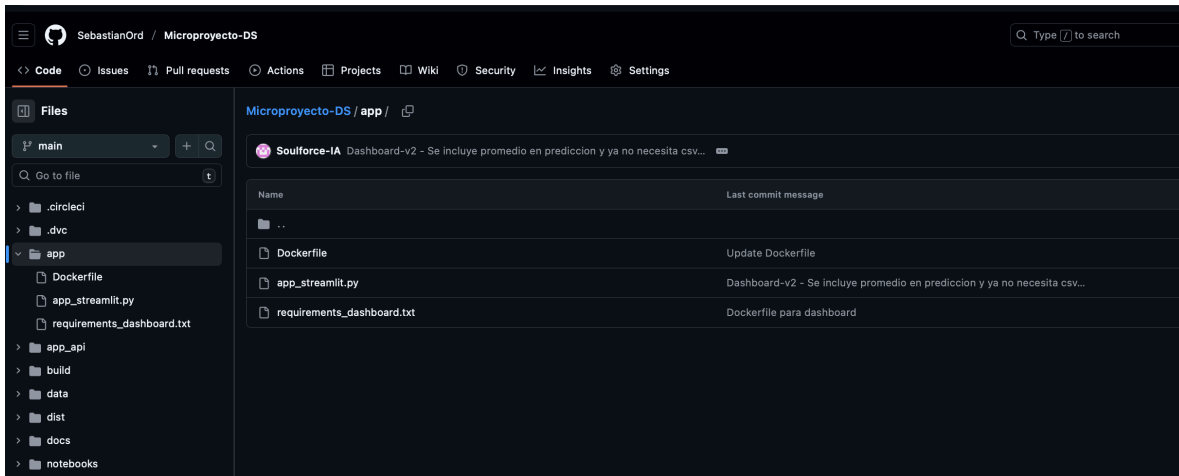


Manual de instalación

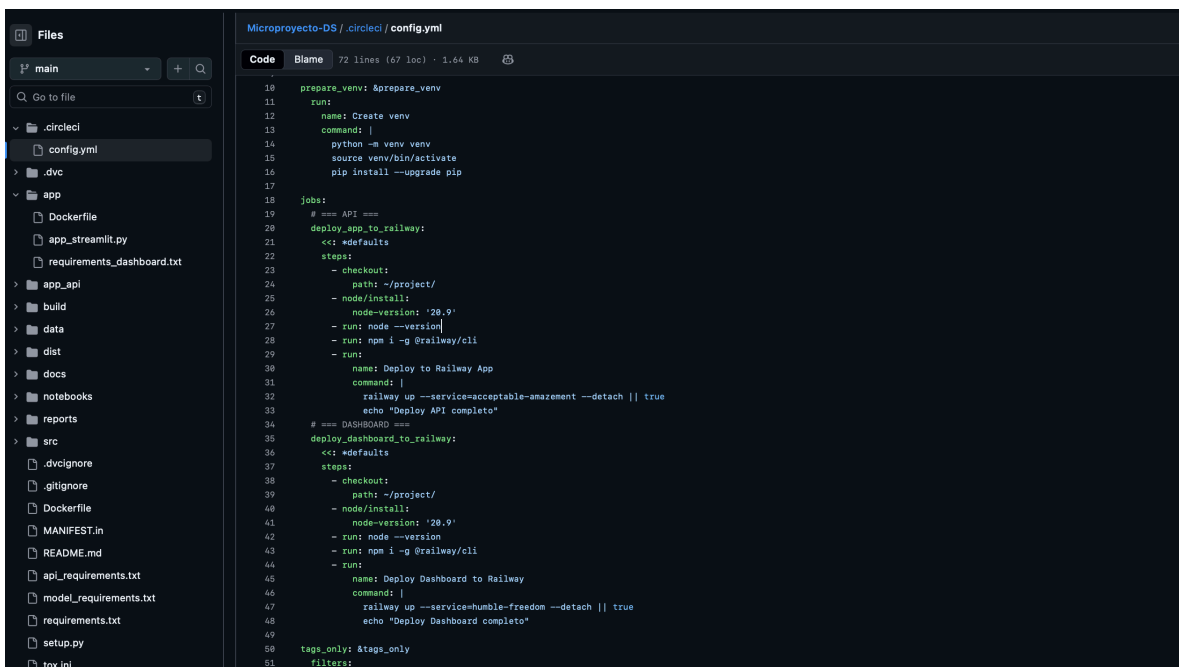
El dashboard se encuentra en el repositorio en la carpeta /app,

el archivo principal es app_streamlit.py y se encuentra acompañado de requirements_dashboard.txt y Dockerfile

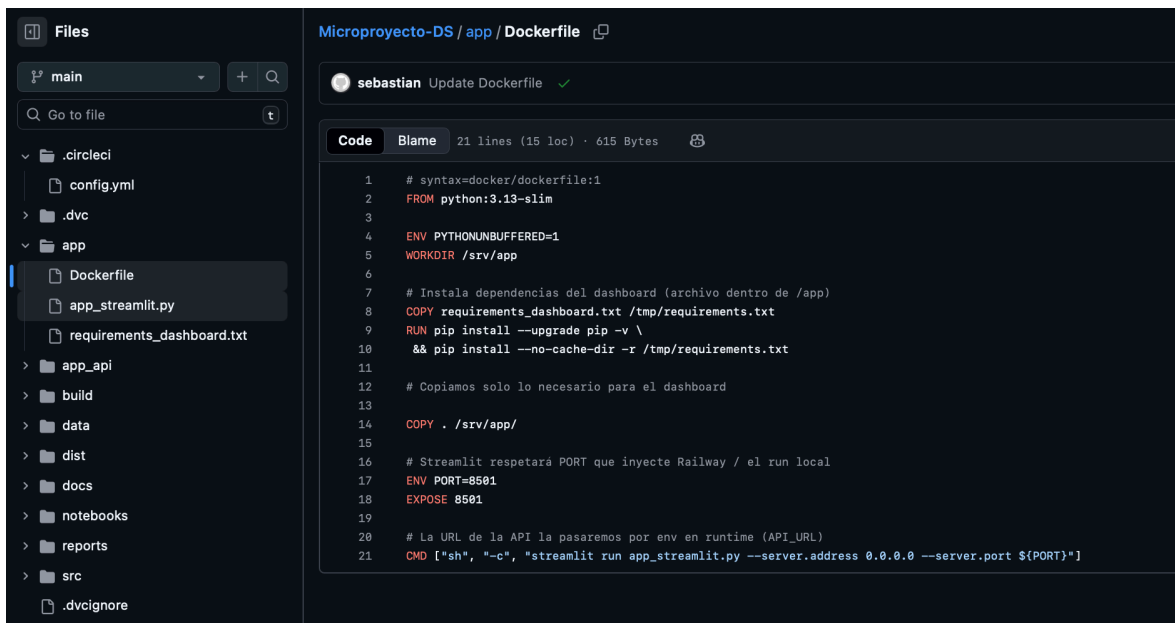


Estos últimos dos archivos son imprescindibles a la hora de realizar el despliegue dado que se realiza automáticamente por medio de circleCI

En la carpeta .circleci se encuentra el archivo config.yml donde se encuentra la tarea que se ejecuta para llevar a railway todo lo necesario para realizar el despliegue de la aplicación



El Dockerfile es el archivo que lee railway y del cual se basa el servicio para desplegar y ejecutar la instalación

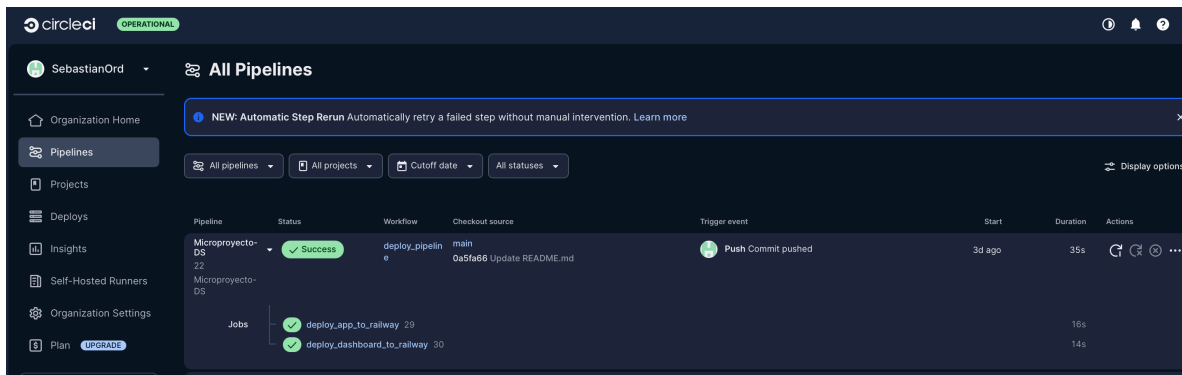


```
1 # syntax=docker/dockerfile:1
2 FROM python:3.13-slim
3
4 ENV PYTHONUNBUFFERED=1
5 WORKDIR /srv/app
6
7 # Instala dependencias del dashboard (archivo dentro de /app)
8 COPY requirements_dashboard.txt /tmp/requirements.txt
9 RUN pip install --upgrade pip -v \
10     && pip install --no-cache-dir -r /tmp/requirements.txt
11
12 # Copiamos solo lo necesario para el dashboard
13
14 COPY . /srv/app/
15
16 # Streamlit respetará PORT que inyecte Railway / el run local
17 ENV PORT=8501
18 EXPOSE 8501
19
20 # La URL de la API la pasaremos por env en runtime (API_URL)
21 CMD ["sh", "-c", "streamlit run app_streamlit.py --server.address 0.0.0.0 --server.port ${PORT}"]
```

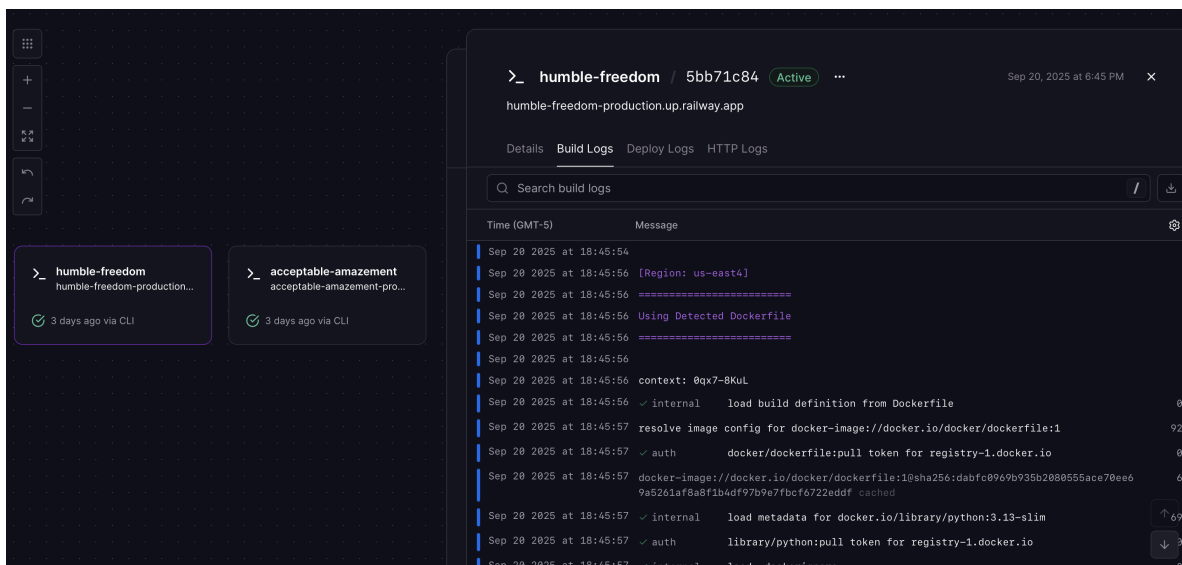
Vamos a explicar paso por paso que hace este dockerfile

1. Define la **imagen base**: Debian “slim” con Python 3.13 ya instalado.
2. Con la variable PYTHONUNBUFFERED Python **no “almacene en buffer” la salida** (stdout/stderr).
3. Establece la **ruta /srv/app como directorio de trabajo** dentro del contenedor.
4. Copia el archivo de dependencias del proyecto al contenedor (a /tmp/requirements.txt).
5. Actualiza pip e instala dependencias (del archivo anterior)
6. Copia todo lo que se encuentre en la carpeta /app (definido en el servidor de railway) a la carpeta /srv/app
7. Define la variable PORT con valor de 8501
8. Expone el puerto 8501
9. Comando de streamlite para ejecutar en la ruta y puerto definido la aplicación a partir del archivo app_streamlit.py

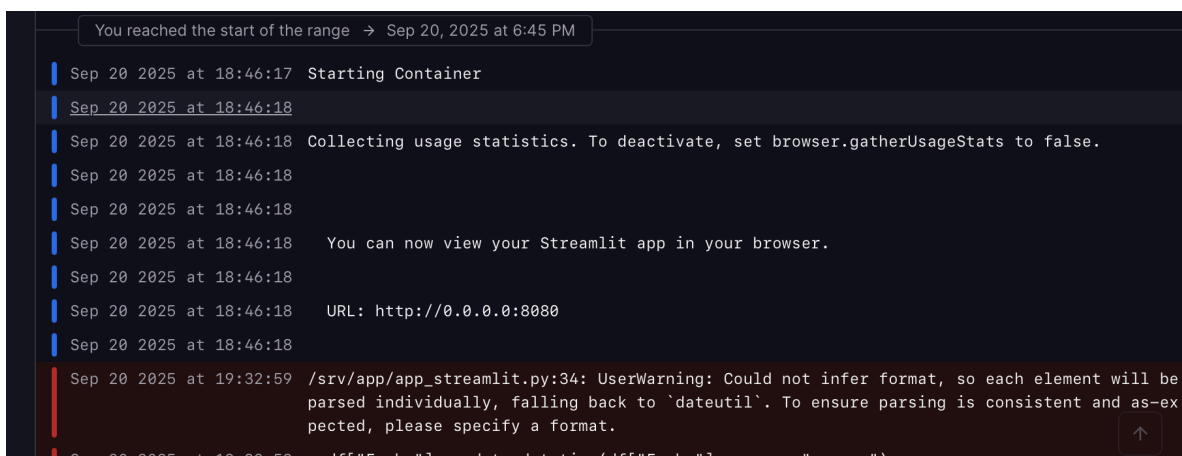
EL flujo de trabajo funciona de la siguiente manera, una vez detectado un commit, circleCI lanza un pipeline que contiene las dos tareas a ejecutar definidas en el archvo config.yml de .circleci



Una vez finalizado el pipeline, se envía a railway (conectado por un token con circleCI) los datos y archivos necesarios para el despliegue, una vez railway recibe esta información empieza el build a partir del dockerfile



Donde finalmente se despliega un contenedor corriendo la aplicación



Por último, railway dispone de un dominio gratuito donde expone el resultado final, en este caso el link es:

<https://humble-freedom-production.up.railway.app>

donde se puede ingresar y probar el dashboard.