



Tecnológico de Estudios Superiores de Ecatepec

**División de Ingeniería en Sistemas
Computacionales**

Academia en Ciencias de la Ingeniería

Materia Programación de Base de Datos

Grupo 5701

Alumno

Campero Granados Luis Daniel

Profesor

Zapiain Cruz Cesar

Practicas 1 a 10

Practica 1 – Creación de una máquina virtual

Sabremos crear una máquina virtual en Azure y administrarla, la maquina será de Ubuntu 18.04 Gen2.

1. Nos vamos a los servicios de Azure a “Máquinas Virtuales”, posterior a ello, le damos al botón crear y nos desplegara dos opciones y le damos en “Máquina virtual” y nos va a mostrar una página donde ya configuraremos la máquina.

Servicios de Azure

Crear un recurso Grupos de recursos Centro de inicio rápido Máquinas virtuales

Recursos recientes

Nombre cloudshell201821479 cloudshell

Navegar

Suscripciones Grupos de recursos

Herramientas

Microsoft Learn Azure Monitor Security Center Administración de costos

Inicio >

Máquinas virtuales

TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ECATEPEC (tese.edu.mx)

+ Crear Cambiar al modo clásico Reservas Administrar vista Actualizar Exportar a CSV Abrir consulta Asignar etiquetas Iniciar Reiniciar Detener Eliminar

+ Máquina virtual Comenzar con una configuración preestablecida

Mostrando de 0 a 0 de 0 registros.

Grupo de recursos == todo Ubicación == todo Agregar filtro

Sin agrupar Vista de lista

Nombre Suscripción Grupo de recursos Ubicación Estado Sistema operativo Tamaño Dirección IP pública Discos

No hay máquinas virtuales para mostrar

Crea una máquina virtual que ejecute Linux o Windows. Seleccione una imagen de Marketplace o use una imagen personalizada propia.

Más información acerca de Windows Virtual Machines Más información sobre Linux Virtual Machines

2. Primero creamos un recurso en donde se guardara todo, este al final de cada practica se deberá borrar para que no gaste el crédito proporcionado para estudiantes.

Inicio > Máquinas virtuales >

Máquinas virtuales

TECNOLOGICO DE ESTUDIOS SUPERIORES DE ECA...

+ Crear ▾ ...

Filtrar por cualquier ca...

Nombre ↑↓ Suscripción ↑↓

No hay máquinas virtuales para mostrar

Crea una máquina virtual que ejecuta Linux o Windows. Seleccione una imagen de Marketplace o use una imagen personalizada propia.

Más información acerca de Windows Virtual Machines ⓘ

Más información sobre Linux Virtual Machines ⓘ

Crear una máquina virtual

Datos básicos | Discos | Redes | Administración | Opciones avanzadas | Etiquetas | Revisar y crear

Cree una máquina virtual que ejecuta Linux o Windows. Seleccione una imagen de Azure Marketplace o use una imagen personalizada propia. Complete la pestaña Conceptos básicos y, después, use Revisar y crear para aprovisionar una máquina virtual con parámetros predeterminados o bien revise cada una de las pestañas para personalizar la configuración. [Más información ⓘ](#)

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costes. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción * ⓘ Azure para estudiantes

Grupo de recursos * ⓘ (Nuevo) Grupo de recursos

[Crear nuevo](#)

Detalles de instancia

Nombre de máquina virtual * ⓘ

Región * ⓘ

Opciones de disponibilidad ⓘ

Revisar y crear

< Anterior

Un grupo de recursos es un contenedor que tiene los recursos relacionados de una solución de Azure.

Nombre * TESE

Aceptar Cancelar

3. Le asignamos un nombre a la máquina, dejamos la región que se estableció al momento de configurar el bash, y seleccionamos la imagen de nuestra máquina virtual, en este caso se va a utilizar Ubuntu Server la versión 18.04 LTS – Gen2.

Detalles de instancia

Nombre de máquina virtual * ⓘ Ubuntu ✓

Región * ⓘ (US) Este de EE. UU. ▾

Opciones de disponibilidad ⓘ No se requiere redundancia de la infraestructura ▾

Imagen * ⓘ Ubuntu Server 18.04 LTS - Gen2 ▾

[Ver todas las imágenes](#) | [Configure VM generation](#)

Instancia de Azure de acceso puntual ⓘ ☐

4. Requerimos 4 GB de memoria RAM como mínimo para poder trabajar con Docker. Así que seleccionamos una máquina que nos soporte Docker con las especificaciones mínimas y que tenga un costo mínimo.

Seleccionar un tamaño de máquina virtual ...

×

Buscar por tamaño de má Mostrar costo : Cada mes vCPU : Todo RAM (GiB) : Todo Agregar filtro

Mostrando 414 tamaños de máquina virtual. Suscripción: Azure para estudiantes Región: Este de EE.UU. Tamaño actual: Standard_D2s_v3 Imagen: Ubuntu Server 18.04 LTS Más información sobre los tamaños de VM Agrupar por serie

Tamaño de VM ↑↓	Familia ↑↓	vCPU ↑↓	RAM (GiB) ↑↓	Discos de datos ↑↓	E/S máxima por s... ↑↓	Almacenamiento tem... ↑↓	Disco premium ↑↓	Costo/mes ↑↓
Más usados por los usuarios de Azure								
Los tamaños más usados por los usuarios en Azure.								
D51_v2	Uso general	1	3.5	4	3200	7	Se admite	MXN 1,124.42
D2s_v3	Uso general	2	8	4	3200	16	Se admite	MXN 1,478.69
D2as_v4	Uso general	2	8	4	3200	16	Se admite	MXN 1,478.69
B2s	Uso general	2	4	4	1280	8	Se admite	MXN 640.76
B1s	Uso general	1	1	2	320	4	Se admite	MXN 160.19
B2ms	Uso general	2	8	4	1920	16	Se admite	MXN 1,281.53
B1ls	Uso general	1	0.5	2	160	4	Se admite	MXN 80.10
DS2_v2	Uso general	2	7	8	6400	14	Se admite	MXN 2,248.84
R4ms	Uso general	4	16	8	2880	22	Se admite	MXN 2,556.00

5. Elegimos esta máquina y posteriormente agregamos un nombre de usuario y una contraseña. Por último, dejamos activo el puerto SSH (22) y finalmente le damos a revisar y crear.

Cuenta de administrador

Tipo de autenticación ⓘ

- ☐ Clave pública SSH
- ☒ Contraseña

Nombre de usuario * ⓘ

usertese ✓

Contraseña * ⓘ

..... ✓

Confirmar contraseña * ⓘ

..... ✓

Reglas de puerto de entrada

Seleccione los puertos de red de máquina virtual que son accesibles desde la red Internet pública. Puede especificar acceso de red más limitado o granular en la pestaña Red.

Puertos de entrada públicos * ⓘ

- ☐ Ninguno
- ☒ Permitir los puertos seleccionados

Seleccionar puertos de entrada *

SSH (22) ✓

- Veremos las especificaciones que tendrá la máquina y el precio de cuanto costará por hora. Le damos a Crear y ya tendríamos nuestra máquina.

PRODUCT DETAILS

B2s estándar

by Microsoft

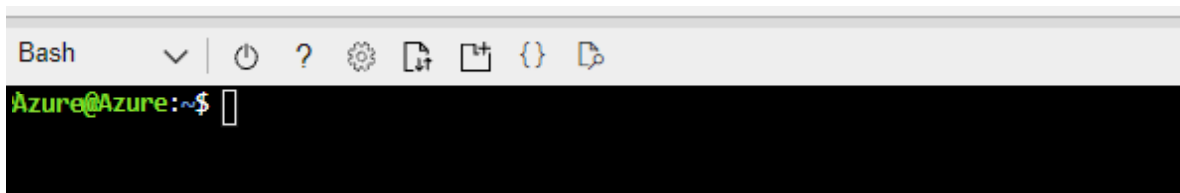
[Terms of use](#) | [Privacy policy](#)

Subscription credits apply ⓘ

0,8778 MXN/hr

[Pricing for other VM sizes](#)

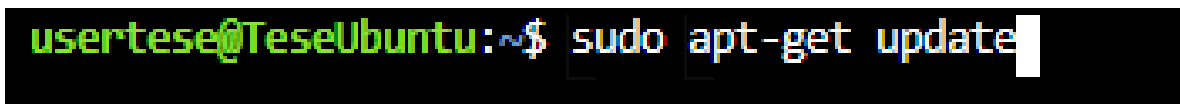
- Al ya tener nuestra máquina virtual creada nos meteremos a la terminal para configurarla y añadirle los paquetes que requerimos para trabajar sobre ella.



```
Bash
Azure@Azure:~$
```

- Ingresamos el siguiente comando junto con nuestra dirección IP de nuestra máquina virtual: SSH (direccion IP) -l (nombre de usuario), nos aparecerá una pregunta de y elegimos yes, colocamos la contraseña y ya estaremos dentro de la máquina. Después nos dirigimos a actualizar los vínculos y la actualización de los paquetes.

- Agregaremos el siguiente código para hacer la actualización.



```
usertese@TeseUbuntu:~$ sudo apt-get update
```

- Después hacemos un upgrade



```
usertese@TeseUbuntu:~$ sudo apt-get upgrade
```

11. Luego copiaremos las instrucciones de la siguiente página:
<https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-18-04-1-es>
12. Por último, hacemos la instalación del Docker que este al igual viene más abajo en la misma página. Y ya tendríamos instalado Docker.

Por último, instale Docker:

```
$ sudo apt install docker-ce
```

13. Después nos haría falta instalar el Azure CLI, los comandos para su instalación están en la siguiente página en la opción 2: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-linux?pivot=apt>
14. Terminamos la configuración

```
user@ubuntu:~$ sudo apt-get install azure-cli
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-159
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  azure-cli
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 65.2 MB of archives.
After this operation, 1024 MB of additional disk space will be used.
Get:1 https://packages.microsoft.com/repos/azure-cli bionic/main amd64 azure-cli all 2.29.0-1~bionic [65.2 MB]
Fetched 65.2 MB in 1s (50.5 MB/s)
Selecting previously unselected package azure-cli.
(Reading database ... 77226 files and directories currently installed.)
Preparing to unpack .../azure-cli_2.29.0-1~bionic_all.deb ...
Unpacking azure-cli (2.29.0-1~bionic) ...
Setting up azure-cli (2.29.0-1~bionic) ...
user@ubuntu:~$
```

Conclusión

Aprendí a hacer una maquina virtual y su respectiva configuración, así como la instalación de un Docker de Azure, esto me ayuda a expandir mis conocimientos e ir aprendiendo mas día con día ya que el uso de esta herramienta nunca lo había llevado a cabo y con esto aprendí sobre ella y sus usos.

Practica 2 - Dockerizacion de una aplicación en NodeJS

Hacemos el mismo procedimiento de la primer practica que es la instalación de la máquina virtual e instalar el Docker y el CLI de Azure, al hacer eso procedimos a ingresar a la siguiente página:

<https://nodejs.org/en/docs/guides/nodejs-docker-webapp/>

Obtendremos la información para hacer la práctica, y se verán los códigos y las formas para trabajar y elaborar nuestro hola mundo en una página;


1. Creamos un archivo vacío para el *package.json*.

```
usertese@TeseUbuntu:~$ mkdir DockerNodeJS
usertese@TeseUbuntu:~$ cd DockerNodeJS/
usertese@TeseUbuntu:~/DockerNodeJS$ touch package.json
usertese@TeseUbuntu:~/DockerNodeJS$ ls
package.json
usertese@TeseUbuntu:~/DockerNodeJS$
```

2. Creamos ahora el archivo *server.js*

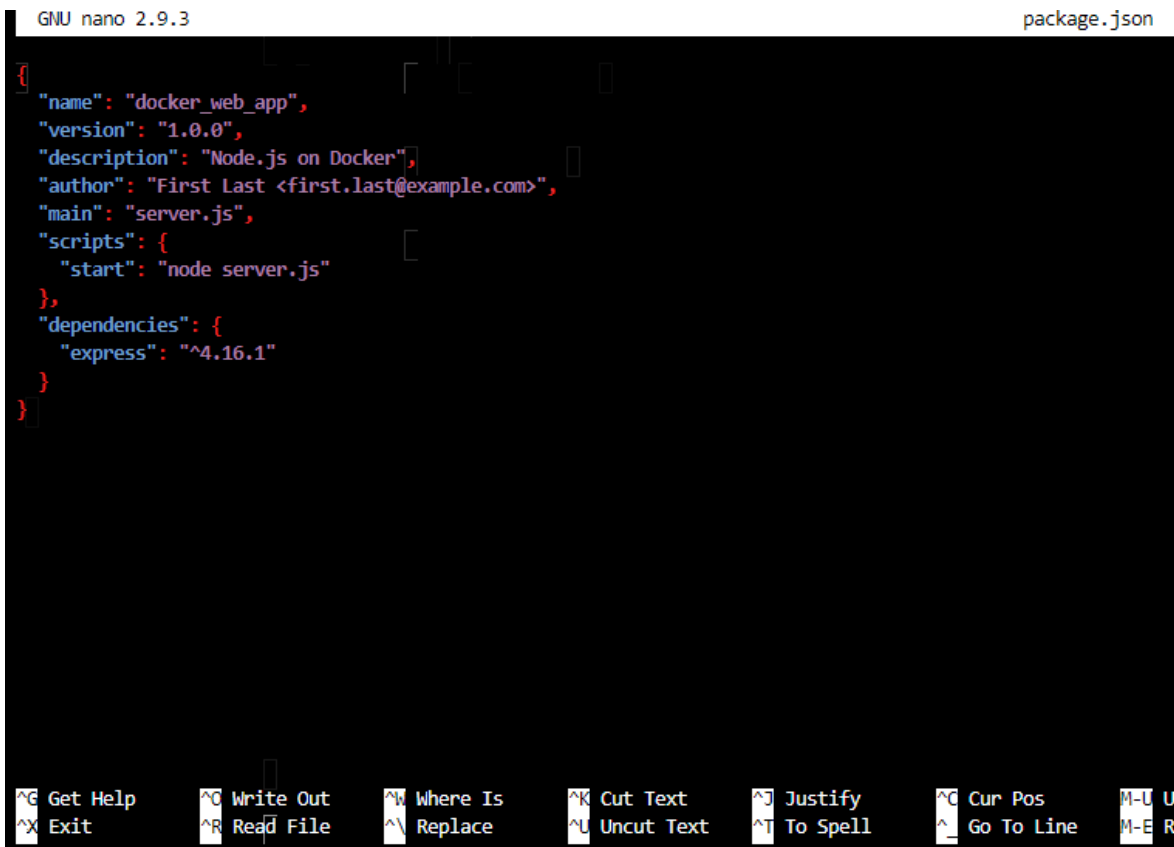
```
usertese@TeseUbuntu:~/DockerNodeJS$ touch server.js
usertese@TeseUbuntu:~/DockerNodeJS$ ls
package.json  server.js
usertese@TeseUbuntu:~/DockerNodeJS$
```

3. Abrimos la sección para insertar los códigos de Node.js



A screenshot of the nano text editor interface. The title bar at the top shows "GNU nano 2.9.3" on the left and "package.json" on the right. The main editing area is a black rectangle with some faint, illegible text. At the bottom, there is a status bar with various keyboard shortcuts and their corresponding actions, such as "Get Help", "Write Out", "Where Is", "Cut Text", "Justify", "Cur Pos", "Go To Line", "Undo", "Mark Text", "To Bracket", "Previous", "Back", "Exit", "Read File", "Replace", "Uncut Text", "To Spell", "Redo", "Copy Text", "WhereIs Next", "Next", and "Forward".

4. Insertamos el primer código de la página y nos salimos con Ctrl+X y le damos que si se guarde.



A screenshot of the nano text editor interface showing the contents of the package.json file. The title bar at the top shows "GNU nano 2.9.3" on the left and "package.json" on the right. The main editing area is a black rectangle with the following JSON code in red text:

```
{  
  "name": "docker_web_app",  
  "version": "1.0.0",  
  "description": "Node.js on Docker",  
  "author": "First Last <first.last@example.com>",  
  "main": "server.js",  
  "scripts": {  
    "start": "node server.js"  
  },  
  "dependencies": {  
    "express": "^4.16.1"  
  }  
}
```

 At the bottom, the status bar shows various keyboard shortcuts and their corresponding actions, such as "Get Help", "Write Out", "Where Is", "Cut Text", "Justify", "Cur Pos", "Go To Line", "Undo", "Mark Text", "To Bracket", "Previous", "Back", "Exit", "Read File", "Replace", "Uncut Text", "To Spell", "Redo", "Copy Text", "WhereIs Next", "Next", and "Forward".

5. Hacemos lo mismo que el anterior solo que el código del `server.js`
6. Después de hacer lo anterior debemos crear nuestro archivo Dockerfile que nos permitirá crear el microservicio.

```
GNU nano 2.9.3

FROM node:14

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied
# where available (npm@5+)
COPY package*.json ./

RUN npm install
# If you are building your code for production
# RUN npm ci --only=production

# Bundle app source
COPY . .

EXPOSE 8080
CMD [ "node", "server.js" ]
```

7. Al ya tener los archivos, vamos a crear la imagen

```
usertese@TeseUbuntu:~/DockerNodeJS$ sudo docker build -t tese/nodejs .
Sending build context to Docker daemon 4.096kB
Step 1/7 : FROM node:14
14: Pulling from library/node
2f0ef4316716: Pull complete
1a43d3c11106: Pull complete
243ae34810fb: Pull complete
d01c447bcebc: Pull complete
1d07840244ef: Pull complete
5df57a36055e: Pull complete
5037d4710e07: Pull complete
714346faf07f: Pull complete
55ce05f20ef8: Pull complete
Digest: sha256:109b118e0d49dd12ca6f5b84a7a9a9c8a147f75567b3ad50620bdacaf5e6320d
Status: Downloaded newer image for node:14
---> 6a64cec3731e
Step 2/7 : WORKDIR /usr/src/app
---> Running in 81eb57592e7d
Removing intermediate container 81eb57592e7d
---> ca75091e93b5
Step 3/7 : COPY package*.json ./
---> b7217f056e45
Step 4/7 : RUN npm install
---> Running in cfa31ba36bc5

usertese@TeseUbuntu:~/DockerNodeJS$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tese/nodejs	latest	4b6c9a0adf90	4 minutes ago	947MB
node	14	6a64cec3731e	3 days ago	944MB


Conclusión

Aprendí a Dockerizar una aplicación en Node JS, aprendí a crear los archivos, copiando el código de la pagina web proporcionada y guardamos la configuración y por ultimo pues creamos la imagen para poder encender nuestra imagen y recibir un mensaje de *Hello World*.

Practica 3 - Ejecución de un contenedor y su ciclo de vida.

Corremos nuestra imagen y haremos los siguientes pasos

1. Nos dirigimos a la máquina virtual y abriremos los puertos, tendremos que agregar una regla.

 **Agregar regla de seguridad de entrada** ×
TeseUbuntu-nsg

Custom ▼

Intervalos de puertos de destino * ⓘ
49160-49170 ✓

Protocolo
☒ Any
☐ TCP
☐ UDP
☐ ICMP

Acción
☒ Permitir
☐ Denegar

Prioridad * ⓘ
310

Agregar


Cancelar

2. Obtendremos la dirección pública.

teseubuntu735

Configuración de IP ⓘ

ipconfig1 (Principal) ▼

 **Interfaz de red: teseubuntu735**

Red virtual/subred: Tese-vnet/default

Reglas de seguridad vigentes

IP pública de NIC: **13.68.189.55**

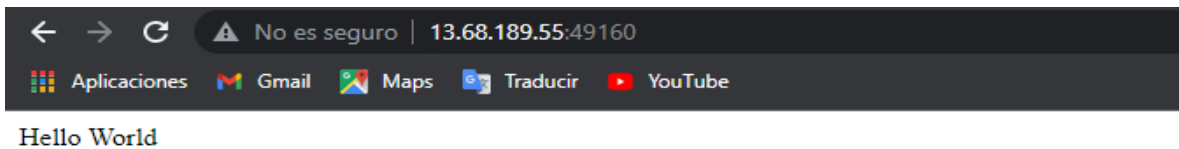
[Solucionar problemas de conexión de VM](#)

IP privada de NIC: **10.0.0.4**

[Topología](#)

Redes aceleradas: **Deshabilitado**

3. Ingresamos con esta dirección y un puerto en el navegador y veremos el Hello World



4. Para detener un puerto usamos el siguiente comando *stop* y el código que genero al activar ese puerto.

```
usertese@TeseUbuntu:~/DockerNodeJS$ sudo docker run -p 49161:8080 -d tese/nodejs:latest
7958370bcf78a5bfb271fa3868458990a54633070e1f6d74b540ab3fd65861c6
usertese@TeseUbuntu:~/DockerNodeJS$ sudo docker stop -p 49161:8080 -d tese/nodejs:latest
unknown shorthand flag: 'p' in -p
See 'docker stop --help'.
usertese@TeseUbuntu:~/DockerNodeJS$ sudo docker stop 7958370bcf78a5bfb271fa3868458990a54633070e1f6d74b540ab3fd65861c6
7958370bcf78a5bfb271fa3868458990a54633070e1f6d74b540ab3fd65861c6
```

5. Para encenderlo lo hacemos con *start*.

```
usertese@TeseUbuntu:~/DockerNodeJS$ sudo docker start 7958370bcf78a5bfb271fa3868458990a54633070e1f6d74b540ab3fd65861c6
7958370bcf78a5bfb271fa3868458990a54633070e1f6d74b540ab3fd65861c6
```

6. Cuando queramos remover el puerto para que no se pueda volver a usar lo hacemos con el comando *rm*

```
usertese@TeseUbuntu:~/DockerNodeJS$ sudo docker rm 7958370bcf78a5bfb271fa3868458990a54633070e1f6d74b540ab3fd65861c6
7958370bcf78a5bfb271fa3868458990a54633070e1f6d74b540ab3fd65861c6
```

7. Para saber cuántas instancias están corriendo, en que puerto y cuánto tiempo tienen de vida lo hacemos con el siguiente comando.

```
usertese@TeseUbuntu:~/DockerNodeJS$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0b402c88f26b	tese/nodejs:latest	"docker-entrypoint.s..."	47 minutes ago	Up 46 minutes	0.0.0.0:49160->8080/tcp, :::49160->8080/tcp	nice_spence

Conclusión

Para esta practica ejecutamos nuestra imagen ya instalada para mostrarnos nuestro mensaje y después abrimos los puertos de la maquina para poder ser utilizados y conocer los ciclos de vida.

Practica 4 – Robótica.

En el tema presentado primeramente se habló sobre que era la inteligencia artificial, el concepto dado es que la inteligencia artificial es la simulación de la inteligencia humana en máquinas, dándoles la capacidad de aprender, razonar, deducir, hacer predicciones, etc.

Sus aplicaciones hoy en día son en la medicina, ya que con lo que se vivió y vive actualmente se dio un uso mayor a estas inteligencias para ayudar a sobrellevar esta situación.

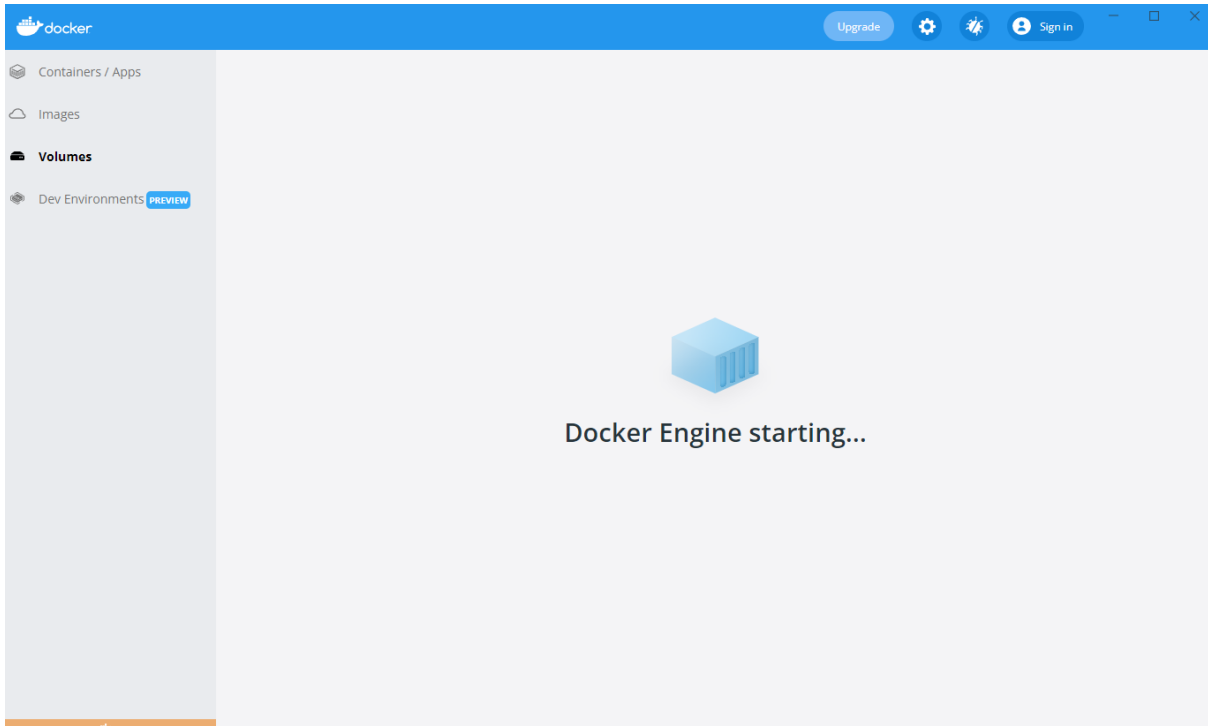
Se hablo sobre la Robótica en el TESE, sus eventos que han tenido y sus antecedentes. Se hablo del evento que se realizó, el tipo de evento y el año en que se realizaron estos eventos y exposiciones. También se habló de los torneos realizados en el TESE en donde participaron alumnos de diferentes carreras.

Otro tema hablado fue presentado por el maestro Ruben Raya Delgado y el tema impartido fue Alexa, disruptores de conectividad en IoT, este tema fue interesantes ya que nos hablaron sobre un dispositivo que hoy en día se utiliza de manera muy común, en este caso un asistente de voz. Esto es debido a que las personas se les facilita consultar y que se les de información de manera más rápida sin tener que escribirlo.

La búsqueda por voz y los altavoces inteligentes sobre la tecnología que permite a los usuarios realizar una exploración en internet por ahora una pregunta del norte esto se realiza a través de un smartphone muy positivo que la gente o un ordenador.

Practica 5 - instalación Docker

Descargamos Docker



Ingresamos el siguiente código en el cmd; *docker pull mcr.microsoft.com/mssql/server:2019-latest*

```
Microsoft Windows [Versión 10.0.19043.1348]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>docker pull mcr.microsoft.com/mssql/server:2019-latest
2019-latest: Pulling from mssql/server
35807b77a593: Downloading [=====>] 12.09MB/28.57MB
be2aa0ec326c: Download complete
912596dfeae3: Downloading [=====>] 15.12MB/56.12MB
84a6a587b3fb: Downloading [==>] 16.74MB/403.6MB
a0ceb3206273: Waiting
```

Ingresamos contraseña y configuramos el puerto con el siguiente código:

```
docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=<YourStrong@Passw0rd>" \ -p 1433:1433 --name sql1 -h sql1 \ -d mcr.microsoft.com/mssql/server:2019-latest
```

```
35807b77a593: Pull complete
be2aa0ec326c: Pull complete
912596dfeab: Pull complete
84a6a587b3fb: Pull complete
a0ceb3206273: Pull complete
Digest: sha256:925bb075b5715b20c2acfb244db857c92afdbfbaefbe27257a4b97817906bef
Status: Downloaded newer image for mcr.microsoft.com/mssql/server:2019-latest
mcr.microsoft.com/mssql/server:2019-latest

C:\Windows\system32> docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=<YourStrong@Passw0rd>" \
docker: invalid reference format.
See 'docker run --help'.

C:\Windows\system32> -p 1433:1433 --name sql1 -h sql1 \
"-p" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Windows\system32> -d mcr.microsoft.com/mssql/server:2019-latest docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=<YourStrong@Passw0rd>" \
"-d" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Windows\system32> -p 1433:1433 --name sql1 -h sql1 \
"-p" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Windows\system32> docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Killzone789" -p 1434:1433 --name sql1 -h sql1 -d mcr.microsoft.com/mssql/server:2019
latest
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
1433:1433/tcp -> 1433:1433/tcp
```

Ingresamos el siguiente código; *Docker ps -a*

```
C:\Windows\system32>docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
AMES					
7a107f51616f	mcr.microsoft.com/mssql/server:2019-latest	"/opt/mssql/bin/per..."	About a minute ago	Up About a minute	0.0.0.0:1434->1433/tcp
sql1					

Nos conectamos a SQL Server a través de Linux.

```
CONTAINER ID    IMAGE
AMES
7a107f51616f    mcr.microsoft.com/mssql/server:2019-latest
sql1

C:\Windows\system32>docker exec -it sql1 "bash"
mssql@sql1:/$
```

E ingresamos la siguiente línea de comandos;

```
7a107f51616f mcr.microsoft.com/mssql/server:2019-latest "/opt/mssql/bin/per
ql1
C:\Windows\system32>docker exec -it sql1 "bash"
mssql@sql1:/$ /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Killzone789"
1>
```

Creamos una Base de Datos:

```
C:\Windows\system32>docker exec -it sql1 "
mssql@sql1:/$ /opt/mssql-tools/bin/sqlcmd
1> create database TESE
2> GO
1> use TESE
2> GO
Changed database context to 'TESE'.
1>
```


Practica 6

Primero tendremos que correr Docker en el terminal después seguimos los siguientes pasos;

1. Copiamos el siguiente comando para Docker y descargamos las imagenes `sudo Docker Pull mcr.microsoft.com/mssql/server:2019-latest`

```
Administrador: Símbolo del sistema - docker pull mcr.microsoft.com/mssql/server:2019-latest
Microsoft Windows [Versión 10.0.19043.1348]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>docker run -d -p 80:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
0d0a0d46f8b: Pull complete
53675450485: Pull complete
0fa1b21e55d: Pull complete
742bedbd98a: Pull complete
55384f200a8: Pull complete
7010ea25d94: Pull complete
5163a27e4cc: Pull complete
fd5d370a5f97: Pull complete
Digest: sha256:c3ccf0a6246c8c8989eea4b54adef90b5e6c99d317a986d5ff0872ab15666251
Status: Downloaded newer image for docker/getting-started:latest
6be2cf9f099a7d710d9dd672a13cd130573bdb6783d663c85f742a930c18372

C:\WINDOWS\system32>sudo docker pull mcr.microsoft.com/mssql/server:2019-latest
'sudo' no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\WINDOWS\system32>docker pull mcr.microsoft.com/mssql/server:2019-latest
2019-latest: Pulling from mssql/server
5807b77a593: Pull complete
be2aa0ec326c: Pull complete
12596dfeaeab: Downloading [=====>] 15.12MB/56.12MB
4a6a587b3fb: Downloading [=====>] 72.97MB/403.6MB
0ceb3206273: Downloading [=====>] 20.19MB/25.01MB
```

2. Después ingresamos el siguiente comando `docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Password123." -p 1434:1433 --name sql1 -h sql1 -d mcr.microsoft.com/mssql/server:2019-latest`

```
Administrador: Símbolo del sistema
C:\WINDOWS\system32>sudo docker pull mcr.microsoft.com/mssql/server:2019-latest
'sudo' no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\WINDOWS\system32>docker pull mcr.microsoft.com/mssql/server:2019-latest
2019-latest: Pulling from mssql/server
35807b77a593: Pull complete
be2aa0ec326c: Pull complete
912596dfeaeab: Pull complete
84a6a587b3fb: Pull complete
a0ceb3206273: Pull complete
Digest: sha256:925bb075b5715b20c2acfbcb244db857c92afdbfbaefbe27257a4b97817906bef
Status: Downloaded newer image for mcr.microsoft.com/mssql/server:2019-latest
mcr.microsoft.com/mssql/server:2019-latest

C:\WINDOWS\system32>docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Password123." -p 1434:1433 --name sql1 -h sql1 \-d mcr
.microsoft.com/mssql/server:2019-latest
docker: invalid reference format.
See 'docker run --help'.

C:\WINDOWS\system32>
```

3. Nos conectamos con SQL con el siguiente comando *docker exec -it sql1 "bash"*



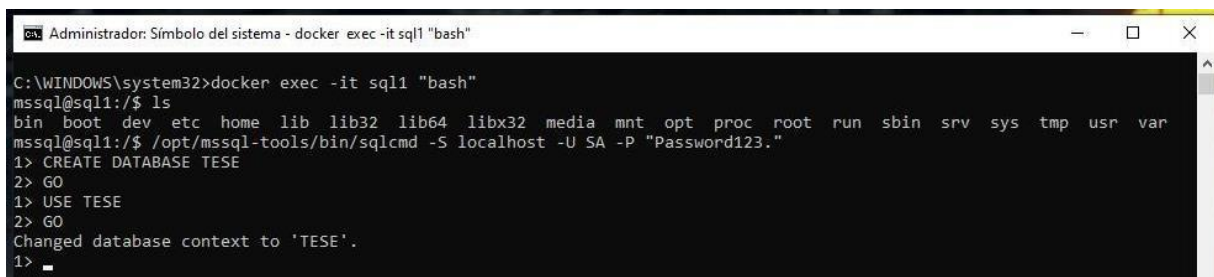
```
Administrador: Símbolo del sistema - docker exec -it sql1 "bash"
C:\WINDOWS\system32>docker exec -it sql1 "bash"
mssql@sql1:/$
```

4. Después ingresamos el siguiente comando */opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password123."*



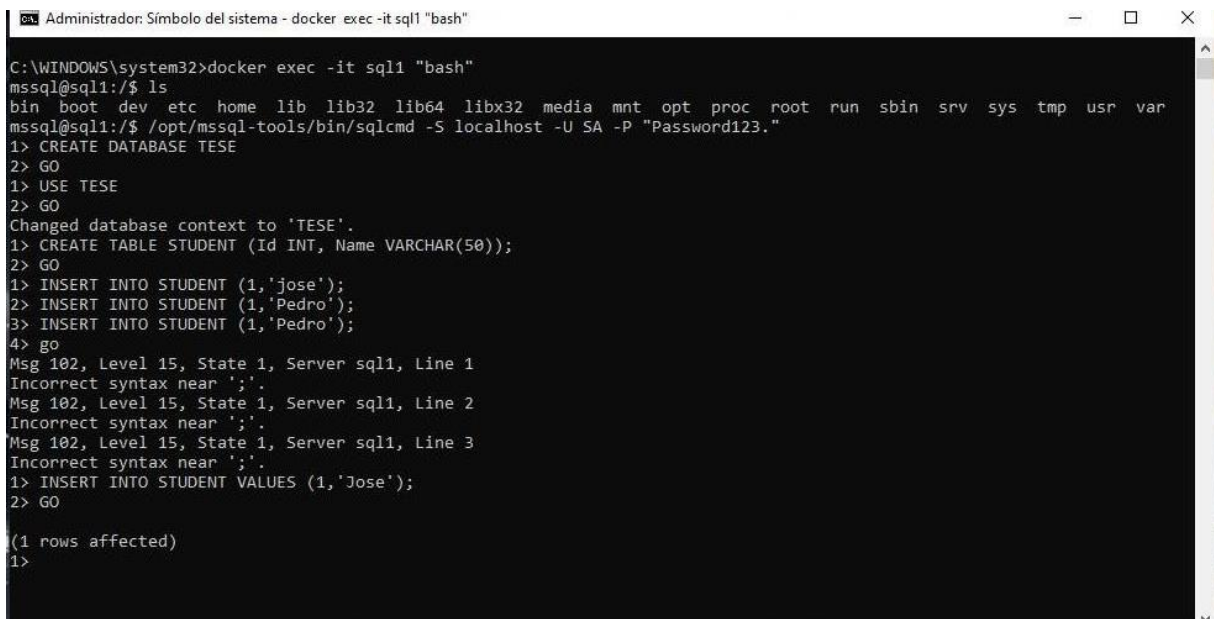
```
Administrador: Símbolo del sistema - docker exec -it sql1 "bash"
C:\WINDOWS\system32>docker exec -it sql1 "bash"
mssql@sql1:/$ ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
mssql@sql1:/$ /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password123."
1>
```

5. Comprobamos su funcionamiento creando una base de datos



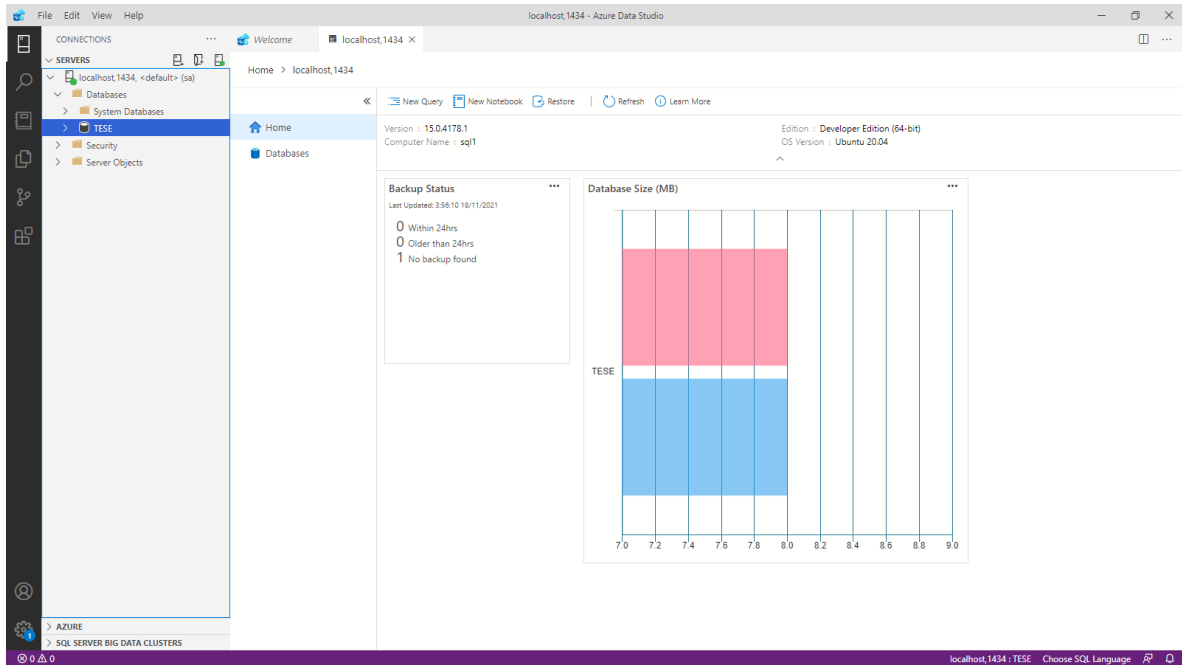
```
Administrador: Símbolo del sistema - docker exec -it sql1 "bash"
C:\WINDOWS\system32>docker exec -it sql1 "bash"
mssql@sql1:/$ ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
mssql@sql1:/$ /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password123."
1> CREATE DATABASE TESE
2> GO
1> USE TESE
2> GO
Changed database context to 'TESE'.
1>
```

6. Agregamos registros a nuestra base de datos para comprobar que esta correcta.

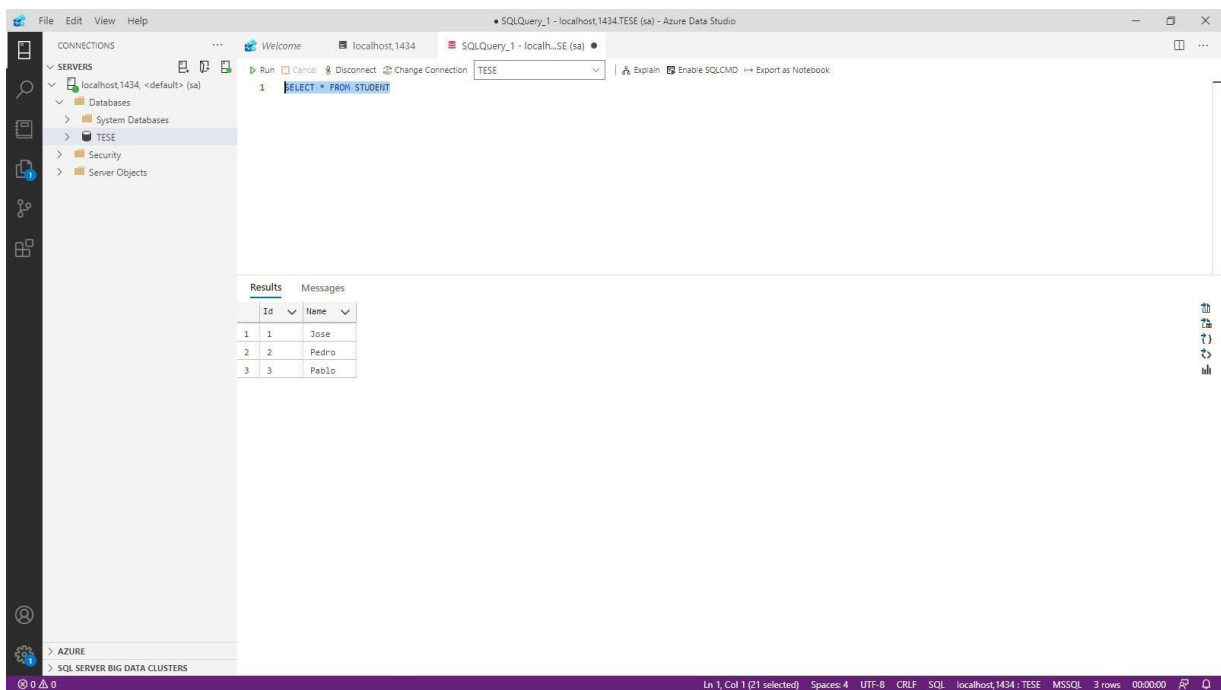


```
Administrador: Símbolo del sistema - docker exec -it sql1 "bash"
C:\WINDOWS\system32>docker exec -it sql1 "bash"
mssql@sql1:/$ ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
mssql@sql1:/$ /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password123."
1> CREATE DATABASE TESE
2> GO
1> USE TESE
2> GO
Changed database context to 'TESE'.
1> CREATE TABLE STUDENT (Id INT, Name VARCHAR(50));
2> GO
1> INSERT INTO STUDENT (1,'jose');
2> INSERT INTO STUDENT (1,'Pedro');
3> INSERT INTO STUDENT (1,'Pedro');
4> go
Msg 102, Level 15, State 1, Server sql1, Line 1
Incorrect syntax near ';'.
Msg 102, Level 15, State 1, Server sql1, Line 2
Incorrect syntax near ';'.
Msg 102, Level 15, State 1, Server sql1, Line 3
Incorrect syntax near ';'.
1> INSERT INTO STUDENT VALUES (1,'Jose');
2> GO
(1 rows affected)
1>
```

7. Abrimos *Azure Data Studio* y entramos con el usuario sa y la contraseña Password123. Y tendremos que verificar nuestra base de datos.



8. Creamos un nuevo query y checamos que estén los datos



9. Tendremos que crear otro servidor

```
Administrador: Símbolo del sistema

Id      Name
-----
1       Jose
2       Pedro
3       Pablo

(3 rows affected)
1> exit
mssql@sql1:/$ exit
exit

C:\WINDOWS\system32>docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Password123." -p 1435:1433 --name sql2 -h sql2 -d mcr.
microsoft.com/mssql/server:2019-latest
1519d9f8bb4cdf0b278d6e1955dc2fe358f4a6d369c31217e62dac9ab1cbb82

C:\WINDOWS\system32>
```

10. Creamos otra conexión

Connection

Recent

Browse

Clear List

localhost,1434, <default> (sa)

Connection Details

Connection type

Microsoft SQL Server

Server

localhost,1435

Authentication type

SQL Login

User name

sa

Password

☐ Remember password

Database

<Default>

Server group

<Default>

Name (optional)

Advanced...

Connect

Cancel

11. Ejecutamos el Bash del segundo servidor creado.

```
Administrador: Símbolo del sistema - docker exec -it sql2 "bash"

1 Jose
2 Pedro
3 Pablo

(3 rows affected)
1> exit
mssql@sql1:/$ exit
exit
C:\WINDOWS\system32>docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Password123." -p 1435:1433 --name sql2 -h sql2 -d mcr.
microsoft.com/mssql/server:2019-latest
1519d9f8bb4cdf0b278d6e1955dc2fe358f4a6d369c31217e62dac9ab1cbb82

C:\WINDOWS\system32>docker exec -it sql2 "bash"
mssql@sql2:/$
```

12. Entramos al contenedor

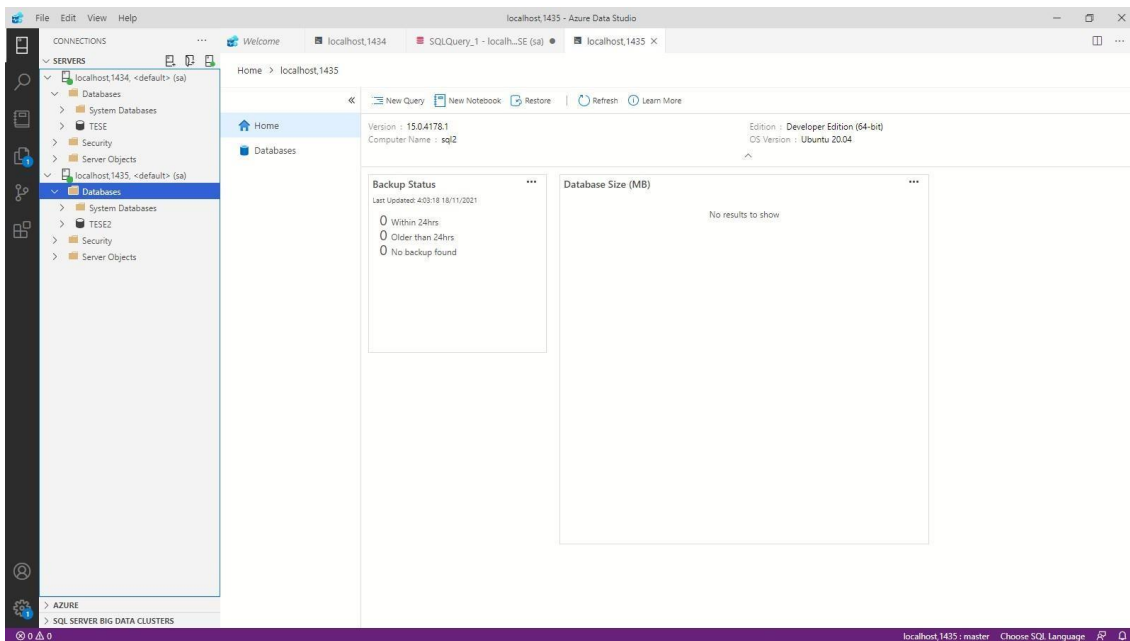
```
Administrador: Símbolo del sistema - docker exec -it sql2 "bash"

1 Jose
2 Pedro
3 Pablo

(3 rows affected)
1> exit
mssql@sql1:/$ exit
exit
C:\WINDOWS\system32>docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Password123." -p 1435:1433 --name sql2 -h sql2 -d mcr.
microsoft.com/mssql/server:2019-latest
1519d9f8bb4cdf0b278d6e1955dc2fe358f4a6d369c31217e62dac9ab1cbb82

C:\WINDOWS\system32>docker exec -it sql2 "bash"
mssql@sql2:/$ /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password123."
1>
```

13. Verificamos las bases de datos



14. Detenemos el servidor y lo borramos

```
Administrador: Símbolo del sistema
C:\WINDOWS\system32>docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Password123." -p 1435:1433 --name sql2 -h sql2 -d mcr.microsoft.com/mssql/server:2019-latest
1519d9f8bb4cdf0b278d6e1955dc2fe358f4a6d369c31217e62dac9ab1cbb82
C:\WINDOWS\system32>docker exec -it sql2 "bash"
mssql@sql2:/$ /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "Password123."
1> CREATE DATABASE TESE2
2> GO
1> quit
mssql@sql2:/$ exit
exit
C:\WINDOWS\system32>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
1519d9f8bb4c   mcr.microsoft.com/mssql/server:2019-latest  "/opt/mssql/bin/per..." 6 minutes ago  Up 6 minutes  0.0.0.0:1435->1433/tcp
3ba35eff203a   mcr.microsoft.com/mssql/server:2019-latest  "/opt/mssql/bin/per..." 24 minutes ago  Up 24 minutes  0.0.0.0:1434->1433/tcp
46be2cf9f099   docker/getting-started                 "/docker-entrypoint..." 40 minutes ago  Up 40 minutes  0.0.0.0:80->80/tcp
C:\WINDOWS\system32>docker stop 1519d9f8bb4c
1519d9f8bb4c
C:\WINDOWS\system32>docker rm 1519d9f8bb4c
1519d9f8bb4c
C:\WINDOWS\system32>
```


Practica 7

Tendremos que poner los siguientes comandos;

- *mkdir Docker*
- *cd Docker*



```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19044.1348]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>mkdir Docker

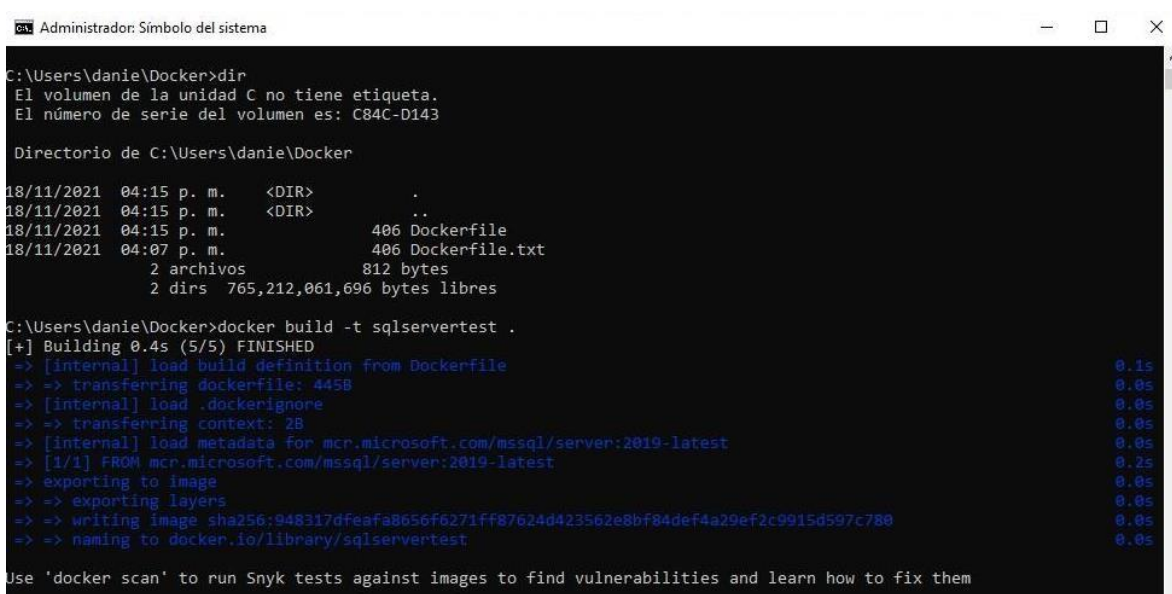
C:\WINDOWS\system32>cd Docker

C:\Windows\System32\Docker>
```

Guardamos el siguiente código en usuarios y carpeta docker

- *FROM mcr.microsoft.com/mssql/server:2019-latest*
- *ENV ACCEPT_EULA=Y*
- *ENV PASSWORD=Password123.*

Verificamos que tengamos nuestro archivo para crear la imagen



```
Administrador: Símbolo del sistema

C:\Users\danie\Docker>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: C84C-D143

Directorio de C:\Users\danie\Docker

18/11/2021  04:15 p. m.      <DIR>          .
18/11/2021  04:15 p. m.      <DIR>          ..
18/11/2021  04:15 p. m.                406 Dockerfile
18/11/2021  04:07 p. m.                406 Dockerfile.txt
                2 archivos                812 bytes
                2 dirs 765,212,061,696 bytes libres

C:\Users\danie\Docker>docker build -t sqlservertest .
[+] Building 0.4s (5/5) FINISHED
=> [internal] load build definition from Dockerfile                                0.1s
=> => transferring dockerfile: 445B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for mcr.microsoft.com/mssql/server:2019-latest      0.0s
=> [1/1] FROM mcr.microsoft.com/mssql/server:2019-latest                       0.2s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:948317dfeafa8656f6271ff87624d423562e8bf84def4a29ef2c9915d597c780 0.0s
=> => naming to docker.io/library/sqlservertest                                0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Verificamos que este el test

```
Administrador: Símbolo del sistema

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\danie\docker>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker/getting-started latest             eb9194091564       6 days ago         28.5MB
sqlservertest        latest             948317dfeafa       7 weeks ago        1.55GB
mcr.microsoft.com/mssql/server 2019-latest       80bdc8efc889       7 weeks ago        1.55GB
```

Ponemos el siguiente comando `docker run -p 1435:1433 --name sql2 -h sql2 -d sqlservertest`

```
Administrador: Símbolo del sistema

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\danie\docker>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker/getting-started latest             eb9194091564       6 days ago         28.5MB
sqlservertest        latest             948317dfeafa       7 weeks ago        1.55GB
mcr.microsoft.com/mssql/server 2019-latest       80bdc8efc889       7 weeks ago        1.55GB

C:\Users\danie\docker>docker run -p 1435:1433 --name sql2 -h sql2 -d sqlservertest
559fbd897ba0034d944b073a702d73e8e59be3e143a9b27a30e74671399818dc

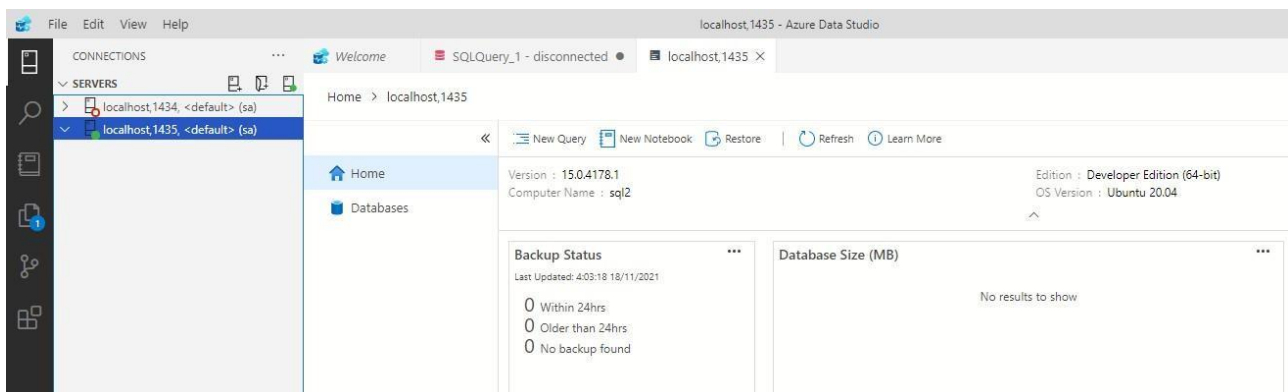
C:\Users\danie\docker>
```

Verificamos que este corriendo

```
Administrador: Símbolo del sistema

C:\Users\danie\docker>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
559fbd897ba0   sqlservertest "/opt/mssql/bin/perm..." 34 seconds ago Up 32 seconds 0.0.0.0:1435->1433/tcp   sql2
```

Verificamos en azure




```

C:\Users\danie\Simbolo del sistema
-> => exporting layers 0.05
-> => writing image sha256:a3667e8fb20727380cf921efa489363069dc4e205d273bdd8ea97c2e9e42fae6 0.05
-> => naming to docker.io/library/sqlservertest 0.05

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\danie\Docker>docker run -p 1435:1433 --name sql2 -h sql2 -d sqlservertest
f8ff577d0963b38c563b5fc6118851c797ef787ec7171e022931b71927a09a3d

C:\Users\danie\Docker>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                    NAMES
f8ff577d0963   sqlservertest  "/opt/mssql/bin/per..."  2 minutes ago  Up 2 minutes  0.0.0.0:1435->1433/tcp    sql2

C:\Users\danie\Docker>docker rm -f f8ff5
f8ff5

C:\Users\danie\Docker>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                    NAMES

C:\Users\danie\Docker>docker run -p 1435:1433 --name sql2 -h sql2 -d sqlservertest
1a6b5ee4cc15147396dfbf6f360027ab97e39b3264127c97cfbfed0032c25175

C:\Users\danie\Docker>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                    NAMES
1a6b5ee4cc15   sqlservertest  "/opt/mssql/bin/per..."  3 minutes ago  Up 2 minutes  0.0.0.0:1435->1433/tcp    sql2

C:\Users\danie\Docker>docker rm -f 1a6b
1a6b

```

```

C:\Administrador: Símbolo del sistema

:\Users\danie\Docker>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
a6b5ee4cc15147396dfbf6f360027ab97e39b3264127c97cfbfed0032c25175

:\Users\danie\Docker>docker run -p 1435:1433 --name sql2 -h sql2 -d sqlservertest
a6b5ee4cc15147396dfbf6f360027ab97e39b3264127c97cfbfed0032c25175

:\Users\danie\Docker>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
a6b5ee4cc15   sqlservertest  "/opt/mssql/bin/per..."  3 minutes ago  Up 2 minutes  0.0.0.0:1435->1433/tcp  sql2

:\Users\danie\Docker>docker rm -f 1a6b
1a6b

:\Users\danie\Docker>docker build -t sqlservertest .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 32B                                                0.0s
=> [internal] load .dockerignore                                                    0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for mcr.microsoft.com/mssql/server:2019-latest        0.0s
=> CACHED [1/1] FROM mcr.microsoft.com/mssql/server:2019-latest                  0.0s
=> exporting to image                                                              0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:a3667e8fb20727380cf921efa489363069dc4e205d273bdd8ea97c2e9e42fae6 0.0s
=> => naming to docker.io/library/sqlservertest                                  0.0s

se 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

:\Users\danie\Docker>

```

Practica 8

Primero tendremos que borrar la imagen de SQL Server Test

```
Administrator: Símbolo del sistema

Manage images

Commands:
  build      Build an image from a Dockerfile
  history    Show the history of an image
  import     Import the contents from a tarball to create a filesystem image
  inspect    Display detailed information on one or more images
  load       Load an image from a tar archive or STDIN
  ls         List images
  prune      Remove unused images
  pull       Pull an image or a repository from a registry
  push       Push an image or a repository to a registry
  rm         Remove one or more images
  save       Save one or more images to a tar archive (streamed to STDOUT by default)
  tag        Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.

C:\Users\danie\Docker>docker images
REPOSITORY              TAG          IMAGE ID      CREATED      SIZE
docker/getting-started  latest      eb9194091564  6 days ago  28.5MB
mcr.microsoft.com/mssql/server  2019-latest  80bdc8efc889  7 weeks ago  1.55GB
<none>                  <none>      948317dfeafa  7 weeks ago  1.55GB
sqlservertest           latest      a3667e8fb207  7 weeks ago  1.55GB

C:\Users\danie\Docker>docker rmi -f sqlservertest
Untagged: sqlservertest:latest
Deleted: sha256:a3667e8fb20727380cf921efa489363069dc4e205d273bdd8ea97c2e9e42fae6
```

Verificamos que ya no este

```
Administrator: Símbolo del sistema

inspect    Display detailed information on one or more images
load       Load an image from a tar archive or STDIN
ls         List images
prune      Remove unused images
pull       Pull an image or a repository from a registry
push       Push an image or a repository to a registry
rm         Remove one or more images
save       Save one or more images to a tar archive (streamed to STDOUT by default)
tag        Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.

C:\Users\danie\Docker>docker images
REPOSITORY              TAG          IMAGE ID      CREATED      SIZE
docker/getting-started  latest      eb9194091564  6 days ago  28.5MB
mcr.microsoft.com/mssql/server  2019-latest  80bdc8efc889  7 weeks ago  1.55GB
<none>                  <none>      948317dfeafa  7 weeks ago  1.55GB
sqlservertest           latest      a3667e8fb207  7 weeks ago  1.55GB

C:\Users\danie\Docker>docker rmi -f sqlservertest
Untagged: sqlservertest:latest
Deleted: sha256:a3667e8fb20727380cf921efa489363069dc4e205d273bdd8ea97c2e9e42fae6

C:\Users\danie\Docker>docker images
REPOSITORY              TAG          IMAGE ID      CREATED      SIZE
docker/getting-started  latest      eb9194091564  6 days ago  28.5MB
<none>                  <none>      948317dfeafa  7 weeks ago  1.55GB
mcr.microsoft.com/mssql/server  2019-latest  80bdc8efc889  7 weeks ago  1.55GB
```

Después crearemos una nueva imagen de SQL

```
Administrador: Símbolo del sistema

mcr.microsoft.com/mssql/server 2019-latest 80bdc8efc889 7 weeks ago 1.55GB
<none> <none> 948317dfeafa 7 weeks ago 1.55GB
sqlservertest latest a3667e8fb207 7 weeks ago 1.55GB

C:\Users\danie\Docker>docker rmi -f sqlservertest
Untagged: sqlservertest:latest
Deleted: sha256:a3667e8fb20727380cf921efa489363069dc4e205d273bdd8ea97c2e9e42fae6

C:\Users\danie\Docker>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
docker/getting-started latest eb9194091564 6 days ago 28.5MB
<none> <none> 948317dfeafa 7 weeks ago 1.55GB
mcr.microsoft.com/mssql/server 2019-latest 80bdc8efc889 7 weeks ago 1.55GB

C:\Users\danie\Docker>docker build -t sqlservertest .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for mcr.microsoft.com/mssql/server:2019-latest 0.0s
=> CACHED [1/1] FROM mcr.microsoft.com/mssql/server:2019-latest 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:a3667e8fb20727380cf921efa489363069dc4e205d273bdd8ea97c2e9e42fae6 0.0s
=> => naming to docker.io/library/sqlservertest 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\danie\Docker>
```

Ejecutamos la imagen

```
Administrador: Símbolo del sistema

=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for mcr.microsoft.com/mssql/server:2019-latest 0.0s
=> CACHED [1/1] FROM mcr.microsoft.com/mssql/server:2019-latest 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:a3667e8fb20727380cf921efa489363069dc4e205d273bdd8ea97c2e9e42fae6 0.0s
=> => naming to docker.io/library/sqlservertest 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\danie\Docker>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
docker/getting-started latest eb9194091564 6 days ago 28.5MB
<none> <none> 948317dfeafa 7 weeks ago 1.55GB
sqlservertest latest a3667e8fb207 7 weeks ago 1.55GB
mcr.microsoft.com/mssql/server 2019-latest 80bdc8efc889 7 weeks ago 1.55GB

C:\Users\danie\Docker>docker run -p 1434:14133 --name sql1 -h sql1 -d sqlservertest
docker: Error response from daemon: Conflict. The container name "/sql1" is already in use by container "3ba35eff203ad21eb527567ae6ee0ac430489f3cc7df2bf505f35420fc4fbef7". You have to remove (or rename) that container to be able to reuse the name.
See 'docker run --help'.

C:\Users\danie\Docker>docker run -p 1434:14133 --name sql2 -h sql2 -d sqlservertest
a78af7c5e7f47937153a500a94df3d49ad15f9148e539400271d84bf667cc569

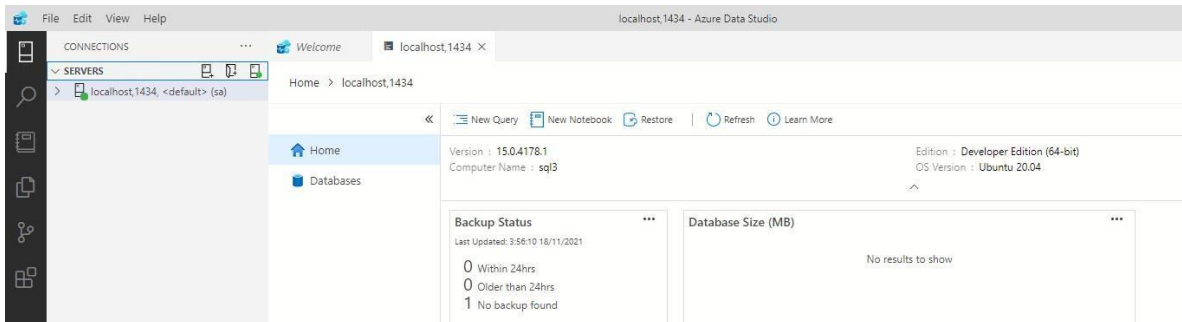
C:\Users\danie\Docker>
```


Verificamos que funcione

```
Administrador: Símbolo del sistema - docker run -p 1434:1433 --name sql3 -h sql3 sqlservertest

2021-11-19 00:43:02.26 spid12s Database 'model' running the upgrade step from version 899 to version 900.
2021-11-19 00:43:02.26 spid9s Database 'msdb' running the upgrade step from version 899 to version 900.
2021-11-19 00:43:02.29 spid12s Database 'model' running the upgrade step from version 900 to version 901.
2021-11-19 00:43:02.30 spid9s Database 'msdb' running the upgrade step from version 900 to version 901.
2021-11-19 00:43:02.36 spid12s Database 'model' running the upgrade step from version 901 to version 902.
2021-11-19 00:43:02.38 spid9s Database 'msdb' running the upgrade step from version 901 to version 902.
2021-11-19 00:43:02.41 spid12s Database 'model' running the upgrade step from version 902 to version 903.
2021-11-19 00:43:02.42 spid12s Database 'model' running the upgrade step from version 903 to version 904.
2021-11-19 00:43:02.55 spid12s Clearing tempdb database.
2021-11-19 00:43:03.39 spid12s [2]. Feature Status: PVS: 0. CTR: 0. ConcurrentPFSUpdate: 1.
2021-11-19 00:43:03.42 spid12s Starting up database 'tempdb'.
2021-11-19 00:43:03.62 spid12s The tempdb database has 1 data file(s).
2021-11-19 00:43:03.69 spid27s The Service Broker endpoint is in disabled or stopped state.
2021-11-19 00:43:03.71 spid27s The Database Mirroring endpoint is in disabled or stopped state.
2021-11-19 00:43:03.75 spid27s Service Broker manager has started.
2021-11-19 00:43:03.77 spid9s Database 'msdb' running the upgrade step from version 902 to version 903.
2021-11-19 00:43:03.78 spid9s Database 'msdb' running the upgrade step from version 903 to version 904.
2021-11-19 00:43:03.95 spid9s Recovery is complete. This is an informational message only. No user action is required.
2021-11-19 00:43:03.98 spid31s The default language (LCID 0) has been set for engine and full-text services.
2021-11-19 00:43:05.27 spid31s The tempdb database has 8 data file(s).
2021-11-19 00:43:16.67 spid52 Attempting to load library 'xpstar.dll' into memory. This is an informational message only. No user action is required.
2021-11-19 00:43:16.72 spid52 Using 'xpstar.dll' version '2019.150.4178' to execute extended stored procedure 'xp_instance_regread'. This is an informational message only; no user action is required.
2021-11-19 00:48:07.13 spid61 Attempting to load library 'xplog70.dll' into memory. This is an informational message only. No user action is required.
2021-11-19 00:48:07.18 spid61 Using 'xplog70.dll' version '2019.150.4178' to execute extended stored procedure 'xp_msver'. This is an informational message only; no user action is required.
```

Verificamos que se creó en la base de datos en Azure



Practica 9

Primero tendremos que crear un registro de contenedores con el programa *Container Registry*

Asignamos los parámetros

[Inicio](#) > [Crear un recurso](#) > [Container Registry](#) >

Crear Registro de contenedor ...

[Datos básicos](#) [Redes](#) [Cifrado](#) [Etiquetas](#) [Revisar y crear](#)

Azure Container Registry permite compilar, almacenar y administrar artefactos e imágenes de contenedor en un registro privado para todos los tipos de implementación de contenedor. Use registros de contenedor de Azure con sus canalizaciones de desarrollo e implementación de contenedores actuales. Use Azure Container Registry Tasks para compilar imágenes de contenedor en Azure a petición, o bien automatizar compilaciones desencadenadas por actualizaciones del código fuente, actualizaciones de la imagen base de un contenedor o temporizadores. [Más información](#)

Detalles del proyecto

Suscripción *

Grupo de recursos * [Crear nuevo](#)

Detalles de instancia

Nombre del Registro * [.azurecr.io](#)

Ubicación *

Zonas de disponibilidad ⓘ ☐ Habilitado

 La característica Availability Zones está habilitada en los registros Premium y en las regiones que admiten zonas de disponibilidad. [Más información](#)

SKU * ⓘ

[Revisar y crear](#) [< Anterior](#) [Siguiente: Redes >](#)

Teniéndolo listo ingresaremos el siguiente comando; *docker tag sqlservertest dany16.azurecr.io/sqlserver2019:v1.0*

```
Símbolo del sistema
pause      Pause all processes within one or more containers
port       List port mappings or a specific mapping for the container
ps         List containers
pull       Pull an image or a repository from a registry
push       Push an image or a repository to a registry
rename     Rename a container
restart    Restart one or more containers
rm         Remove one or more containers
rmi        Remove one or more images
run        Run a command in a new container
save       Save one or more images to a tar archive (streamed to STDOUT by default)
search     Search the Docker Hub for images
start      Start one or more stopped containers
stats      Display a live stream of container(s) resource usage statistics
stop       Stop one or more running containers
tag        Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top        Display the running processes of a container
unpause    Unpause all processes within one or more containers
update     Update configuration of one or more containers
version    Show the Docker version information
wait       Block until one or more containers stop, then print their exit codes

Run 'docker COMMAND --help' for more information on a command.

To get more help with docker, check out our guides at https://docs.docker.com/go/guides/
```

Verificamos que este la imagen

```
C:\Users\danie\docker>docker images
REPOSITORY              TAG          IMAGE ID      CREATED      SIZE
docker/getting-started  latest       eb9194091564  6 days ago  28.5MB
mcr.microsoft.com/mssql/server  2019-latest  80bdc8efc889  7 weeks ago  1.55GB
dany16.azurecr.io/sqlserver2019  v1.0        a3667e8fb207  7 weeks ago  1.55GB
sqlservertest           latest       a3667e8fb207  7 weeks ago  1.55GB
```

Haremos un push

```
The push refers to repository [dany16.azurecr.io/sqlserver2019]
a68c83c36a8d: Pushing [====>] 4.349MB/70.82MB
7fd44c220a99: Pushing [>] 2.726MB/1.236GB
08c020a644f6: Pushing [====>] 12.2MB/168.2MB
c8a9d6642668: Pushed
4942a1abcbfa: Pushing [=====>] 10.83MB/72.78MB
```

Vemos que ya se hizo todo el proceso

```
The push refers to repository [dany16.azurecr.io/sqlserver2019]
a68c83c36a8d: Pushed
7fd44c220a99: Pushed
08c020a644f6: Pushed
c8a9d6642668: Pushed
4942a1abcbfa: Pushed
v1.0: digest: sha256:171e619f6f460e0408f620b48a0fe3b64bada92ce3fc840c2d3f5375ebce06a4 size: 1373
```

Ejecutamos la instancia

sqlserver2019

Actualizar Eliminar repositorio

Información esencial

Repositorio: sqlserver2019

Recuento de etiquetas: 1

Fecha de la última actualización: 18/11/2021 22:34 GMT-6

Recuento de manifiesto: 1

Buscar la opción para filtrar etiquetas...

Etiquetas

v1.0

- Crear webhook
- Eliminar
- Quitar etiqueta
- Ejecutar instancia**
- Implementar en aplicación web

Configuramos la instancia

Inicio > Dany16 > sqlserver2019

sqlserver2019

Actualizar Eliminar repositorio

Información esencial Vista JSON

Repositorio: sqlserver2019

Fecha de la última actualización: 18/11/2021 22:34 GMT-6

Recuento de etiquetas: 1

Recuento de manifiesto: 1

Buscar la opción para filtrar etiquetas...

Etiquetas

v1.0

Crear instancia de contenedor

Nombre de contenedor *: sqlservertestv1 ✓

Imagen de contenedor: dany16.azurecr.io/sqlserver2019:v1.0

Tipo de SO: ☒ Linux ☐ Windows

Suscripción *: Azure para estudiantes

Grupo de recursos *: Tese

Ubicación *: Centro-Sur de EE. UU.

Número de núcleos: 2

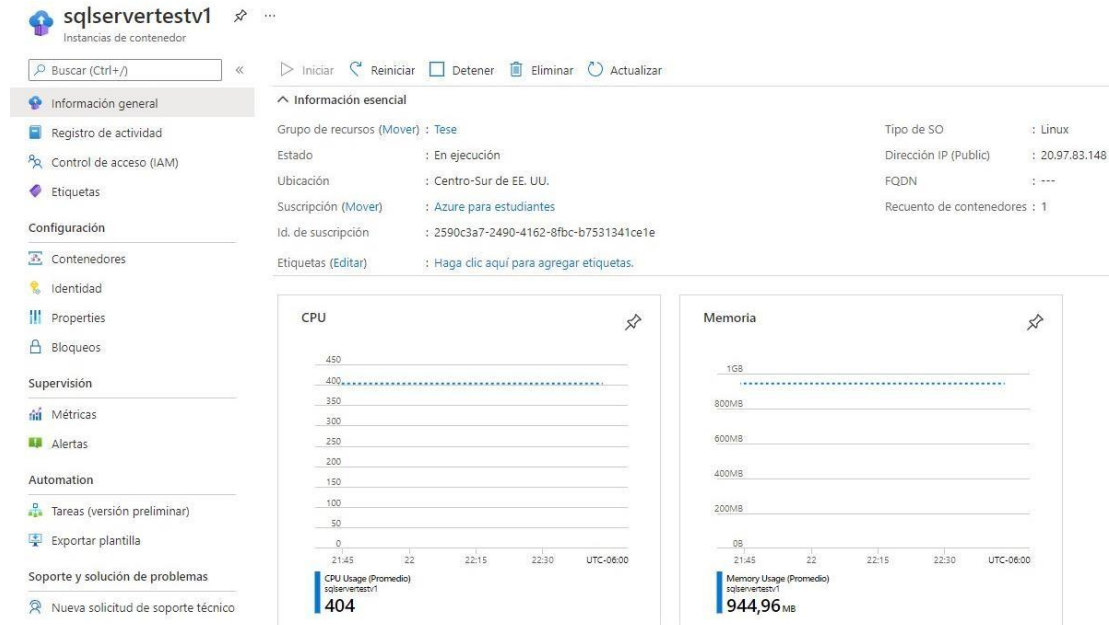
Memoria (GB) *: 4 ✓

Dirección IP pública: ☒ Sí ☐ No

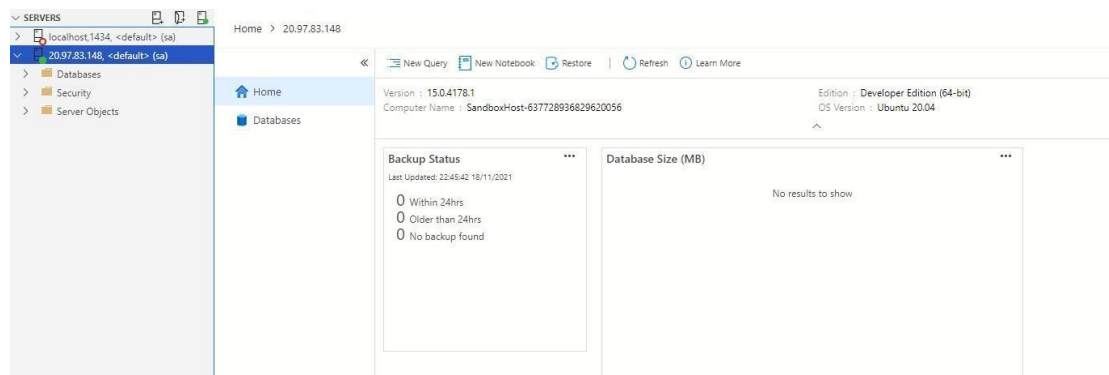
Puerto *: 1433 ✓

Ejecutar instancia

Verificamos que corra



Nos conectaremos al servidor



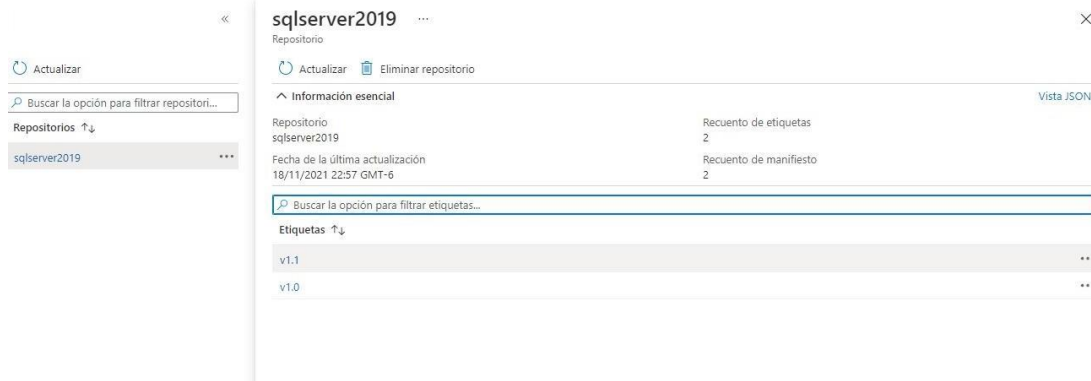
Reconstruimos la imagen con el siguiente código

```
+ Building 3.5s (4/5)
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 700B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for mcr.microsoft.com/mssql/server:2019-latest 0.0s
=> CACHED [1/2] FROM mcr.microsoft.com/mssql/server:2019-latest 0.0s
=> [2/2] RUN /opt/mssql/bin/sqlservr --accept-eula & ) | grep -q "Service Broker manager has started" && /opt/m 3.4s
```


Ingresamos el tag y el push con la versión 1.1

```
The push refers to repository [dany16.azurecr.io/sqlserver2019]
7aef8c37bc59: Pushing [==>] 3.312MB/73.67MB
a68c83c36a8d: Preparing
7fd44c220a99: Layer already exists
08c020a644f6: Preparing
c8a9d6642668: Layer already exists
4942a1abcbfa: Waiting
```

Checamos que este el repositorio con la versión 1.1



sqlserver2019

Repositorio

Actualizar Eliminar repositorio

Información esencial

Repositorio	Recuento de etiquetas
sqlserver2019	2

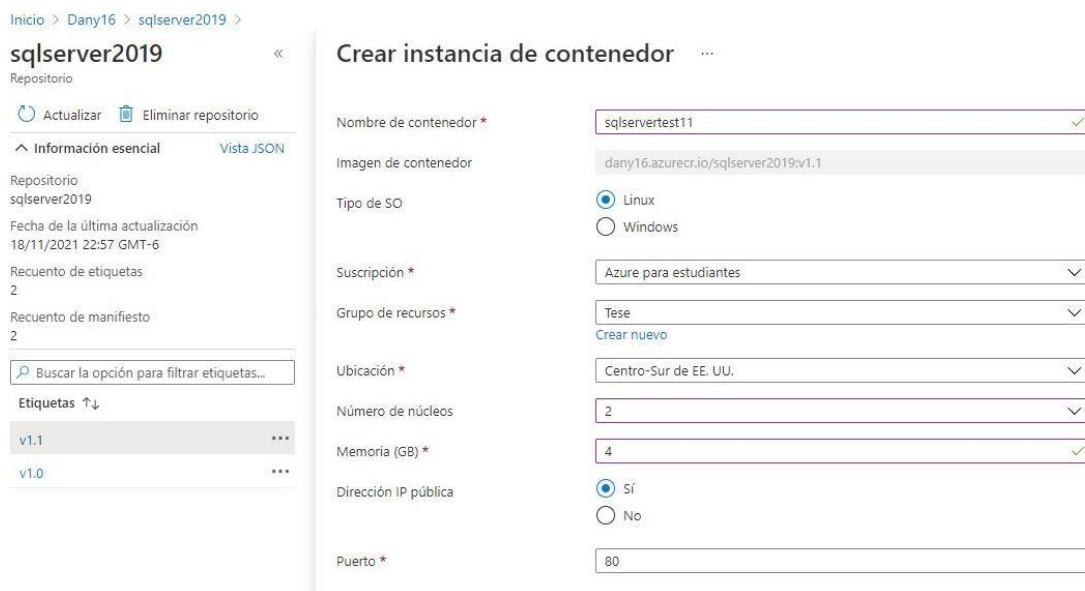
Fecha de la última actualización	Recuento de manifiesto
18/11/2021 22:57 GMT-6	2

Buscar la opción para filtrar etiquetas...

Etiquetas

v1.1	...
v1.0	...

Creamos una nueva instancia para esta nueva versión



Inicio > Dany16 > sqlserver2019 >

sqlserver2019

Repositorio

Actualizar Eliminar repositorio

Información esencial Vista JSON

Repositorio

sqlserver2019

Fecha de la última actualización

18/11/2021 22:57 GMT-6

Recuento de etiquetas

2

Recuento de manifiesto

2

Buscar la opción para filtrar etiquetas...

Etiquetas

v1.1	...
v1.0	...

Crear instancia de contenedor

Nombre de contenedor *

sqlservertest11

Imagen de contenedor

dany16.azurecr.io/sqlserver2019:v1.1

Tipo de SO

☒ Linux

☐ Windows

Suscripción *

Azure para estudiantes

Grupo de recursos *

Tese

Crear nuevo

Ubicación *

Centro-Sur de EE. UU.

Número de núcleos

2

Memoria (GB) *

4

Dirección IP pública

☒ Sí

☐ No

Puerto *

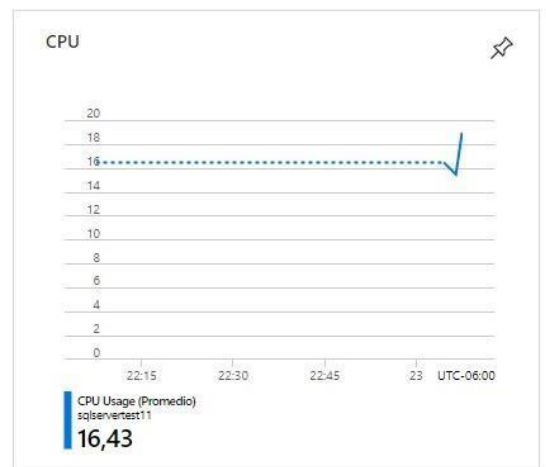
80

Esperamos a que se cree

[▶ Iniciar](#) [↺ Reiniciar](#) [□ Detener](#) [🗑 Eliminar](#) [🔄 Actualizar](#)

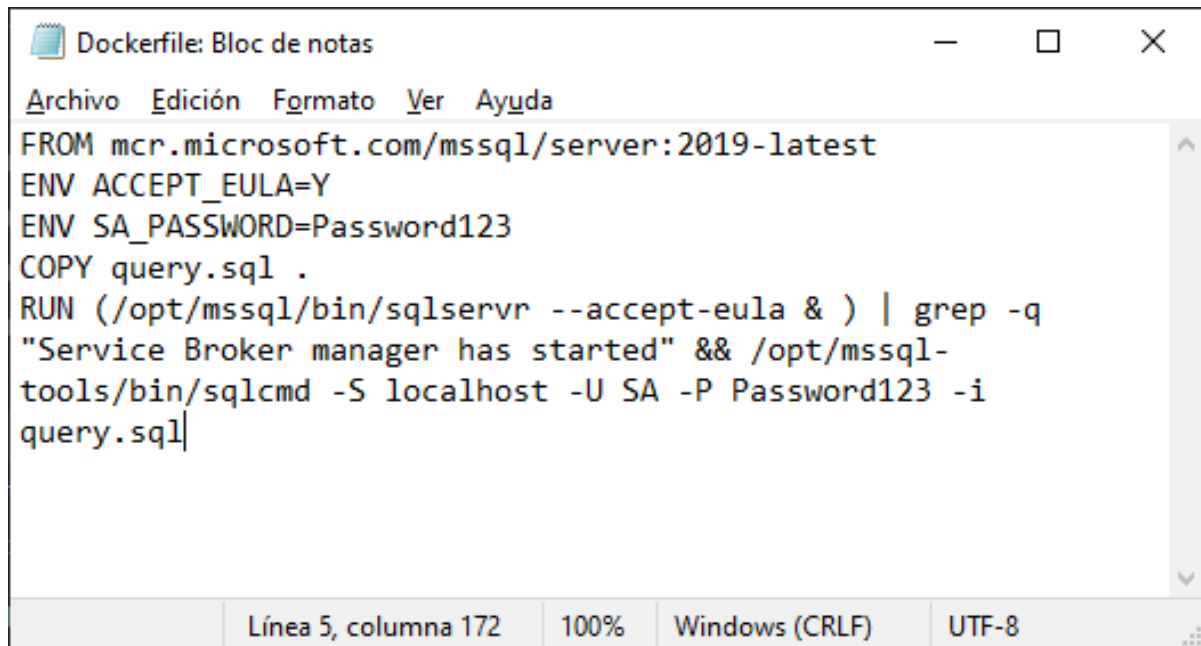
^ Información esencial

Grupo de recursos (Mover)	: Tese	Tipo de SO	: Linux
Estado	: En ejecución	Dirección IP (Public)	: 20.189.27.130
Ubicación	: Centro-Sur de EE. UU.	FQDN	: ---
Suscripción (Mover)	: Azure para estudiantes	Recuento de contenedores	: 1
Id. de suscripción	: 2590c3a7-2490-4162-8fbc-b7531341ce1e		
Etiquetas (Editar)	: Haga clic aquí para agregar etiquetas.		



Practica 10 – Carga de un Docker con un archivo

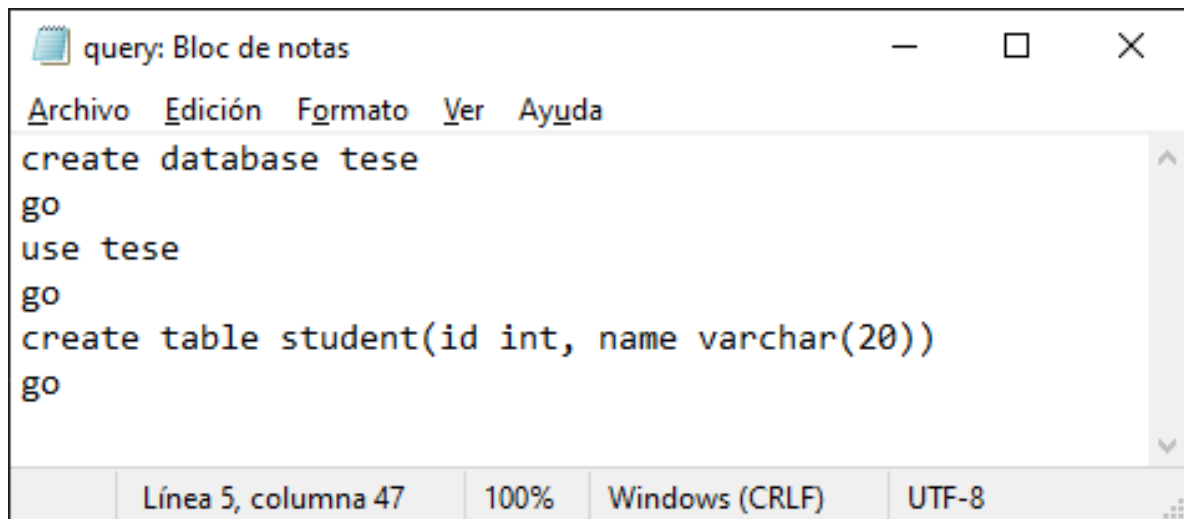
Primero tendremos que editar el Dockerfile de la siguiente manera:



```
FROM mcr.microsoft.com/mssql/server:2019-latest
ENV ACCEPT_EULA=Y
ENV SA_PASSWORD=Password123
COPY query.sql .
RUN (/opt/mssql/bin/sqlservr --accept-eula & ) | grep -q
"Service Broker manager has started" && /opt/mssql-
tools/bin/sqlcmd -S localhost -U SA -P Password123 -i
query.sql
```

Línea 5, columna 172 100% Windows (CRLF) UTF-8

En donde -i nos indica cuál será el archivo de entrada, en este caso query.sql:



```
create database tese
go
use tese
go
create table student(id int, name varchar(20))
go
```

Línea 5, columna 47 100% Windows (CRLF) UTF-8

Actualizamos nuestra imagen con el nuevo Dockerfil y el archivo SQL:

```
Seleccionar Símbolo del sistema

C:\User\l1dan>Docker>docker build -t inv .
[+] Building 48.9s (8/8) FINISHED
=> [internal] load build definition from Dockerfile 0.4s
=> => transferring dockerfile: 325B 0.0s
=> [internal] load .dockerignore 0.6s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for mcr.microsoft.com/mssql/server:2019-latest 0.0s
=> [internal] load build context 0.9s
=> => transferring context: 126B 0.3s
=> CACHED [1/3] FROM mcr.microsoft.com/mssql/server:2019-latest 0.0s
=> [2/3] COPY query.sql . 0.9s
=> [3/3] RUN (/opt/mssql/bin/sqlservr --accept-eula & ) | grep -q "Service Broker manager has started" && /opt/ 40.6s
=> exporting to image 4.6s
=> => exporting layers 4.1s
=> => writing image sha256:6074dfde5189e3ac5eea914fab2496feaf31a20eda1454fffeebf0bfb24ed4ea 0.1s
=> => naming to docker.io/library/inv 0.1s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\User\l1dan>Docker>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
inv latest 6074dfde5189 About a minute ago 1.64GB
edgarcr.azurecr.io/sqlserver2019 v1.0 d07746e11da5 2 hours ago 1.64GB
docker101tutorial latest 7ea0bc62d57e 13 hours ago 28.5MB
alpine/git latest 0deb7380d708 4 weeks ago 27.4MB
mcr.microsoft.com/mssql/server 2019-latest 80bdc8efc889 7 weeks ago 1.55GB

C:\User\l1dan>Docker>
```

Registramos e insertamos la imagen a CR:

```
Símbolo del sistema

=> [3/3] RUN (/opt/mssql/bin/sqlservr --accept-eula & ) | grep -q "Service Broker manager has started" && /opt/ 40.6s
=> exporting to image 4.6s
=> => exporting layers 4.1s
=> => writing image sha256:6074dfde5189e3ac5eea914fab2496feaf31a20eda1454fffeebf0bfb24ed4ea 0.1s
=> => naming to docker.io/library/inv 0.1s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\User\l1dan>Docker>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
inv latest 6074dfde5189 About a minute ago 1.64GB
edgarcr.azurecr.io/sqlserver2019 v1.0 d07746e11da5 2 hours ago 1.64GB
docker101tutorial latest 7ea0bc62d57e 13 hours ago 28.5MB
alpine/git latest 0deb7380d708 4 weeks ago 27.4MB
mcr.microsoft.com/mssql/server 2019-latest 80bdc8efc889 7 weeks ago 1.55GB

C:\User\l1dan>Docker>docker tag inv edgarcr.azurecr.io/sqlserver2019:v1.4

C:\User\l1dan>Docker>docker push edgarcr.azurecr.io/sqlserver2019:v1.4
The push refers to repository [edgarcr.azurecr.io/sqlserver2019]
4faa341f7d89: Pushed
90bf64261bdf: Pushed
a68c83c36a8d: Layer already exists
7fd44c220a99: Layer already exists
08c020a644f6: Layer already exists
c8a9d6642668: Layer already exists
4942a1abcbfa: Layer already exists
v1.4: digest: sha256:e4f13710673b835b7dc044d74e2d66ab4cd9bd721d4d08b36d79bdb56fb70c2e size: 1791

C:\User\l1dan>Docker>
```

Creamos su instancia:

The screenshot shows the 'Crear instancia de contenedor' (Create container instance) page in the Azure portal for the 'sqlserver2019' repository. The left sidebar shows repository details: 'Actualizar', 'Eliminar repositorio', 'Información esencial', 'Vista JSON', 'Repositorio: sqlserver2019', 'Fecha de la última actualización: 19/11/2021 01:26 GMT-6', 'Recuento de etiquetas: 2', and 'Recuento de manifiesto: 2'. The main form fields are: 'Nombre de contenedor *' (practica10), 'Imagen de contenedor' (azurecr.azurecr.io/sqlserver2019v1.4), 'Tipo de SO' (Linux selected), 'Suscripción *' (Azure para estudiantes), 'Grupo de recursos *' (TESE), 'Ubicación *' (Centro-Sur de EE. UU.), 'Número de núcleos' (2), 'Memoria (GB) *' (4), 'Dirección IP pública' (Sí selected), and 'Puerto *' (1433).

Nos conectamos a través de Azure Data Studio:

The screenshot shows the 'Connection Details' dialog box in Azure Data Studio. The fields are: 'Connection type' (Microsoft SQL Server), 'Server' (20.97.82.221), 'Authentication type' (SQL Login), 'User name' (SA), 'Password' (masked with dots), 'Remember password' (unchecked), 'Database' (<Default>), 'Server group' (<Default>), and 'Name (optional)' (empty). There is an 'Advanced...' button at the bottom right of the dialog. At the very bottom of the image, there are 'Connect' and 'Cancel' buttons.

Se creó la base de datos y la tabla solicitada

