Desplegar Reflex en Render

Lo primero es crearse una cuenta en Render: https://render.com/

Imagen 1. Ir al dashboard de Render

Si es la primera vez que se ingresa, pedirá crear una cuenta ya sea con Google, Git y que mediante el mail completando un formulario. La cuestión es que cuando se complete la creación de la cuenta, la hacer click en ***Dashboard*** e iniciar sesión aparecerá algo similar a esto:

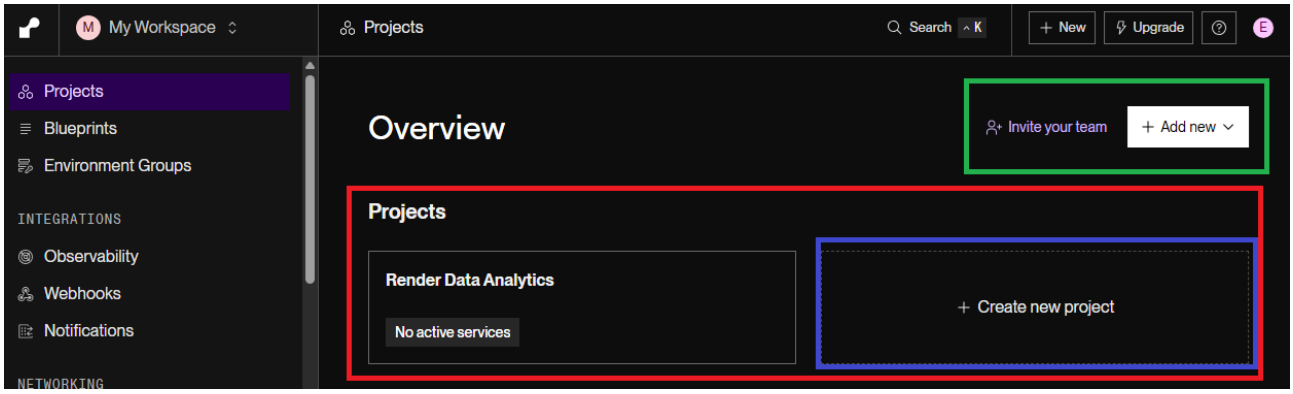

Imagen 2. Dashboard principal de Render.

La parte indicada con verde en la imagen 2 es para invitar gente al proyecto. La parte señalada en rojo son los proyectos que tenemos activos y la parte azul es para crear un nuevo proyecto. Si no se tiene ningún proyecto activo, debe irse a la opción marcada en azul.
Pero, antes de crear nuestro proyecto nuevo vamos a ver qué necesitamos tener en el proyecto de Reflex para subirlo a Render.
Lo primero es crear un repositorio en Github (yo lo probé en Github, pero debería funcionar también En Git Bucket o en Gitlab).
Cuando el repositorio esté creado, debemos subir los archivos de nuestro proyecto. Mi proyecto en Reflex se ve de la siguiente manera:

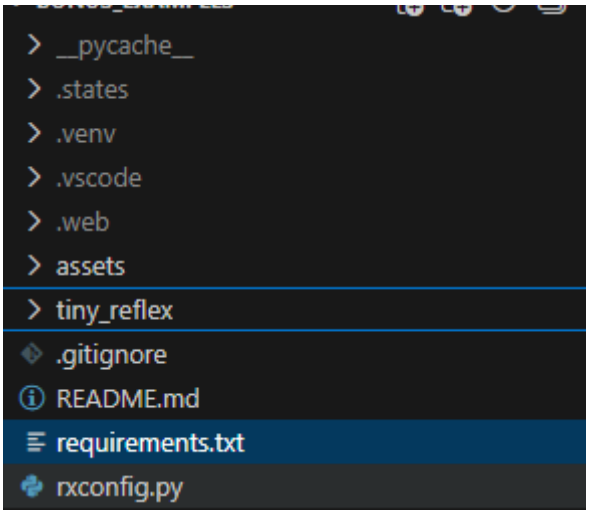

Imagen 3. Archivos de Render

Estos archivos deben estar en el entorno virtual creado con ***env*** para Python y allí ya se debió instalar Reflex y haber ejecutado ***Reflex init***.
Si ejecutamos ***pip install -r requirements.txt***
Nos creará el archivo ***requirements.txt***. Si comente el error que cometí yo de crear el proyecto fuera de la máquina virtual, les saldrá un archivo extremadamente largo. Por lo tanto, lo necesario para realizar el despliegue es que el archivo ***requirements.txt*** contenga lo siguiente:

```
# Framework web
reflex==0.8.21

# Data analysis
pandas>=2.0
numpy>=1.24

# Visualización
plotly>=5.20
kaleido>=0.2

# Base de datos
SQLAlchemy>=2.0
psycopg2-binary>=2.9

# Utilidades usadas por Reflex
httpx>=0.28
starlette>=0.50
typing_extensions>=4.0
```

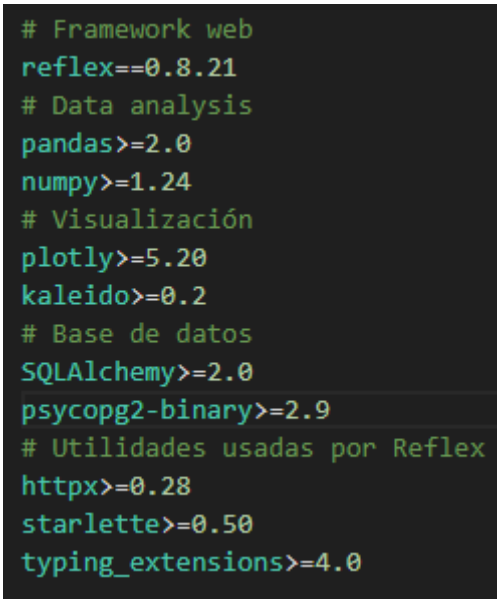

Imagen 4. Contenido del archivo requirements.txt

En el archivo ***.gitignore*** tengo lo siguiente

Imagen 5. Contenido del archivo .gitignore

Allí deben colocar todo lo que no quiere que se suba. Algunas carpetas como ***.web y .state*** se crearán en Render cuando despleguemos el proyecto, así que deben ser ignorados para subirlos al repositorio.
A continuación, debemos crear el archivo ***render.yaml*** que tendrá el siguiente contenido:

```
services:
  - type: web
    name: reflex-app
    env: python
    plan: free
    buildCommand: |
      pip install --upgrade pip
      pip install -r requirements.txt
      reflex export
    startCommand: reflex run --env prod --host 0.0.0.0 --port $PORT
    envVars:
      - key: PYTHON_VERSION
        value: 3.11
```

Originalmente, el archivo ***rxconfig.py*** se veía así

```python
import reflex as rx

config = rx.Config(
    app_name="tiny_reflex",
    plugins=[rx.plugins.SitemapPlugin()],

)
```

Imagen 6. Contenido original del archivo rxconfig.py

Debe modificarse para que quede así

```python
import reflex as rx

config = rx.Config(
    app_name="tiny_reflex",
    plugins=[rx.plugins.SitemapPlugin()],
    backend_host="0.0.0.0",
    backend_port=8000
)
```

Imagen 7. Contenido modificado del archivo rxconfig.py

Luego, necesitamos el archivo de conexión a la base de datos. Para ello, debemos abrir la carpeta ***tiny_reflex*** y veremos la siguiente lista de archivos:

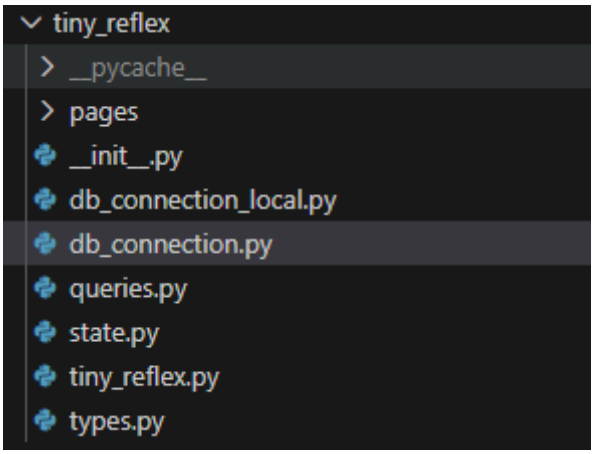

Imagen 8. Contenido de la carpeta tiny_reflex

El archivo ***db_connection_local.py*** tiene la información de conexión para conectarse a PostgreSQL de manera local. En el archivo ***db_connection.py*** colocaremos la información para conectarnos a la base de datos que tenemos levantada en ***Neon***.
El archivo ***db_connection.py*** debería quedar así:

```python
import os
from sqlalchemy import create_engine

def get_engine():
    database_url = os.getenv("DATABASE_URL")
    if not database_url:
        raise ValueError("DATABASE_URL is not set")
    return create_engine(database_url, pool_pre_ping=True)
```

Imagen 9. Contenido del archivo db_connectio.py

En el archivo ***queries.py*** tenemos la información de las consultas realizadas a la base de datos. Originalmente, la tenemos así para trabajar en local:

```python
"""Database query functions."""

import pandas as pd
from typing import cast
from tiny_reflex.db_connection_local import get_engine
from tiny_reflex.types import DimCustomerData,SalesForCustomersData, SalesForStateCustomerData

def load_sales_for_customers()->list[SalesForCustomersData]:
    """Load average and total sales for customers"""
```

Imagen 10. Contenido del archivo queries.py

La línea:

```
from tiny_reflex.db_connection_local import get_engine
```

Debemos cambiar a

```
from tiny_reflex.db_connection import get_engine
```

Para que use la conexión a Neon.

Con todos los cambios realizados, aplicamos el commit a la rama que queramos y ahora nos dirigimos primero Render. En el dashboard de Render, si ya tenemos creado un proyecto, no podremos tener otro. El plan gratis sólo te deja tener un proyecto. Así que yo elegiré el proyecto que ya tengo.

Al seleccionarlo, nos mostrará lo siguiente:

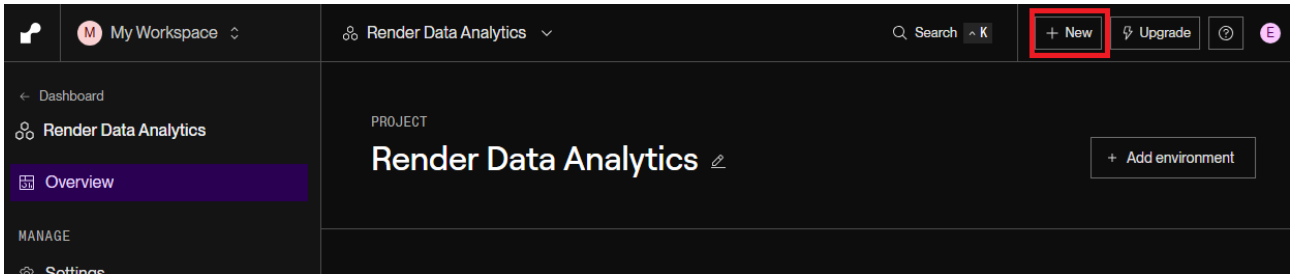

Imagen 11. Nuevo proyecto de Render

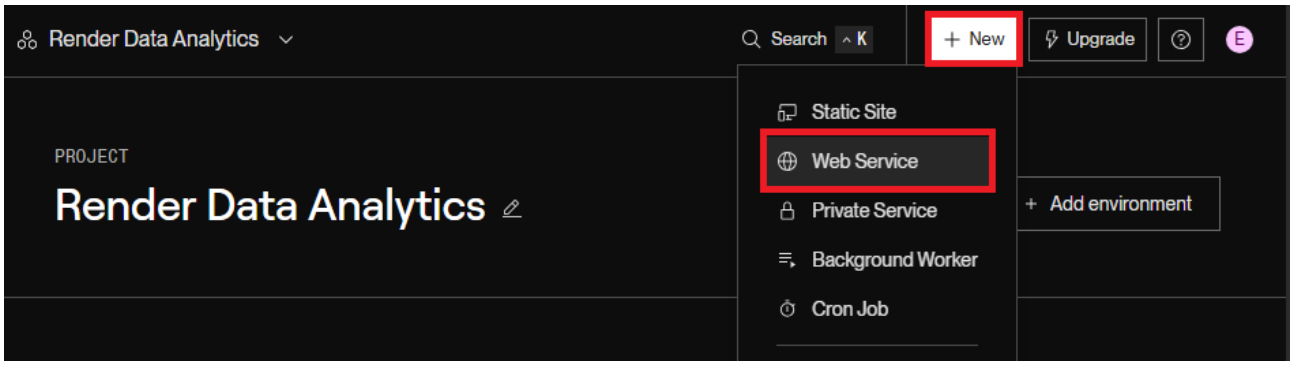

Imagen 12. Crear un nuevo proyecto web en Render

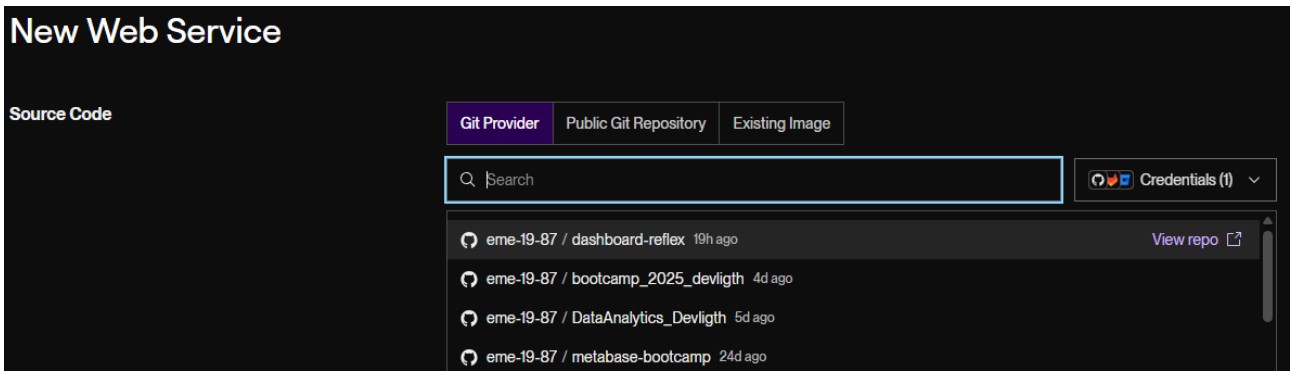

Imagen 13. Elegir la fuente de nuestro repositorio mediante el proveedor de git

Si alguna vez iniciaron con su cuenta de Github, les aparecerán sus repositorios como se muestra en la imagen 13. Si no les aparece, pueden elegir la opción ***Public Git Repository***. Allí les pedirá que coloquen su dirección del repositorio git que quieren desplegar. Eso pueden encontrarlo en su proyecto de Github

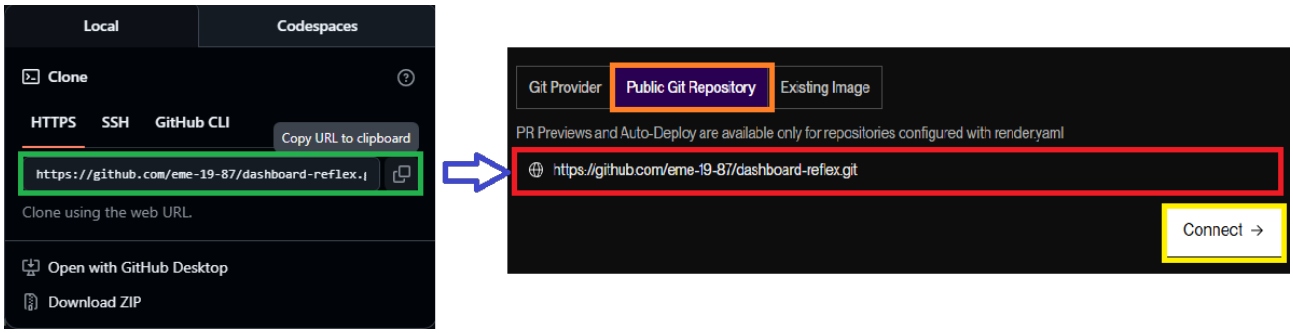

Imagen 14. Copiar la dirección del proyecto Github en Render

Cuando le den al botón ***Connect*** de la imagen 14, les aparecerá la siguiente ventana con las opciones

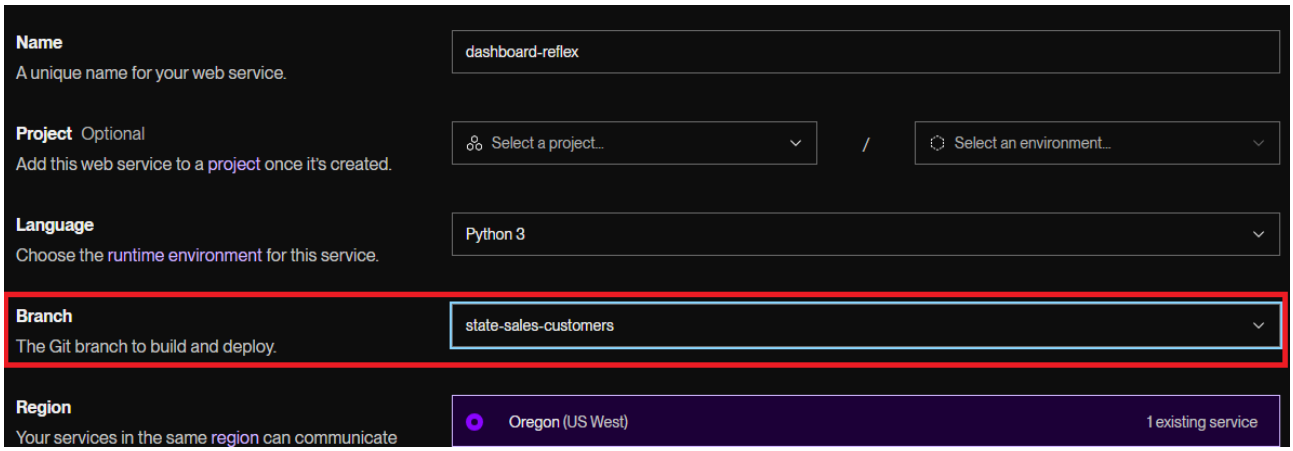

Imagen 15. Primeras opciones del despliegue.

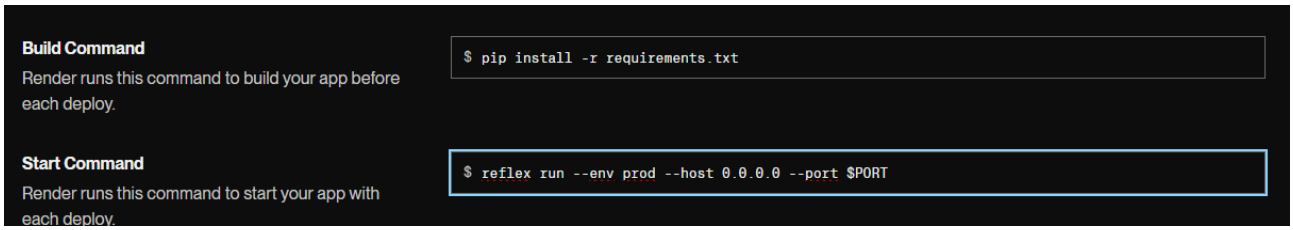

Imagen 16. Comandos de construcción y ejecución iniciales.

Fe De Erratas: El comando correcto que debe colocarse en ***Start Command*** es:

reflex run --env prod --backend-host 0.0.0.0 --backend-port $PORT

Primero, conviene elegir la rama desde la cual se va a desplegar el proyecto. Normalmente debería ser main, porque esa es la rama con el producto que se le va a presentar al cliente. Yo elijo otra rama porque estoy practicando.

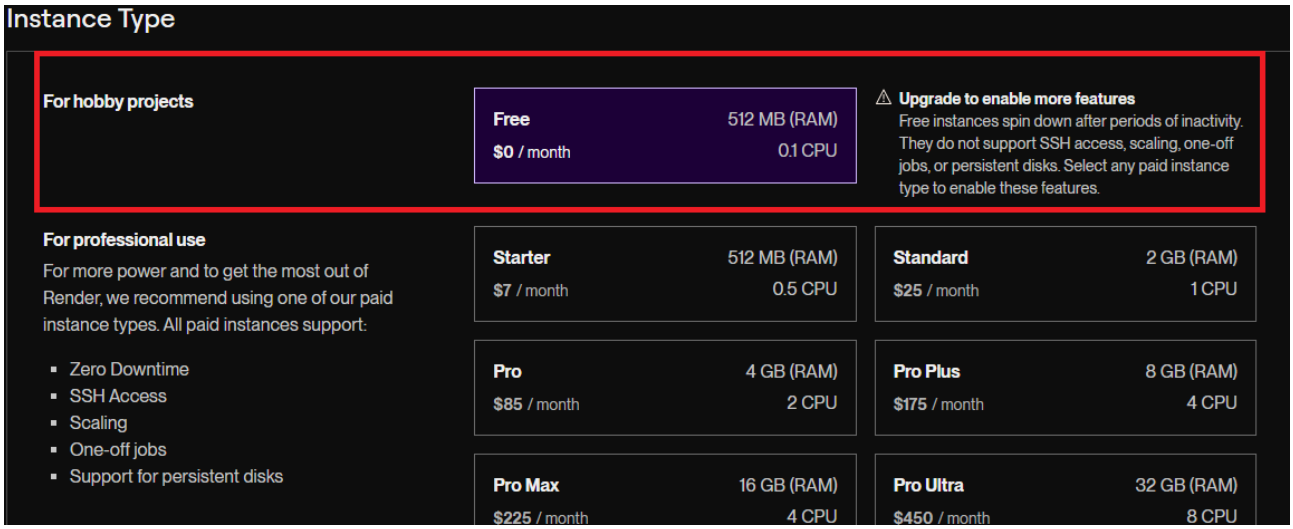

Imagen 17. Selección del plan gratuito.

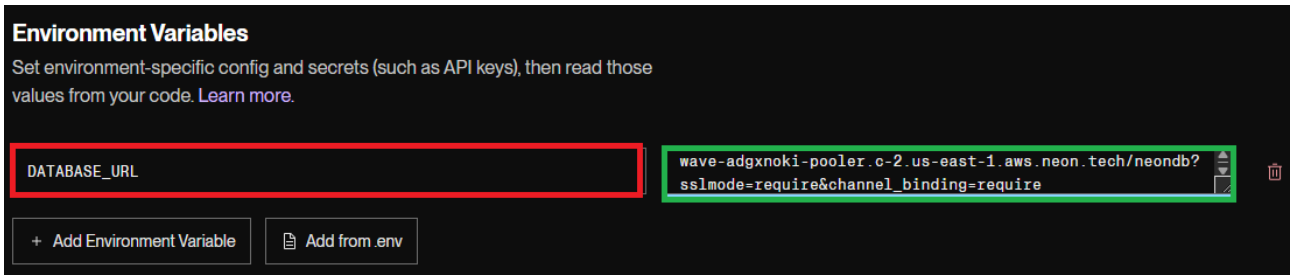

Imagen 18. Creación de la variable para conectarse a Neon.

La parte indicada en verde es la conexión que debemos obtener de Neon. Al final de todas la opciones habrá una que dice ***Deploy Web Service***. Le damos click y empezará el proceso de despliegue.

Nota Muy Importante

Al momento de realizar el build en Render, todo funciona correctamente. Pero, debido a las limitaciones de mi plan gratuito, no me permite terminar de desplegar la aplicación y me lanza un error.
Todos los pasos están correctos y el único problema son los recursos del servicio gratuito. Para quien quiera probarlo teniendo los recursos disponibles y ver si despliega correctamente, allí están las instrucciones. Obviamente, pueden ser mejoradas.
Por lo pronto, dejo una mejor alternativa para trabajar con la página oficial de Reflex.

Nota Adicional:

Recordar que esto es un entorno de prueba, no conviene que se suba el archivo con la información de la conexión. Lo recomendable es crear un archivo .env y no subir ese archivo a github.

Probando El Despliegue En Reflex

Debemos tener una cuenta creada en https://reflex.dev/
En la página principal, si nos dirigimos hacia el final de todo. Veremos una opción que dice ***Hosting***

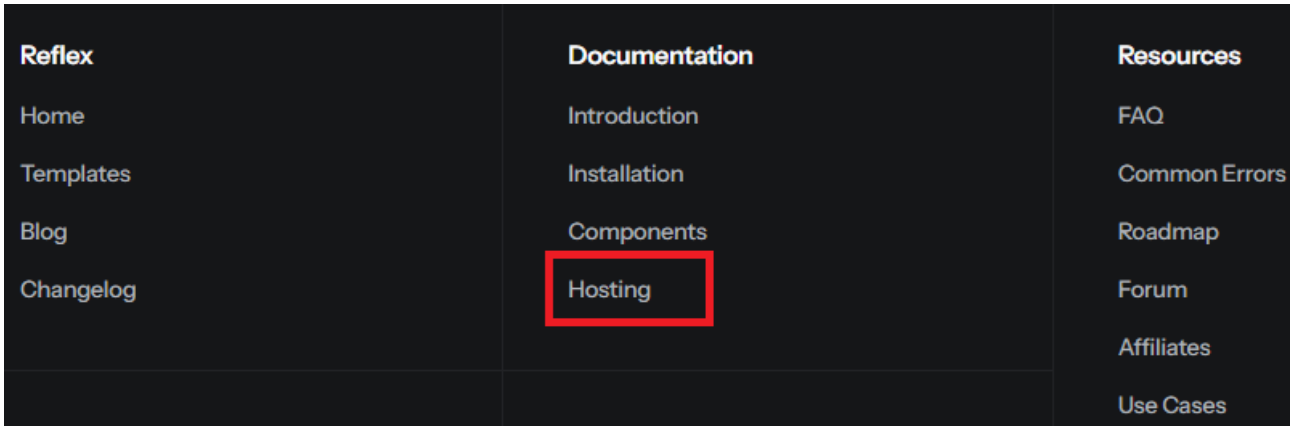

Imagen R1. Opción de Hosting

Cuando hagamos click en esa opción, se abrirá la documentación que indica cómo realizar el despliegue. Las instrucciones que estamos buscando son las siguientes:

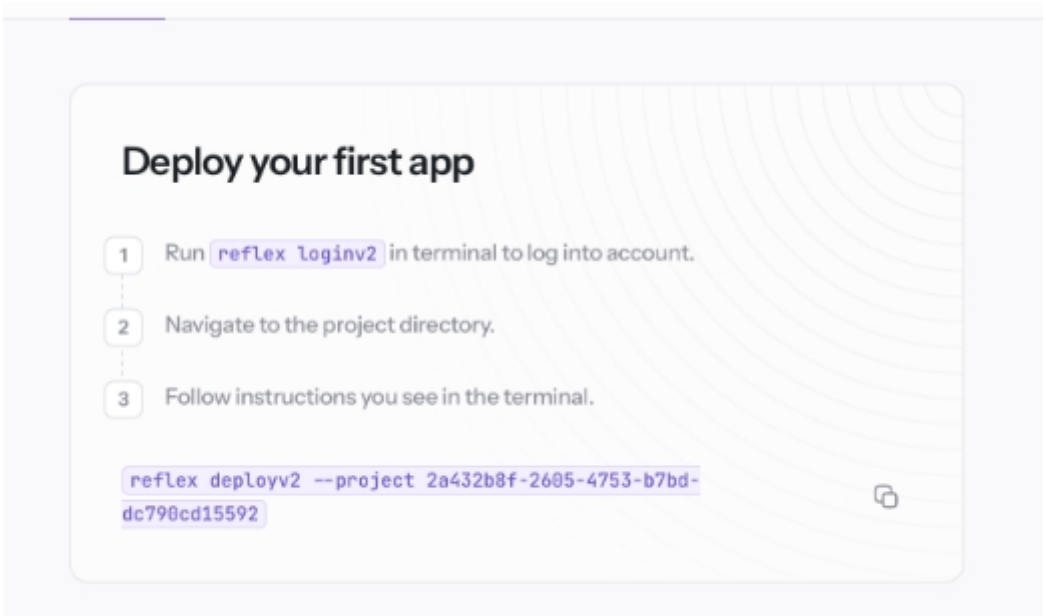

Imagen R1.1. Pasos para el logueo

Allí dice ***reflex loginv2***. Pero yo me loguee con ***reflex login***. Eso debemos hacerlo en la terminal cuando estamos en la carpeta de nuestro proyecto.

Para mi caso, estos serían los pasos para el despliegue

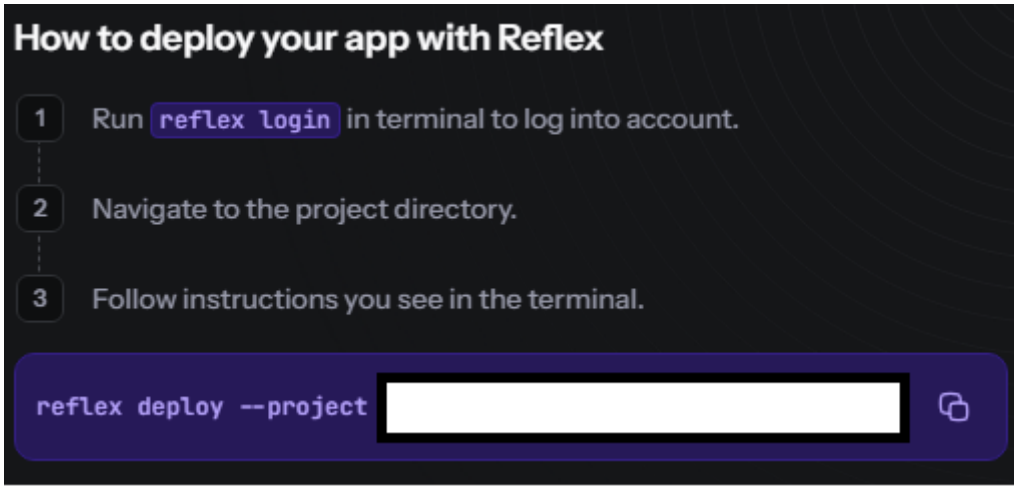

Imagen R2. Pasos para levantar la aplicación.

Cuando se esté en el directorio del proyecto, sólo debemos copiar la instrucción indicada en azul en la imagen R2 (lo blanco es sólo para ocultar el código que muestra, nada más).
Eso lo pegamos en nuestra terminal (recuerda, es necesario estar en el directorio del proyecto, el mismo directorio donde normalmente se ejecuta ***reflex run***).
Inicialmente, nos aparecerá esto:

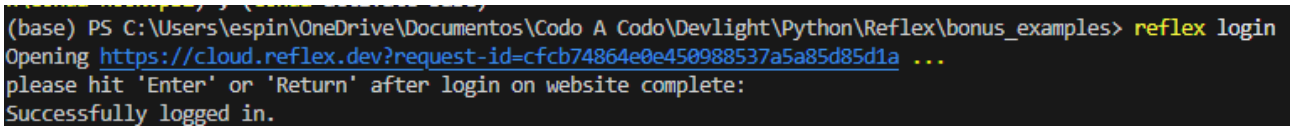

Imagen R3. Pasos para levantar la aplicación.

Esto nos abrirá una ventana donde nos pedirá autentificación en la página de Reflex. Una vez que la autentificación esté realizada, usamos el comando marcado como 3 en la imagen R2

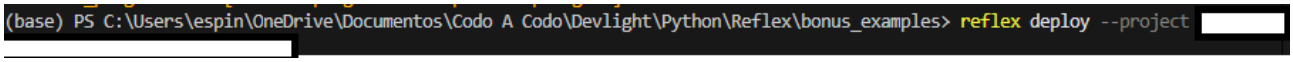

Imagen R4. Segundo Paso para levantar la aplicación.

Nos solicitará confirmación para seguir. Escribimos ***Yes,*** le damos ***Enter***.
Si es la primera vez que desplegamos nuestra aplicación, veremos lo siguiente:

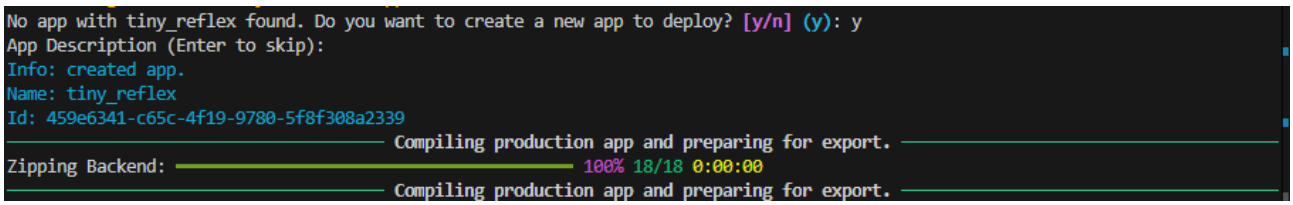

Imagen R5. Tercer Paso para levantar la aplicación.

Debemos colocar ***y luego Enter*** y empezará el proceso de construcción del proyecto. A la vez, comenzará a subir el proyecto a la aplicación creada en la página de Reflex.

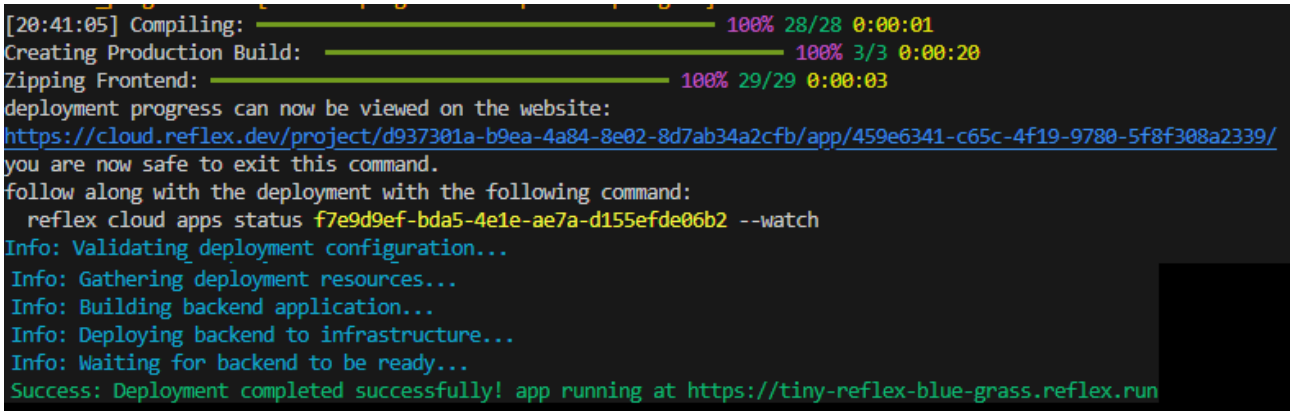

Imagen R6. Logs de la creación del proyecto.

Si nos vamos a la página de ***Reflex*** con la sesión iniciada, veremos algo como esto

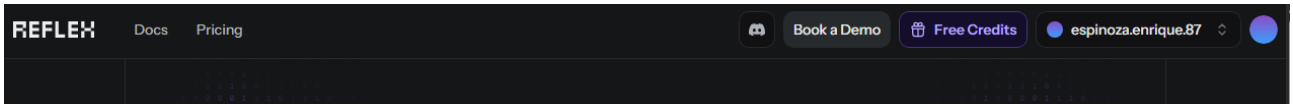

Imagen R6. Logs de la creación del proyecto.

Cuando estemos en una ventana así, bajamos hasta el final para encontrar nuevamente la opción ***Hosting***

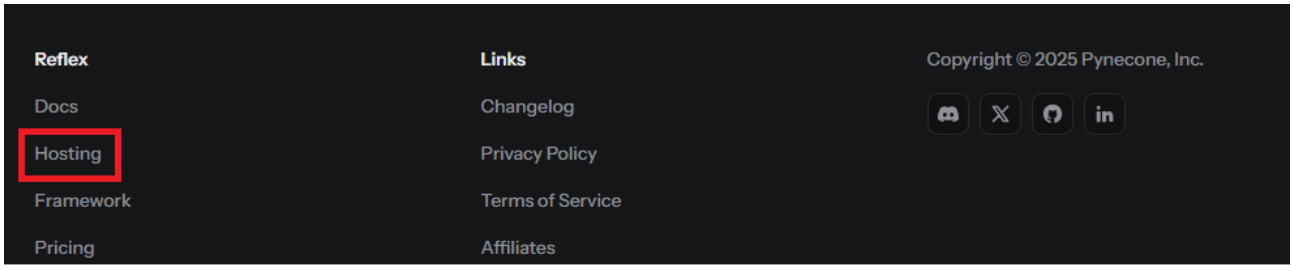

Imagen R7. Opción de Hosting cuando tenemos iniciada la sesión.

Esa opción no nos llevará a la documentación, sino a la próxima ventana:

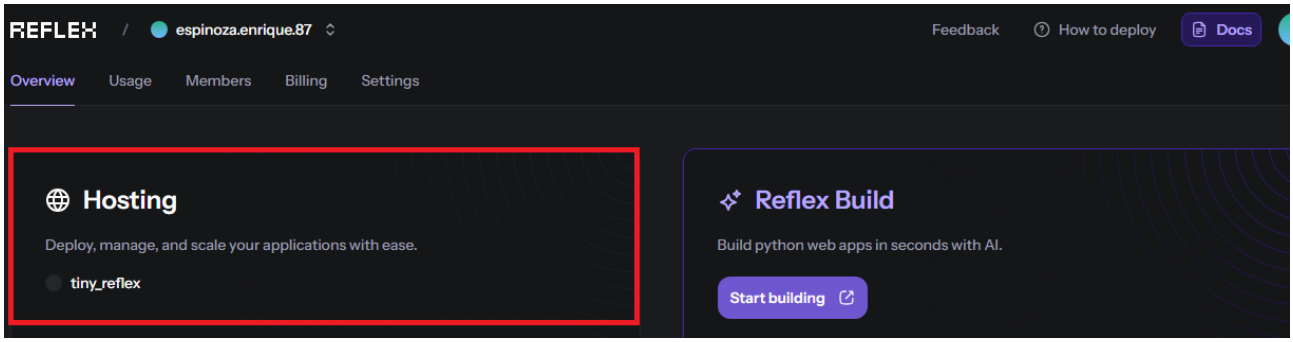

Imagen R8. Ventana con el despliegue creado.

Notamos en la parte señalada en rojo para la imagen R8 que aparece el despliegue que creamos con la consola de comandos. Si hacemos click sobre esa opción, veremos sus detalles

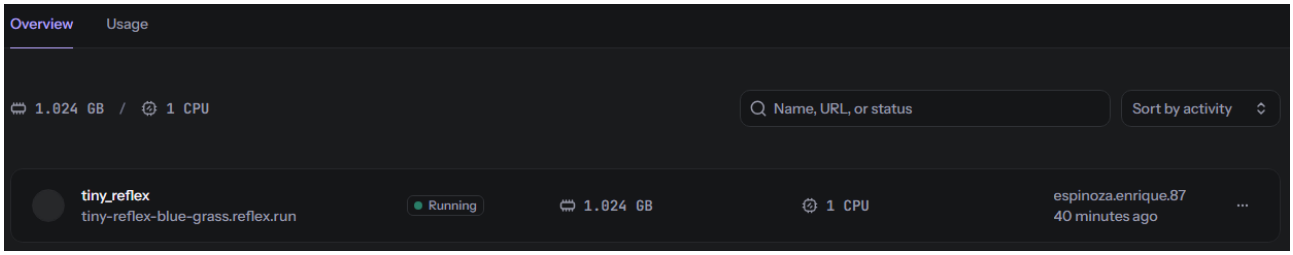

Imagen R9. Detalles del despliegue creado.

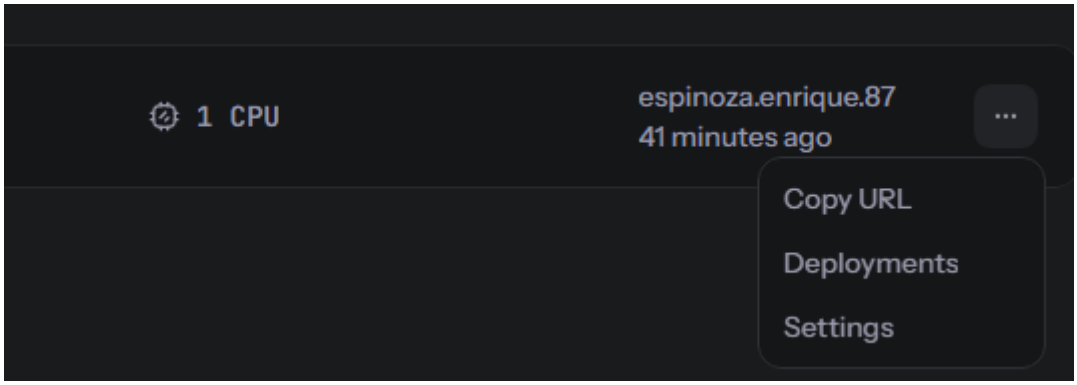

Imagen R10. Opciones del despliegue.

Si elegimos la opción mostrada en la imagen R10, notaremos que se despliegan más opciones. Elijamos ***Deployments***

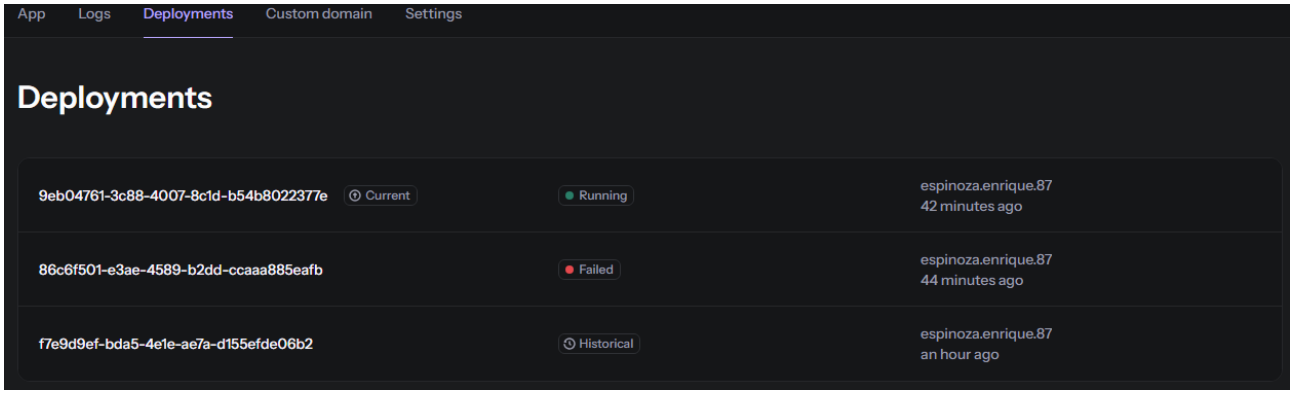

Imagen R11. Lista de deployments.

Si elegimos la opción ***App*** de la imagen R11, veremos los detalles de nuestra aplicación:

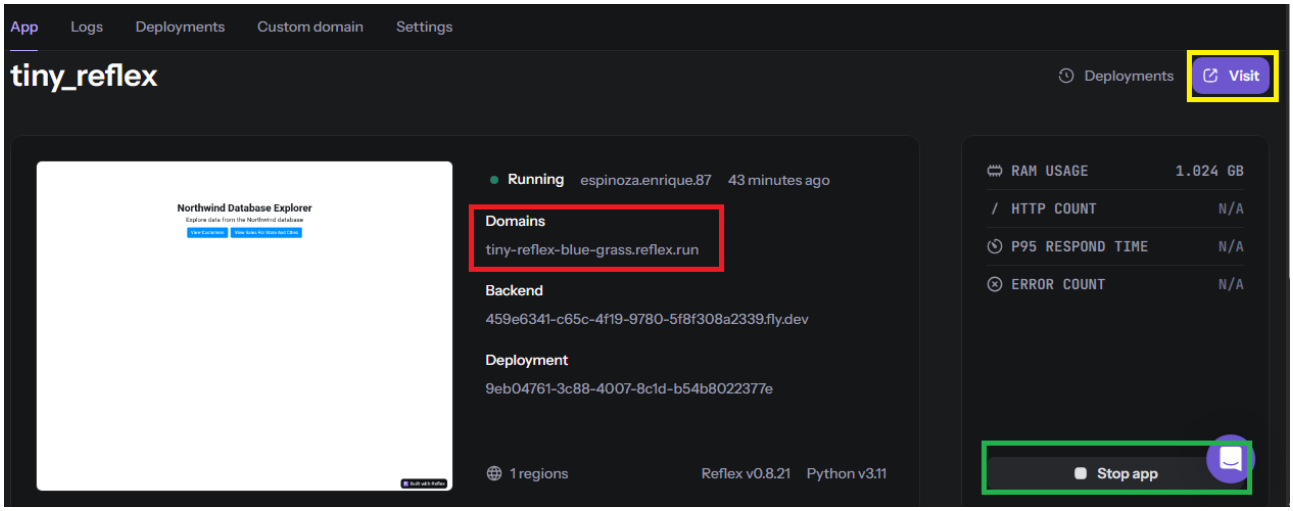

Imagen R12. Detalles de nuestra aplicación.

La parte indicada en rojo en la imagen R12 es la url de nuestra aplicación. Podemos visitar también nuestra aplicación con la opción indicada en amarillo.
La opción en verde puede ser útil para detener nuestra aplicación para que no nos consuma recursos.

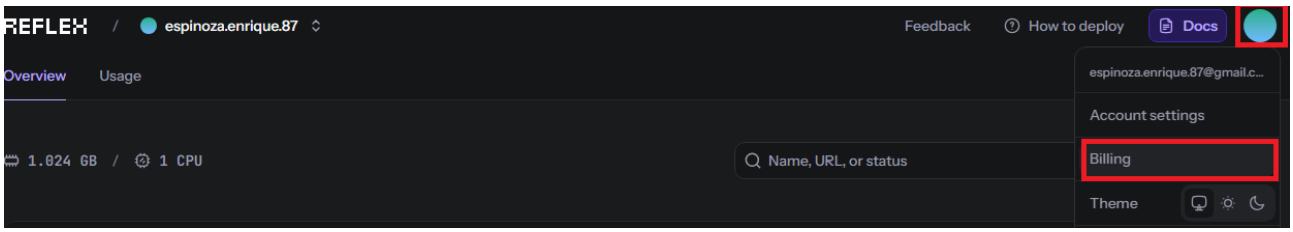

Imagen R13. Opciones para ver la facturación de nuestra cuenta.

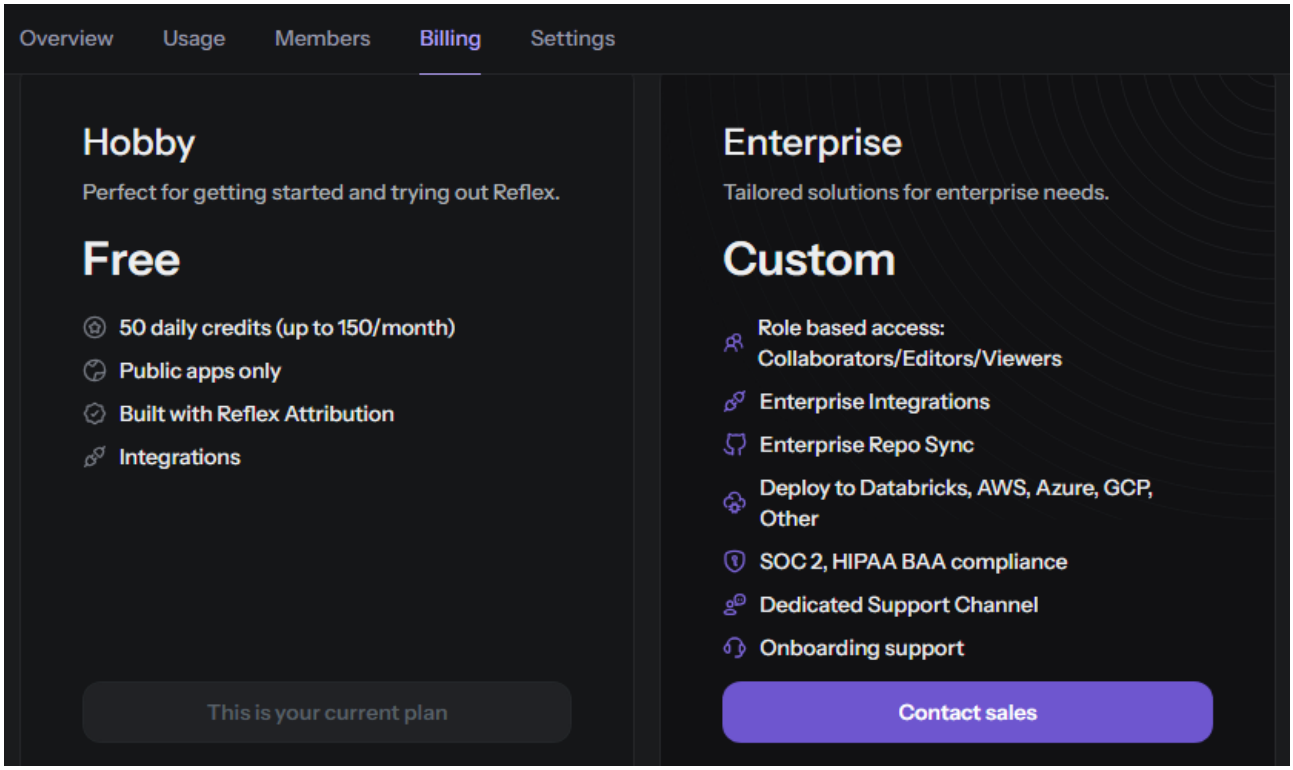

Imagen R14. Opciones de prestaciones y pagos.

De las modificaciones que se realizaron a los documentos para subirlo a ***Render***, debemos realizar una modificación en el documento ***db_connection.py***

```python
def get_engine():
    """
    Create and return a SQLAlchemy engine for Neon PostgreSQL database.
    Uses environment variables or defaults for connection.
    """

    user = os.getenv("POSTGRES_USER", "neondb_owner")
    password = os.getenv("POSTGRES_PASSWORD", [redacted])
    host = os.getenv("POSTGRES_HOST", "ep-polished-wave-adgxnok1-pooler.c-2.us-east-1.aws.neon.tech")
    port = os.getenv("POSTGRES_PORT", "5432")
    database = os.getenv("POSTGRES_DB", "neondb")

    # IMPORTANT: Neon requires SSL mode
    connection_string = (
        f"postgresql://{user}:{password}@{host}:{port}/{database}"
        f"?sslmode=require"
    )

    engine = create_engine(connection_string)
    return engine
```

Imagen R15. Modificación del archivo de conexión.

Con esto logré la conexión a Neon y la información que se muestra allí debe ser obtenida del string de conexión de Neon.
Obviamente esto debe ser mejorado porque estamos exponiendo nuestras claves.

Un problema que tuve es el siguiente. Quise realizar una modificación y aplicar el cambio en la página de Reflex. Cuando quise realizar nuevamente el comando 3 de la imagen R2, me dio el siguiente error:

```
  reflex cloud apps status 86c6f501-e3ae-4589-b2dd-ccaaa885eafb --watch
Info: Validating deployment configuration...
Info: Gathering deployment resources...
Info: Deployment failed: [Errno 28] No space left on device: '/tmp/tmpk1x210ah'
Warning: Deployment failed: Internal server error occurred. Please try again or contact support if the issue persists.
```

Imagen R16. Error en el despliegue.

Lo que me indica es que no hay espacio suficiente en el dispositivo. Lo arreglé eliminando la aplicación de la página de Reflex y volviendo a realizar los pasos para desplegar la aplicación.

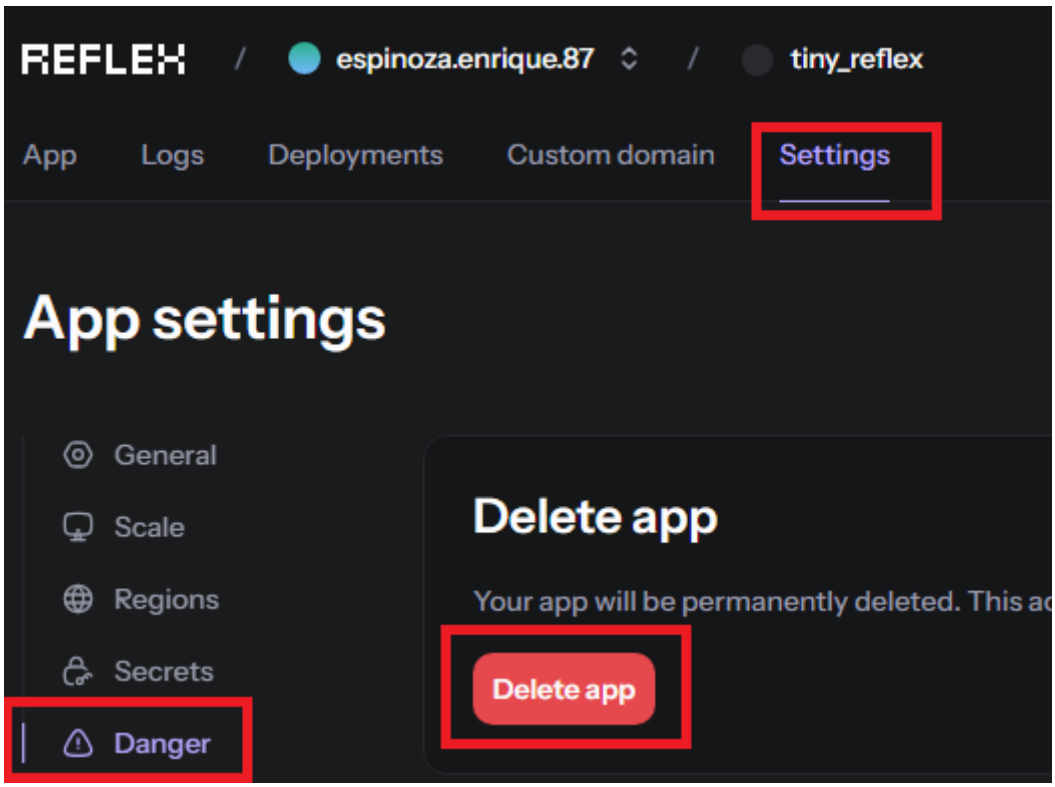

Imagen R17. Eliminar la aplicación.