Meter Flask en Contenedor DOCKER

A.D.G.

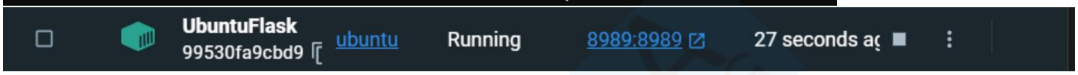
1) Crea el contendor desde un ubuntu con toda la configuración necesaria.

Descargamos la imagen del Ubuntu de Docker:

```
C:\Users\Andrei>docker pull ubuntu
```

Ejecutamos un contenedor de nombre UbuntuFlask en el puerto 8989 del contenedor y puerto 8989 del docker para inicializar la terminal interactiva para el ubuntu:

```
C:\Users\Andrei>docker run --name UbuntuFlask -p 8989:8989 -ti ubuntu
```



Actualizamos la imagen “Ubuntu” ya que puede estar muy desactualizada:

```
root@99530fa9cbd9:/# apt update
```

Instalamos el Python:

```
root@99530fa9cbd9:/# apt install python3
```

Instalamos el Administrador de paquetes de Python:

```
root@99530fa9cbd9:/# apt install python3-pip
```

Creamos la carpeta donde estará el Flask:

```
root@99530fa9cbd9:/# mkdir CarpetaFlask
```

Copio mi Flask (me situé antes en donde está la carpeta para mejor accesibilidad) y especificamos un nombre para el Contenedor que será “UbuntuFlask” y se copiará a la “CarpetaFlask” que tengo dentro de Downloads:

```
PS C:\Users\Andrei\Downloads> docker cp .\ADGFlask\ UbuntuFlask:/CarpetaFlask
```

Arrancamos antes el contenedor y accedemos al intérprete bash del contenedor de mi flask:

```
PS C:\Users\Andrei\Downloads> docker exec -ti UbuntuFlask bash
```

Nos situamos en la carpeta que está en el contenedor donde se encuentra mi proyecto flask, entramos en lo que sería la carpeta que se ha copiado (donde está mi flask) y comprobamos que esté todo bien:

```
root@99530fa9cbd9:/# cd CarpetaFlask/
root@99530fa9cbd9:/CarpetaFlask# ls
ADGFlask
root@99530fa9cbd9:/CarpetaFlask# cd ADGFlask/
root@99530fa9cbd9:/CarpetaFlask/ADGFlask# ls
__pycache__  app.py  bd.py  requirements.txt  static  tareas_productos.py  templates
```

Instalamos los requirements.txt necesarios para la ejecución e interpretación correcta de los códigos:

```
root@99530fa9cbd9:/CarpetaFlask/ADGFlask# pip install -r requirements.txt
Collecting Flask==2.2.2
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.5/101.5 KB 3.9 MB/s eta 0:00:00
Collecting PyMySQL==1.0.2
  Downloading PyMySQL-1.0.2-py3-none-any.whl (43 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.8/43.8 KB 2.3 MB/s eta 0:00:00
Collecting Werkzeug>=2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.7/232.7 KB 4.3 MB/s eta 0:00:00
Collecting click>=8.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 96.6/96.6 KB 4.3 MB/s eta 0:00:00
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2>=3.0
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.1/133.1 KB 3.7 MB/s eta 0:00:00
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Installing collected packages: PyMySQL, MarkupSafe, itsdangerous, click, Werkzeug, Jinja2, Flask
Successfully installed Flask-2.2.2 Jinja2-3.1.2 MarkupSafe-2.1.2 PyMySQL-1.0.2 Werkzeug-2.2.2 click-8.1.3 itsdangerous-2
.1.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packa
ge manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

Cambio el puerto de app.py de 3000 (que tenía antes) ya que he abierto para el puerto 8989:

```
root@99530fa9cbd9:/CarpetaFlask/ADGFlask# apt install nano
```

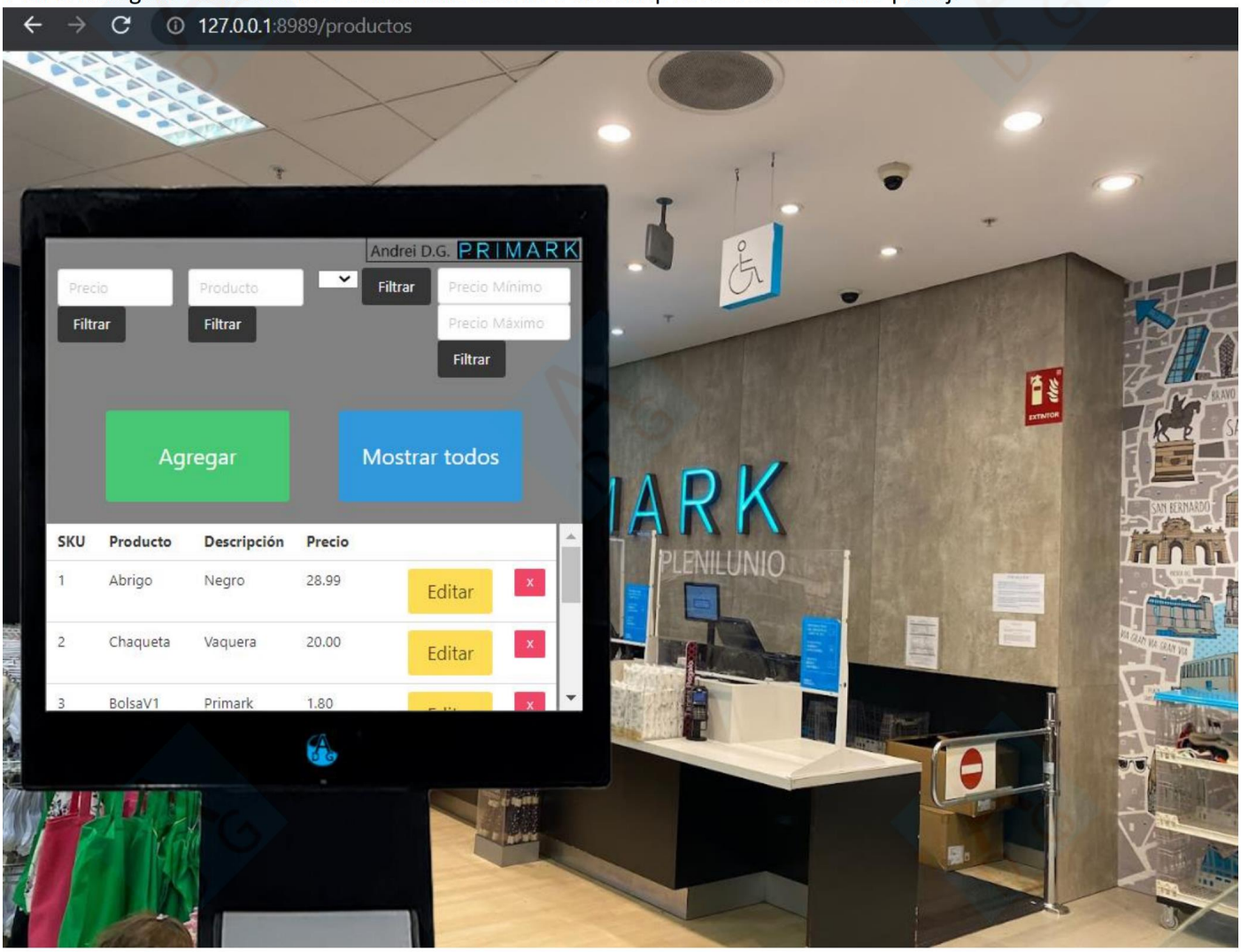
porque necesito para modificar en el app.py abajo el puerto y el host en 0.0.0.0 para vincular todas las interfaces. De lo contrario, vincularía solo un host si especificamos una IP en concreto:

```
root@99530fa9cbd9: /Carpeta
  GNU nano 6.2                                        app.py *
    precio_min = request.form["precio_min"]
    precio_max = request.form["precio_max"]
    rangosfiltrados = tareas_productos.filtro_productos_rango(precio_min, precio_max)
    return render_template("productos.html", productos=rangosfiltrados)


if __name__ == "__main__":
    app.run(port=8989, host="0.0.0.0", debug=True)

^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo     M-A Set Mark
^X Exit      ^R Read File  ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line M-E Redo     M-6 Copy
```

Arrancamos la aplicación flask desde:

```
root@99530fa9cbd9:/CarpetaFlask/ADGFlask# python3 ./app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8989
 * Running on http://172.17.0.2:8989
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 545-227-929
```

Y en el navegador accederíamos con localhost:8989 ó con mi ip 192.168.1.138:8989 por ej:



2) Crea la imagen, salvala y mandasela a otro compañero que la ejecute.

Salgo del intérprete de comandos bash y hacemos un commit para crear la imágen especificando el ENTRYPOINT que es como el archivo/iniciador que arrancará por defecto y usará python3 ejecutando la aplicación de “app.py” del contenedor “UbuntuFlask” y la “imagenubuntuflask”:

```
root@99530fa9cbd9:/CarpetaFlask/ADGFlask# exit
exit
PS C:\Users\Andrei\Downloads> docker commit -m "Commit del Flask 1" --change="ENTRYPOINT python3 ./CarpetaFlask/ADGFlask
/app.py" UbuntuFlask imagenubuntuflask
sha256:2a92efa7b9ce9559142b5cad6b6c471baf6130c6910c5fce39ad1dfc7d497c3f
```

Podemos ver las imágenes del docker por la interfaz gráfica ó por comando:

```
PS C:\Users\Andrei\Downloads> docker images
REPOSITORY                                TAG       IMAGE ID       CREATED          SIZE
imagenubuntuflask                         latest    2a92efa7b9ce   24 seconds ago   473MB
ubuntu                                    latest    58db3edaf2be   13 days ago      77.8MB
mysql                                     latest    b939d379d46e   3 weeks ago      514MB
gophish/gophish                           latest    35133b15ff61   2 months ago     230MB
portainer/portainer-docker-extension      2.16.2    5f11582196a4   2 months ago     287MB
phpmyadmin/phpmyadmin                     latest    4a4023c7e22a   9 months ago     510MB
mysql                                     8.0.16    de764ad211de   3 years ago      443MB
mariadb                                   10.3.9    23edf799b823   4 years ago      363MB
```

Arrancamos otravez el contenedor en los mismos puertos, otro nombre de contenedor e imágen (cuidado con puertos, nombre contenedor o imagen repetida y tal):

```
PS C:\Users\Andrei\Downloads> docker run -p 8989:8989 --name=ContenedorPruebaImagen1 imagenubuntuflask
docker: Error response from daemon: driver failed programming external connectivity on endpoint ContenedorPruebaImagen1
(23bd02d89ce358e0d55c549f1061ab93bed55695c32e9e306892009d4c1c57c8): Bind for 0.0.0.0:8989 failed: port is already alloca
ted.
```

Para probar: Paro el UbuntuFlask que usa 8989, y ejecuto otravez el anterior comando, también cambio el nombre del contenedor que liberé ese nombre ya:

```
PS C:\Users\Andrei\Downloads> docker run -p 8989:8989 --name=ContenedorPruebaImagen imagenubuntuflask
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8989
 * Running on http://172.17.0.2:8989
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 732-485-567
```

Guardamos la imagen “imagenubuntuflask” en el archivo “NombreImagen.tar”:

```
PS C:\Users\Andrei\Downloads\DescargaImagen> docker save imagenubuntuflask -o NombreImagen.tar
PS C:\Users\Andrei\Downloads\DescargaImagen> ls


    Directorio: C:\Users\Andrei\Downloads\DescargaImagen


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        08/02/2023     23:06      483846144 NombreImagen.tar
```

3) Ejecuta la imagen que te ha pasado otro compañero.

Descargamos el .tar del compañero e iniciamos el contenedor con el puerto e imagen que ellos dijeron haber usado, el nombre del contenedor el que yo quiera sin repetirse en mis contenedores locales:

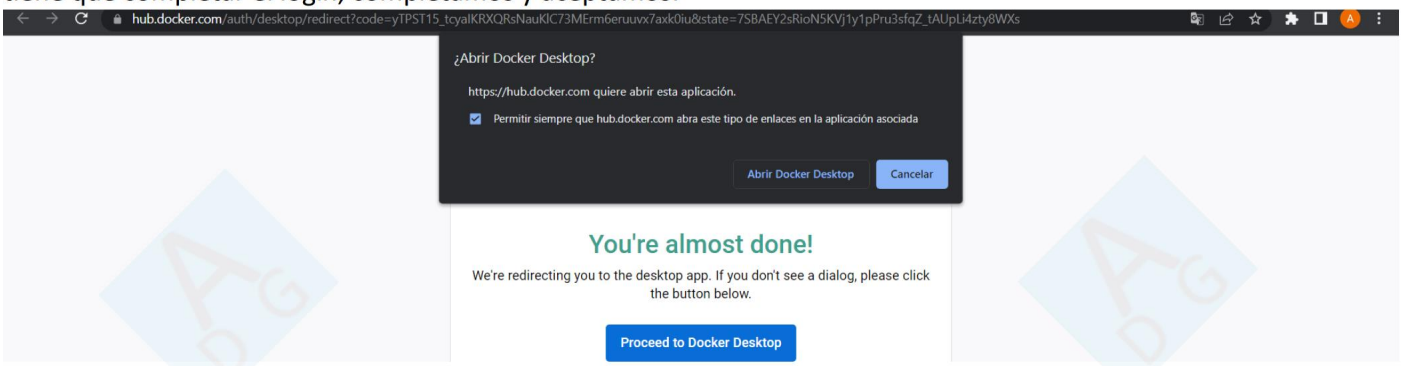
```
PS C:\Users\Andrei\Downloads> docker load -i .\flaskAlvaroDani.tar
778becc82b32: Loading layer [==================================================>]  514.1MB/514.1MB
Loaded image: flaskdanieloalvaro:latest
PS C:\Users\Andrei\Downloads> docker run -p 5000:5000 --name=ContenedorAlvaroDani flaskdanieloalvaro
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.3:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 143-436-881
```

Accedemos mediante localhost o mi ip otravez y el puerto, y ahí está el resultado:

4) Sube la imagen a un registro publico, dockerhub y a github.

DOCKERHUB:

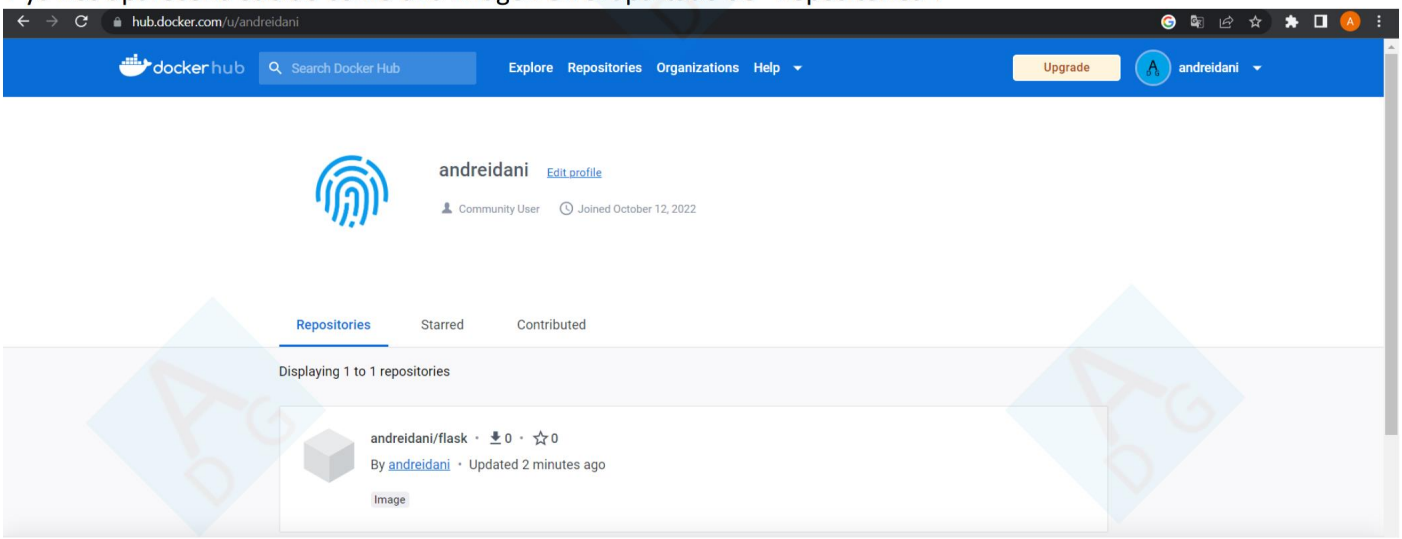
Desde el programa Docker clicko en la parte que pone “Sign In” y accedo. Me redirecciona al navegador donde se tiene que completar el login, completamos y aceptamos:



Se hace un commit para crear la imagen y otravez haremos que se ejecute el ENTRYPOINT que hará usar el python3 y ejecutará la “app.py” de la ubicación indicada que es la carpeta de dentro de nuestro contenedor. Especificamos el Contenedor y luego el formato “andreidani/flask” que sería “usuariodockerhub/nombreimagen”. Luego hacemos un push de la imagen creada para enviarla hacia DockerHub:

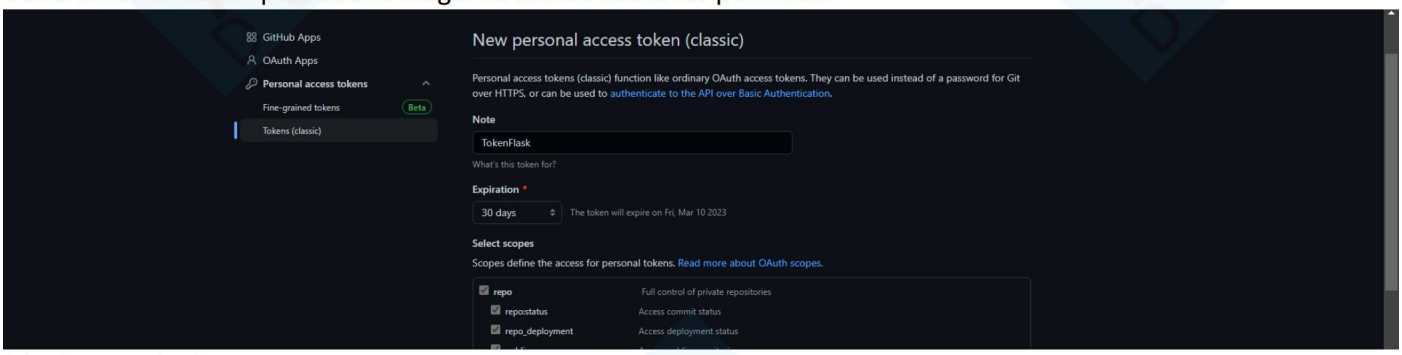
```
PS C:\Users\Andrei\Downloads> docker commit --change="ENTRYPOINT python3 ./CarpetaFlask/ADGFlask/app.py" UbuntuFlask andreidani/flask
sha256:6e004e667bbb48bb00a8c8131af9e8c285819f17091cb5dce777569694a6654d
PS C:\Users\Andrei\Downloads> docker push andreidani/flask
Using default tag: latest
The push refers to repository [docker.io/andreidani/flask]
89c3cc28f6d8: Pushed
c5ff2d88f679: Mounted from library/ubuntu
latest: digest: sha256:da81ad22dfce4daae86da804cce26a49eeae0809e2ae72a12af2daf64ab6fa64 size: 742
```

Y ya nos aparecería subido como una imagen en el apartado de “Repositories”:



GITHUB:

Generamos un token para usarlo luego. Le damos todos los permisos:

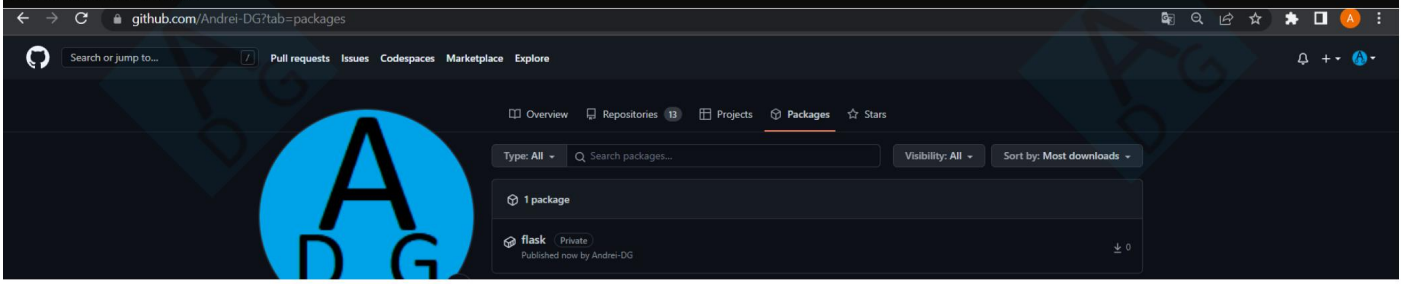


Iniciamos sesión en GitHub desde la terminal en github con "ghcr.io" que será del sitio donde se podrá usar la función de las autenticaciones con token. Primero el usuario, y luego de password el token generado.

Luego hacemos un commit para generar la imagen haciendo que ENTRYPOINT inicie el Python y a su vez luego la "app.py" que está dentro del contenedor UbuntuFlask y con la imagen ghcr.io/andrei-dg/flask en este formato: "ghcr.io/usuariogithub/nombreparaelpackage":

Luego hacemos un push con el mismo formato anterior:

```
PS C:\Users\Andrei\Downloads> docker login ghcr.io
Username: Andrei-DG
Password:
Login Succeeded
PS C:\Users\Andrei\Downloads> docker commit --change="ENTRYPOINT python3 ./CarpetaFlask/ADGFlask/app.py" UbuntuFlask ghcr.io/andrei-dg/flask
sha256:997fcaf7c70d1f87160b9ca965781329ead8573f383d7c76f084538441ddddaa
PS C:\Users\Andrei\Downloads> docker push ghcr.io/andrei-dg/flask
Using default tag: latest
The push refers to repository [ghcr.io/andrei-dg/flask]
89c3cc28f6d8: Pushed
c5ff2d88f679: Pushed
```

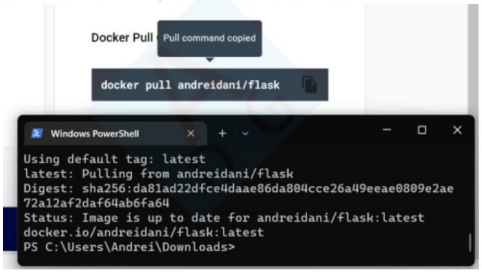


5) Borra la imagen y bajala del docker hub y montala.

Si se quiere ver las imágenes del docker. Luego borrar la imagen según el "IMAGE ID" que tenga:

```
PS C:\Users\Andrei\Downloads> docker images
REPOSITORY                                TAG       IMAGE ID       CREATED          SIZE
ghcr.io/andrei-dg/flask                   latest    997fcaf7c70d   3 minutes ago    473MB
andreidani/flask                          latest    6e004e667bbb   17 minutes ago   473MB
imagenubuntuflask                         latest    2a92efa7b9ce   47 minutes ago   473MB
flaskdanieloalvaro                        latest    1d87d8d99fa5   7 days ago       584MB
ubuntu                                    latest    58db3edaf2be   13 days ago      77.8MB
mysql                                     latest    b939d379d46e   3 weeks ago      514MB
gophish/gophish                           latest    35133b15ff61   2 months ago     230MB
portainer/portainer-docker-extension      2.16.2    5f11582196a4   2 months ago     287MB
phpmyadmin/phpmyadmin                     latest    4a4023c7e22a   9 months ago     510MB
mysql                                     8.0.16    de764ad211de   3 years ago      443MB
mariadb                                   10.3.9    23edf799b823   4 years ago      363MB
PS C:\Users\Andrei\Downloads> docker rmi 997fcaf7c70d
Untagged: ghcr.io/andrei-dg/flask:latest
Untagged: ghcr.io/andrei-dg/flask@sha256:3538f33d99302cce751b9a1853f585aaa59f235aa7079e22bcd42e86bd10f12d
Deleted: sha256:997fcaf7c70d1f87160b9ca965781329ead8573f383d7c76f084538441ddddaa
```

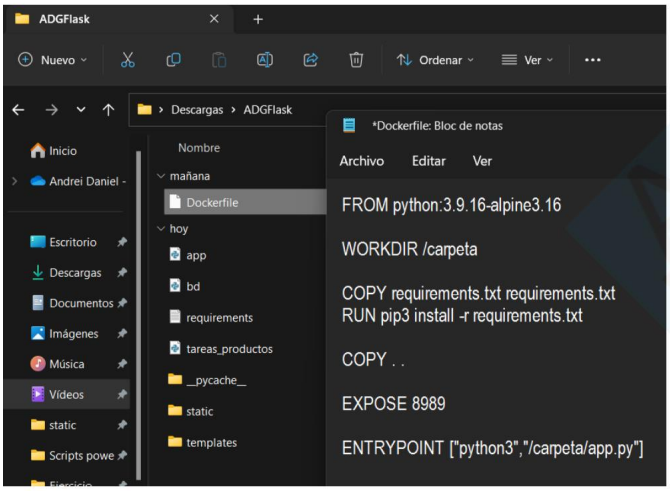
Hacemos un pull en la terminal con el mismo comando que nos da ya en la página:



6) Crea un archivo Dockerfile para crear la imagen.

Paramos los contenedores que afectan y luego escribimos la configuración en un archivo “Dockerfile” sin extensión.

FROM (para usar la imagen base del Python)
WORKDIR (la ubicación donde estarán nuestros archivos)
COPY (se copiará en la misma carpeta desde donde tengo el Dockerfile los requerements.txt)
RUN (se ejecuta el comando de instalar con el pip los requirements)
COPY (copia lo de la ubicación en la que está a la misma ubicación a la vez)
EXPOSE (el puerto que expondremos que será el 8989)



Creamos la imagen y le damos un nombre a la imagen copiada mediante el dockerfile:

```
PS C:\Users\Andrei\Downloads\ADGFlask> docker build . -t imagencopiadadockerfile
[+] Building 12.5s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                                 0.1s
 => => transferring dockerfile: 240B                                                                 0.0s
 => [internal] load .dockerignore                                                                    0.1s
 => => transferring context: 2B                                                                      0.0s
 => [internal] load metadata for docker.io/library/python:3.9.16-alpine3.16                          1.8s
 => [auth] library/python:pull token for registry-1.docker.io                                        0.0s
 => [1/5] FROM docker.io/library/python:3.9.16-alpine3.16@sha256:2528b232fb5e903b5c7ca86dc252468d3a  3.0s
 => => resolve docker.io/library/python:3.9.16-alpine3.16@sha256:2528b232fb5e903b5c7ca86dc252468d3a  0.1s
 => => sha256:2528b232fb5e903b5c7ca86dc252468d3a77b9ced64cd6bc482afc3eb43fd1fe 1.65kB / 1.65kB       0.0s
 => => sha256:85e878ecbf46651ad4c2782841bc583696a11e35fe2499427411bcaca729d371 1.37kB / 1.37kB       0.0s
 => => sha256:4d790a35f806ae8acff15631f97ec645aa482a286b7579622d021f34076c4432 7.31kB / 7.31kB       0.0s
 => => sha256:ca7dd9ec2225f2385955c43b2379305acd51543c28cf1d4e94522b3d94cce3ce 2.81MB / 2.81MB       0.4s
 => => sha256:9e124a36b9ab9caae9b9b43bffcb6f6d9f126ab89cf2a8b20b9a6f912df03d36 661.83kB / 661.83kB   0.2s
 => => sha256:c190fdfd218605c204b73c8c331aa91fe9c4c46ec4690861c5893dc2975870f8 11.99MB / 11.99MB     1.2s
 => => sha256:cee789e46502423bf34f16c58250ca027c098339b6f687bf3a5946b2a2c90214 230B / 230B           0.5s
 => => extracting sha256:ca7dd9ec2225f2385955c43b2379305acd51543c28cf1d4e94522b3d94cce3ce            0.1s
 => => sha256:c63313197313d995f338f49a8831bcc66f36790cb2052386b2d2a0df2d8b1f38 2.88MB / 2.88MB       0.9s
 => => extracting sha256:9e124a36b9ab9caae9b9b43bffcb6f6d9f126ab89cf2a8b20b9a6f912df03d36            0.2s
 => => extracting sha256:c190fdfd218605c204b73c8c331aa91fe9c4c46ec4690861c5893dc2975870f8            0.7s
 => => extracting sha256:cee789e46502423bf34f16c58250ca027c098339b6f687bf3a5946b2a2c90214            0.0s
 => => extracting sha256:c63313197313d995f338f49a8831bcc66f36790cb2052386b2d2a0df2d8b1f38            0.3s
 => [internal] load build context                                                                    0.1s
 => => transferring context: 1.47MB                                                                  0.0s
 => [2/5] WORKDIR /carpeta                                                                           0.2s
 => [3/5] COPY requirements.txt requirements.txt                                                     0.1s
 => [4/5] RUN pip3 install -r requirements.txt                                                       6.4s
```

Iniciamos el contenedor con el puerto 8989 para él y para el docker, con ese nombre de contenedor nuevo pero con el mismo nombre de imagen creada anteriormente y otravez podríamos acceder desde el navegador a la aplicación:

```
PS C:\Users\Andrei\Downloads\ADGFlask> docker run -p 8989:8989 --name=ContenedorDockerFile imagencopiadadockerfile
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8989
 * Running on http://172.17.0.2:8989
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 980-225-408
```