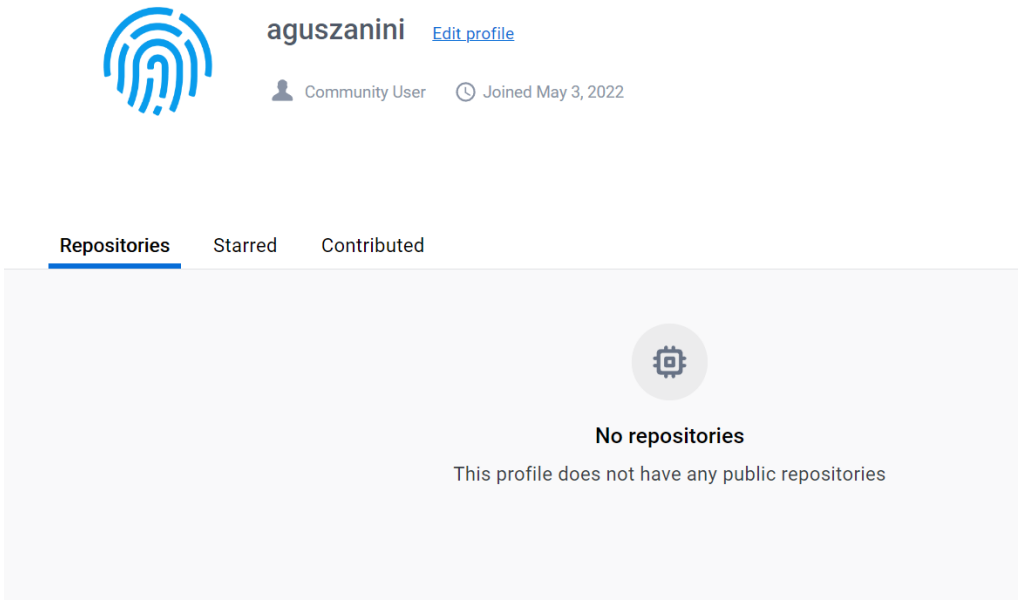
TP 2 INGENIERÍA DE SOFTWARE 3

1. Instalar Docker

```
C:\Users\Usuario>docker version
Client:
 Cloud integration: 1.0.17
 Version:           20.10.7
 API version:       1.41
 Go version:        go1.16.4
 Git commit:        f0df350
 Built:             Wed Jun  2 12:00:56 2021
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.7
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.13.15
  Git commit:       b0f5bc3
  Built:            Wed Jun  2 11:54:58 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.6
  GitCommit:        d71fcd7d8303cbf684402823e425e9dd2e99285d
 runc:
  Version:          1.0.0-rc95
  GitCommit:        b9ee9c6314599f1b4a7f497e1f1f856fe433d3b7
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

2. Registrarse en DockerHub



3. Obtener imagen BusyBox y mostrar imagenes

```
C:\Users\Usuario\Desktop\facu\4to\inge de software 3\prueba\SimpleWebAPI>docker image ls
REPOSITORY                          TAG        IMAGE ID       CREATED       SIZE
mywebapi                            latest     d0b71f091ccb   7 days ago    216MB
k8s.gcr.io/kube-apiserver           v1.21.2    106ff58d4308   2 years ago   126MB
k8s.gcr.io/kube-proxy               v1.21.2    a6ebd1c1ad98   2 years ago   131MB
k8s.gcr.io/kube-scheduler           v1.21.2    f917b8c8f55b   2 years ago   50.6MB
k8s.gcr.io/kube-controller-manager  v1.21.2    ae24db9aa2cc   2 years ago   120MB
docker/desktop-vpnkit-controller    v2.0       8c2c38aa676e   2 years ago   21MB
docker/desktop-storage-provisioner  v2.0       99f89471f470   2 years ago   41.9MB
k8s.gcr.io/pause                    3.4.1      0f8457a4c2ec   2 years ago   683kB
k8s.gcr.io/coredns/coredns          v1.8.0     296a6d5035e2   2 years ago   42.5MB
k8s.gcr.io/etcd                     3.4.13-0   0369cf4303ff   2 years ago   253MB

C:\Users\Usuario\Desktop\facu\4to\inge de software 3\prueba\SimpleWebAPI>
```

Símbolo del sistema

```
C:\Users\Usuario>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:3fbc632167424a6d997e74f52b878d7cc478225cffac6bc977eedfe51c7f4e79
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest

C:\Users\Usuario>
```

4. Ejecutando contenedores

```
C:\Users\Usuario>docker run busybox

C:\Users\Usuario>
```

No se obtiene ningun resultado porque busybox no devuelve nada, solo se prende y se apaga.

Símbolo del sistema

```
C:\Users\Usuario>docker run busybox echo "hola mundo"
hola mundo

C:\Users\Usuario>
```

```
C:\Users\Usuario>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS                      NAMES
74ed129291d7   mywebapi   "dotnet SimpleWebAPI…"   39 seconds ago   Up 39 seconds   80/tcp, 443/tcp, 5254/tcp  myapi

C:\Users\Usuario>
```

Símbolo del sistema

```
C:\Users\Usuario>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\Usuario>docker ps -a
CONTAINER ID   IMAGE               COMMAND                  CREATED             STATUS                          PORTS

                                NAMES
7fd0373891c8   busybox             "echo 'hola mundo'"      About a minute ago   Exited (0) About a minute ago

                                stupefied_maxwell
ef17b73869b9   busybox             "sh"                     6 minutes ago        Exited (0) 6 minutes ago
```

Con docker ps –a obtenemos todos los contenedores, ya sea que estén prendidos o no.

5. Ejecutando en modo interactivo

Símbolo del sistema - docker run -it busybox sh

```
C:\Users\Usuario>docker run -it busybox sh
/ #
/ # ps
PID   USER     TIME  COMMAND
    1 root      0:00 sh
    7 root      0:00 ps
/ # uptime
 01:05:00 up 15 min,  0 users,  load average: 0.81, 0.42, 0.29
/ # free
              total        used        free      shared  buff/cache   available
Mem:       13046260     1266876     9866688      415936     1912696    11129152
Swap:       4194304           0     4194304
/ # ls -l /
total 40
drwxr-xr-x    2 root     root         12288 Jul 17 18:30 bin
drwxr-xr-x    5 root     root           360 Aug 16 01:04 dev
drwxr-xr-x    1 root     root          4096 Aug 16 01:04 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  270 root     root             0 Aug 16 01:04 proc
drwx------    1 root     root          4096 Aug 16 01:04 root
dr-xr-xr-x   11 root     root             0 Aug 16 01:04 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
/ #
```

6. Borrando contenedores terminados

```
C:\Users\Usuario>docker rm nifty_davinci
nifty_davinci

C:\Users\Usuario>
```

7. Construir una imagen

```
C:\Windows\System32\cmd.exe
C:\Users\Usuario\Desktop\facu\4to\inge de software 3\prueba\SimpleWebAPI>docker build -t mywebapi .
[+] Building 1.4s (18/18) 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/dotnet/sdk:7.0                                       1.1s
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0                                    1.2s
 => [internal] load build context                                                                       0.1s
 => => transferring context: 5.94kB                                                                     0.0s
 => [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:b4da8b1973fedf6261096a01450dd09a3ac327ff1236798ac7  0.0s
 => => resolve mcr.microsoft.com/dotnet/aspnet:7.0@sha256:b4da8b1973fedf6261096a01450dd09a3ac327ff1236798ac7b3019  0.0s
 => [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:746fb86d08ca172ae0880c585ea86d8682e297fca5e5ff269ebd  0.0s
 => => resolve mcr.microsoft.com/dotnet/sdk:7.0@sha256:746fb86d08ca172ae0880c585ea86d8682e297fca5e5ff269ebd31d793  0.0s
 => CACHED [base 2/2] WORKDIR /app                                                                      0.0s
 => CACHED [final 1/2] WORKDIR /app                                                                     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
 => CACHED [build 5/7] COPY . .                                                                         0.0s
 => CACHED [build 6/7] WORKDIR /src/SimpleWebAPI                                                        0.0s
 => CACHED [build 7/7] RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build                  0.0s
 => CACHED [publish 1/1] RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false  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:d0b71f091ccba478c18c518ab675c7e904f19a3b55f313dc88b04d7edc496654            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
```

8. Publicando puertos

```
Símbolo del sistema

C:\Users\Usuario>docker run --name myapi -d mywebapi
74ed129291d7fad7aabcf6ea9887e1186e8d56449f81b7f98e234c9eff5c14f0

C:\Users\Usuario>
```

```
Símbolo del sistema

C:\Users\Usuario>docker run --name myapi -d -p 80:80 -p 5254:5254 mywebapi
e510421f1219f47029703d67bce924a12be1e641e8adbf0ccfad724fc55a6bf5

C:\Users\Usuario>
```

Para poder acceder a la url exitosamente, tenemos que hacer el mapeo del puerto del host y el puerto que expone el contenedor.

9. Modificar Dockerfile

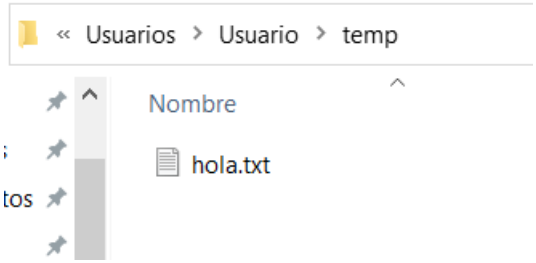
```
Dockerfile M ×
Dockerfile > ...
11  COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]
12  RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"
13  COPY . .
14  WORKDIR "/src/SimpleWebAPI"
15  RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build
16
17  FROM build AS publish
18  RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false
19
20  FROM base AS final
21  WORKDIR /app
22  COPY --from=publish /app/publish .
23  ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]
24  CMD ["/bin/bash"]
```

(en la captura no aparece comentada la línea del entrypoint, la comenté después)

```
root@7d605c55717f:/app# dotnet SimpleWebAPI.dll
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
```

10. Montando volúmenes

```
C:\>docker run -it --rm -p 80:80 -v C:\Users\Usuario\temp:/var/temp  mywebapi
root@7d605c55717f:/app# ls
Microsoft.AspNetCore.OpenApi.dll  SimpleWebAPI.pdb                      Swashbuckle.AspNetCore.SwaggerUI.dll
Microsoft.OpenApi.dll             SimpleWebAPI.runtimeconfig.json       appsettings.Development.json
SimpleWebAPI.deps.json            Swashbuckle.AspNetCore.Swagger.dll    appsettings.json
SimpleWebAPI.dll                  Swashbuckle.AspNetCore.SwaggerGen.dll web.config
root@7d605c55717f:/app# touch /var/temp/hola.txt
root@7d605c55717f:/app# ls /var/temp
hola.txt
root@7d605c55717f:/app#
```



11. Utilizando una base de datos

```
C:\Users\Usuario>docker exec -it my-postgres /bin/bash
root@3295d34335b7:/# psql -h localhost -U postgres
psql (9.4.26)
Type "help" for help.

postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
-----------+----------+----------+------------+------------+-----------------------
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
           |          |          |            |            | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
           |          |          |            |            | postgres=CTc/postgres
(3 rows)

postgres=# create database test;
CREATE DATABASE
postgres=# \connect test
You are now connected to database "test" as user "postgres".
test=# create table tabla_a (mensaje varchar(50));
CREATE TABLE
test=# insert into tabla_a (mensaje) values('hola mundo!');
INSERT 0 1
test=# select * from tabla_a;
   mensaje
-------------
 hola mundo!
(1 row)

test=# \q
root@3295d34335b7:/#
```

Con docker run lo que logramos es crear y ejecutar un nuevo contenedor a partir de una imagen, mientras que con docker exec lo que hace es permitirnos ejecutar comandos dentro del contenedor

12. No se pudo iniciar una conexión con Microsoft sqlserver

```
C:\Users\Usuario>docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=root" -p 1433:1433 --name sql-server-container -v C:\User
s\Usuario\.sqlserver:/var/opt/mssql -d mcr.microsoft.com/mssql/server:2019-latest
d07d5d99f502859e106b3b3b6aca9486bfb5c6e2413a989e028de2cdae9b2df8

C:\Users\Usuario>docker exec -it sql-server-container /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P root
Sqlcmd: Error: Microsoft ODBC Driver 17 for SQL Server : Login timeout expired.
Sqlcmd: Error: Microsoft ODBC Driver 17 for SQL Server : TCP Provider: Error code 0x2749.
Sqlcmd: Error: Microsoft ODBC Driver 17 for SQL Server : A network-related or instance-specific error has occurred while
 establishing a connection to SQL Server. Server is not found or not accessible. Check if instance name is correct and i
f SQL Server is configured to allow remote connections. For more information see SQL Server Books Online..

C:\Users\Usuario>
```