Diplomatura de Posgrado en Desarrollo de Soluciones de Inteligencia Artificial Generativa en la Nube

Curso II- Infraestructura Tecnológica para Inteligencia Artificial

Docente: David M. Petrocelli

Alumnos: Nicolás S. Vidal, José E. Racker

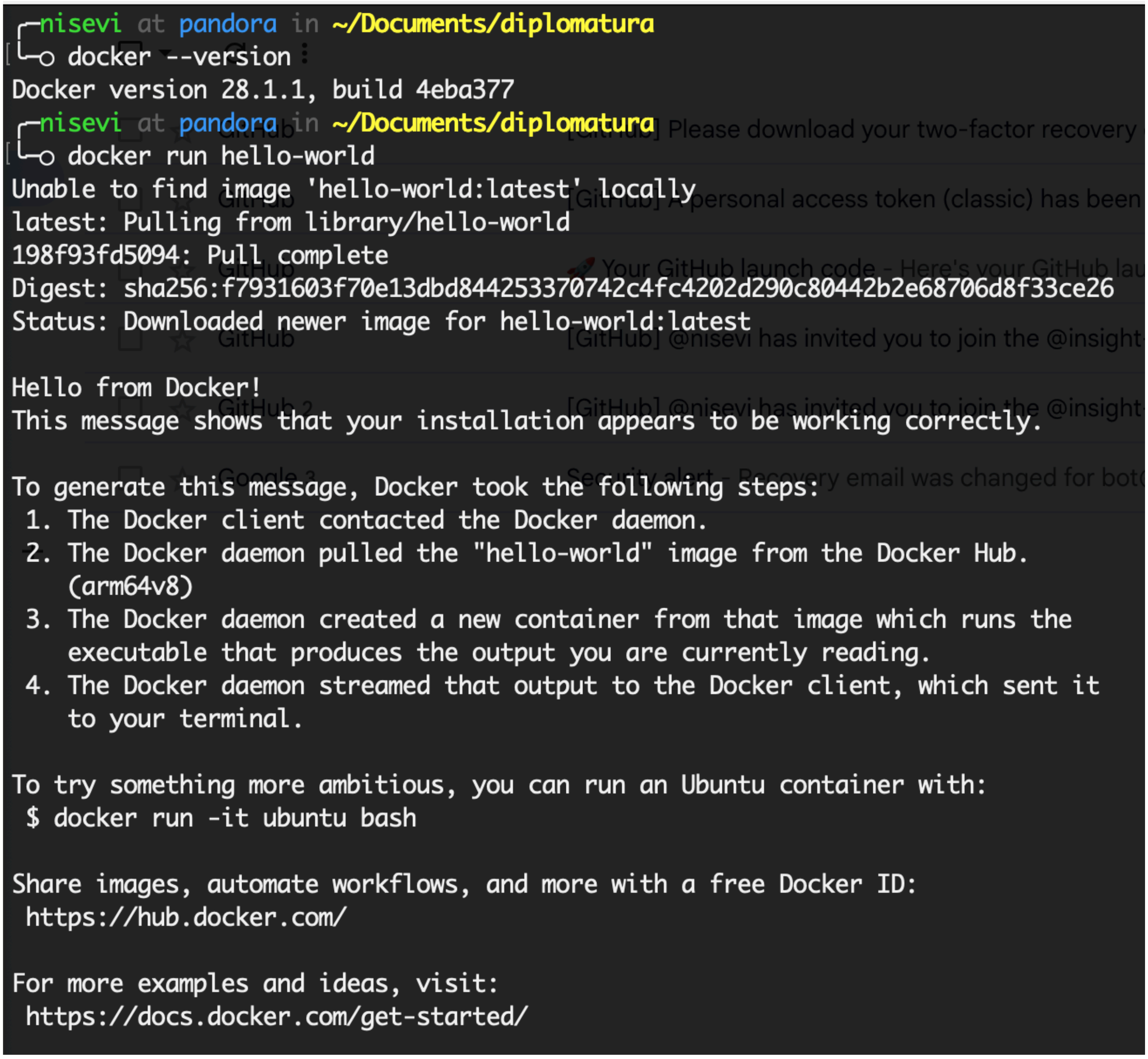
Unidad 4 – "Contenedores (Docker)"

Laboratorio 1 - Contenedores - Introducción práctica a contenedores

U4 - Contenedores (Docker)

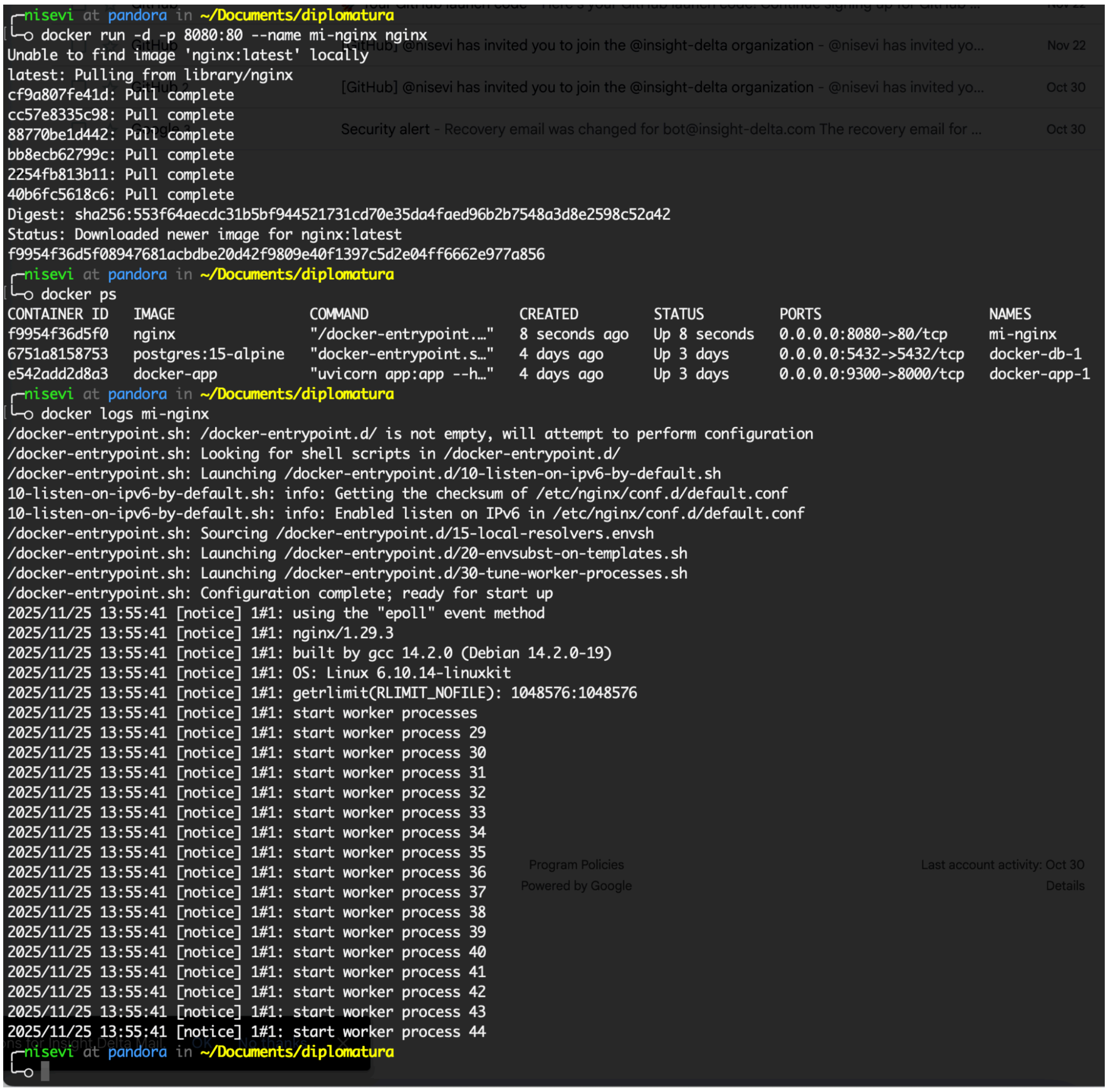
Lab 1 - Contenedores - Introducción práctica a contenedores

Parte 1 — Instalación de Docker



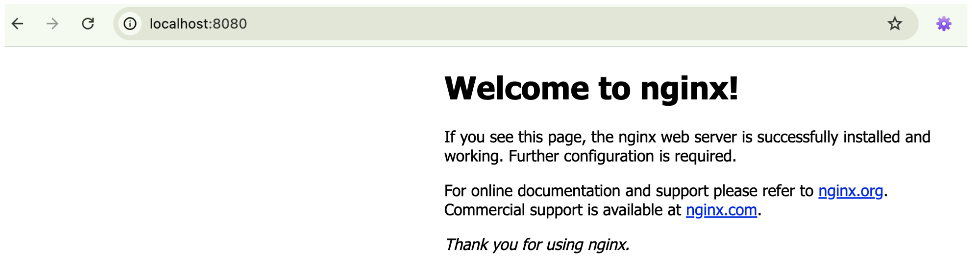
```
╭─nisevi at pandora in ~/Documents/diplomatura
╰─○ docker --version
Docker version 28.1.1, build 4eba377
╭─nisevi at pandora in ~/Documents/diplomatura
╰─○ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
198f93fd5094: Pull complete
Digest: sha256:f7931603f70e13dbd844253370742c4fc4202d290c80442b2e68706d8f33ce26
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Parte 2 — Ejecutar Contenedores Existentes



```
┌─nisevi at pandora in ~/Documents/diplomatura
└─○ docker run -d -p 8080:80 --name mi-nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
cf9a807fe41d: Pull complete
cc57e8335c98: Pull complete
88770be1d442: Pull complete
bb8ecb62799c: Pull complete
2254fb813b11: Pull complete
40b6fc5618c6: Pull complete
Digest: sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c52a42
Status: Downloaded newer image for nginx:latest
f9954f36d5f08947681acbdbe20d42f9809e40f1397c5d2e04ff6662e977a856
┌─nisevi at pandora in ~/Documents/diplomatura
└─○ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS         PORTS                    NAMES
f9954f36d5f0   nginx                "/docker-entrypoint.…"   8 seconds ago   Up 8 seconds   0.0.0.0:8080->80/tcp     mi-nginx
6751a8158753   postgres:15-alpine   "docker-entrypoint.s…"   4 days ago      Up 3 days      0.0.0.0:5432->5432/tcp   docker-db-1
e542add2d8a3   docker-app           "uvicorn app:app --h…"   4 days ago      Up 3 days      0.0.0.0:9300->8000/tcp   docker-app-1
┌─nisevi at pandora in ~/Documents/diplomatura
└─○ docker logs mi-nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/11/25 13:55:41 [notice] 1#1: using the "epoll" event method
2025/11/25 13:55:41 [notice] 1#1: nginx/1.29.3
2025/11/25 13:55:41 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2025/11/25 13:55:41 [notice] 1#1: OS: Linux 6.10.14-linuxkit
2025/11/25 13:55:41 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/11/25 13:55:41 [notice] 1#1: start worker processes
2025/11/25 13:55:41 [notice] 1#1: start worker process 29
2025/11/25 13:55:41 [notice] 1#1: start worker process 30
2025/11/25 13:55:41 [notice] 1#1: start worker process 31
2025/11/25 13:55:41 [notice] 1#1: start worker process 32
2025/11/25 13:55:41 [notice] 1#1: start worker process 33
2025/11/25 13:55:41 [notice] 1#1: start worker process 34
2025/11/25 13:55:41 [notice] 1#1: start worker process 35
2025/11/25 13:55:41 [notice] 1#1: start worker process 36
2025/11/25 13:55:41 [notice] 1#1: start worker process 37
2025/11/25 13:55:41 [notice] 1#1: start worker process 38
2025/11/25 13:55:41 [notice] 1#1: start worker process 39
2025/11/25 13:55:41 [notice] 1#1: start worker process 40
2025/11/25 13:55:41 [notice] 1#1: start worker process 41
2025/11/25 13:55:41 [notice] 1#1: start worker process 42
2025/11/25 13:55:41 [notice] 1#1: start worker process 43
2025/11/25 13:55:41 [notice] 1#1: start worker process 44
┌─nisevi at pandora in ~/Documents/diplomatura
└─○
```

Nginx funcionando en el navegador:



localhost:8080

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

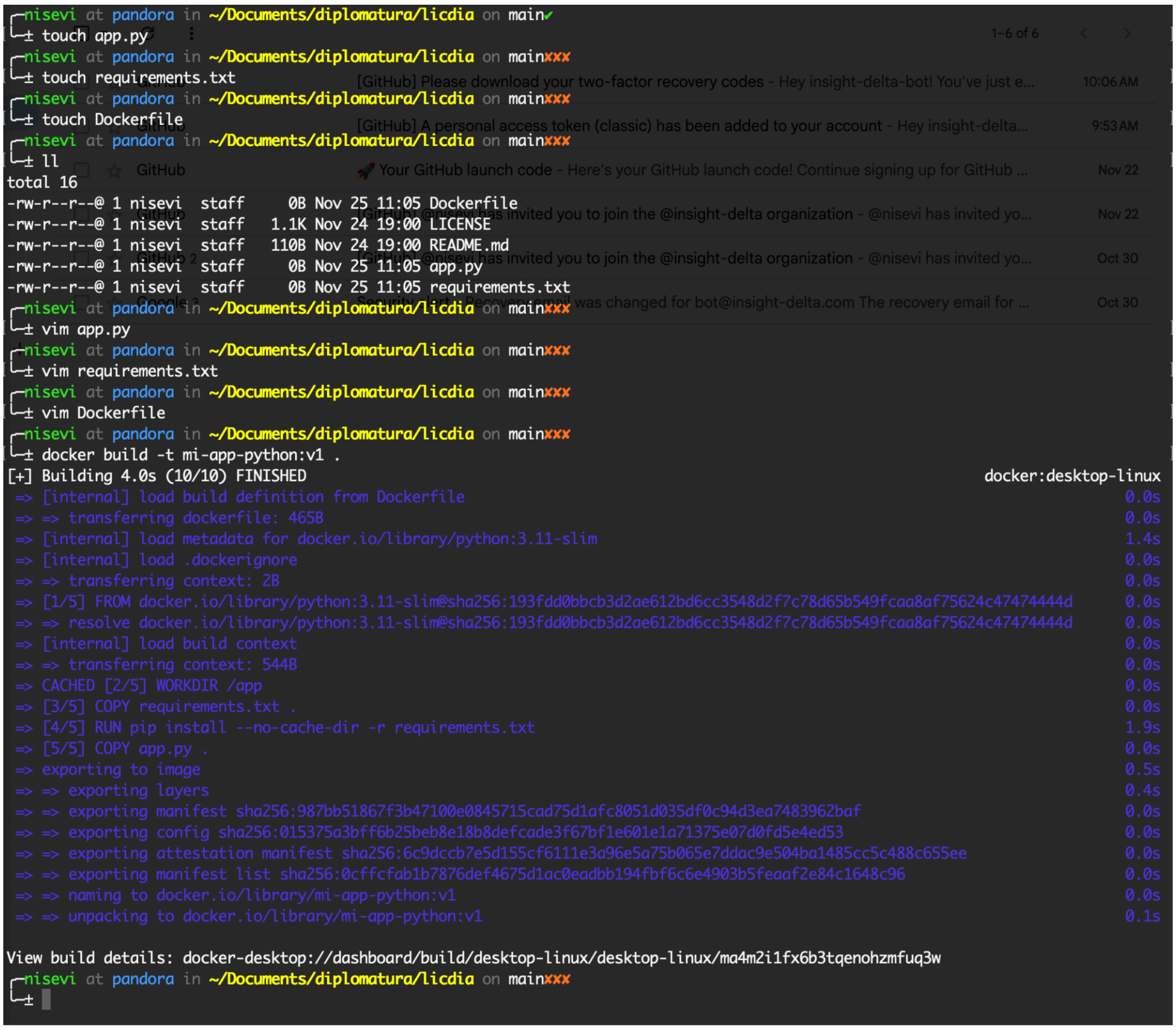
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

¿Qué diferencia hay entre docker stop y docker rm?

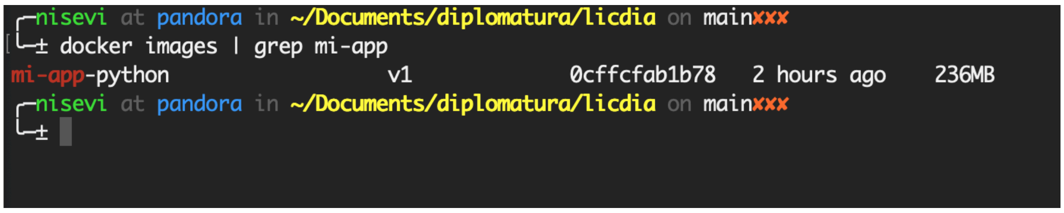
- **docker stop**: Detiene la ejecución de un contenedor en ejecución, pero no lo elimina;
- **docker rm**: Elimina permanentemente un contenedor que ya ha sido detenido;

Parte 3 — Crear Dockerfile para Aplicación Python



```
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on main✓
└─± touch app.py
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± touch requirements.txt
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± touch Dockerfile
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± ll
total 16
-rw-r--r--@ 1 nisevi  staff     0B Nov 25 11:05 Dockerfile
-rw-r--r--@ 1 nisevi  staff   1.1K Nov 24 19:00 LICENSE
-rw-r--r--@ 1 nisevi  staff   110B Nov 24 19:00 README.md
-rw-r--r--@ 1 nisevi  staff     0B Nov 25 11:05 app.py
-rw-r--r--@ 1 nisevi  staff     0B Nov 25 11:05 requirements.txt
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± vim app.py
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± vim requirements.txt
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± vim Dockerfile
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± docker build -t mi-app-python:v1 .
[+] Building 4.0s (10/10) FINISHED                                                                      docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                   0.0s
 => => transferring dockerfile: 465B                                                                                   0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                                    1.4s
 => [internal] load .dockerignore                                                                                      0.0s
 => => transferring context: 2B                                                                                        0.0s
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c47474444d   0.0s
 => => resolve docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c47474444d   0.0s
 => [internal] load build context                                                                                      0.0s
 => => transferring context: 544B                                                                                      0.0s
 => CACHED [2/5] WORKDIR /app                                                                                          0.0s
 => [3/5] COPY requirements.txt .                                                                                      0.0s
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt                                                           1.9s
 => [5/5] COPY app.py .                                                                                                0.0s
 => exporting to image                                                                                                 0.5s
 => => exporting layers                                                                                                0.4s
 => => exporting manifest sha256:987bb51867f3b47100e0845715cad75d1afc8051d035df0c94d3ea7483962baf                      0.0s
 => => exporting config sha256:015375a3bff6b25beb8e18b8defcade3f67bf1e601e1a71375e07d0fd5e4ed53                        0.0s
 => => exporting attestation manifest sha256:6c9dccb7e5d155cf6111e3a96e5a75b065e7ddac9e504ba1485cc5c488c655ee          0.0s
 => => exporting manifest list sha256:0cffcfab1b7876def4675d1ac0eadbb194fbf6c6e4903b5feaaf2e84c1648c96                 0.0s
 => => naming to docker.io/library/mi-app-python:v1                                                                    0.0s
 => => unpacking to docker.io/library/mi-app-python:v1                                                                 0.1s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ma4m2i1fx6b3tqenohzmfuq3w
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─±
```

docker images output:



```
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± docker images | grep mi-app
mi-app-python                v1           0cffcfab1b78    2 hours ago     236MB
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─±
```

¿Por qué copiamos requirements.txt antes que app.py? ¿Qué ventaja tiene para el caché de Docker?

La razón del orden: Aprovechamiento del caché de capas

Cuando se copia requirements.txt antes que app.py, se esta optimizando el build aprovechando cómo funciona el caché de capas de Docker:

Patrón óptimo:

```
Dockerfile
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY app.py .
```

Ventajas principales:

1. Caché inteligente de dependencias
    a. Las dependencias (requirements.txt) cambian poco frecuentemente
    b. El código de tu app (app.py) cambia constantemente durante desarrollo
    c. Al separarlos, Docker cachea la instalación de dependencias

2. Builds mucho más rápidos
    a. Si solo modificas app.py: Docker reutiliza la capa de dependencias ya instaladas;
    b. Si modificas requirements.txt: Solo entonces reinstala las dependencias;

3. Ahorro de tiempo significativo
    a. Instalar dependencias puede tomar minutos;
    b. Copiar tu código toma segundos;

Comparación:

Forma ineficiente:

```
COPY . .  # Copia TODO junto
RUN pip install -r requirements.txt
```

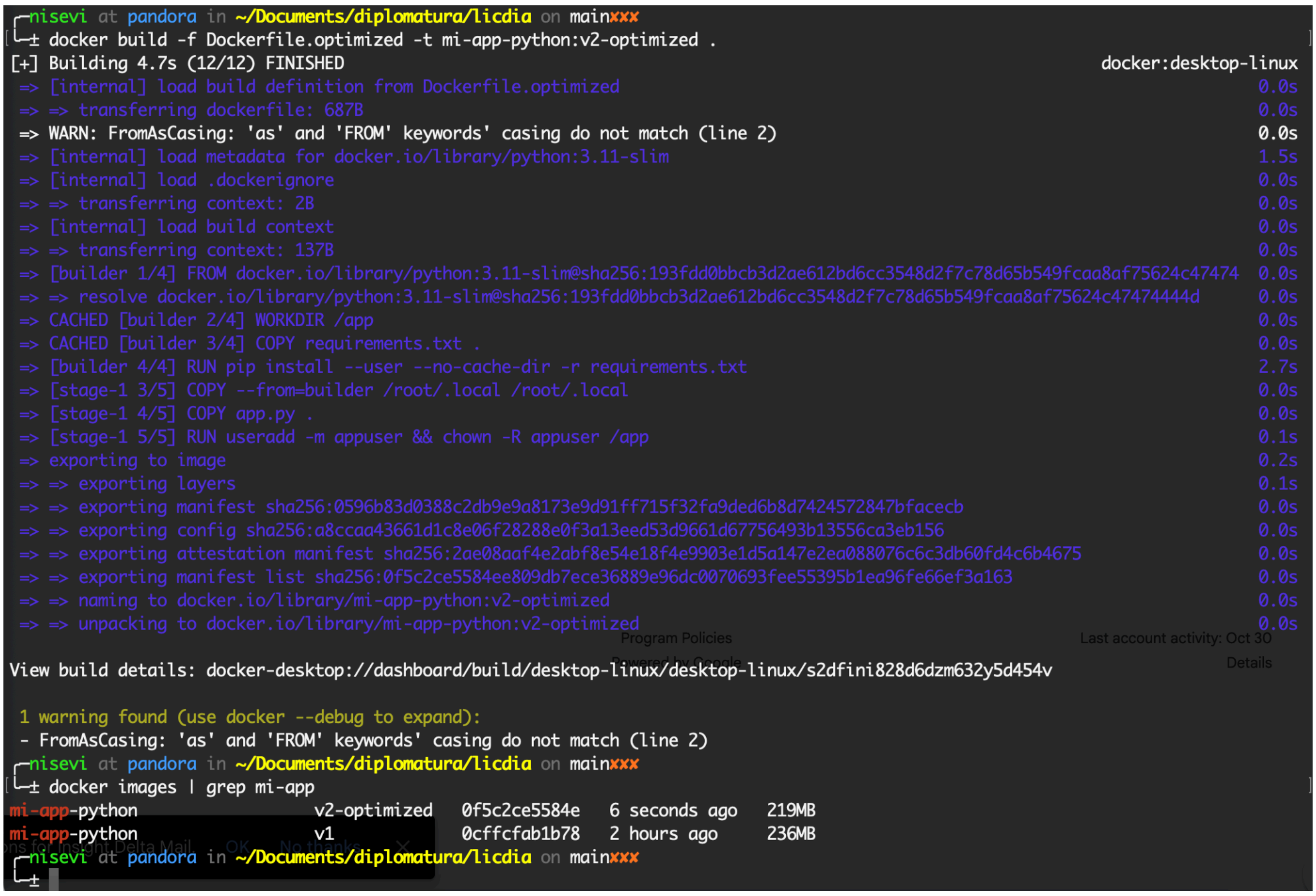
Cada cambio en app.py invalida el caché y reinstala TODO

Forma eficiente:

```
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY app.py .  # Solo esto se reconstruye si cambias tu código
```

Solo se recopia app.py, las dependencias quedan en caché.

Parte 4 — Optimizar Imagen (Multi-stage Build)



```
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± docker build -f Dockerfile.optimized -t mi-app-python:v2-optimized .
[+] Building 4.7s (12/12) FINISHED                                                                       docker:desktop-linux
 => [internal] load build definition from Dockerfile.optimized                                                          0.0s
 => => transferring dockerfile: 687B                                                                                    0.0s
 => WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 2)                                          0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                                     1.5s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                         0.0s
 => [internal] load build context                                                                                       0.0s
 => => transferring context: 137B                                                                                       0.0s
 => [builder 1/4] FROM docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c47474  0.0s
 => => resolve docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c47474444d   0.0s
 => CACHED [builder 2/4] WORKDIR /app                                                                                   0.0s
 => CACHED [builder 3/4] COPY requirements.txt .                                                                        0.0s
 => [builder 4/4] RUN pip install --user --no-cache-dir -r requirements.txt                                             2.7s
 => [stage-1 3/5] COPY --from=builder /root/.local /root/.local                                                         0.0s
 => [stage-1 4/5] COPY app.py .                                                                                         0.0s
 => [stage-1 5/5] RUN useradd -m appuser && chown -R appuser /app                                                       0.1s
 => exporting to image                                                                                                  0.2s
 => => exporting layers                                                                                                 0.1s
 => => exporting manifest sha256:0596b83d0388c2db9e9a8173e9d91ff715f32fa9ded6b8d7424572847bfacecb                       0.0s
 => => exporting config sha256:a8ccaa43661d1c8e06f28288e0f3a13eed53d9661d67756493b13556ca3eb156                         0.0s
 => => exporting attestation manifest sha256:2ae08aaf4e2abf8e54e18f4e9903e1d5a147e2ea088076c6c3db60fd4c6b4675           0.0s
 => => exporting manifest list sha256:0f5c2ce5584ee809db7ece36889e96dc0070693fee55395b1ea96fe66ef3a163                  0.0s
 => => naming to docker.io/library/mi-app-python:v2-optimized                                                           0.0s
 => => unpacking to docker.io/library/mi-app-python:v2-optimized                                                        0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/s2dfini828d6dzm632y5d454v

 1 warning found (use docker --debug to expand):
 - FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 2)
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± docker images | grep mi-app
mi-app-python                     v2-optimized   0f5c2ce5584e   6 seconds ago   219MB
mi-app-python                     v1             0cffcfab1b78   2 hours ago     236MB
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─±
```

¿Qué beneficios tiene usar multi-stage builds? Menciona al menos 3.

Los multi-stage builds son una técnica poderosa en Docker que ofrece múltiples ventajas:

1. Imágenes finales mucho más pequeñas 📦
   a. Separas las herramientas de build de la imagen final
   b. Solo copias los artefactos necesarios, no todo el toolchain

      Ejemplo: Una imagen de compilación de Go puede pesar 800MB, pero la imagen final solo 10MB

      # Stage 1: Build (pesado)
      FROM golang:1.21 AS builder
      COPY . .
      RUN go build -o app

      # Stage 2: Runtime (liviano)
      FROM alpine:latest

```
COPY --from=builder /app/app .
CMD ["./app"]
```

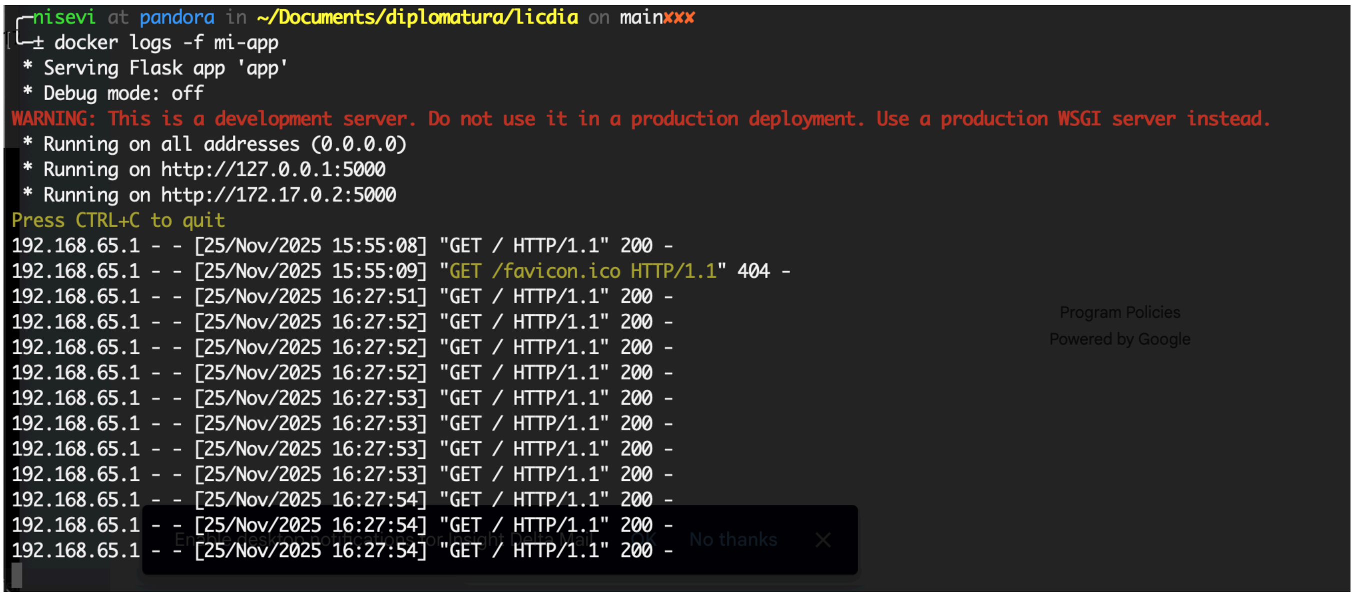
2. Mayor seguridad
    a. Reduces la superficie de ataque eliminando herramientas de desarrollo
    b. No incluyes compiladores, git, build tools en producción
    c. Menos paquetes significa menos vulnerabilidades potenciales

3. Mejor separación de responsabilidades 🎯
    a. Cada stage tiene un propósito específico;
    b. Build, test, y runtime están claramente separados;
    c. Facilita el debugging: puedes construir stages específicos para testing
    Ejemplo: **docker build --target test** para solo ejecutar tests

4. Optimización de caché más granular
    a. Cada stage tiene su propio caché independiente;
    b. Puedes reconstruir solo el stage que cambió;
    c. Paralelización de builds cuando es posible;

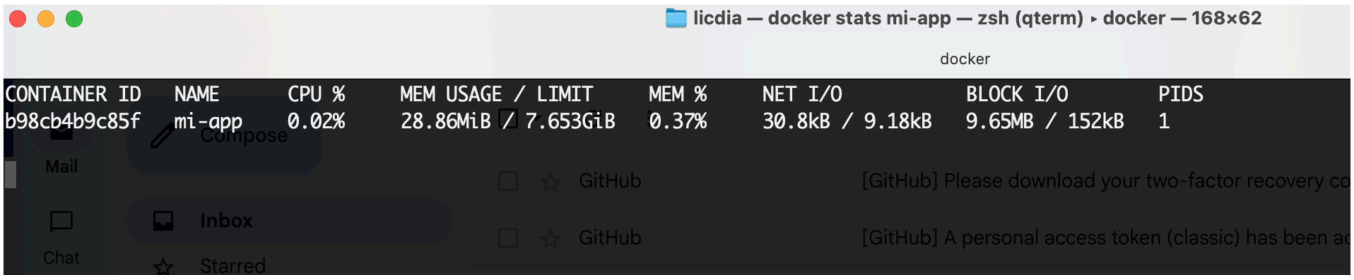
Parte 5 — Inspeccionar Logs y Contenedores

docker logs -f mi-app



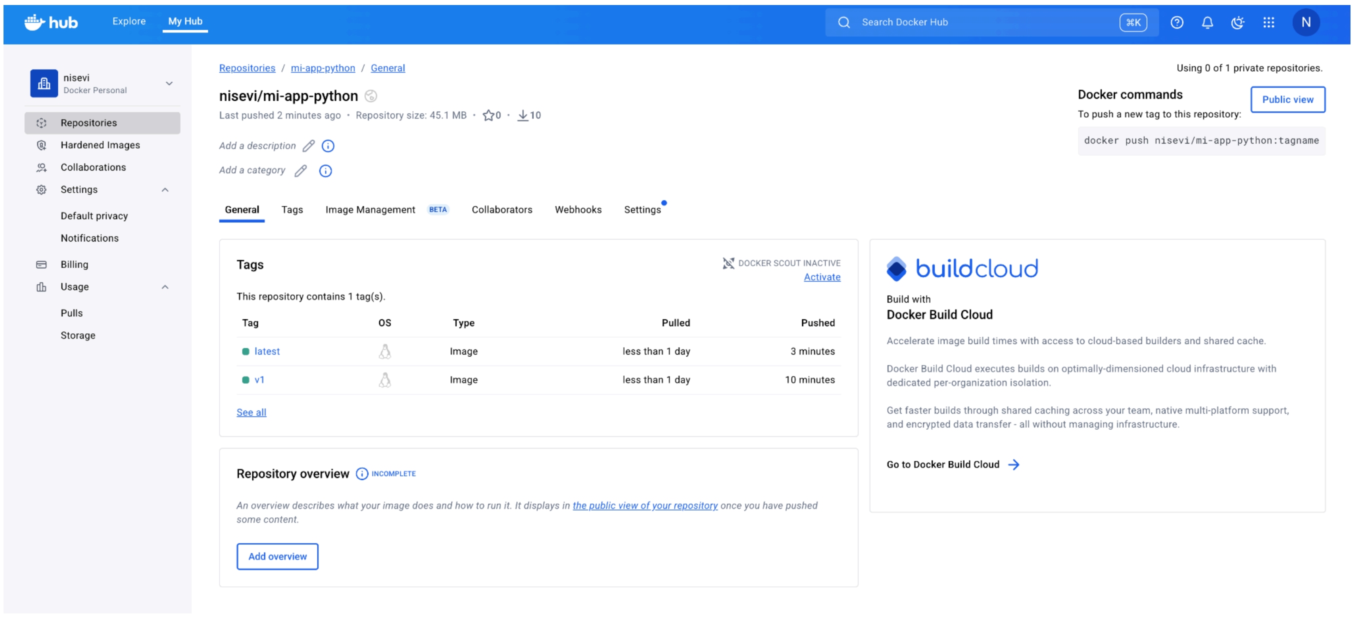
```
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on mainxxx
└─± docker logs -f mi-app
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
192.168.65.1 - - [25/Nov/2025 15:55:08] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 15:55:09] "GET /favicon.ico HTTP/1.1" 404 -
192.168.65.1 - - [25/Nov/2025 16:27:51] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:52] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:52] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:52] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:53] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:53] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:53] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:53] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:54] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:54] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [25/Nov/2025 16:27:54] "GET / HTTP/1.1" 200 -
```

docker stats


licdia — docker stats mi-app — zsh (qterm) ▸ docker — 168×62
docker

```
CONTAINER ID   NAME     CPU %     MEM USAGE / LIMIT     MEM %     NET I/O          BLOCK I/O        PIDS
b98cb4b9c85f   mi-app   0.02%     28.86MiB / 7.653GiB   0.37%     30.8kB / 9.18kB  9.65MB / 152kB   1
```

Parte 6 — Publicar en Docker Hub



Repositories / mi-app-python / General

Using 0 of 1 private repositories.

nisevi/mi-app-python

Last pushed 2 minutes ago · Repository size: 45.1 MB · ☆0 · ↓10

Add a description

Add a category

Docker commands

To push a new tag to this repository:

```
docker push nisevi/mi-app-python:tagname
```

Tags

DOCKER SCOUT INACTIVE

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	3 minutes
v1		Image	less than 1 day	10 minutes

Repository overview INCOMPLETE

An overview describes what your image does and how to run it. It displays in the public view of your repository once you have pushed some content.

buildcloud

Build with
Docker Build Cloud

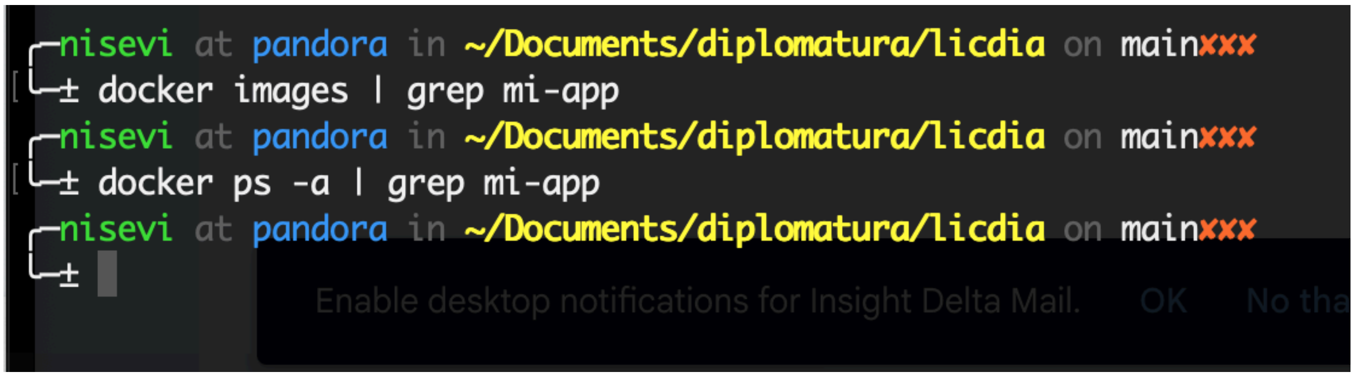
Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without managing infrastructure.

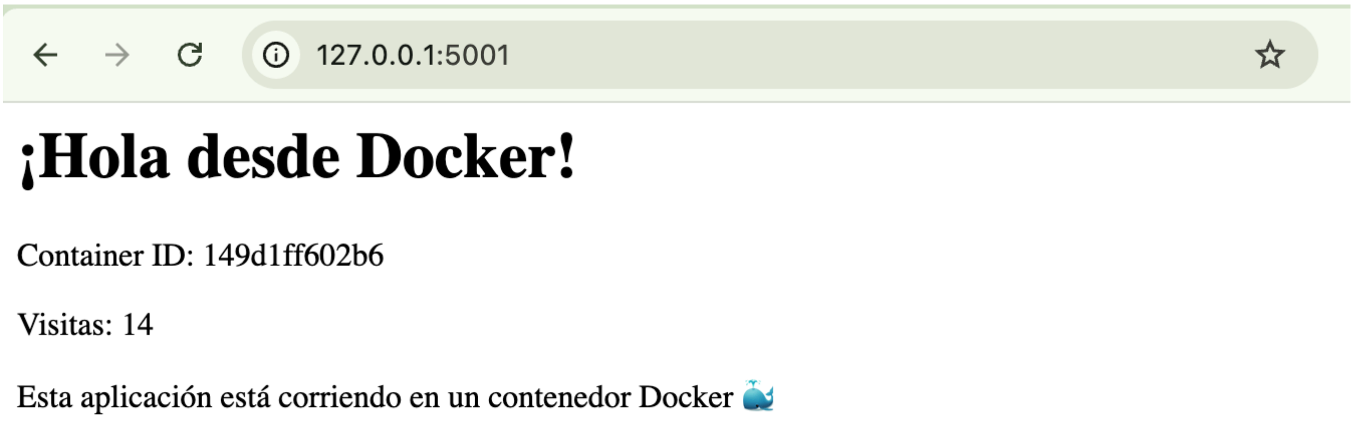
Docker image URL: https://hub.docker.com/r/nisevi/mi-app-python

Parte 7 — Limpieza de Recursos



```
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on main✘✘✘
└─± docker images | grep mi-app
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on main✘✘✘
└─± docker ps -a | grep mi-app
┌─nisevi at pandora in ~/Documents/diplomatura/licdia on main✘✘✘
└─±
```

Desafío Extra



¡Hola desde Docker!

Container ID: 149d1ff602b6

Visitas: 14

Esta aplicación está corriendo en un contenedor Docker 🐳