


1)

```
ignacioachaval@Ignacios-MacBook-Pro ~ % docker version
Client:
 Cloud integration: v1.0.22
 Version:          20.10.13
 API version:      1.41
 Go version:       go1.16.15
 Git commit:       a224086
 Built:            Thu Mar 10 14:08:44 2022
 OS/Arch:          darwin/amd64
 Context:          default
 Experimental:     true

Server: Docker Desktop 4.6.1 (76265)
Engine:
 Version:          20.10.13
 API version:      1.41 (minimum version 1.12)
 Go version:       go1.16.15
 Git commit:       906f57f
 Built:            Thu Mar 10 14:06:05 2022
 OS/Arch:          linux/amd64
 Experimental:     false
containerd:
 Version:          1.5.10
 GitCommit:       2a1d4dbdb2a1030dc5b01e96fb110a9d9f150ecc
runc:
 Version:          1.0.3
 GitCommit:       v1.0.3-0-gf46b6ba
docker-init:
 Version:          0.19.0
 GitCommit:       de40ad0
```

2)




ignacioachaval94 [Edit profile](#)
Community User Joined October 18, 2023

Repositories

Starred

Contributed



No repositories
This profile does not have any public repositories

3)

```
ignacioachaval@Ignacios-MacBook-Pro ~ % 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
```

```
ignacioachaval@Ignacios-MacBook-Pro ~ % docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
jenkins-with-dotnetcore  latest             9c8ae4841c60       4 weeks ago        1.27GB
jenkins/jenkins       lts-jdk11          2a8f0d8e7b28       7 weeks ago        478MB
busybox               latest             a416a98b71e2       3 months ago       4.26MB
```

4)

Se corrió el comando "docker run busybox" y no se obtuvo ningun resultado ya que si bien tenemos la imagen, no esta creado el contenedor.

```
ignacioachaval@Ignacios-MacBook-Pro ~ % docker run busybox echo "Hola Mundo"
Hola Mundo
ignacioachaval@Ignacios-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
ignacioachaval@Ignacios-MacBook-Pro ~ % docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED      STATUS      PORTS   NAMES
c01e8b6a28c8   busybox   "echo 'Hola Mundo'"   32 seconds ago   Exited (0) 31 seconds ago   beautiful_torvalds
ae7b03dfa3ca   busybox   "sh"      2 minutes ago   Exited (0) 2 minutes ago   musing_hertz
ignacioachaval@Ignacios-MacBook-Pro ~ %
```

Con este comando se listan los contenedores levantados actualmente

5)

```
ignacioachaval@Ignacios-MacBook-Pro ~ % docker run -it busybox sh
/ #
/ # ps
PID   USER     TIME  COMMAND
    1   root      0:00   sh
    9   root      0:00   ps
/ # uptime
 02:53:05 up 28 min,  0 users,  load average: 0.00, 0.00, 0.00
/ # free
             total        used        free      shared  buff/cache   available
Mem:      4029192       213656      3198748       335920       616788      3259612
Swap:      1048572           0       1048572
/ # ls -l /
total 40
drwxr-xr-x  2 root    root          12288 Jul 17 18:30 bin
drwxr-xr-x  5 root    root           360 Oct 18 02:39 dev
drwxr-xr-x  1 root    root          4096 Oct 18 02:39 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 171 root    root           0 Oct 18 02:39 proc
drwx----- 1 root    root          4096 Oct 18 02:52 root
dr-xr-xr-x 13 root    root           0 Oct 18 02:39 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
ignacioachaval@Ignacios-MacBook-Pro ~ %
```

6)

```
/ # exit
ignacioachaval@Ignacios-MacBook-Pro ~ % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS          NAMES
7d5de4064fbc   busybox   "sh"                    About an hour ago   Exited (0) About an hour ago   nervous_goldstine
c01e8b6a28c8   busybox   "echo 'Hola Mundo'"    About an hour ago   Exited (0) About an hour ago   beautiful_torvalds
ae7b03dfa3ca   busybox   "sh"                    About an hour ago   Exited (0) About an hour ago   musing_hertz
ignacioachaval@Ignacios-MacBook-Pro ~ % docker rm musing_hertz
musing_hertz
ignacioachaval@Ignacios-MacBook-Pro ~ % docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
7d5de4064fbc1cf62f9f7d4f186589ca1c406185e2f2388b2df33e2a9ed4249d
c01e8b6a28c85f22bfa5cf7e8d19a4a465d7292c3e6b91896f755de73d11f68
Total reclaimed space: 28B
```

7)

```
built an image from a Dockerfile
ignacioachaval@Ignacios-MacBook-Pro: SimpleWebAPI % docker build -t mywebapi .
[+] Building 182.4s (18/18) FINISHED
=> [internal] load build definition from Dockerfile                                0.1s
=> => transferring dockerfile: 815B                                              0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0            2.9s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0               3.0s
=> [internal] load build context                                                0.1s
=> => transferring context: 80.44kB                                             0.0s
=> [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:d0755fbf1ed34aec79c127fd1dd01b80587acdb8ff50e5a68d1adf8b076922b 138.9s
=> => resolve mcr.microsoft.com/dotnet/sdk:7.0@sha256:d0755fbf1ed34aec79c127fd1dd01b80587acdb8ff50e5a68d1adf8b076922b 0.0s
=> => sha256:d0755fbf1ed34aec79c127fd1dd01b80587acdb8ff50e5a68d1adf8b076922b 1.79kB / 1.79kB 0.0s
=> => sha256:0ab66724116f089079aab4d1403ce98586595f13ca4b4163005ec3733a5dab94 14.97MB / 14.97MB 9.7s
=> => sha256:fbf7eed1efc4cd2571e40f0c34949a09474f4a96594b0501b4e24b7f0767bcb 2.01kB / 2.01kB 0.0s
=> => sha256:bd36eac352d8b2d27a63f4714b9063fc8c7e9c615e430006ddeb7e3448cd5780 5.28kB / 5.28kB 0.0s
=> => sha256:e67fdae3559346105027c63e7fb032bba57e62b1fe9f2da23e6f6dfb56384e00b 31.42MB / 31.42MB 28.0s
=> => sha256:14cdddebb1bc93dd113768cf39770236e666888c5d08fc5a554d2e47a0fe930e 32.46MB / 32.46MB 23.8s
=> => sha256:5e265b51b431d80b30bffa2e2f867af78d559418b418ea7b3edffaead2221b244 156B / 156B 11.0s
=> => sha256:3bda6efdfc5b92b1a9760092efa9eb1a4f50a0374789af843c6842acaf848b8f 10.12MB / 10.12MB 19.9s
=> => sha256:9a6d473c86f69f795e0ff2b6014ed32f2c9b35e743068b430a4df1c1448176b 25.38MB / 25.38MB 37.6s
=> => sha256:5464f27bea88b4d6d19d0c0e15a62a967f65d74b9753611d0a174526bc08b1da 180.99MB / 180.99MB 113.5s
=> => sha256:3cc0e9253374a2ce8dd07c45550ebb8a207cc9898aa397219369e970962f7ab6 13.98MB / 13.98MB 43.0s
=> => extracting sha256:e67fdae3559346105027c63e7fb032bba57e62b1fe9f2da23e6f6dfb56384e00b 150.7s
=> => extracting sha256:0ab66724116f089079aab4d1403ce98586595f13ca4b4163005ec3733a5dab94 141.5s
=> => extracting sha256:14cdddebb1bc93dd113768cf39770236e666888c5d08fc5a554d2e47a0fe930e 139.1s
=> => extracting sha256:3bda6efdfc5b92b1a9760092efa9eb1a4f50a0374789af843c6842acaf848b8f 134.0s
=> => extracting sha256:9a6d473c86f69f795e0ff2b6014ed32f2c9b35e743068b430a4df1c1448176b 6.3s
=> => extracting sha256:5464f27bea88b4d6d19d0c0e15a62a967f65d74b9753611d0a174526bc08b1da 23.5s
=> => extracting sha256:3cc0e9253374a2ce8dd07c45550ebb8a207cc9898aa397219369e970962f7ab6 1.3s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:77ea66eb200fa06d2929cc937b75724bb2f5cdf2bcf6dc6aaf0851bb75bb631f 47.1s
=> => resolve mcr.microsoft.com/dotnet/aspnet:7.0@sha256:77ea66eb200fa06d2929cc937b75724bb2f5cdf2bcf6dc6aaf0851bb75bb631f 0.0s
=> => sha256:b06452c8ee916a8c0a9863aad05f86b728ec72a1e43bcfbd8e7bd304953285a1 2.36kB / 2.36kB 0.0s
=> => sha256:e67fdae3559346105027c63e7fb032bba57e62b1fe9f2da23e6f6dfb56384e00b 31.42MB / 31.42MB 28.0s
=> => sha256:77ea66eb200fa06d2929cc937b75724bb2f5cdf2bcf6dc6aaf0851bb75bb631f 1.79kB / 1.79kB 0.0s
=> => sha256:4cecfef3ad7531e38b0915ad8f4e297a888d260634162f899e7bc83f25e2ef 1.37kB / 1.37kB 0.0s
=> => sha256:0ab66724116f089079aab4d1403ce98586595f13ca4b4163005ec3733a5dab94 14.97MB / 14.97MB 9.7s
=> => sha256:14cdddebb1bc93dd113768cf39770236e666888c5d08fc5a554d2e47a0fe930e 32.46MB / 32.46MB 23.8s
=> => sha256:5e265b51b431d80b30bffa2e2f867af78d559418b418ea7b3edffaead2221b244 156B / 156B 11.0s
```

8)

Utiliza una imagen base de ASP.NET Core para la versión 7.0
FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base

Establece el directorio de trabajo en /app
WORKDIR /app

Expone los puertos 80, 443 y 5254
EXPOSE 80
EXPOSE 443
EXPOSE 5254

Define una nueva etapa llamada "build" que utiliza una imagen SDK de .NET Core 7.0
FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build

Establece el directorio de trabajo en /src
WORKDIR /src

Copia el archivo de proyecto SimpleWebAPI.csproj al directorio de trabajo
COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]

Ejecuta dotnet restore para restaurar las dependencias del proyecto
RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"

Copia todos los archivos del proyecto al directorio de trabajo actual (/src)
COPY . .

Cambia el directorio de trabajo a /src/SimpleWebAPI

WORKDIR "/src/SimpleWebAPI"

Compila la aplicación en modo Release y la guarda en /app/build

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

Define una nueva etapa llamada "publish"

FROM build AS publish

Publica la aplicación en modo Release y la guarda en /app/publish

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

Define una nueva etapa llamada "final" que utiliza la etapa "base" como base

FROM base AS final

Cambia el directorio de trabajo a /app

WORKDIR /app

Copia los archivos publicados desde la etapa "publish" al directorio actual

COPY --from=publish /app/publish .

Establece el comando de entrada para el contenedor para iniciar la aplicación

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

```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
mywebapi             latest          350c60fdbbaa   4 minutes ago   216MB
jenkins-with-dotnetcore latest          9c8ae4841c60   4 weeks ago     1.27GB
jenkins/jenkins      lts-jdk11      2a8f0d8e7b28   7 weeks ago     478MB
busybox              latest          a416a98b71e2   3 months ago    4.26MB
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run -d -p 8080:80 mywebapi
f30cb9e70d7407cdea7ac682dd9e3e0bc720366ff93419424ee1bd8acbff8778
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker tag mywebapi ignacioachaval94/mywebapi:latest

ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % 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: ignacioachaval94
Password:
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/
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker tag mywebapi ignacioachaval94/mywebapi:latest
```

8)

```
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run --name myapi -d mywebapi
b9b08c79c98a21afa9a3c6ea60498f393d888bdb21f394a69a268065655bd6d0
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker kill myapi
myapi
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker rm myapi
myapi
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run --name myapi -d -p 80:80 -p 5254:5254 mywebapi
01a53f42949c0d25b24d764d137afc26b3653fa1c5eca8b7a4f583caf3f1a675
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI %
```

levanta <http://localhost/WeatherForecast> que al acceder devuelve datos del clima

No logra levantar <http://localhost/swagger/index.html>

9)

```
#See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build images for container fast mode

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"]
```

```
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker build -t mywebapi .

[+] Building 20.7s (18/18) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 816B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0                 1.3s
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0              1.3s
=> [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:d0755fbf1ed34aecd79c127fd1dd01b80587acdb8ff50e5a68d1adf8b076922b 0.0s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:77ea66eb200fa06d2929cc937b75724bb2f5cdf2bcf6dc6aaf0851bb75bb631f 0.0s
=> [internal] load build context                                                  0.0s
=> => transferring context: 3.50kB                                                0.0s
=> CACHED [build 2/7] WORKDIR /src                                                0.0s
=> CACHED [build 3/7] COPY [SimpleWebAPI/SimpleWebAPI.csproj, SimpleWebAPI/]     0.0s
=> CACHED [build 4/7] RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"      0.0s
=> [build 5/7] COPY . .                                                           0.1s
=> [build 6/7] WORKDIR /src/SimpleWebAPI                                          0.0s
=> [build 7/7] RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build   13.8s
=> [publish 1/1] RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false 5.1s
=> CACHED [base 2/2] WORKDIR /app                                                0.0s
=> CACHED [final 1/2] WORKDIR /app                                               0.0s
=> CACHED [final 2/2] COPY --from=publish /app/publish .                       0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:cf4f9927e467cc7309cfd13779ad21da60ed4cac08a94aa2d35334d87dc0ff3f 0.0s
=> => naming to docker.io/library/mywebapi                                       0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run -it --rm -p 80:80 mywebapi
```

```

use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run --name myapi -d -p 80:80 -p 5254:5254 mywebapi
docker: Error response from daemon: Conflict. The container name "/myapi" is already in use by container "b4a2e9c64ad962eb00ac003f412efdd4a511edbe5ed2a1b703e2f31f489170e8". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % dotnet SimpleWebAPI.dll
Could not execute because the specified command or file was not found.
Possible reasons for this include:
 * You misspelled a built-in dotnet command.
 * You intended to execute a .NET program, but dotnet-SimpleWebAPI.dll does not exist.
 * You intended to run a global tool, but a dotnet-preferred executable with this name could not be found on the PATH.
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run -it --rm -p 80:80 -v /Users/miuser/temp:/var/tmp mywebapi
docker: Error response from daemon: driver failed programming external connectivity on endpoint suspicious_haslett (13ca470b4f6cb6dac4bf6339771430d65231aa9065b6789e3a54bc3200e0313d): Bind for 0.0.0.0:80 failed: port is already allocated.
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run -it --rm -p 80:81 -v /Users/miuser/temp:/var/tmp mywebapi
docker: Error response from daemon: driver failed programming external connectivity on endpoint lucid_rosalind (fe97ec12760611b25dbab83dc27870433f3643562dcbf29fd3a2f657abd3361): Bind for 0.0.0.0:80 failed: port is already allocated.
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run -it --rm -p 81:80 -v /Users/miuser/temp:/var/tmp mywebapi
docker: Error response from daemon: error while creating mount source path '/Users/miuser/temp': mkdir /Users/miuser: operation not permitted.
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker kill myapi
myapi
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker rm myapi
myapi
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI %

```

10)

```

ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % docker run -it --rm -p 80:80 -v /Users/ignacioachaval/tmp:/var/tmp 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
ls -l /var/tmp

```

```

ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % ls -l /var/tmp

total 152
drwxr-xr-x  3 root      wheel   96 Jul  1 03:27 aud
-rw-r--r--  1 root      wheel 24576 Oct  5 11:20 bc3902d8132f43e3ae086a009979fa88.db
-rw-r--r--  1 root      wheel 32768 Oct 17 15:28 bc3902d8132f43e3ae086a009979fa88.db-shm
-rw-r--r--  1 root      wheel  4152 Oct 17 15:28 bc3902d8132f43e3ae086a009979fa88.db-wal
-rw-r--r--  1 root      wheel   51 Oct  5 10:28 bc3902d8132f43e3ae086a009979fa88.db.ses
srwxrwxrwx  1 root      wheel    0 Oct 12 09:00 com.cleverfiles.cfbackd.chief
-rw-rw-rw-  1 root      wheel    3 Oct 12 09:00 com.cleverfiles.cfbackd.pid
srw-r--r--  1 ignacioachaval wheel    0 Oct 12 09:17 filesystemui.socket
drwxr-xr-x  2 root      wheel    64 Sep 19 20:37 kernel_panics
-rw-r--r--  1 root      wheel    0 Oct  5 10:28 mat-debug-12570.log
-rw-r--r--  1 root      wheel  1253 Oct 18 02:35 mat-debug-4874.log
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI %

```

```

ignacioachaval@Ignacios-MacBook-Pro ~ % touch /var/tmp/hola.txt

```

```

ignacioachaval@Ignacios-MacBook-Pro tmp % ls
aud                                bc3902d8132f43e3ae086a009979fa88.db.ses hola.txt
bc3902d8132f43e3ae086a009979fa88.db com.cleverfiles.cfbackd.chief             kernel_panics
bc3902d8132f43e3ae086a009979fa88.db-shm com.cleverfiles.cfbackd.pid                mat-debug-12570.log
bc3902d8132f43e3ae086a009979fa88.db-wal filesystemui.socket                       mat-debug-4874.log
ignacioachaval@Ignacios-MacBook-Pro tmp %

```

```

ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % mkdir $HOME/.postgres
ignacioachaval@Ignacios-MacBook-Pro SimpleWebAPI % 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: Extracting [=====>] 16.29MB/22.52MB
7ee0fe6664f6: Download complete
9ca7ba8f7764: Download complete
9e1155d037e2: Download complete
febcbf7f8870: Download complete
8c78c79412b5: Download complete
5a35744405c5: Download complete
27717922e067: Download complete
36f0c5255550: Downloading [=====>] 16.11MB/53.62MB
dbf0a396f422: Download complete
ec4c06ea33e5: Download complete
e8dd33eba6d1: Download complete
51c81b3b2c20: Download complete
2a03dd76f5d7: Waiting

```

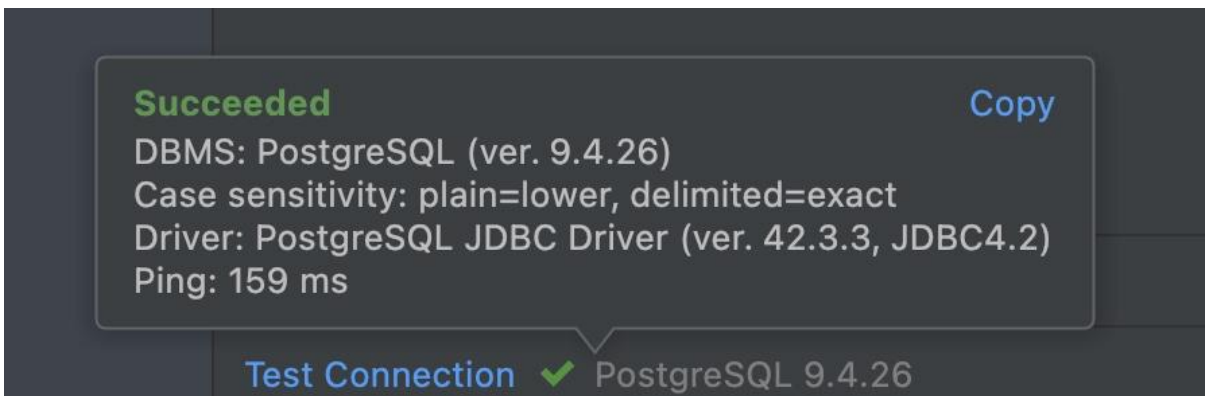
```

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)

```



Los comandos `docker run` y `docker exec` son esenciales en la gestión de contenedores Docker y se utilizan para diferentes propósitos:

1. **`docker run`**:

- **Creación y Ejecución de Contenedores**: El comando `docker run` se utiliza principalmente para crear y ejecutar nuevos contenedores a partir de una imagen de Docker. Puede ejecutar un contenedor en modo interactivo (`-it`) o en segundo plano según sus necesidades.

- **Mapeo de Puertos**: Con la opción `-p`, como en `docker run -p 80:80`, se pueden mapear puertos del host a puertos del contenedor. Esto permite que las aplicaciones en el contenedor se comuniquen con el mundo exterior a través de los puertos mapeados en el host.

- **Montaje de Volúmenes**: La opción `-v`, como en `docker run -v /ruta/host:/ruta/contenedor`, permite montar volúmenes desde el host al contenedor. Esto es útil para compartir datos o archivos entre el host y el contenedor.

- **Administración de Recursos**: Puede especificar la cantidad de CPU, memoria y otros recursos disponibles para el contenedor utilizando opciones como `--cpu` y `--memory`.

- **Limpieza de Contenedores**: La opción `--rm` puede utilizarse para eliminar automáticamente el contenedor después de que termine su ejecución.

2. **docker exec**:

- **Ejecución de Comandos en Contenedores en Ejecución**: El comando `docker exec` se utiliza para ejecutar comandos en contenedores en ejecución. A diferencia de `docker run`, que inicia un nuevo contenedor, `docker exec` se utiliza para interactuar con contenedores que ya están en funcionamiento.

- **Acceso a la Shell del Contenedor**: Puede utilizar `docker exec -it <contenedor_id> /bin/sh` (o `/bin/bash` u otra shell según la imagen) para acceder a una shell interactiva dentro de un contenedor en ejecución. Esto es útil para realizar tareas de depuración o mantenimiento en el contenedor.

En el ejercicio que mencionaste, `docker run` se utilizó para crear y ejecutar un contenedor a partir de la imagen `mywebapi`. Este contenedor se ejecutó en modo interactivo (`-it`), mapeó el puerto 80 del host al puerto 80 del contenedor y montó un volumen desde el host al contenedor.

Luego, `docker exec` se utilizó para ejecutar comandos dentro del contenedor en ejecución, lo que permitió listar el contenido de un directorio y crear un archivo en el contenedor. Esto demuestra cómo `docker exec` facilita la interacción con un contenedor que ya está en funcionamiento, lo que es útil para realizar tareas específicas dentro del entorno del contenedor sin necesidad de crear uno nuevo.