh
    7  root      0:00   ps
/ # uptime
 23:11:03 up 19 min,  0 users,  load average: 0.00, 0.05, 0.22
/ # free
              total        used        free      shared  buff/cache   available
Mem:        12967604        698836       10725300          1912       1543468       12009916
Swap:        4194304           0          4194304
/ # ls -l /
total 40
drwxr-xr-x  2 root    root          12288 Jul 17 18:30 bin
drwxr-xr-x  5 root    root           360 Aug 23 23:10 dev
drwxr-xr-x  1 root    root          4096 Aug 23 23:10 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 263 root    root           0 Aug 23 23:10 proc
drwx----- 1 root    root          4096 Aug 23 23:10 root
dr-xr-xr-x 11 root    root           0 Aug 23 23:10 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

```

Borramos los conetenedores terminados

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker rm suspicious_volhard
suspicious_volhard

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker rm awesome_hopper
awesome_hopper

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker rm gracious_rhodes
gracious_rhodes

```

Borro todos los contenedores que no están corriendo

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker rm $(docker ps -a -q -f status=exited)
4601009621d1
8f5b383251d2
17698d63bb5c
4d3c7ac483f0
87007e4364ca
a6d4a6e71395
176b22695544
312a20f6ad71
c7933bf85484
5e54664e7959
70f2cc6f7257
70a5a8c2cbdc
0383ef910fa3
a025bdec8201
8e9b3b62e3a1

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
```

A partir del código <https://github.com/ingsoft3ucc/SimpleWebAPI> crear imagen etiquetándola con un nombre. El punto final le indica a Docker que use el dir actual

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker build -t mywebapi .
[+] Building 5.6s (5/17)
=> => transferring dockerfile: 32B
0.0s
```

Dockerfile: Bloc de notas

Archivo Edición Formato Ver Ayuda

#See <https://aka.ms/containerfastmode> to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.

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"]

Dockerfile

FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base:

Se usa una imagen base de Docker que proporciona el entorno **ASP.NET**. Específicamente se usa la imagen *mcr.microsoft.com/dotnet/aspnet:7.0* como la etapa base. Esta etapa se llama **"base"**. El **AS** base asigna un nombre a esta etapa para que puedas hacer referencia a ella más adelante.

WORKDIR /app:

Cambia el directorio de trabajo dentro de esta etapa a **/app**. Esto significa que los comandos subsiguientes se ejecutarán en este directorio.

EXPOSE 80, EXPOSE 443, EXPOSE 5254:

Expone los puertos 80, 443 y 5254 en el contenedor. Esto no significa que los puertos estén directamente disponibles en el host, pero indica que la aplicación dentro del contenedor puede escuchar en estos puertos.

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build:

Se está comenzando una nueva etapa de construcción utilizando una imagen **SDK** de **.NET**. Esto se utiliza para compilar la aplicación.

WORKDIR /src:

Establece el directorio de trabajo a **/src**.

COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]:

Copia el archivo de proyecto **SimpleWebAPI.csproj** y el contenido del directorio **SimpleWebAPI/** al directorio de trabajo actual (**/src** en esta etapa).

RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj":

Ejecuta el comando **dotnet restore** para restaurar las dependencias de la aplicación.

COPY . .:

Copia todo el contenido del contexto (los archivos del proyecto y otros archivos) al directorio de trabajo actual.

WORKDIR "/src/SimpleWebAPI":

Cambia el directorio de trabajo a la ubicación de los archivos del proyecto dentro de la carpeta **SimpleWebAPI/**.

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

Compila la aplicación utilizando el comando **dotnet build**, especificando que se compilará en modo **Release** y se colocarán los archivos de salida en **/app/build**.

FROM build AS publish:

Inicia una nueva etapa llamada **"publish"** basada en la etapa **"build"** definida anteriormente.

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

Publica la aplicación utilizando el comando **dotnet publish**, nuevamente en modo **Release**, y coloca los archivos publicados en **/app/publish**. La opción **/p:UseAppHost=false** deshabilita la creación del archivo de host de la aplicación.

FROM base AS final:

Inicia una nueva etapa llamada **"final"** basada en la etapa **"base"** definida anteriormente.

WORKDIR /app:

Cambia el directorio de trabajo al directorio **/app**.

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

Copia los archivos publicados desde la etapa **"publish"** al directorio de trabajo actual.

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

Establece el punto de entrada de la aplicación cuando el contenedor se ejecuta. En este caso, la aplicación **.NET SimpleWebAPI.dll** se ejecutará utilizando el comando **dotnet**.

Imágenes disponibles

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker images -a
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mywebapi	latest	75e5f9f62c60	37 minutes ago	216MB
example-voting-app_result	latest	1c926a633706	8 days ago	266MB
example-voting-app_worker	latest	de453cb0460e	8 days ago	194MB
example-voting-app_vote	latest	6b0a1432b5da	8 days ago	164MB
postgres	15-alpine	ab8fb914369e	12 days ago	237MB
redis	alpine	6c09b0364aa8	2 weeks ago	30.2MB
<none>	<none>	0152193216f5	2 weeks ago	216MB
busybox	latest	a416a98b71e2	5 weeks ago	4.26MB
mongo	latest	7e32c3979b02	2 months ago	653MB
solr	latest	ee92e4df3117	2 months ago	590MB
dockersamples/examplevotingapp_result	latest	6ce23a8ce243	8 months ago	256MB
dockersamples/examplevotingapp_vote	latest	04e406d349f5	8 months ago	142MB
dockersamples/examplevotingapp_worker	latest	03edceb3a0f9	8 months ago	194MB
mysql	latest	3842e9cdfd2	9 months ago	538MB
rabbitmq	3-management	6b94498c1b2f	11 months ago	262MB
mongo	5.0	ae98e14b339d	12 months ago	697MB
memcached	1.6.16	fdff4547b1b7	12 months ago	89.2MB
docker/getting-started	latest	cb90f98fd791	16 months ago	28.8MB
alexisfr/flask-app	latest	5f184752a58e	3 years ago	700MB
mongo	3.4	f76f959b2a49	3 years ago	431MB
weaveworksdemos/load-test	0.1.1	3ab3fd5cd04e	6 years ago	570MB
weaveworksdemos/catalogue	0.3.5	0bd359b6d6e8	6 years ago	41.2MB
rabbitmq	3.6.8	8cdcbee37f62	6 years ago	179MB
weaveworksdemos/user	0.4.4	ab8af7050996	6 years ago	35.3MB
weaveworksdemos/payment	0.4.3	4f2c23055dcd	6 years ago	32.5MB
weaveworksdemos/front-end	0.3.12	b54402ef78a5	6 years ago	120MB
weaveworksdemos/shipping	0.4.8	4fc533e8180a	6 years ago	199MB
weaveworksdemos/carts	0.4.8	c00473736118	6 years ago	198MB
weaveworksdemos/orders	0.4.7	8275c5b9181b	6 years ago	198MB
weaveworksdemos/queue-master	0.3.1	76f0de7a12ac	6 years ago	179MB
weaveworksdemos/edge-router	0.1.1	584dac9095de	6 years ago	21.9MB
weaveworksdemos/catalogue-db	0.3.0	9d0c5eb88949	6 years ago	400MB
weaveworksdemos/user-db	0.4.0	196601f91030	6 years ago	717MB

Ejecutar un contenedor con nuestra imagen

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker run -d -p 8080:80 mywebapi
f5373579b32e413d1d034bc9dd412030394fd3079ca75ea85f78f18b0d06d5db
```

Showing 1 items

Search

NAME

IMAGE ↓

STATUS

PORT(S)

STARTED

nice_banach

f5373579b32e

mywebapi:latest

Running

8080

3 minutes ago

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ 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/

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker tag mywebapi belenaguilarv/mywebapi

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker push belenaguilarv/mywebapi
Using default tag: latest
The push refers to repository [docker.io/belenaguilarv/mywebapi]
1ffc6397357f: Pushed
5f70bf18a086: Pushed
2a7bc811171b: Pushed
b260d977135d: Pushed
c65e011b79f6: Pushed
7af65d5c9e2a: Pushed
a5511d7cb706: Pushed
63290f9c9e52: Pushed
latest: digest: sha256:17048348c418396f89e72a0404d46b14503f02d45d63a568e1e0b4789ac6f8e4 size: 1996

```

Showing 1 items

	NAME	IMAGE	STATUS	PORT(S)	STARTED	
	nice_banach f5373579b32e	belenaguilarv/mywebapi:latest	Running	8080	11 minutes ago	   

Puertos

```

Seleccionar MINGW64:/c/Users/BELU/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker run --name myapi -d mywebapi
6a6e004e7bf41b5601a83e37a8232a7f82bda95db9332303578c37d6f5cb9031

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
6a6e004e7bf4   mywebapi  "dotnet SimpleWebAPI..." 4 seconds ago  Up 3 seconds  80/tcp, 443/tcp, 5254/tcp          myapi

```

Paramos y removemos el contenedor

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker kill myapi
myapi

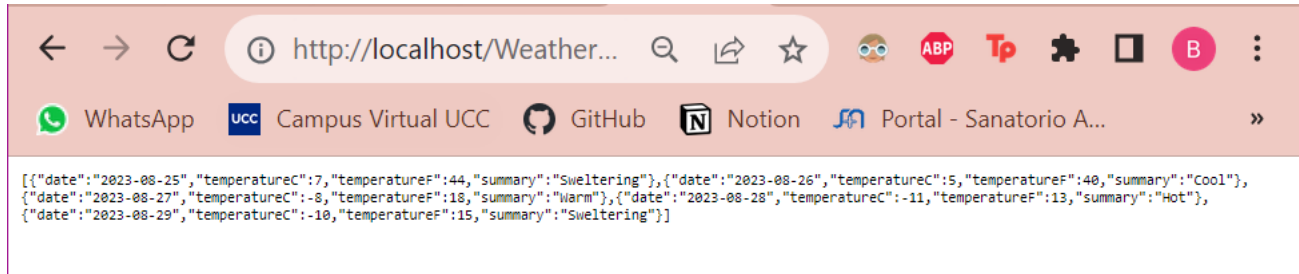
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker rm myapi
myapi

```


Intenté acceder a <http://localhost/WeatherForecast> pero no pasaba nada porque los puertos del contenedor no estaban expuestos ni mapeados en el sistema anfitrión. Entonces no puedo acceder a los servicios desde afuera del contenedor.

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker/SimpleWebAPI (main)
$ docker run --name myapi -d -p 80:80 -p 5254:5254 mywebapi
9f519d6bebfe4185297a76d175f49ca20008e63923af224e868e362bda48a9a0
```

Mapeamos los puertos del contenedor a los puertos del sistema anfitrión. El puerto 80 del contenedor se mapea al puerto 80 del host, y el puerto 5254 del contenedor se mapea al puerto 5254 del host.

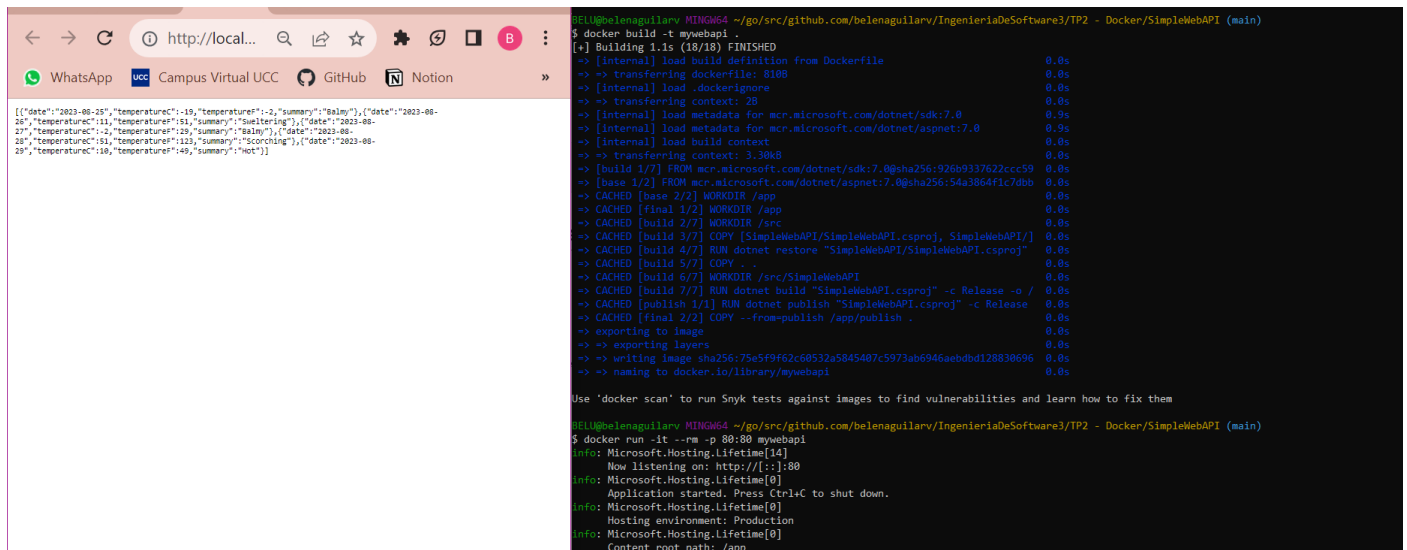


Ahora puedo acceder a los servicios del contenedor. Esto se debe a que los puertos están mapeados y expuestos en el host, lo que permite que los servicios sean accesibles desde fuera del contenedor.

Cuando se expone y mapea los puertos del contenedor al host, podemos acceder a los servicios dentro del contenedor a través del sistema anfitrión utilizando las direcciones y puertos adecuados. Esto es especialmente útil al interactuar con aplicaciones web u otros servicios que se ejecutan en contenedores desde mi compu o cualquier otra herramienta externa.

Modificar Dockerfile para soportar bash

Vimos que no se ejecuta automáticamente después de modificar el Dockerfile, volvemos a hacerlo como antes



Postgres

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ mkdir /c/Users/BELU/.postgres

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker run --name my-postgres -e POSTGRES_PASSWORD=mysecretpassword -v $HOME/.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
619014d83c02: Pull complete
7ec0fe6664f6: Pull complete
9ca7ba8f7764: Pull complete
9e1155d037e2: Pull complete
febcbf7f8870: Pull complete
8c78c79412b5: Pull complete
5a35744405c5: Pull complete
27717922e067: Pull complete
36f0c5255550: Pull complete
dbf0a396f422: Pull complete
ec4c06ea33e5: Pull complete
e8dd33eba6d1: Pull complete
51c81b3b2c20: Pull complete
2a03dd76f5d7: Pull complete
Digest: sha256:42a7a6a647a602efa9592edd1f56359800d079b93fa52c5d92244c58ac4a2ab9
Status: Downloaded newer image for postgres:9.4
64777253bd2aa6bd22cc9b094c59d7a12da0b7e7d6f95a2cd3601ff79900e3e4
```

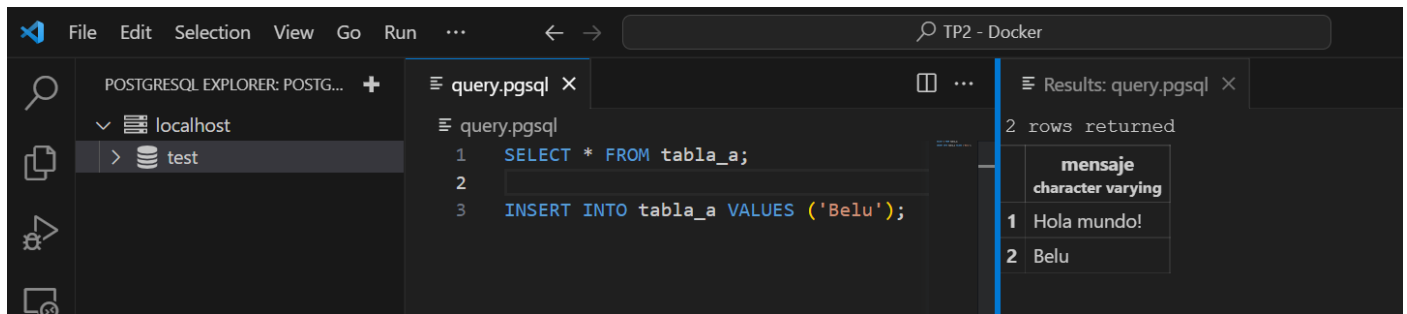
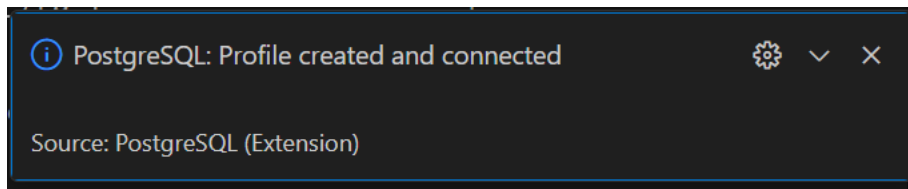
```
MINGW64:/c/Users/BELU/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2...
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP2 - Docker (main)
$ docker exec -it my-postgres psql -U postgres
psql (9.4.26)
Type "help" for help.

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=# \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
 test       | postgres | UTF8      | en_US.utf8 | en_US.utf8 | 
(4 rows)

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)

test=# \q
```

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.