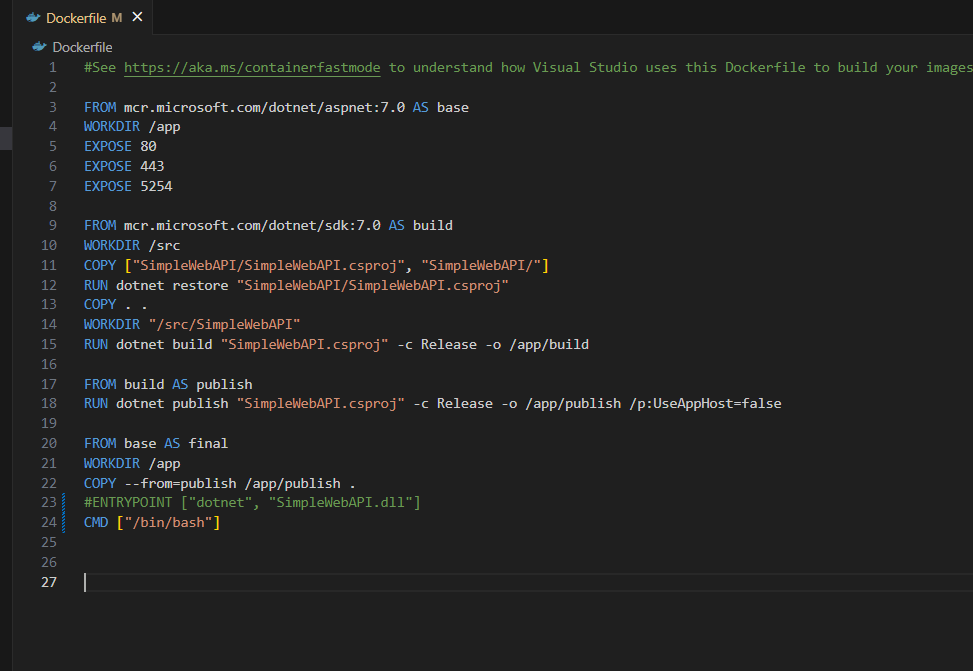
**[9- Modificar Dockerfile para soportar bash](https://github.com/ingsoft3ucc/TPs/blob/master/trabajos/02-introduccion-docker.md" \l "9--modificar-dockerfile-para-soportar-bash)**

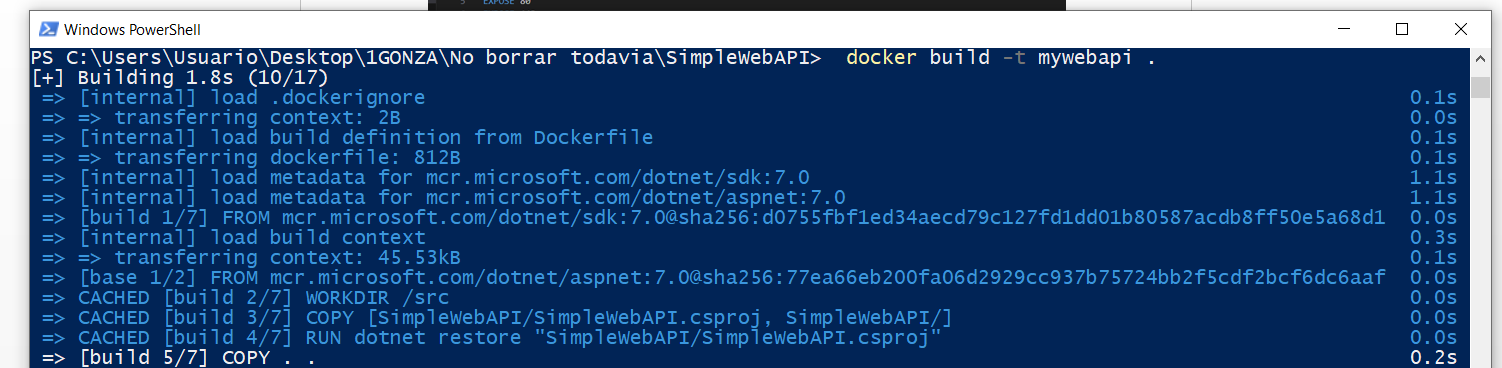
* Modificamos dockerfile para que entre en bash sin ejecutar automaticamente la app

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

CMD ["/bin/bash"]

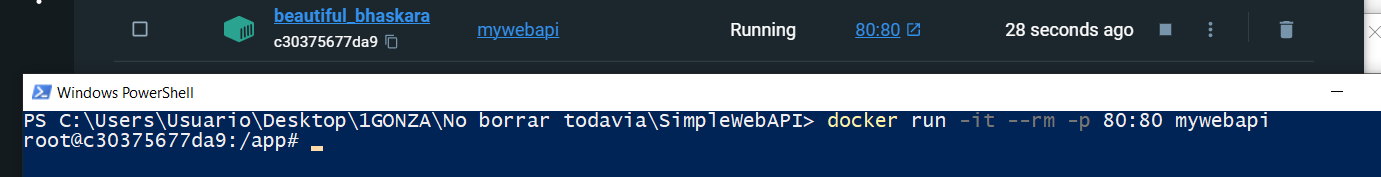


* Rehacemos la imagen



* Corremos contenedor en modo interactivo exponiendo puerto

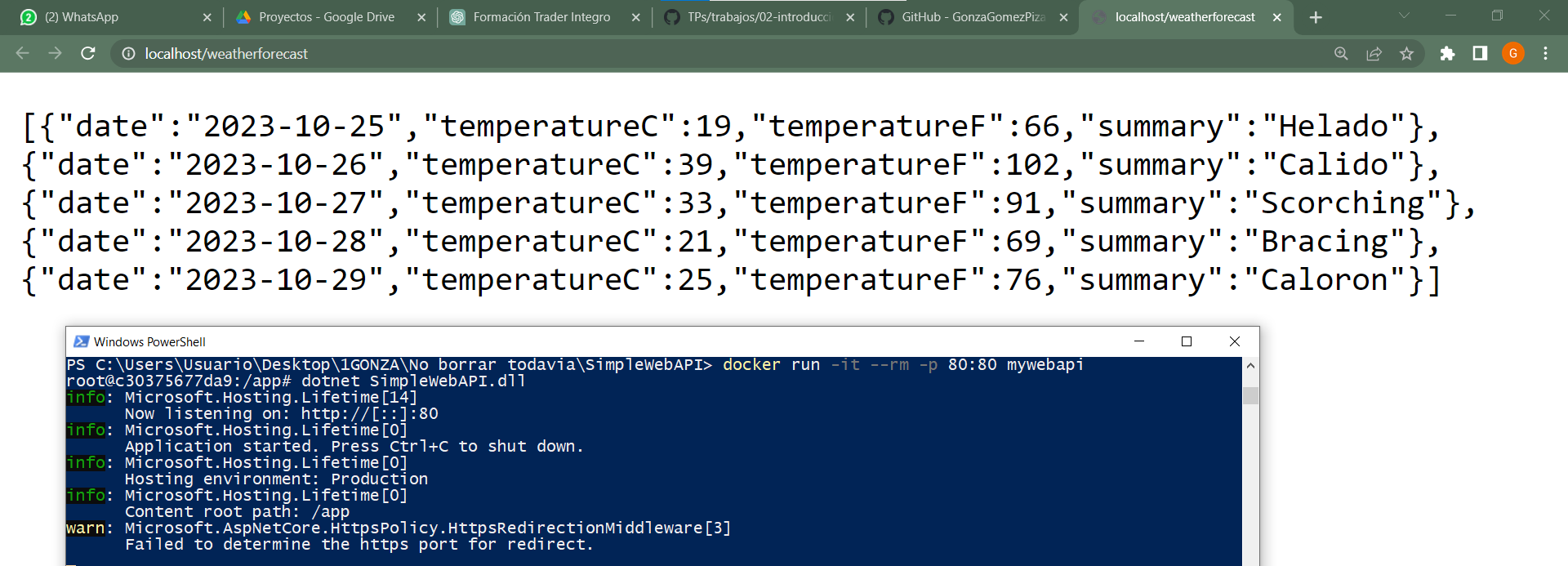
docker run -it --rm -p 80:80 mywebapi



* Navegamos a <http://localhost/weatherforecast>
* Vemos que no se ejecuta automaticamente
* Ejecutamos app:

dotnet SimpleWebAPI.dll

-Volvemos a navegar a <http://localhost/weatherforecast>



Ahora si =)

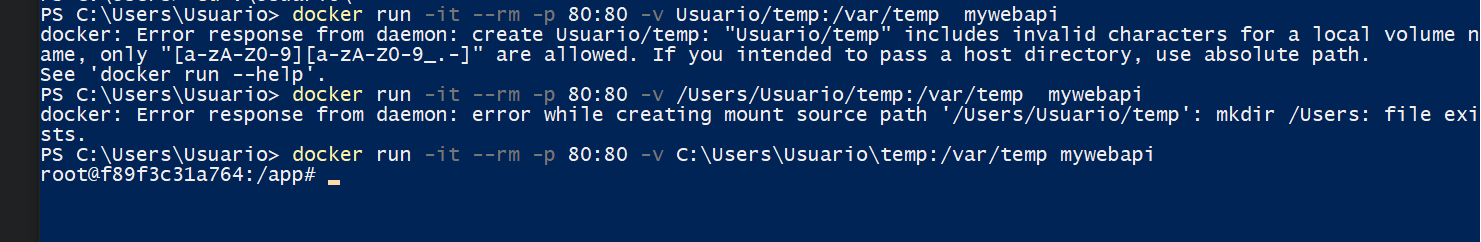
* Salimos del contenedor…

[**10- Montando volúmenes**](https://github.com/ingsoft3ucc/TPs/blob/master/trabajos/02-introduccion-docker.md#10--montando-vol%C3%BAmenes)

Hasta este punto los contenedores ejecutados no tenían contacto con el exterior, ellos corrían en su propio entorno hasta que terminaran su ejecución. Ahora veremos cómo montar un volumen dentro del contenedor para visualizar por ejemplo archivos del sistema huésped:

* Ejecutar el siguiente comando, cambiar myusuario por el usuario que corresponda. En Mac puede utilizarse /Users/miusuario/temp):

docker run -it --rm -p 80:80 -v /Users/miuser/temp:/var/temp  mywebapi

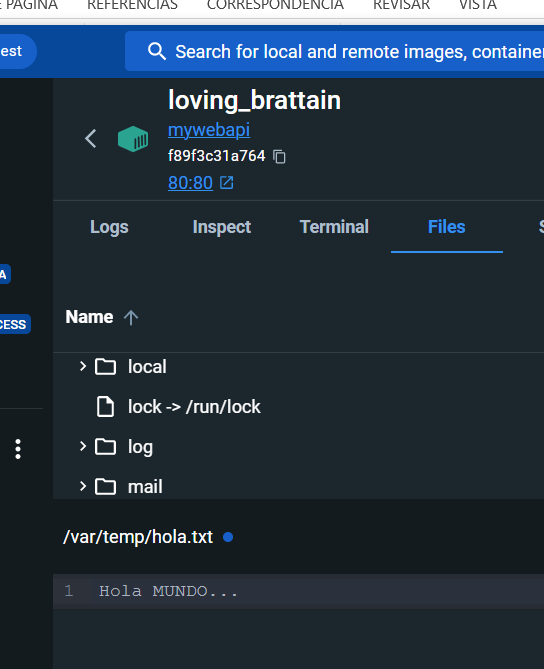


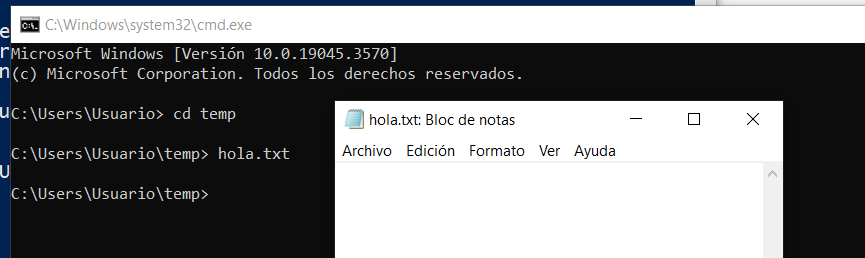
* Dentro del contenedor correr

ls -l /var/temp

touch /var/temp/hola.txt

* Verificar que el Archivo se ha creado en el directorio del guest y del host.



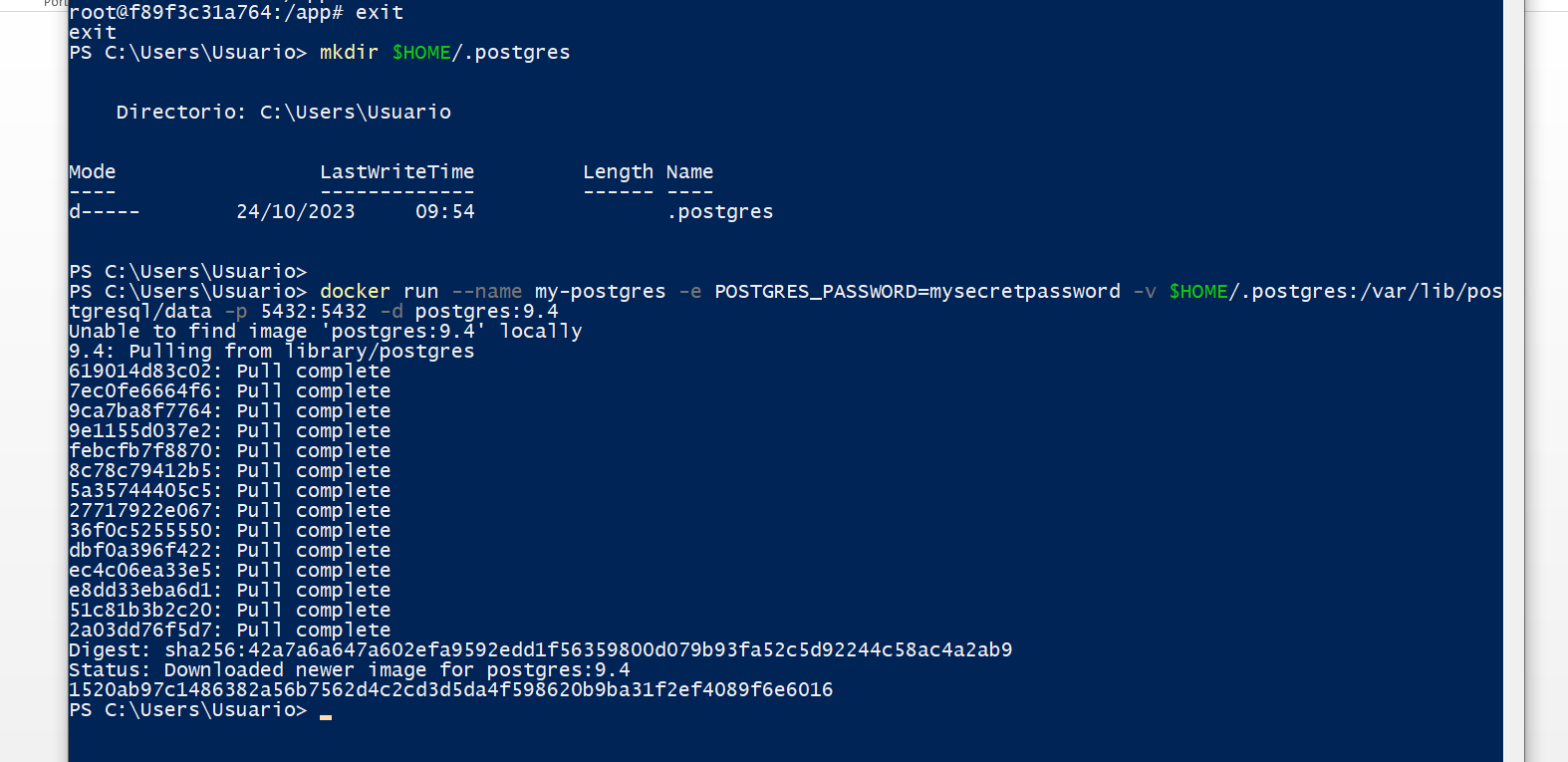


[**11- Utilizando una base de datos**](https://github.com/ingsoft3ucc/TPs/blob/master/trabajos/02-introduccion-docker.md#11--utilizando-una-base-de-datos)

* Levantar una base de datos PostgreSQL

mkdir $HOME/.postgres

docker run --name my-postgres -e POSTGRES\_PASSWORD=mysecretpassword -v $HOME/.postgres:/var/lib/postgresql/data -p 5432:5432 -d postgres:9.4



* Ejecutar sentencias utilizando esta instancia

docker exec -it my-postgres /bin/bash

psql -h localhost -U postgres

#Estos comandos se corren una vez conectados a la base

\l

create database test;

\connect test

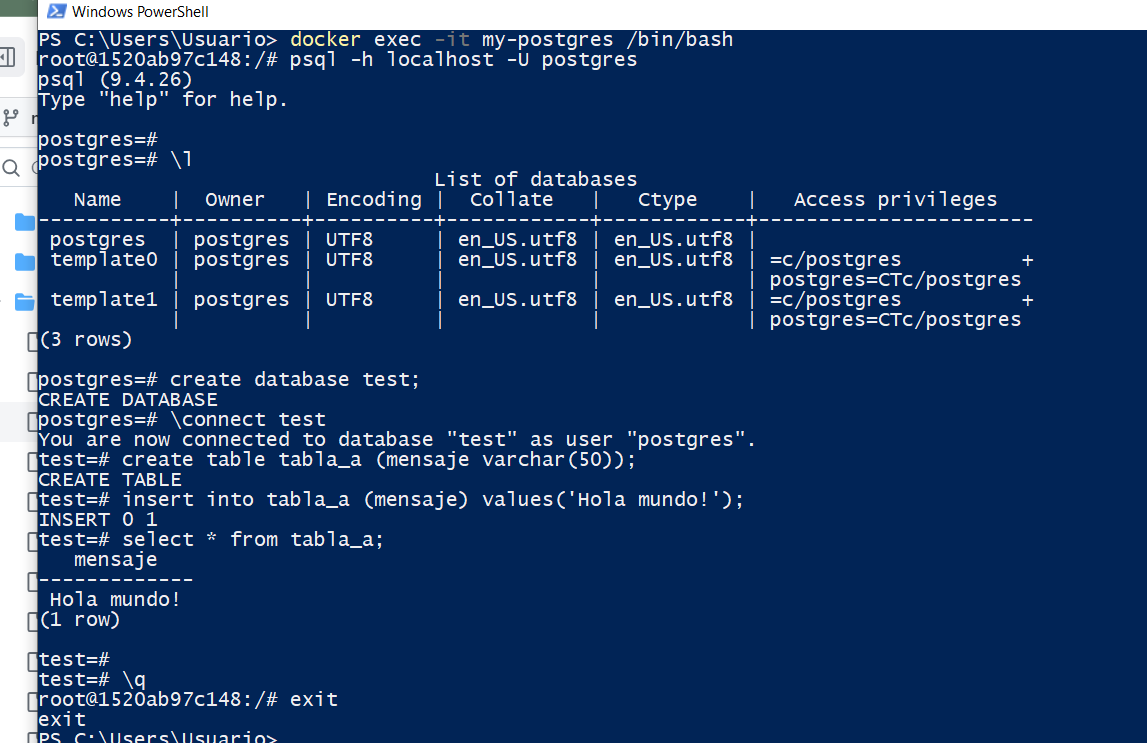
create table tabla\_a (mensaje varchar(50));

insert into tabla\_a (mensaje) values('Hola mundo!');

select \* from tabla\_a;

\q

Exit



* Conectarse a la base utilizando alguna IDE (Dbeaver - <https://dbeaver.io/>, eclipse, IntelliJ, etc...). Interactuar con los objectos objectos creados.

…

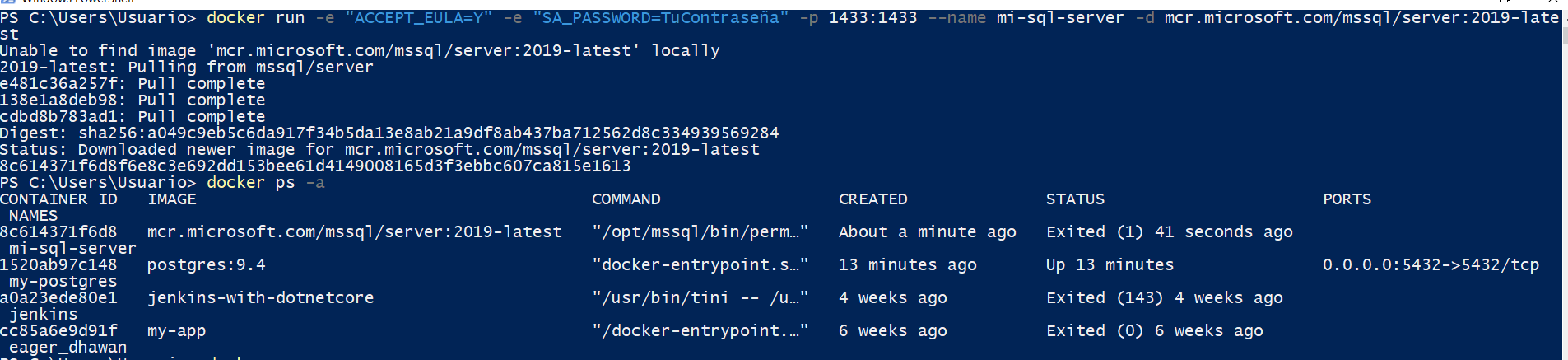
* Explicar que se logro con el comando docker run y docker exec ejecutados en este ejercicio.

**docker run**: Este comando crea un contenedor Docker de PostgreSQL con una contraseña definida y con datos persistentes almacenados en un directorio local. El servidor PostgreSQL dentro del contenedor se ejecuta en segundo plano y es accesible en el puerto 5432 de tu sistema local.

**docker exec:** Este comando te permite acceder a una terminal interactiva dentro del contenedor "my-postgres" y ejecutar comandos dentro del entorno del contenedor. Puedes usarlo para administrar y realizar tareas dentro del contenedor PostgreSQL.

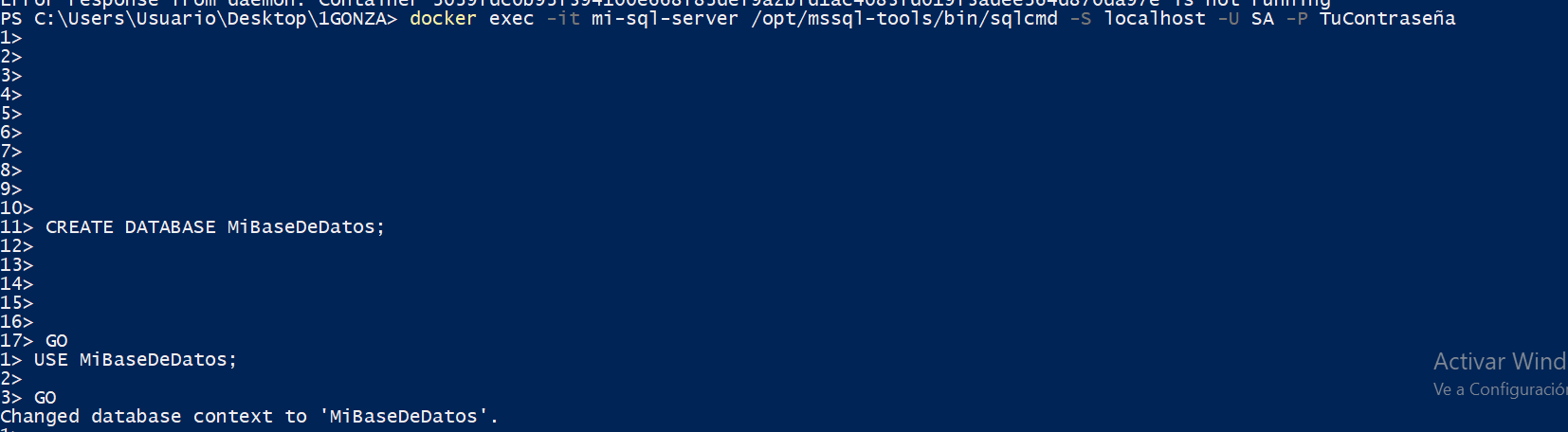
[**12- Hacer el punto 11 con Microsoft SQL Server**](https://github.com/ingsoft3ucc/TPs/blob/master/trabajos/02-introduccion-docker.md#12--hacer-el-punto-11-con-microsoft-sql-server)

* Armar un contenedor con SQL Server

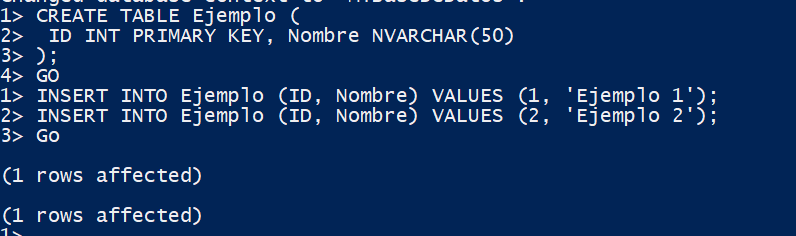


* Crear BD, Tablas y ejecutar SELECT

Crear Base de Datos



Crear Tabla



Ejecutar SELECT…

