Contenido

[Instalación de WSL 2 1](#_Toc184275499)

[Creando una aplicación con flask 1](#_Toc184275500)

[Docker 4](#_Toc184275501)

[¿Qué es docker? 4](#_Toc184275502)

[Instalación de Docker 4](#_Toc184275503)

[Manejando imáges con docker images (para listar), docker pull nombre\_imagen(para descargar), docker image rm nombre\_imagen (para borrar) 9](#_Toc184275504)

[Docker images 9](#_Toc184275505)

[Creando nuestra imagen a partir de el Python-alpine 12](#_Toc184275506)

# Instalación de WSL 2

He creado un artículo único sobre esto:

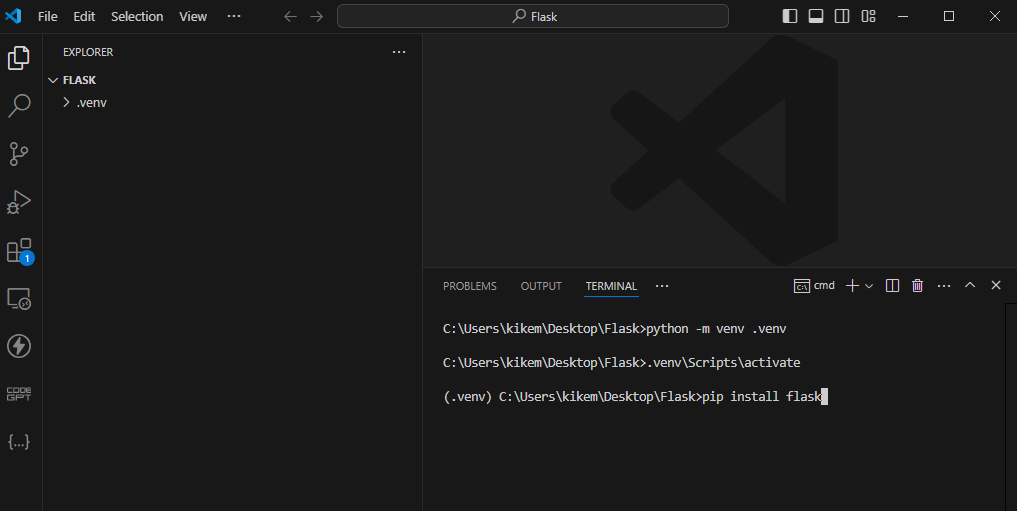
# Creando una aplicación con flask

Escribimos en el cmd o terminal

python -m venv .venv

.venv\Scripts\activate

pip install flask



Creamos una carpeta llamada src y dentro nuestro main.py, la carpeta templates con el archivo plantilla.html y index.html, vamos a pasarle a la vista unos usuarios:

archivo: main.py

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

# Routes

@app.route("/")

@app.route("/index")

def index():

    usuarios=["Fran", "Kike", "Pablo","Teo"]

    return render\_template("index.html", usuarios=usuarios)

# Start the Server

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host="0.0.0.0", port=3000, debug=True)

archivo templates\plantilla.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>{% block title %}{% endblock %}</title>

</head>

<body>

    <h1>Curso avanza de Python</h1>

    {% block content %}{% endblock %}

</body>

</html>

archivo templates\index.html

{% extends "plantilla.html" %}

{% block title %}Home{% endblock %}

{% block content %}

    <h2>Bienvenidos</h2>

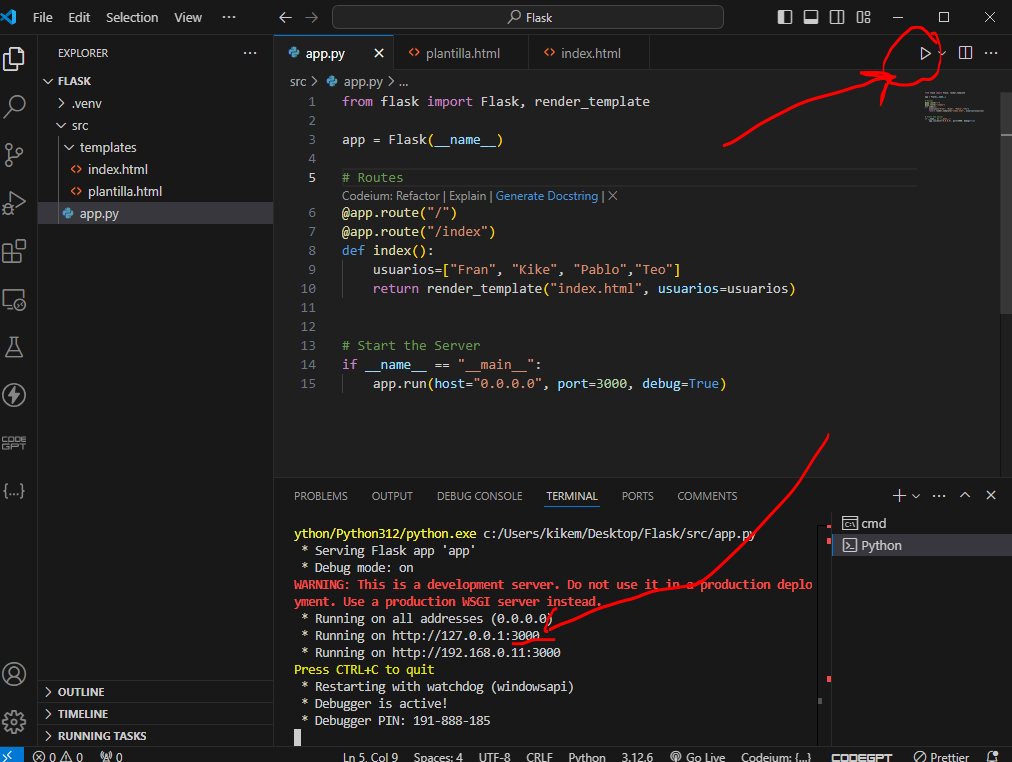
    {% for user in usuarios %}

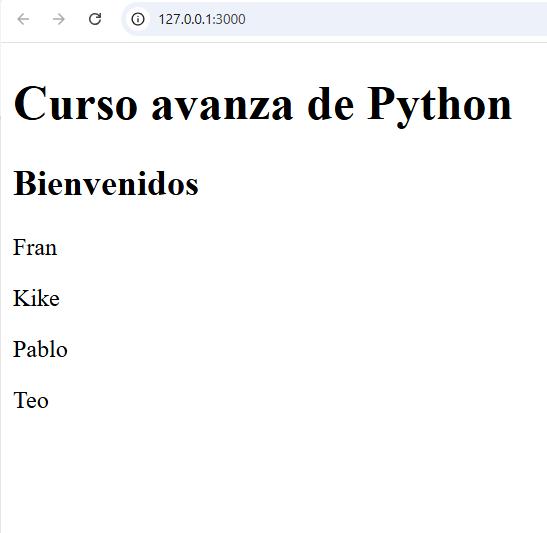
        <p>{{ user }}</p>

    {% endfor %}

{% endblock %}

Pinchamos en el play para ver que funciona, tomamos nota que el flask se ejecuta en el puerto 3000:





# Docker

## ¿Qué es docker?

Docker es un gestor de contenedores, pero ¿Qué es un contenedor?

Un contenedor es una máquina virtual especial, un contendor se compone de:

1. Una imagen
2. Una configuración

Aunque si es posible, los contenedores no tienen interface gráfica y son casi siempre utilizados para crear entorno de desarrollo de páginas web y entorno de cdi o testeo y despliegue de tu app web.

Aunque pueda parecerle realmente trabaja con el núcleo de Linux y las imágenes que te descargas son librerías que se añaden a este núcleo de linux que esta dentro de tu pc, por lo tanto para arrancarlos son tarda pocos segundos, al correr aislados son ideales para microservicios y para seguridad.

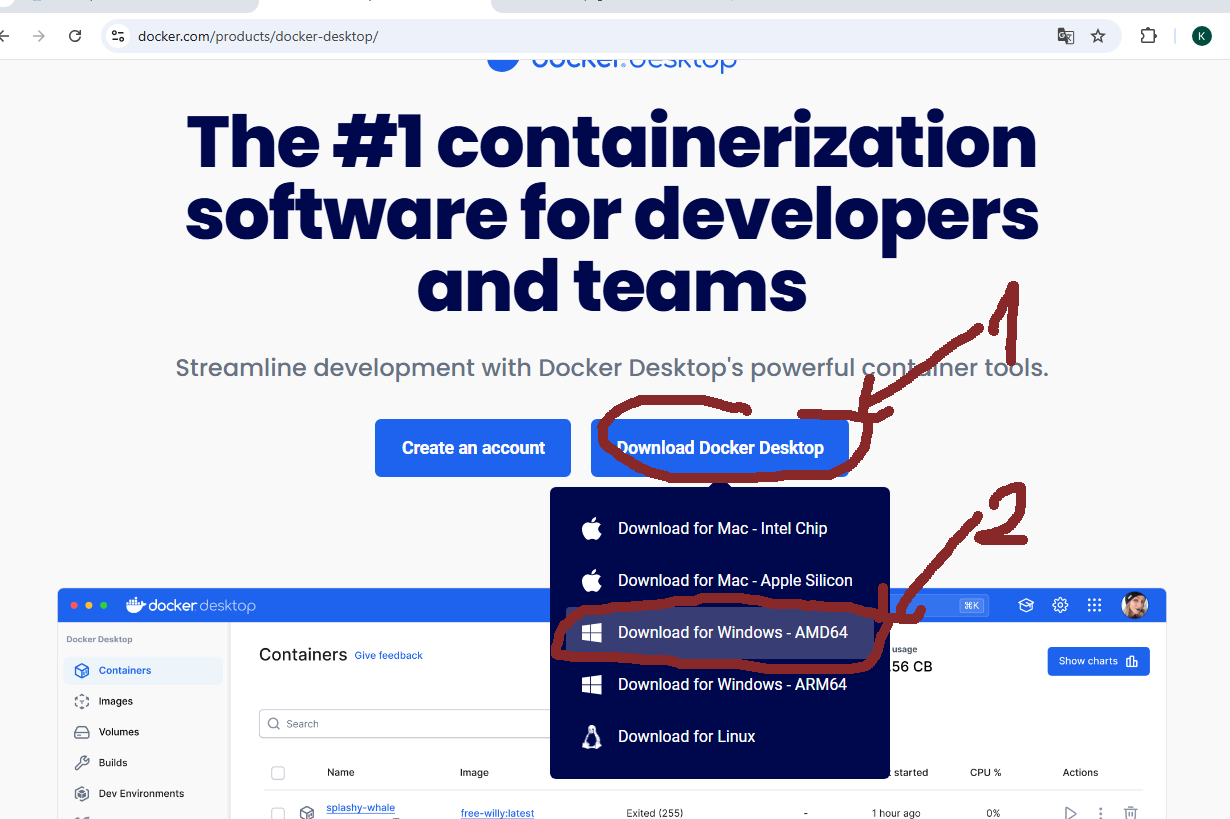
¿Pero que es una imagen?

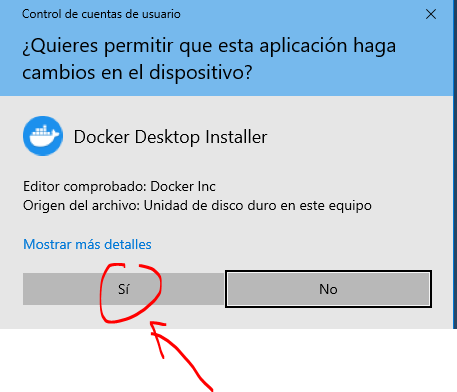
Una imagen son las librerías que necesita docker para correr tu servido java ee, tu flask o tu node y que está todas en docker hub.

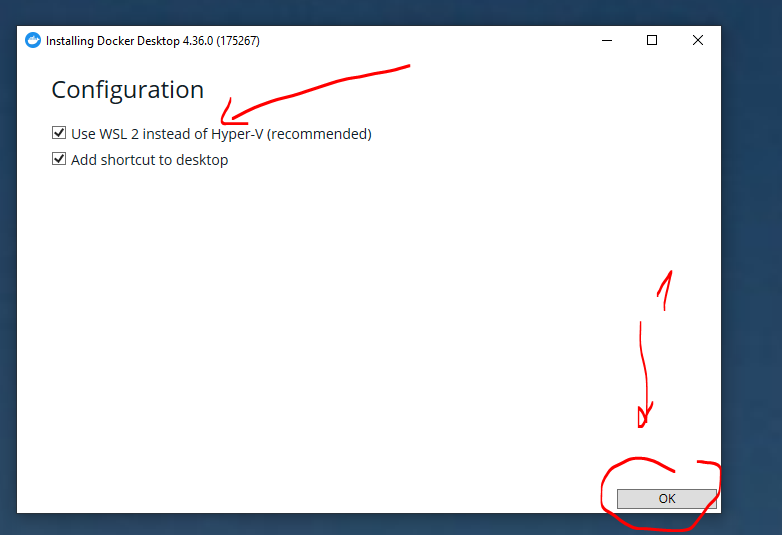
## Instalación de Docker

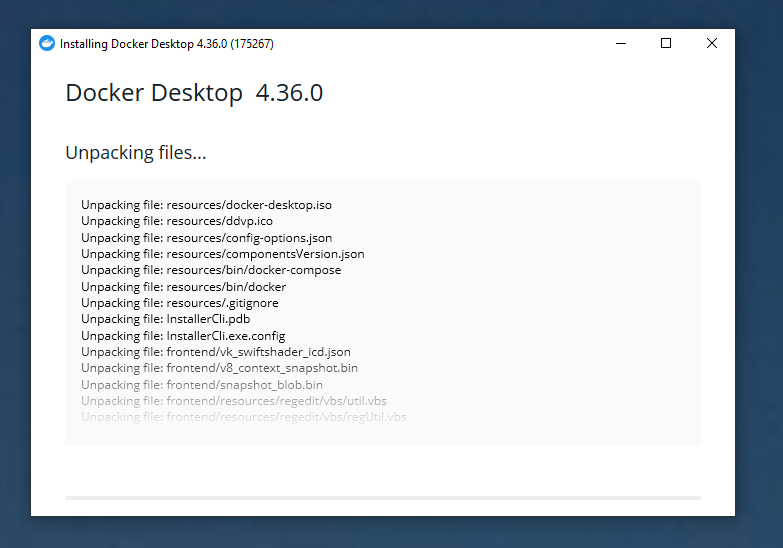
Vamos a la web de Docker y nos lo descargamos:

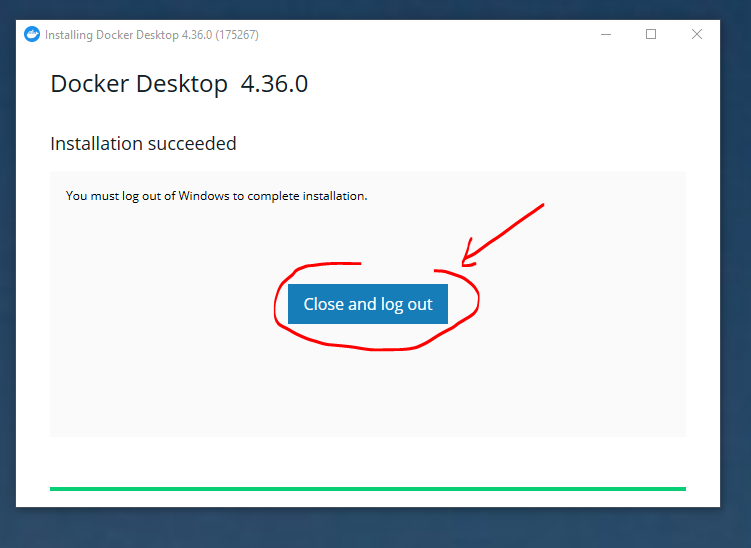
<https://www.docker.com/products/docker-desktop/>



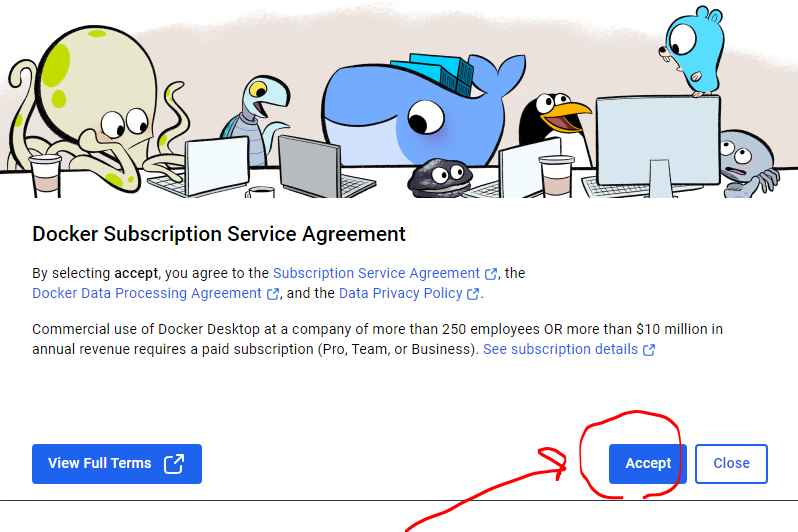


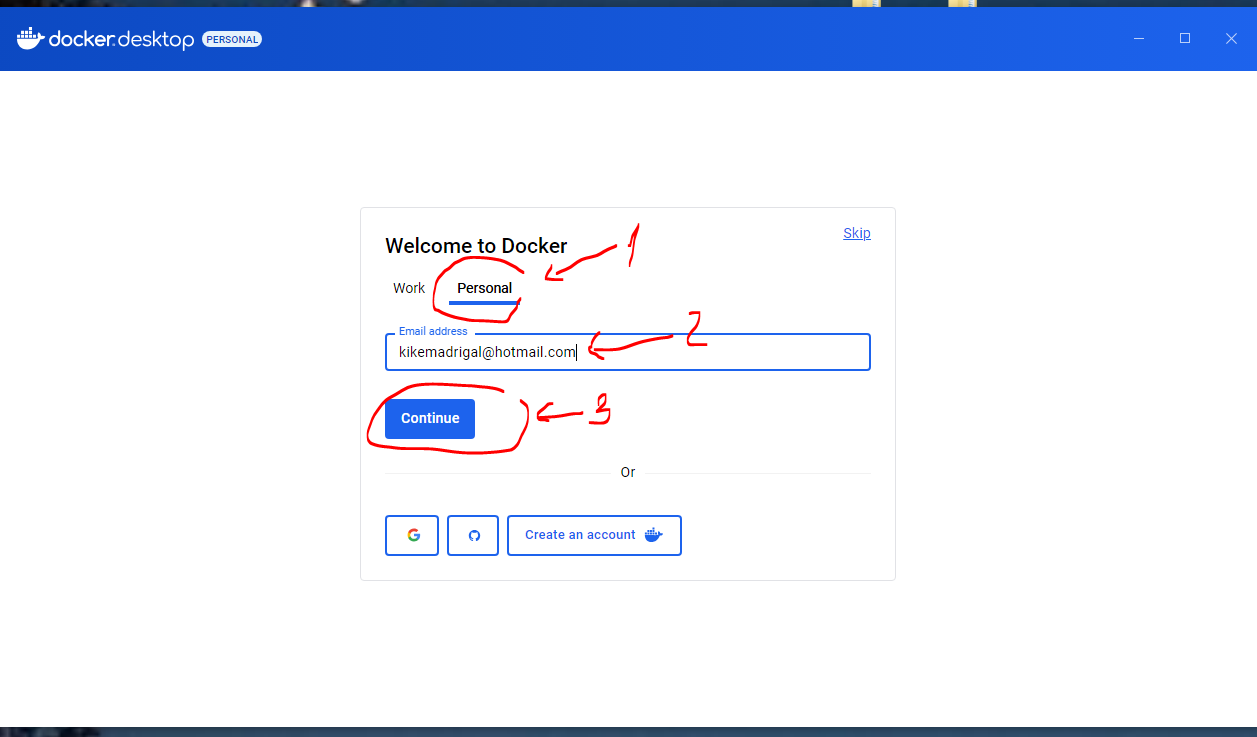




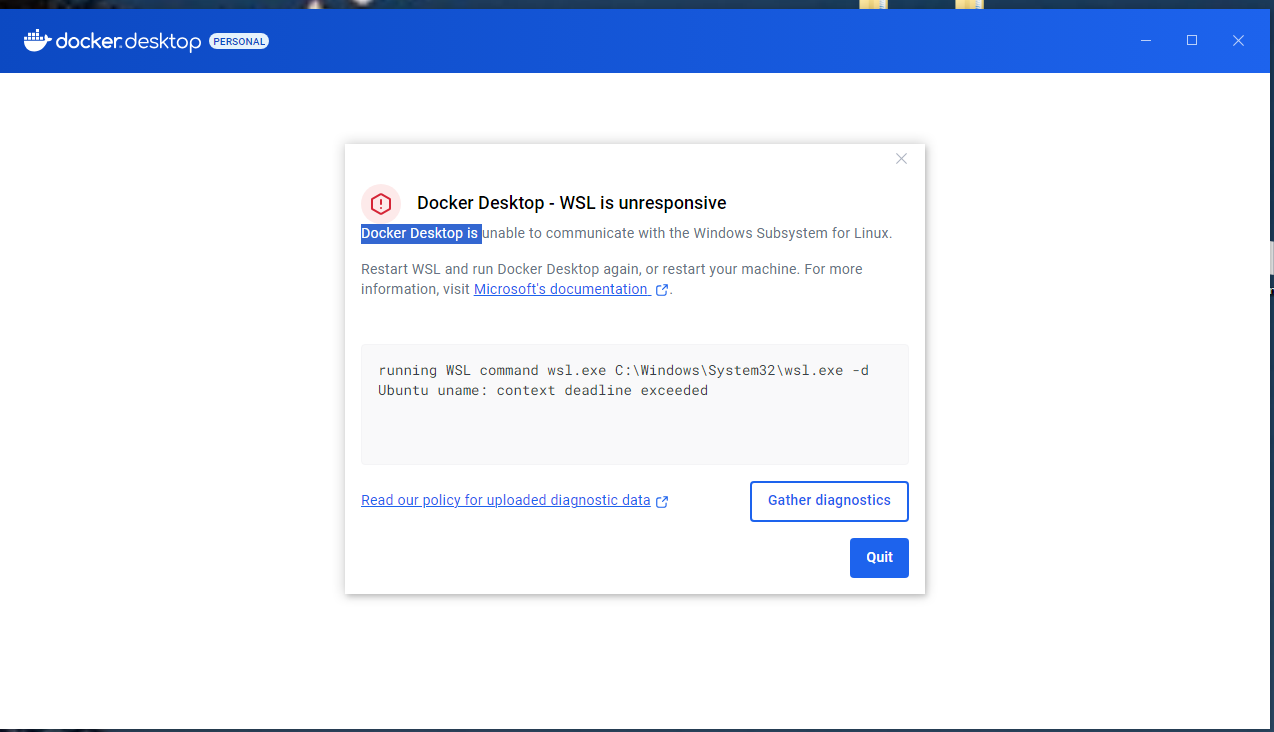


Se reiniciará el ordenador, al reiniciar saldrá de nuevo Docker, pincha en Accept:





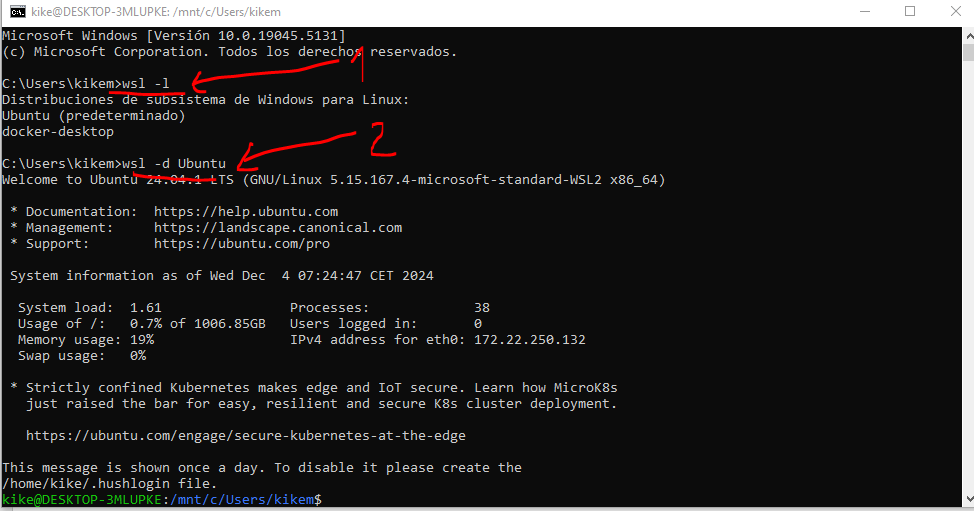
Si sale este mensaje es que hay que activar el wsl:

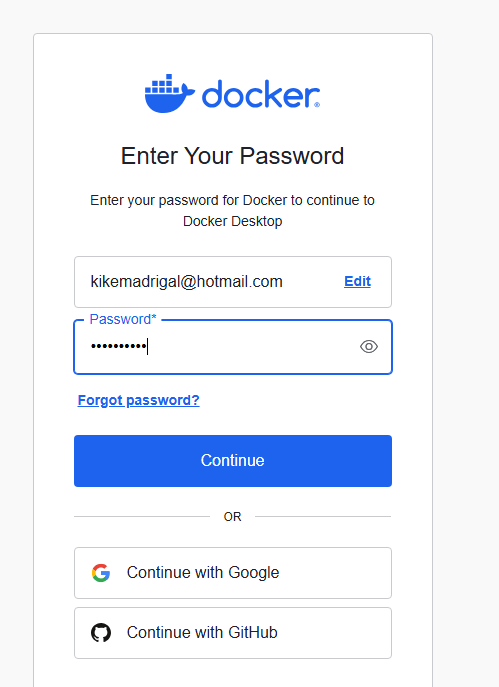


Ve al cmd y escribe:

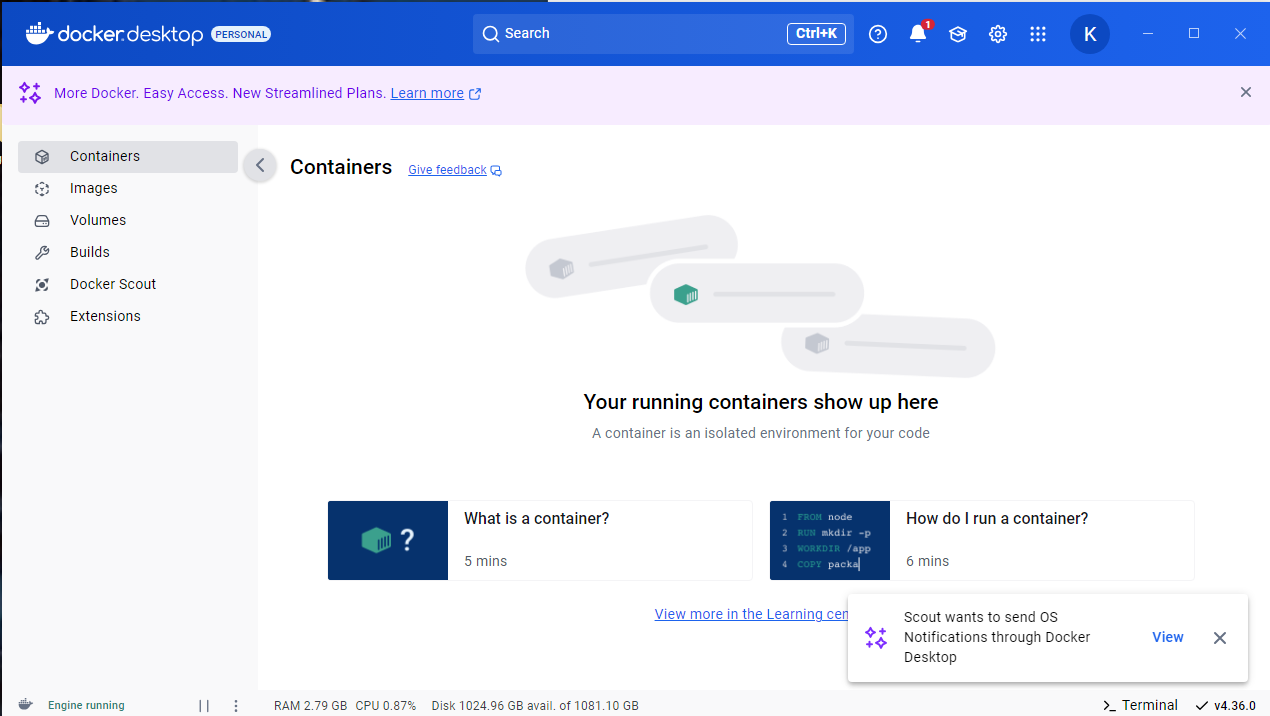
wsl -l -v Para ver las distribuciones que tenemos.

wsl -d nombre\_distribución para arrancarla

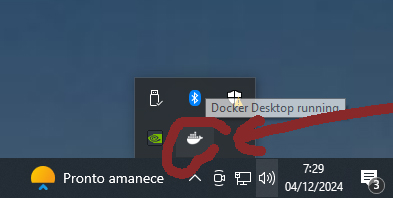




Una vez que tenemos esta pantalla, la cerramos:



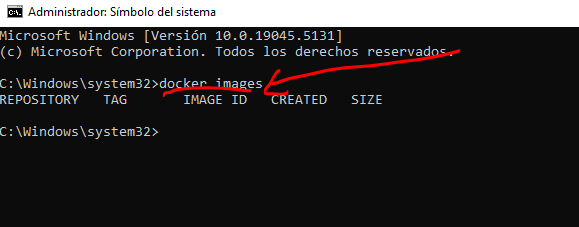
Docker se quedará abajo en los programas activos:



## Manejando imáges con docker images (para listar), docker pull nombre\_imagen(para descargar), docker image rm nombre\_imagen (para borrar)

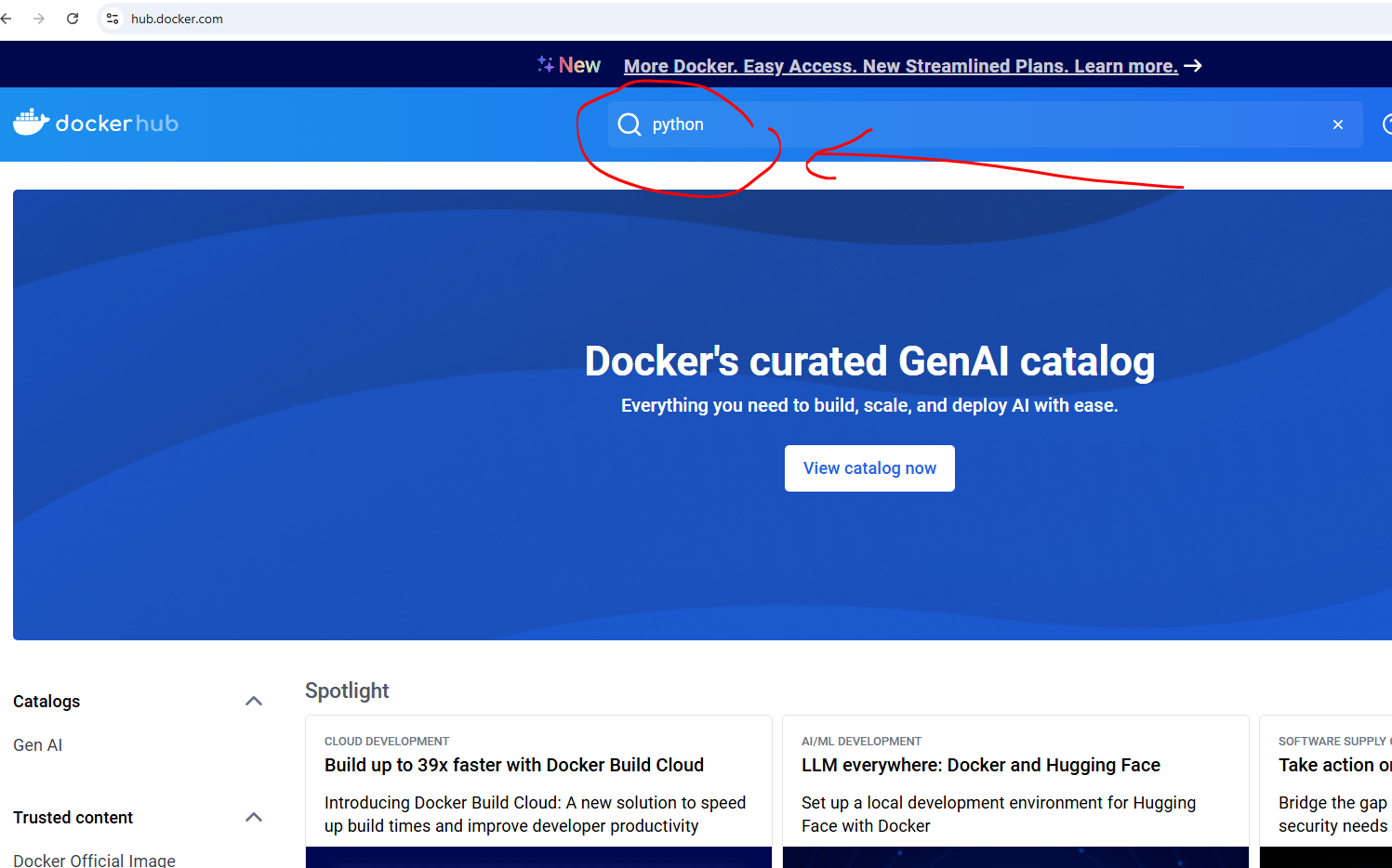
### Docker images

Si escribimos Docker images vemos que no tenemos ninguna descargada:

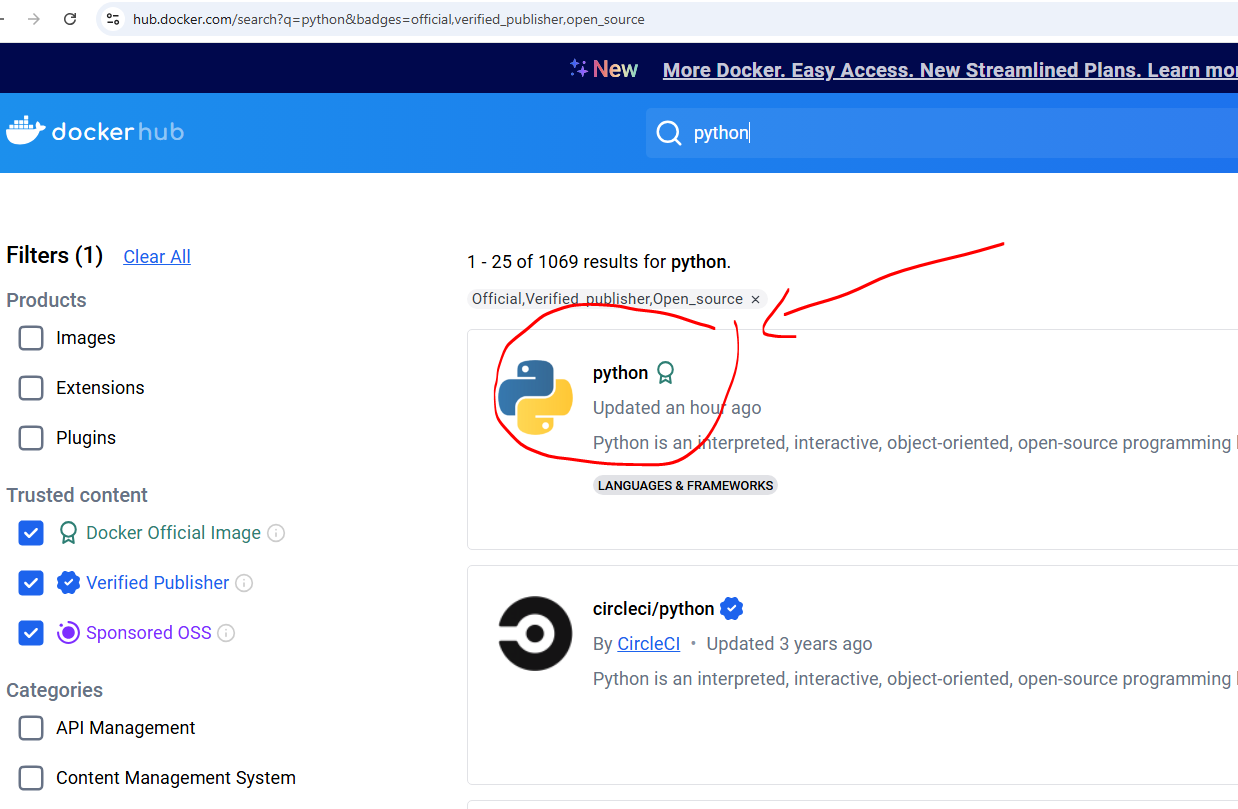


Descargando imágenes de docker hub

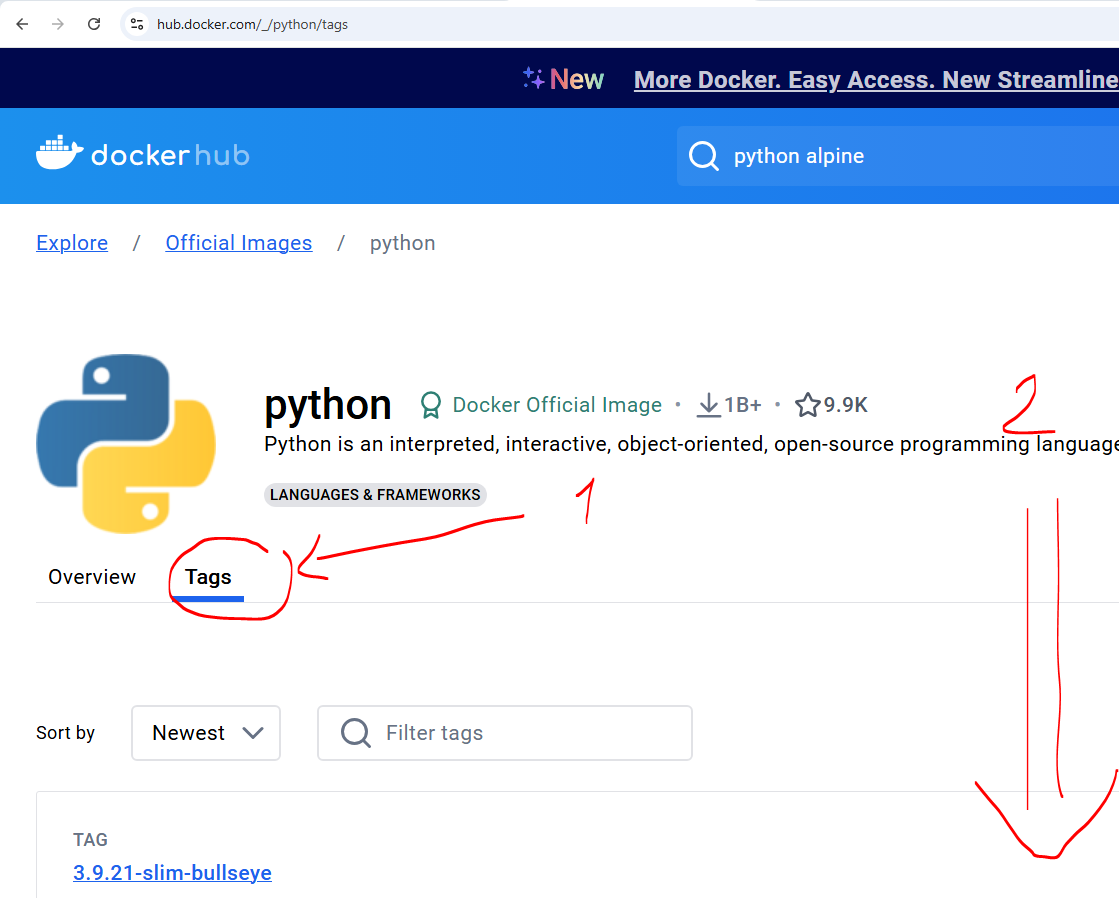
Vamos a docker hub y en el buscardor escribimos Python:

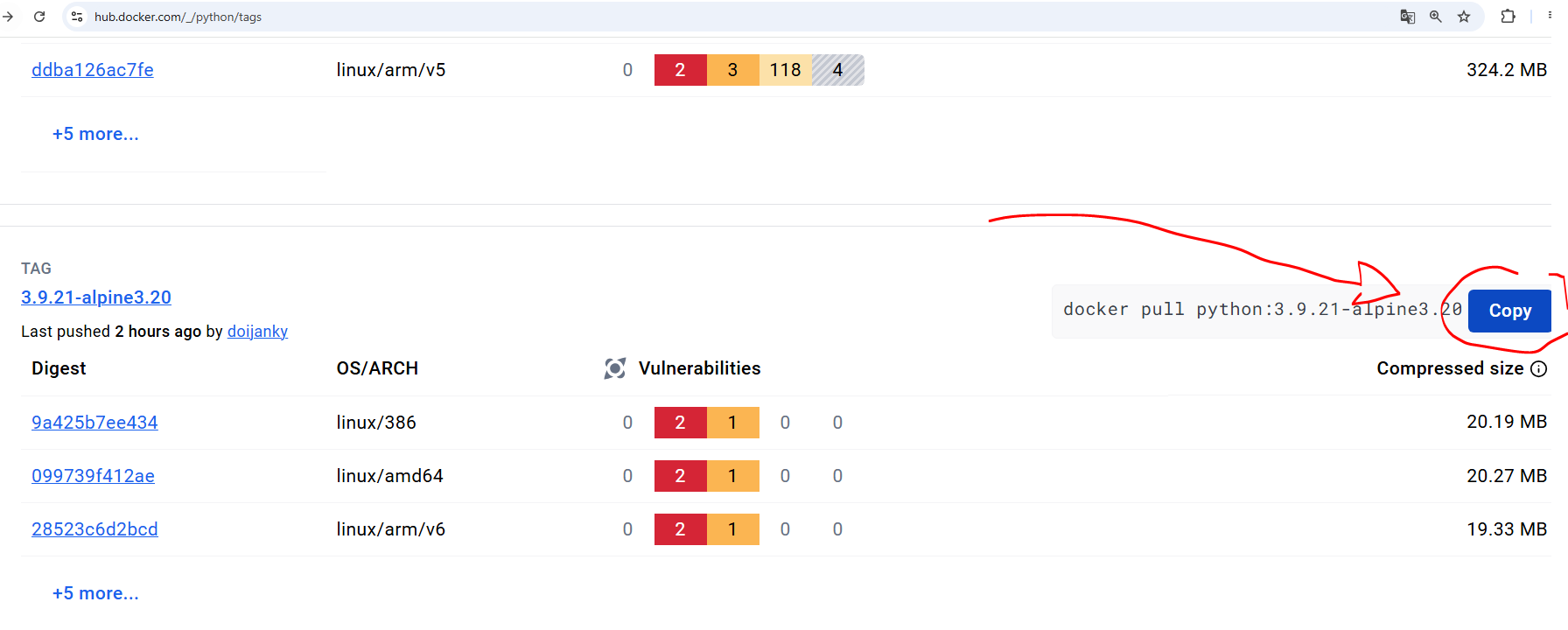


Pinchamos en la primera

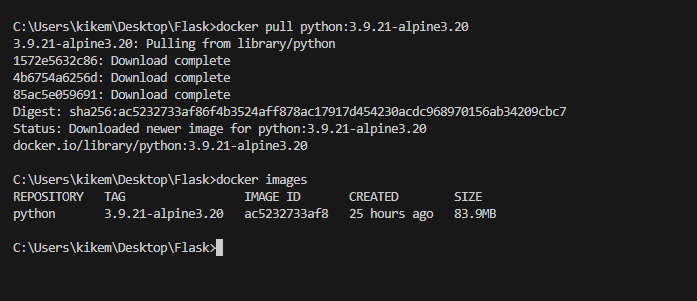


Elegimos la versión de alpine que son las más ligeras:



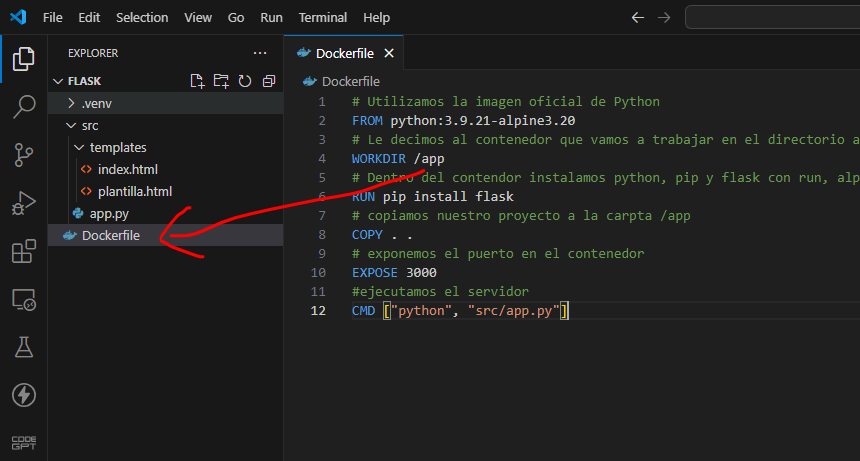


Descargamos la imagen, al escribir docker images la veremos descargada:



## Creando nuestra imagen a partir de el Python-alpine

Creamos un archivo en la raíz llamado Dockerfile:



Dentro le ponemos este texto:

# Utilizamos la imagen oficial de Python

FROM python:3.9.21-alpine3.20

# Le decimos al contenedor que vamos a trabajar en el directorio app

WORKDIR /app

# Dentro del contendor instalamos python, pip y flask con run, alpine utiliza el administrador de paquetes apk

RUN pip install flask

# copiamos nuestro proyecto a la carpta /app

COPY . .

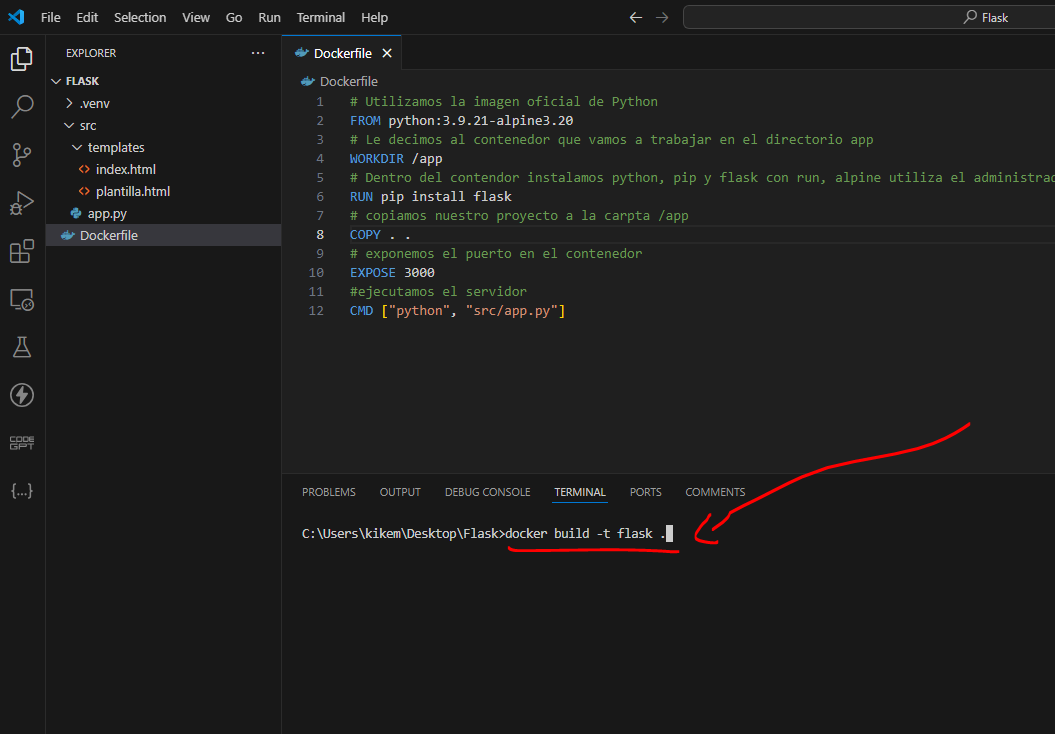
# exponemos el puerto en el contenedor

EXPOSE 3000

#ejecutamos el servidor

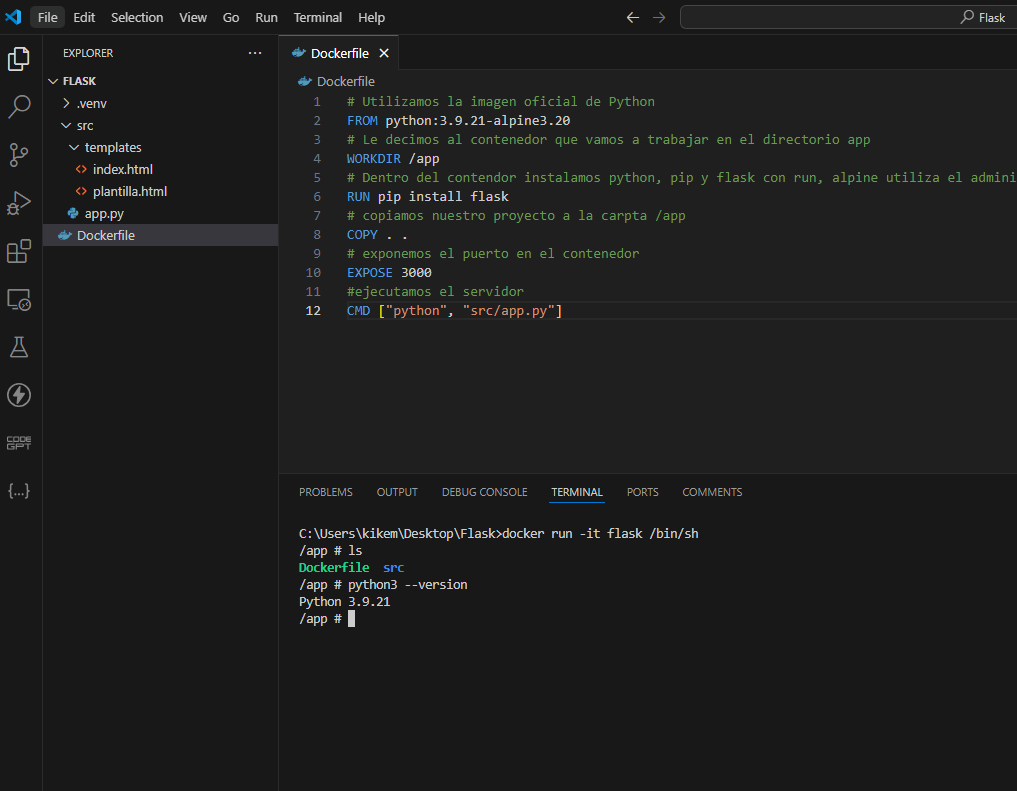
CMD ["python", "src/app.py"]

Escribimos docker build -t nombre\_de\_nuestra\_imagen . (el punto es para decirle que se ubique en el directorio actual):



Escribimos docker run -it nombre\_de\_nuestra\_imagen para crear, correr nuestro conetendor y ejecutarlo en modo interactivo, de esta forma nunca termina

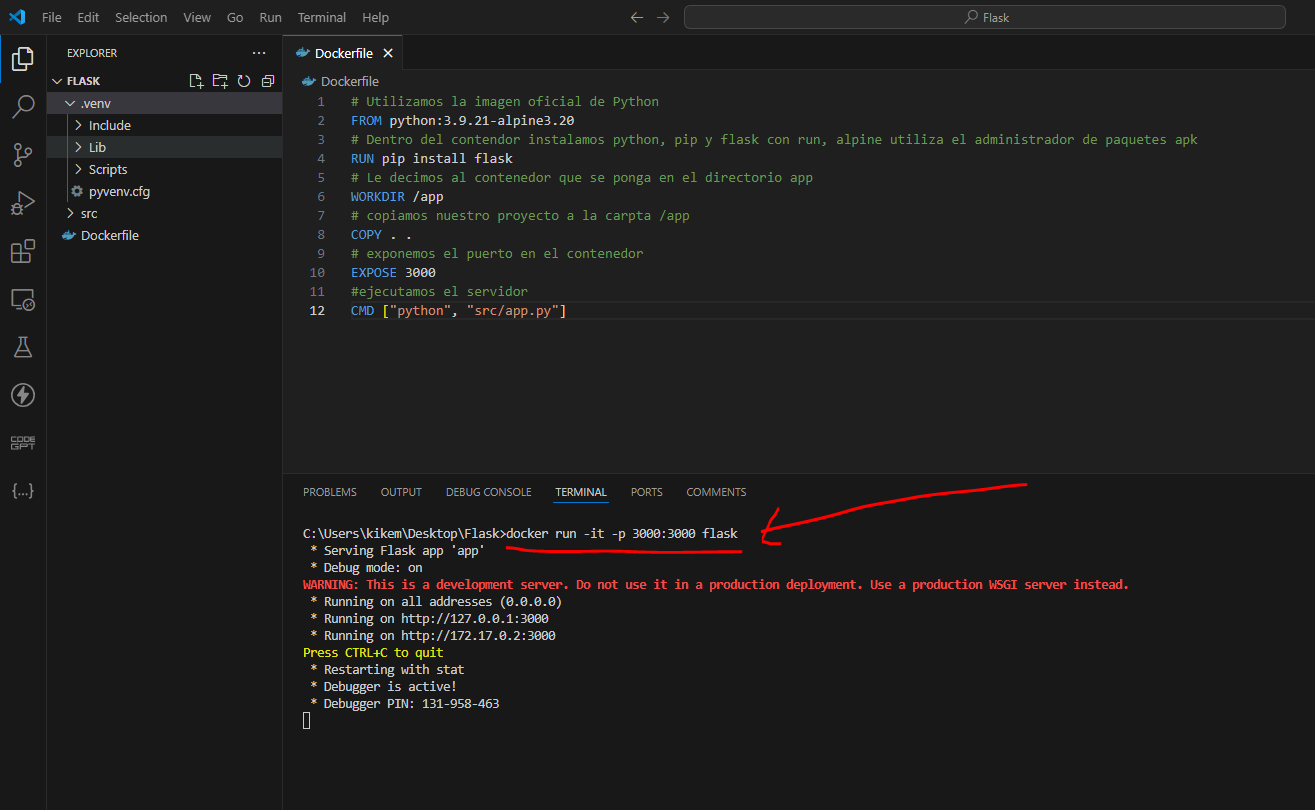
docker run -it -p 3000:3000 flask



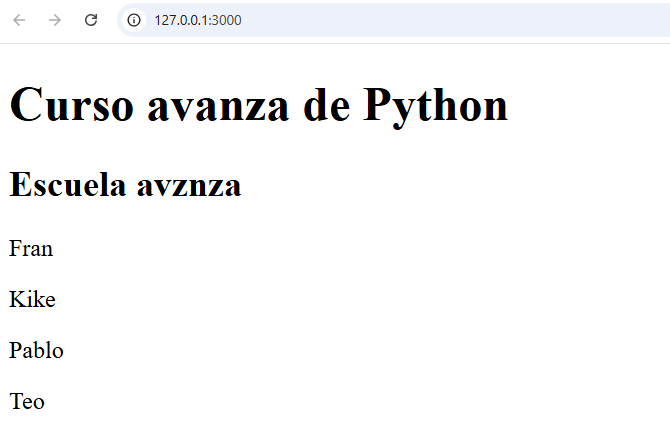
Escribimos exit para salir del contenedor, ahora escribimos:

Docker run -it -p 3000:3000 flask

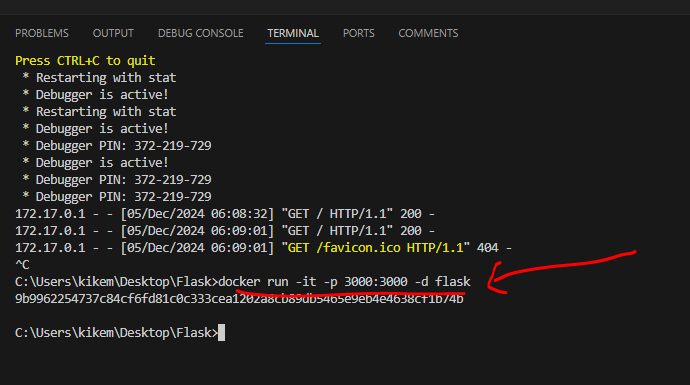
El -p es para decirle que redirecciones el puerto 3000 de nuestro contenedor al puerto 3000 de nuestro pc



Ya tenemos nuestro flask dockerizado:



Hay otro paso que está bien que seamos y es que podemos dejar a nuestro servidor Python en background, esto se hace con el flag -d, de esta forma nunca se parará:



Para pararlo tenemos que poner docker stop y el nombre o el id del contendor

