

# Práctica 6.2. Creación de imágenes propias

---

Gabriel Polo Merlo



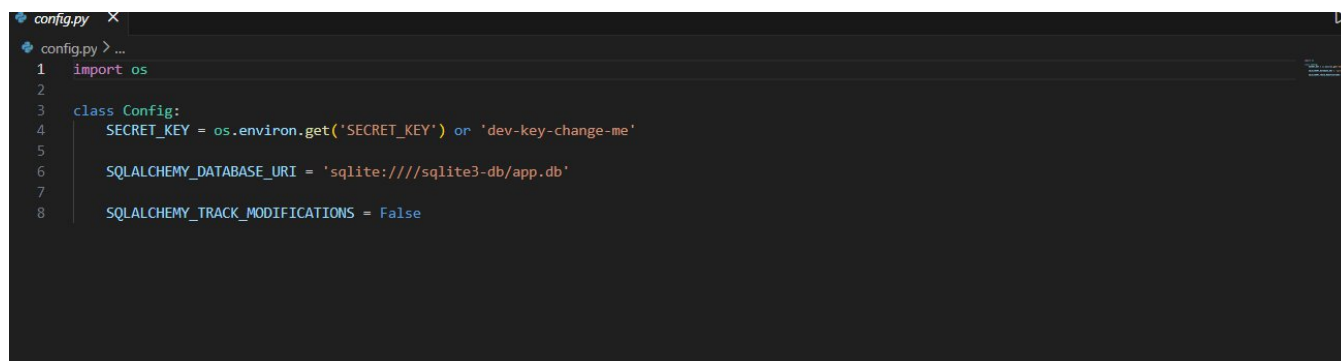
## INDICE

<b>Parte 1. Aplicación Flask.....</b>	<b>Pag 3</b>
<b>Parte 2. Dockerfile.....</b>	<b>Pag 3</b>
<b>Parte 3. Docker compose.....</b>	<b>Pag 4</b>
<b>Parte 4. Compartir la imagen.....</b>	<b>Pag 5</b>

## Parte 1. Aplicación Flask

**Adapta la aplicación para que la base de datos esté en un directorio distinto a la app. Por ejemplo en /sqlite3-db**

Cambiamos el archivo config.py para que use la base de datos en otro directorio



```
config.py
1 import os
2
3 class Config:
4     SECRET_KEY = os.environ.get('SECRET_KEY') or 'dev-key-change-me'
5
6     SQLALCHEMY_DATABASE_URI = 'sqlite:///sqlite3-db/app.db'
7
8     SQLALCHEMY_TRACK_MODIFICATIONS = False
```

Y en app.py al arrancar el pagina web creamos el repositorio de la base de datos si no existe y arrancamos la pagina web.

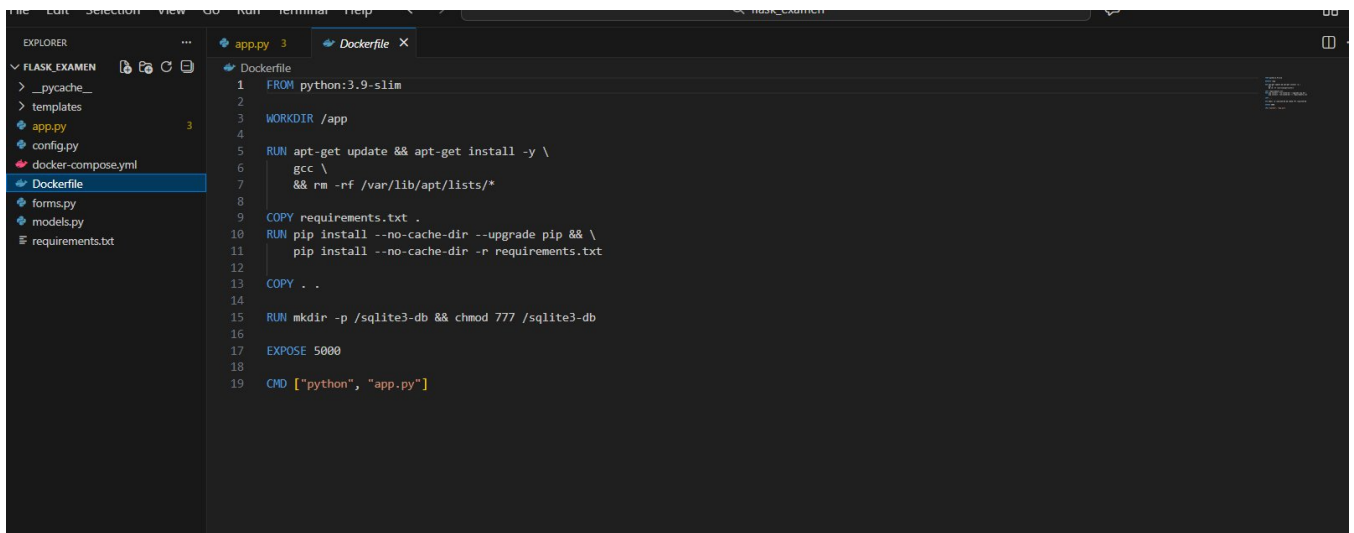


```
160
161 if __name__ == "__main__":
162     app = create_app()
163
164
165     os.makedirs('/sqlite3-db', exist_ok=True)
166
167     app.run(host="0.0.0.0", port=5000, debug=True)
```

## Parte 2. Dockerfile

**Crea una imagen propia de la aplicación Flask que hicisteis por parejas para el examen del trimestre pasado teniendo en cuenta que ya la base de datos tiene que estar en un directorio distinto a la app y que la base de datos no se puede incorporar al contenedor ya que el contenedor no incluye persistencia.**

Ahora creamos nuestro archivo Dockerfile y añadimos el siguiente contenido en el que instalamos gcc, borramos todo el contenido el directorio /var/lib/apt/lists, copiamos nuestro requirements.txt para las librerías, instalamos pip y con este instalamos las librerías, luego copiamos todo el contenido del directorio padre y creamos el directorio de la base de datos luego le damos permisos total y abrimos el puerto 5000 por ejemplo y con cmd ejecutamos con python nuestro app.py

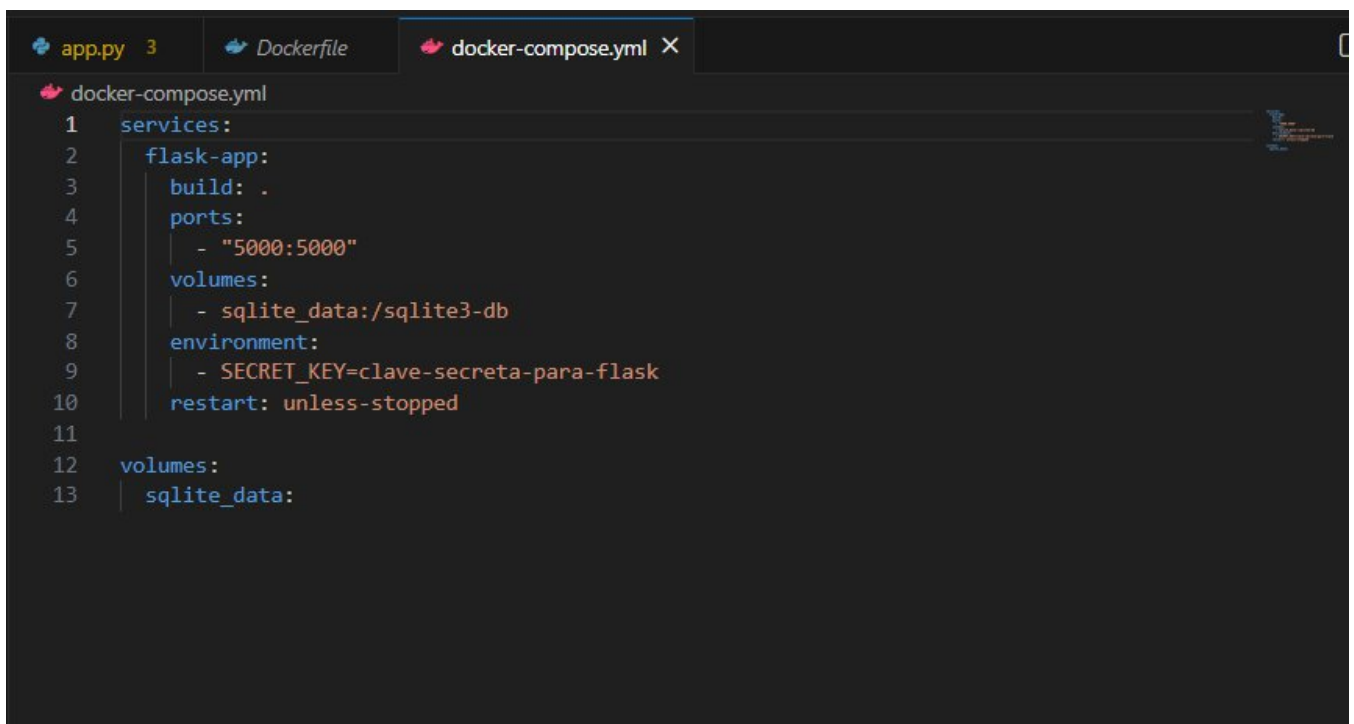
A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a project named 'FLASK\_EXAMEN' with files like '\_\_pycache\_\_', 'templates', 'app.py', 'config.py', 'docker-compose.yml', 'Dockerfile', 'forms.py', 'models.py', and 'requirements.txt'. The 'Dockerfile' is selected and its content is displayed in the main editor. The Dockerfile starts with 'FROM python:3.9-slim', sets 'WORKDIR /app', runs 'apt-get update' and 'apt-get install' for 'gcc', copies 'requirements.txt', runs 'pip install' for the requirements, copies the application files, creates a 'sqlite3-db' directory with permissions, sets 'EXPOSE 5000', and defines the 'CMD' as '["python", "app.py"]'.

```
1 FROM python:3.9-slim
2
3 WORKDIR /app
4
5 RUN apt-get update && apt-get install -y \
6     gcc \
7     && rm -rf /var/lib/apt/lists/*
8
9 COPY requirements.txt .
10 RUN pip install --no-cache-dir --upgrade pip && \
11     pip install --no-cache-dir -r requirements.txt
12
13 COPY . .
14
15 RUN mkdir -p /sqlite3-db && chmod 777 /sqlite3-db
16
17 EXPOSE 5000
18
19 CMD ["python", "app.py"]
```

### Parte 3. Docker compose

Para automatizar la creación de contenedores utiliza la herramienta Docker Compose para levantar la aplicación, publicar el puerto de la aplicación y hacer uso de un volumen donde vamos a almacenar el archivo que contiene la base de datos sqlite3

Aquí decimos nuestro puerto y el volumen de la app también el volumen de la database

A screenshot of the Visual Studio Code editor showing the 'docker-compose.yml' file. The file defines a service named 'flask-app' with build context '.', ports '5000:5000', a volume 'sqlite\_data' mapped to 'sqlite3-db', an environment variable 'SECRET\_KEY', and a restart policy of 'unless-stopped'. A separate 'volumes' section defines the 'sqlite\_data' volume.

```
1 services:
2   flask-app:
3     build: .
4     ports:
5       - "5000:5000"
6     volumes:
7       - sqlite_data:/sqlite3-db
8     environment:
9       - SECRET_KEY=clave-secreta-para-flask
10    restart: unless-stopped
11
12 volumes:
13   sqlite_data:
```

Luego hacemos un docker-compose build

```
PS C:\Users\Gabriel_Polo\Documents\Gabriel\ASIR\2º ASIR\Implemetacion aplicaciones web\Unidad 6\Práctica 6.2. Creación de imágenes propias\flask_examen> docker-compose build
time="2026-02-02T22:02:41+01:00" level=warning msg="C:\\Users\\Gabriel_Polo\\Documents\\Gabriel\\ASIR\\2º ASIR\\Implemetacion aplicaciones web\\Unidad 6\\Práctica 6.2. Creación de imágenes propias\\flask_examen\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Building 35.8s (7/14)
=> [internal] load local bake definitions 0.1s
=> => reading from stdin 800B 0.1s
=> [internal] load build definition from Dockerfile 0.2s
=> => transferring dockerfile: 672B 0.1s
=> [internal] load metadata for docker.io/library/python:3.11-slim 2.7s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 2B 0.0s
=> [internal] load build context 0.2s
=> => transferring context: 28.17kB 0.1s
=> [1/8] FROM docker.io/library/python:3.11-slim@sha256:5be45dbade29bebd6886af6b438fd7e0b4eb7b611f39ba62b430263f82de36d2 15.1s
=> => resolve docker.io/library/python:3.11-slim@sha256:5be45dbade29bebd6886af6b438fd7e0b4eb7b611f39ba62b430263f82de36d2 0.1s
=> => sha256:0b2bf04f68e9f306a8a83f57c6ced322a23968bf3d5acebc07e055c090240826 250B / 250B 0.3s
=> => sha256:3e731abb5c1dd05aef62585d392d31ad26089dc4c031730e5ab0225aef80b3f2 14.36MB / 14.36MB 4.9s
=> => sha256:5b09819094bb89d5b2416ff2fb03f68666a5372c358cfd22f2b62d7f6660d906 1.29MB / 1.29MB 1.7s
=> => sha256:119d43eec815e5f9a47da3a7d59454581b1e204b0c34db86f171b7ceb3336533 29.77MB / 29.77MB 8.2s
=> => extracting sha256:119d43eec815e5f9a47da3a7d59454581b1e204b0c34db86f171b7ceb3336533 3.3s
=> => extracting sha256:5b09819094bb89d5b2416ff2fb03f68666a5372c358cfd22f2b62d7f6660d906 0.5s
=> => extracting sha256:3e731abb5c1dd05aef62585d392d31ad26089dc4c031730e5ab0225aef80b3f2 2.5s
=> => extracting sha256:0b2bf04f68e9f306a8a83f57c6ced322a23968bf3d5acebc07e055c090240826 0.1s
=> [2/8] RUN apt-get update && apt-get install -y --no-install-recommends gcc && rm -rf /var/lib/apt/lists/* 15.9s
=> # Get:23 http://deb.debian.org/debian trixie/main amd64 libtsan2 amd64 14.2.0-19 [2460 kB]
=> # Get:24 http://deb.debian.org/debian trixie/main amd64 libubsan1 amd64 14.2.0-19 [1074 kB]
=> # Get:25 http://deb.debian.org/debian trixie/main amd64 libhwasa0 amd64 14.2.0-19 [1488 kB]
=> # Get:26 http://deb.debian.org/debian trixie/main amd64 libquadmath0 amd64 14.2.0-19 [145 kB]
=> # Get:27 http://deb.debian.org/debian trixie/main amd64 libgcc-14-dev amd64 14.2.0-19 [2672 kB]
=> # Get:28 http://deb.debian.org/debian trixie/main amd64 gcc-14-x86-64-linux-gnu amd64 14.2.0-19 [21.4 MB]
```

Y un docker-compose up

```
PS C:\Users\Gabriel_Polo\Documents\Gabriel\ASIR\2º ASIR\Implemetacion aplicaciones web\Unidad 6\Práctica 6.2. Creación de imágenes propias\flask_examen> docker-compose down
[+] Running 2/2
✓ Container flask_examen2-flask-app-1 Removed 0.1s
✓ Network flask_examen2_default Removed 0.4s
PS C:\Users\Gabriel_Polo\Documents\Gabriel\ASIR\2º ASIR\Implemetacion aplicaciones web\Unidad 6\Práctica 6.2. Creación de imágenes propias\flask_examen> docker-compose up
[+] Running 2/2
✓ Network flask_examen2_default Created 0.2s
✓ Container flask_examen2-flask-app-1 Created 0.4s
Attaching to flask-app-1
flask-app-1 | * Serving Flask app 'app'
flask-app-1 | * Debug mode: on
flask-app-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
flask-app-1 | * Running on all addresses (0.0.0.0)
flask-app-1 | * Running on http://127.0.0.1:5000
flask-app-1 | * Running on http://172.18.0.2:5000
flask-app-1 | Press CTRL+C to quit
flask-app-1 | * Restarting with stat
flask-app-1 | * Debugger is active!
flask-app-1 | * Debugger PIN: 104-645-893
flask-app-1 | 172.18.0.1 - - [02/Feb/2026 22:02:55] "GET / HTTP/1.1" 200 -
```

## Parte 4. Compartir la imagen

Para finalizar vamos a subir la imagen de tu aplicación a tu repositorio Docker haciendo los cambios necesarios en los apartados anteriores para que incluyan el username correspondiente.

Añadimos esta línea a nuestro docker-compose.yml

```
flask-app:
  image: gabrielpolo05/flask-sqlite-app:latest
  build: .
```

Luego nos logueamos en docker localmente y comprobamos con docker login



```
Administrador: Windows Pow x + v
PS C:\Users\Gabriel_Polo> docker login
Authenticating with existing credentials... [Username: gabrielpolo05]

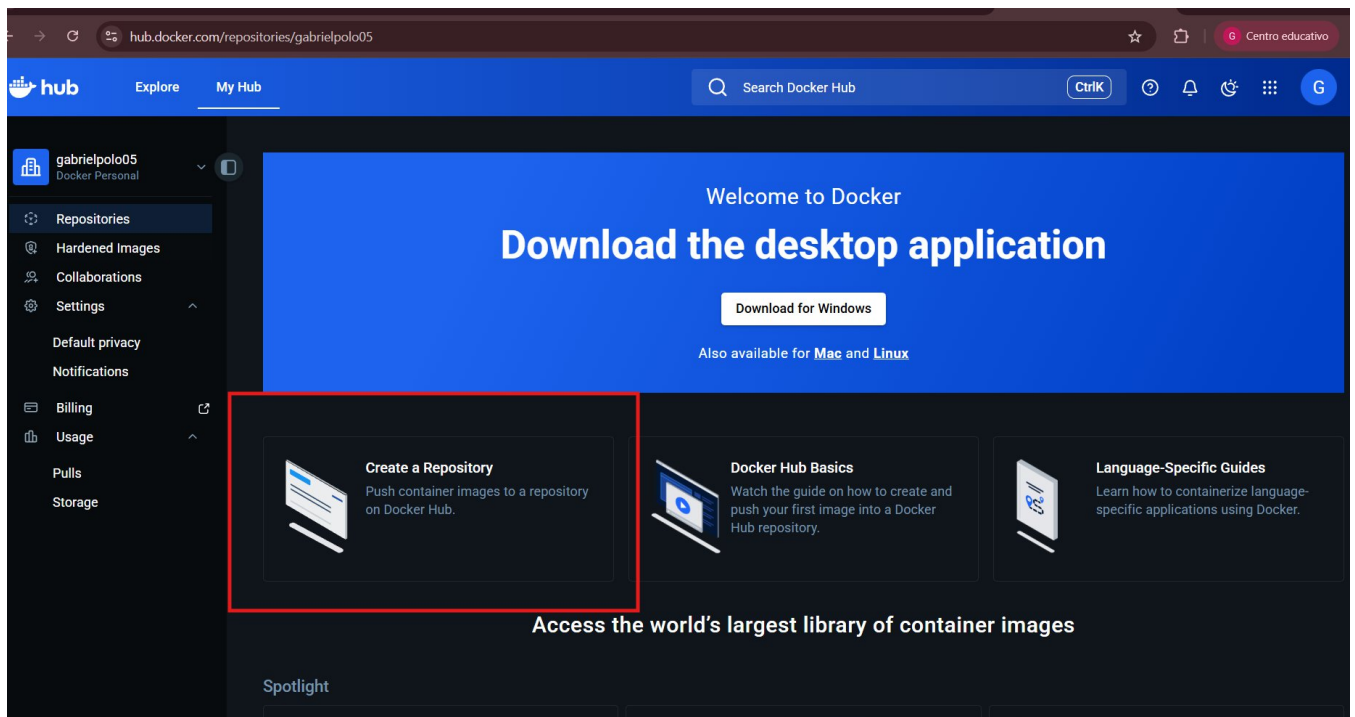
Info → To login with a different account, run 'docker logout' followed by
'docker login'

Login Succeeded
PS C:\Users\Gabriel_Polo> |
```

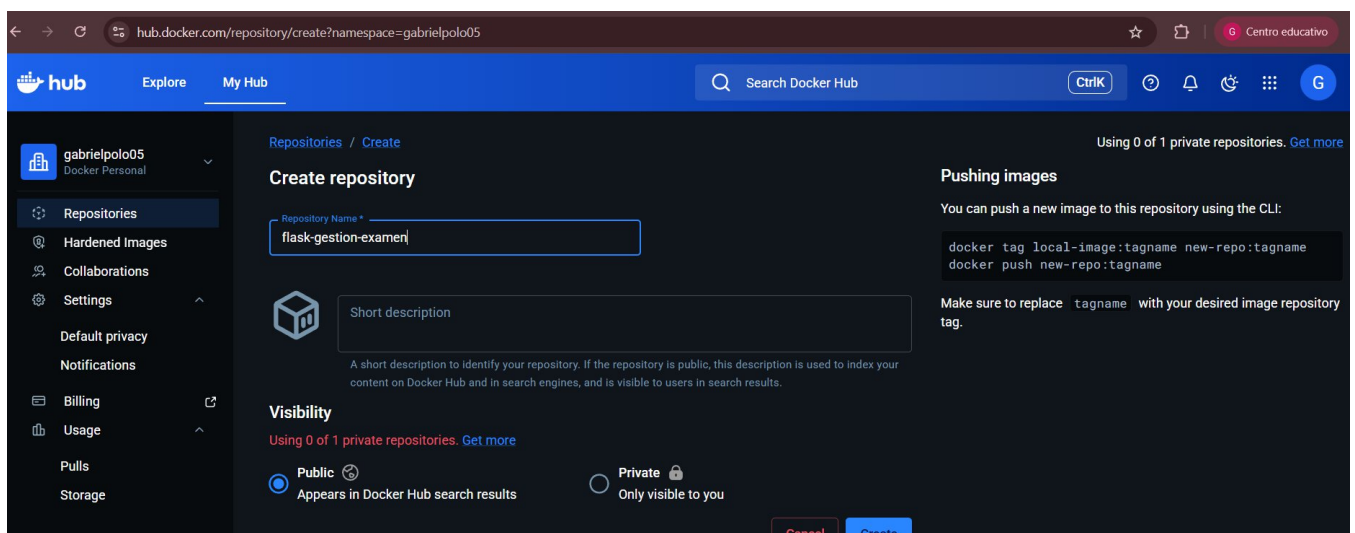
Ahora construimos la imagen

```
PS C:\Users\Gabriel_Polo\Documents\Gabriel\ASIR\2º ASIR\Implemetacion aplica
ciones web\Unidad 6\Práctica 6.2. Creación de imágenes propias\flask_examen>
docker build -t tunombreusuario/flask-gestion-examen .
[+] Building 4.5s (13/13) FINISHED          docker:desktop-linux
=> [internal] load build definition from Dockerfile      0.0s
=> => transferring dockerfile: 406B                      0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim 1.9s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore                        0.0s
=> => transferring context: 2B                            0.0s
=> [1/7] FROM docker.io/library/python:3.9-slim@sha256:2d97f6910b16b 0.1s
=> => resolve docker.io/library/python:3.9-slim@sha256:2d97f6910b16b 0.1s
=> [internal] load build context                        0.1s
=> => transferring context: 9.29kB                         0.1s
=> CACHED [2/7] WORKDIR /app                            0.0s
=> CACHED [3/7] RUN apt-get update && apt-get install -y gcc 0.0s
=> CACHED [4/7] COPY requirements.txt .                  0.0s
=> CACHED [5/7] RUN pip install --no-cache-dir --upgrade pip && 0.0s
=> [6/7] COPY . .                                       0.1s
=> [7/7] RUN mkdir -p /sqlite3-db && chmod 777 /sqlite3-db 0.7s
=> exporting to image                                  1.1s
=> => exporting layers                                    0.4s
=> => exporting manifest sha256:66f2fd300fb77b88b10b66f61610f856db66 0.1s
=> => exporting config sha256:48e95c8f7e4905fda55367669e75e03e9a09b2 0.1s
=> => exporting attestation manifest sha256:7719d8fa0e40cd65542cfb84 0.1s
=> => exporting manifest list sha256:a82d9d8c63ad100ef645e311002f689 0.1s
=> => naming to docker.io/tunombreusuario/flask-gestion-examen:lates 0.0s
=> => unpacking to docker.io/tunombreusuario/flask-gestion-examen:la 0.2s
PS C:\Users\Gabriel_Polo\Documents\Gabriel\ASIR\2º ASIR\Implemetacion aplica
ciones web\Unidad 6\Práctica 6.2. Creación de imágenes propias\flask_examen>
|
```

Creamos el repositorio en nuestra cuenta de dockerhub



Y le ponemos el nombre y lo creamos.



Y acabamos subiendolo a docker hub

