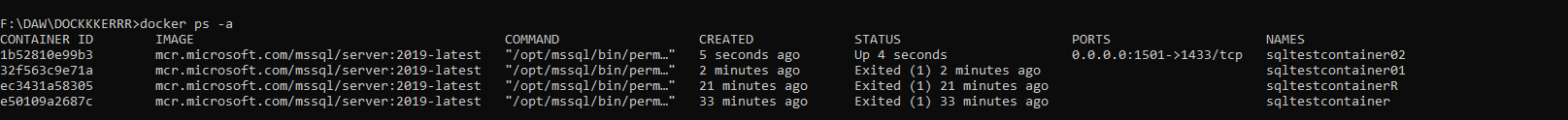
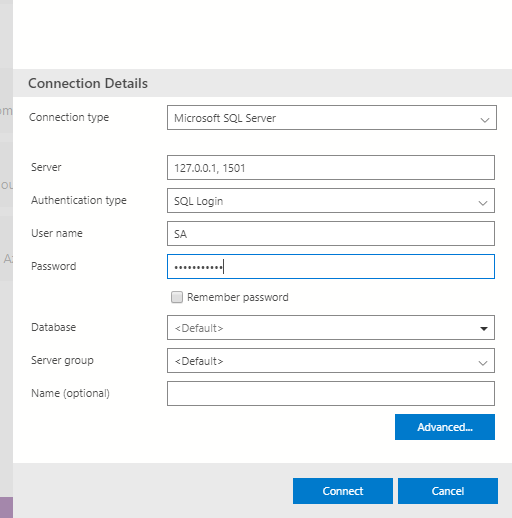
LAB3:

Ejercicio 3

- Ejecuta un contenedor de SQL Server 2019.

- Conéctate a él a través de Azure Data studio o SSMS.





COMANDOS: docker pull mcr.microsoft.com/mssql/server:2019-latest

docker run -e 'ACCEPT\_EULA=Y' -e 'SA\_PASSWORD=Pa$$w0rdSQL' -p 1500:1433 --name sqltestcontainer -d mcr.microsoft.com/mssql/server:2019-latest

Ejercicio 4

- Ejecuta un contenedor de un servidor web.

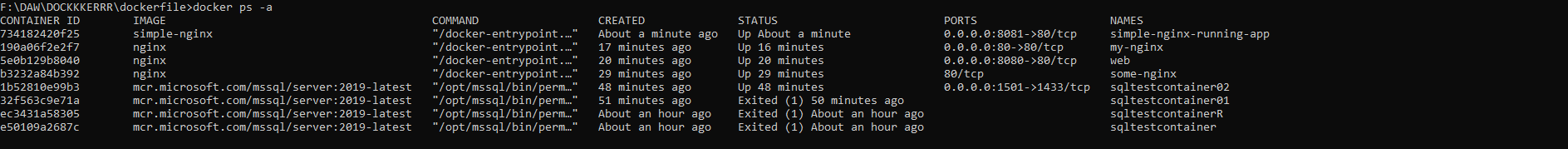
- Aloja en el servidor una web y consúmela a través del navegador.

- Se puede usar la imagen que se desee: nginx; apache; halverneus/static-file-

serve (opción más fácil).

-docker container run nginx

**docker** **container** **run** **--name** **my-nginx** **-p** 80:80 **nginx**



Abrimos el navegador <http://127.0.0.1/> y podemos comprobar que efectivamente se está ejecutando el servidor Web:

**Sesión interactiva**

Ejecutamos el contenedor de forma desacoplada para poder seguir usando la línea de comandos con el parámetro -d, podemos comprobar después que esta ejecutandose con docker ps:

PS> docker run --name docker-nginx-01 -p 80:80 -d nginx

PS> docker **exec** -it 90c5b531d19b bash

root@90c5b531d19b:/#

## Mi primer sitio Web

Fichero Dockerfile para crear la imagen:

FROM nginx:alpine

COPY **index**.html /usr/share/nginx/html/**index**.html

Creo un index.html personalizado en mi ordenador local:

<!DOCTYPE html>

<**html** lang="en">

<**head**>

<**meta** charset="UTF-8" />

<**meta** name="viewport" content="width=device-width, initial-scale=1.0" />

<**meta** http-equiv="X-UA-Compatible" content="ie=edge" />

<**title**>Hello World - Nginx Docker</**title**>

<**style**>

**h1** {

font-weight: lighter;

font-family: Arial, Helvetica, sans-serif;

}

</**style**>

</**head**>

<**body**>

<**h1**>

Hello World

</**h1**>

</**body**>

</**html**>

Crea una imagen a partir de un Dockerfile, la opción -t (o –tag) añade un tag opcional, el último “.” indica la ruta del Dockerfile.

PS> docker build -t simple-nginx .

PS> docker images

Ejecuto la imagen con consola interactiva con la imagen recién creada con el Dockerfile, -i (o --interactive) mantiene la entrada STDIN abierta:

**PS**> **docker** **run** **-it** **-p** 8080:80 **--rm** **--name** **simple-nginx-running-app** **simple-nginx**

La opción --rm borra automaticamente el contenedor cuando este se detiene.

Ahora ya puedo abrir el navegador <http://127.0.0.1:8080/> y ver el resultado:



Ejercicio 5

- Ejecuta un contenedor que aloje un proyecto de .net

- Aloja en el contenedor una api de .net y consúmela a través del navegador.

- Se puede usar la imagen que se desee: una compile y ejecute; una imagen que

solo ejecute el proyecto copiado (opción más fácil).

Para el ejercicio 5 he usado la aplicación de la asignatura de Jaime. Lo primero:

Hay que crear el Dockerfile en la carpeta .Api

FROM mcr.microsoft.com/dotnet/aspnet:3.1

COPY bin/Release/netcoreapp3.1/publish/ App/

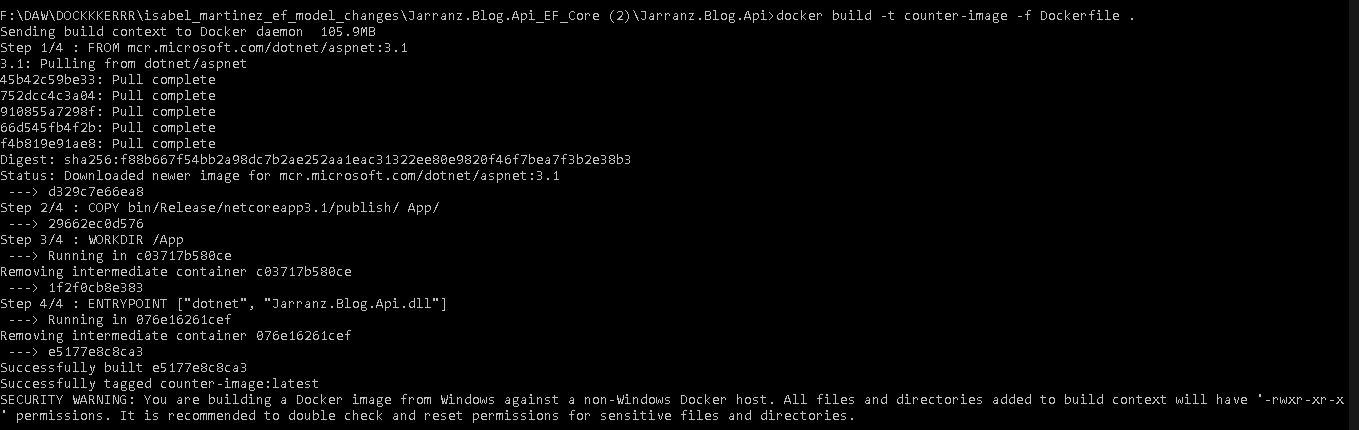
WORKDIR /App

ENTRYPOINT ["dotnet", "Jarranz.Blog.Api.dll"]

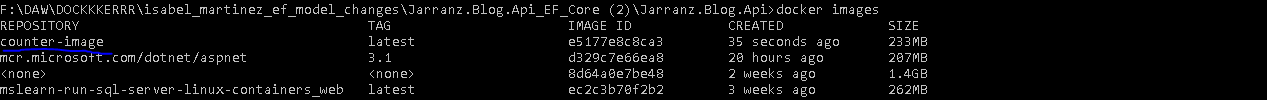
IMPORTANTE: tiene que estar prublicado el proyecto, asi cojemos la ultima publicación en mi casi JarranzBlogApi.dll

Una vez tenemos el dokcerfile creado lo ejecutamos:

docker build -t counter-image -f Dockerfile .



Comprobamos la imagen con docker images



Creamos el contenedor:

docker create --name core-counter counter-image.



Comprobamos con docker ps –a



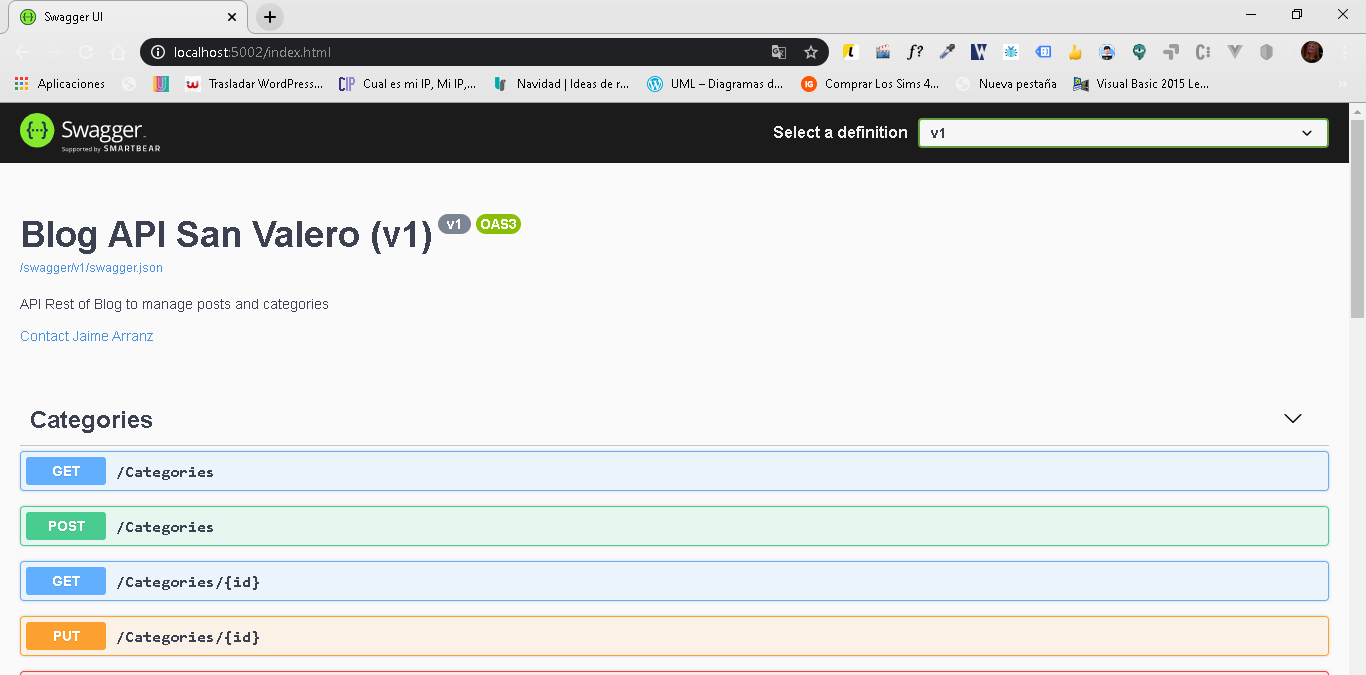
Por ultimo lanzamos el contenedor



Y comprobamos que esta up



Abrimos navegador con localhost:5002 y comprobamos que nos sale el swagger que eso significa que estamos ejecutando la api



Pa$$w0rdSQL

|  |
| --- |
| **Ejercicio 6** |
| * Orquesta los tres contenedores anteriores mediante docker-compose. * Solo es necesario gestionar su ejecución de manera simultánea, no hay que interconectarlos si no se desea. |

Creamos el docker-compose.yml

version: '3'

services:

sqltestcontainerr:

image: mcr.microsoft.com/mssql/server:2019-latest

ports:

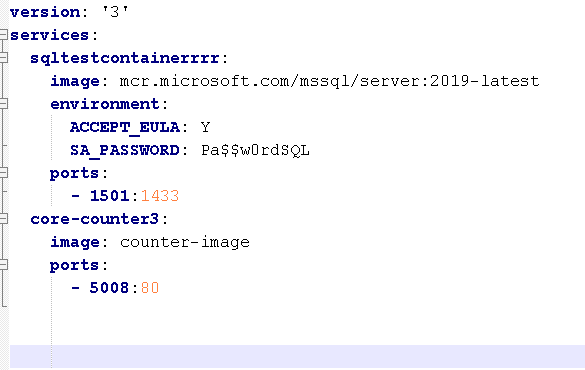
- "1501:1433"

core-counter3:

image: counter-image

ports:

- "5008:80"



Una vez que lo tenemos creado con las aplicaciones que quremos orquestar lanzamos el docker-compose up

