Documento de Despliegue

Proyecto Rector simulator

Versión 1.0

| Versión | Autores | Descripción | Fecha |
| --- | --- | --- | --- |
| 1.0 | Diego Gamarra  Karlo Toro | Primera versión | 14/11/2024 |

ÍNDICE

[**1. Introducción 4**](#_iue8k08wf9a6)

[**2. Requisitos previos 4**](#_kn4f9jdwa6f)

[2.1. Software Necesario 4](#_v0kdgtyqpce5)

[2.2. Credenciales 5](#_33d1qeeswcm8)

[**3. Despliegue del Frontend 5**](#_d3jtu3l01838)

[3.1. Configuración inicial 5](#_7m0jlig2e10v)

[3.2. Script necesario 5](#_5k9iuu3mju74)

[3.3. Despliegue en Railway 6](#_g2nlxvd21szd)

[**4. Despliegue del Backend 8**](#_eilby3vu08p9)

[4.1. Configuración inicial 8](#_pr652tya9y0q)

[4.2. Script necesario 9](#_pasyxxdetlhg)

[4.3. Despliegue en Railway 10](#_ikquh3yq6jbn)

[**5. Despliegue de Base de datos 10**](#_fljo5vjvk86v)

# Introducción

El presente documento especifica las configuraciones y requisitos necesarios para el despliegue de una aplicación frontend React y backend Django Rest Framework utilizando Railway como plataforma de despliegue.

# Requisitos previos

## Software Necesario

* + 1. React: Framework frontend que genera la interfaz de la aplicación.
    2. Tailwind: Framework css que proporciona clases de estilo aplicables que utiliza React para construir la interfaz.
    3. Python: Lenguaje de programación, utilizado en el backend junto con django rest framework para la creación de la API que interactúa con la base de datos y maneja la lógica de la aplicación.
    4. Node: Entorno de ejecución de JavaScript, utilizado para la gestión de dependencias en React y para la compilación de código React.
    5. Git: Es un sistema de control de versiones que administra el historial de versiones del desarrollo y permite la colaboración y manejo de cambios.
    6. Github: Plataforma de alojamiento de código y colaboración, basado en git, el cual permite integraciones con varias plataformas de despliegue.
    7. Railway: Es una plataforma de despliegue que actúa como entorno de producción de una aplicación.

## Credenciales

* + 1. API Key de OpenAI: Clave de acceso proporcionada por OpenAI para utilizar su API e interactuar con diversos modelos de inteligencia artificial. Es necesario configurar el mismo como una variable de entorno en Railway para que la aplicación funcione en producción.

# Despliegue del Frontend

## Configuración inicial

El único paquete imprescindible para el despliegue del frontend es serve. Este paquete crea un servidor simple que entrega los archivos estáticos (HTML, CSS, y JavaScript) generados por el proyecto React para que los navegadores puedan acceder a ellos.



## Script necesario

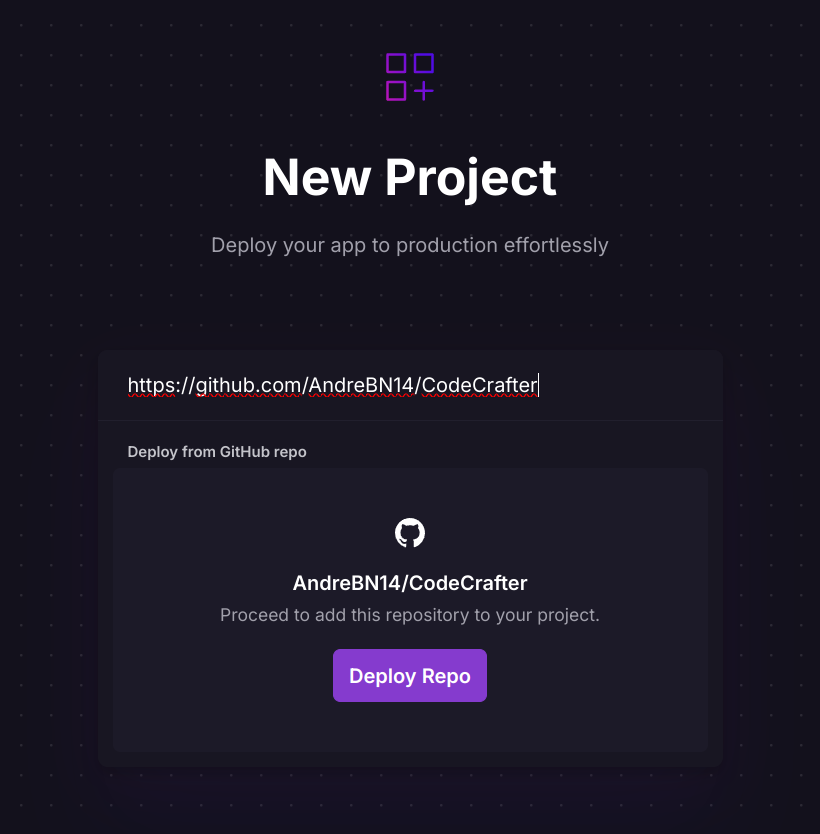
Para que Railway pueda usar el paquete serve es necesario definir el siguiente script



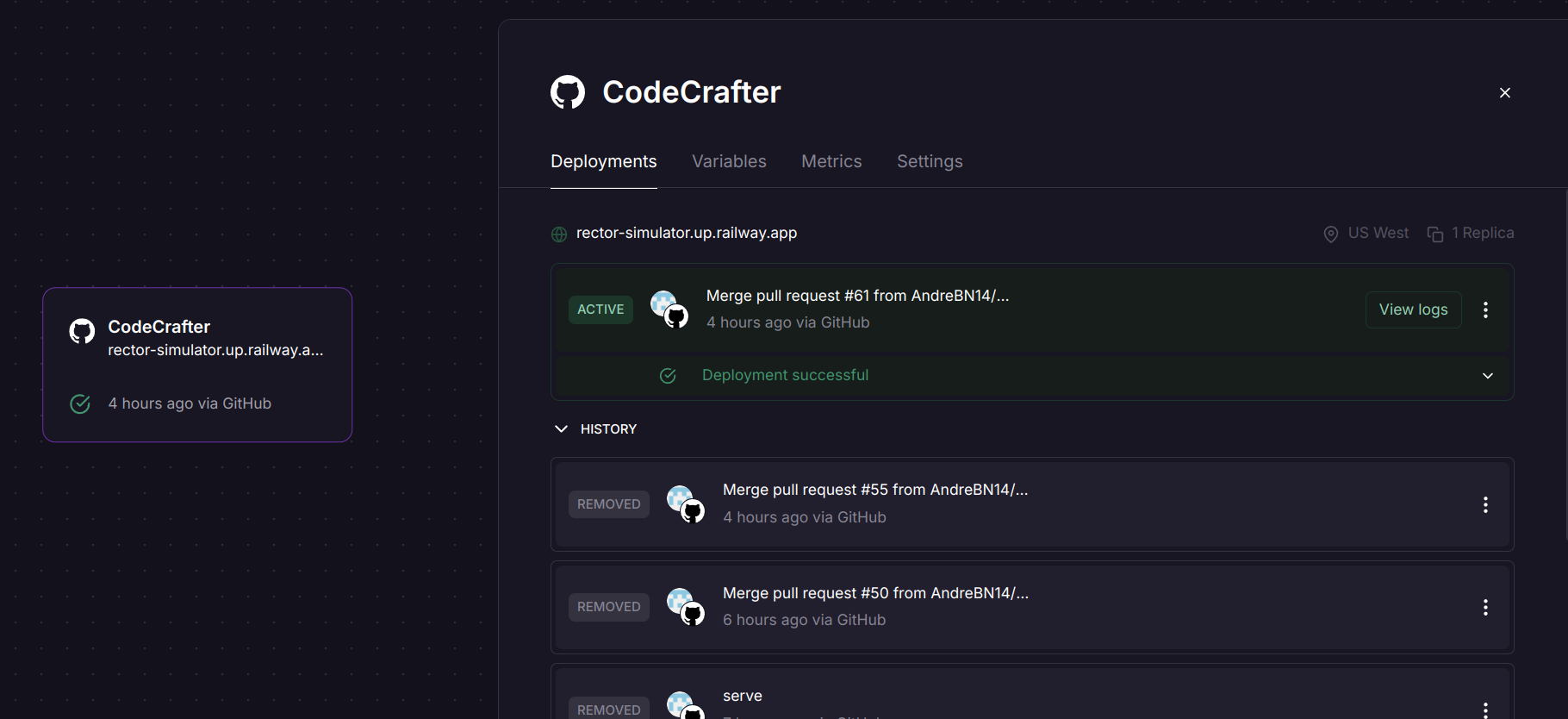
Este script especifica donde se aplicara el serve que seria en la carpeta dist también se ponen unos cuantos flags donde se especifica que el proyecto es un single-page y que se debe desactivar el comportamiento por defecto de serve (-s -n). Asimismo, se especifica que el puerto donde se ejecutará la aplicación será elegido por Railway (-p $PORT).

## Despliegue en Railway

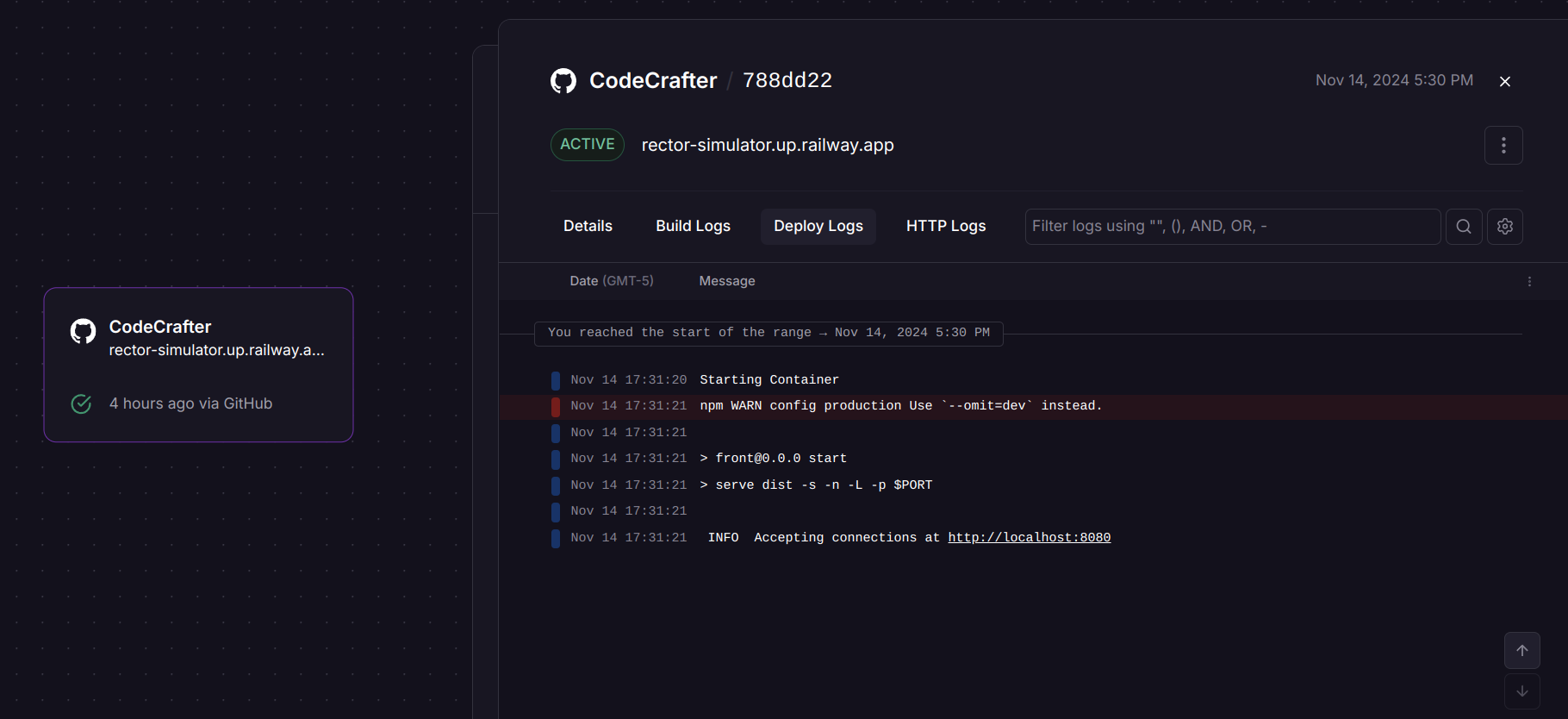
Lo primero que debemos hacer es elegir el proyecto que vamos a desplegar. Esto lo hacemos eligiendo un repositorio de github y poniendo el link donde está alojado.



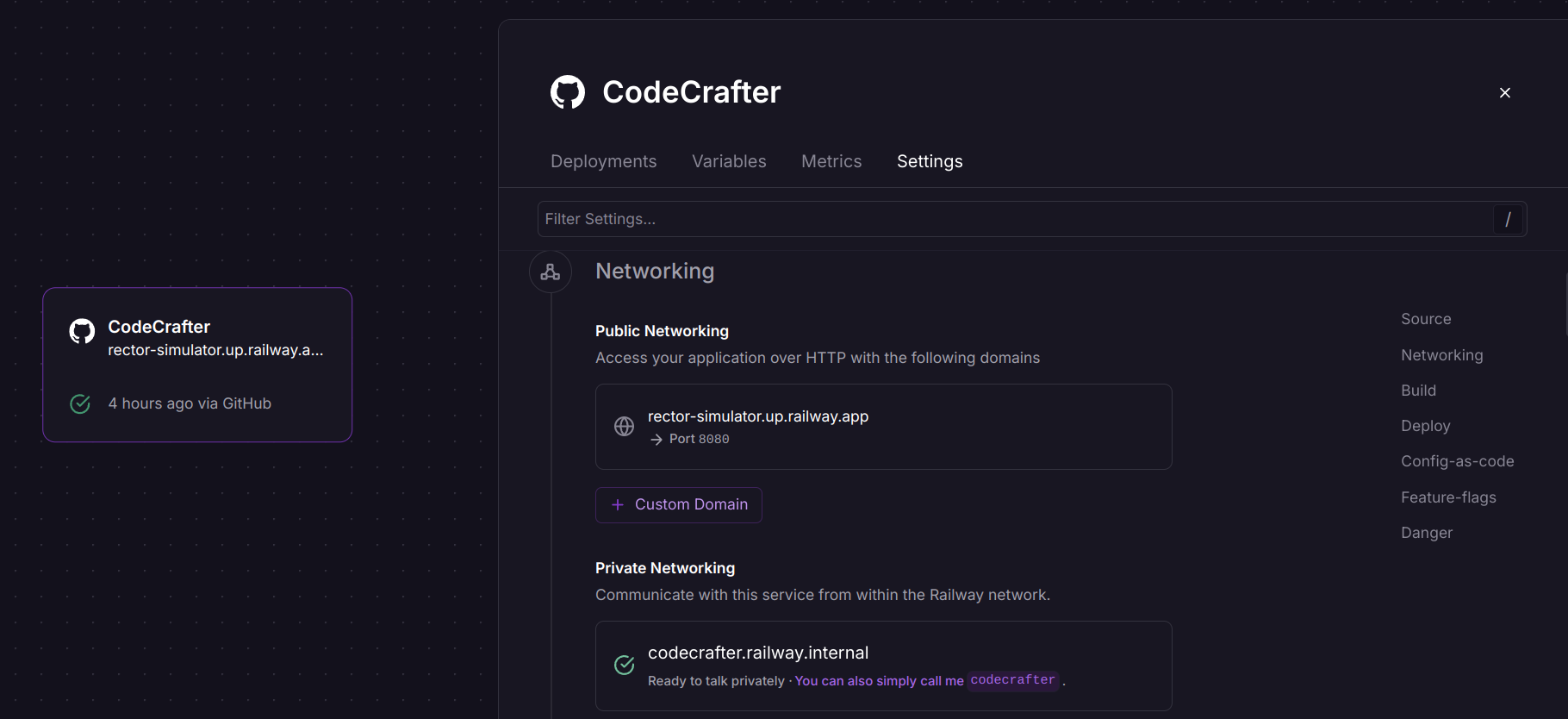
Una vez hecho esto nos aparecerá la siguiente ventana donde aparecerán los últimos commits una vez desplegado el proyecto.



Debemos darle click al que sale primero y ahí veremos una terminal que indica el procedimiento del despliegue. En ello nos daremos cuenta si ha sido exitoso o no.



Si se ha desplegado correctamente el siguiente paso es darle un dominio, Railway te da la posibilidad de generar un dominio automáticamente o personalizarlo, y con ello ya estaría lista la aplicación para ser usada.



# Despliegue del Backend

## Configuración inicial

Los paquetes imprescindibles para el despliegue son:

* Gunicorn: Es un servidor HTTP para aplicaciones Python, y actúa como un servidor robusto que puede manejar múltiples solicitudes simultáneamente.
* Whitenoise: Es parecido a serve ya que se encarga de ser un servidor simple que entrega los archivos estáticos de django.

Además de instalar esos paquetes es fundamental tener los siguientes archivos:

* requirements.txt: Especifica los paquetes instalados en el entorno virtual.
* Procfile: Especifica que scripts debe hacer Railway.
* runtime.txt: Especifica la versión de python usada.

Para el correcto funcionamiento de whitenoise es necesario incluir las siguientes lineas de codigo en settings.py:



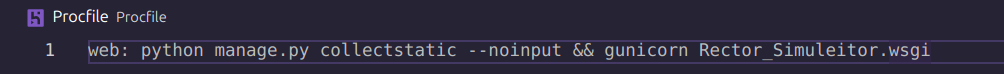
Para la generación de los archivos estáticos es necesario la siguiente línea de código:



Lo que hace el código es especificar en qué documento se encontrarán dichos archivos.

## Script necesario

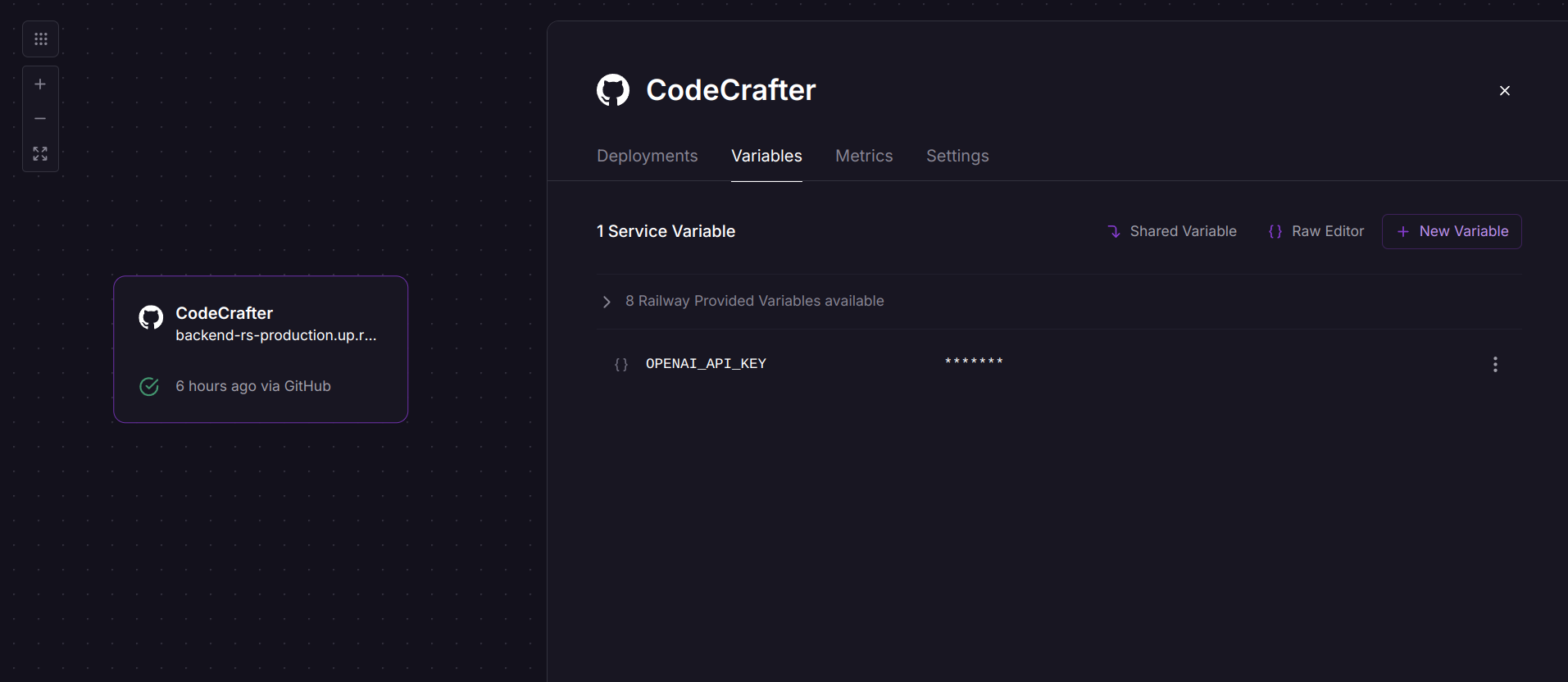
El script que debe hacer Railway para el correcto funcionamiento de la aplicación se debe encontrar en Procfile.



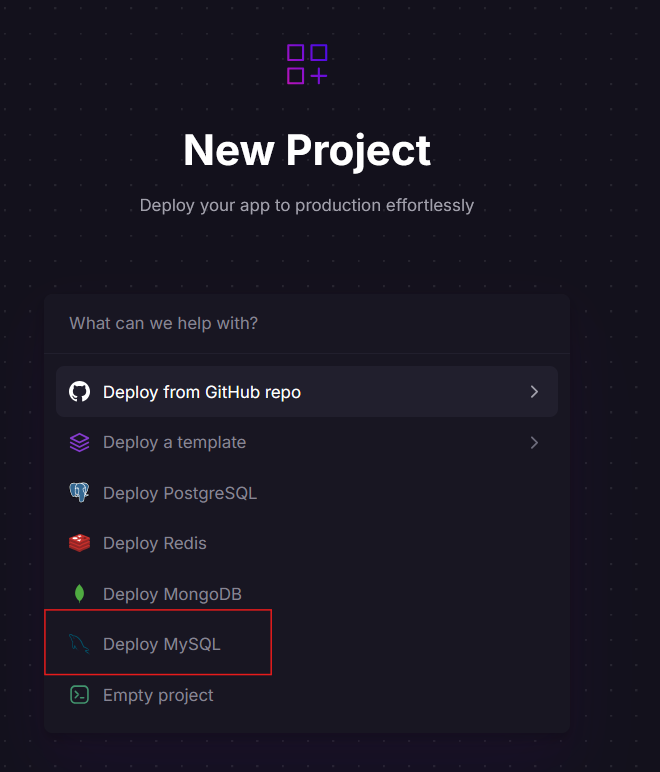
En este caso, lo que hace el script es generar los archivos estáticos del proyecto y generar un servidor que responda las solicitudes http.

## Despliegue en Railway

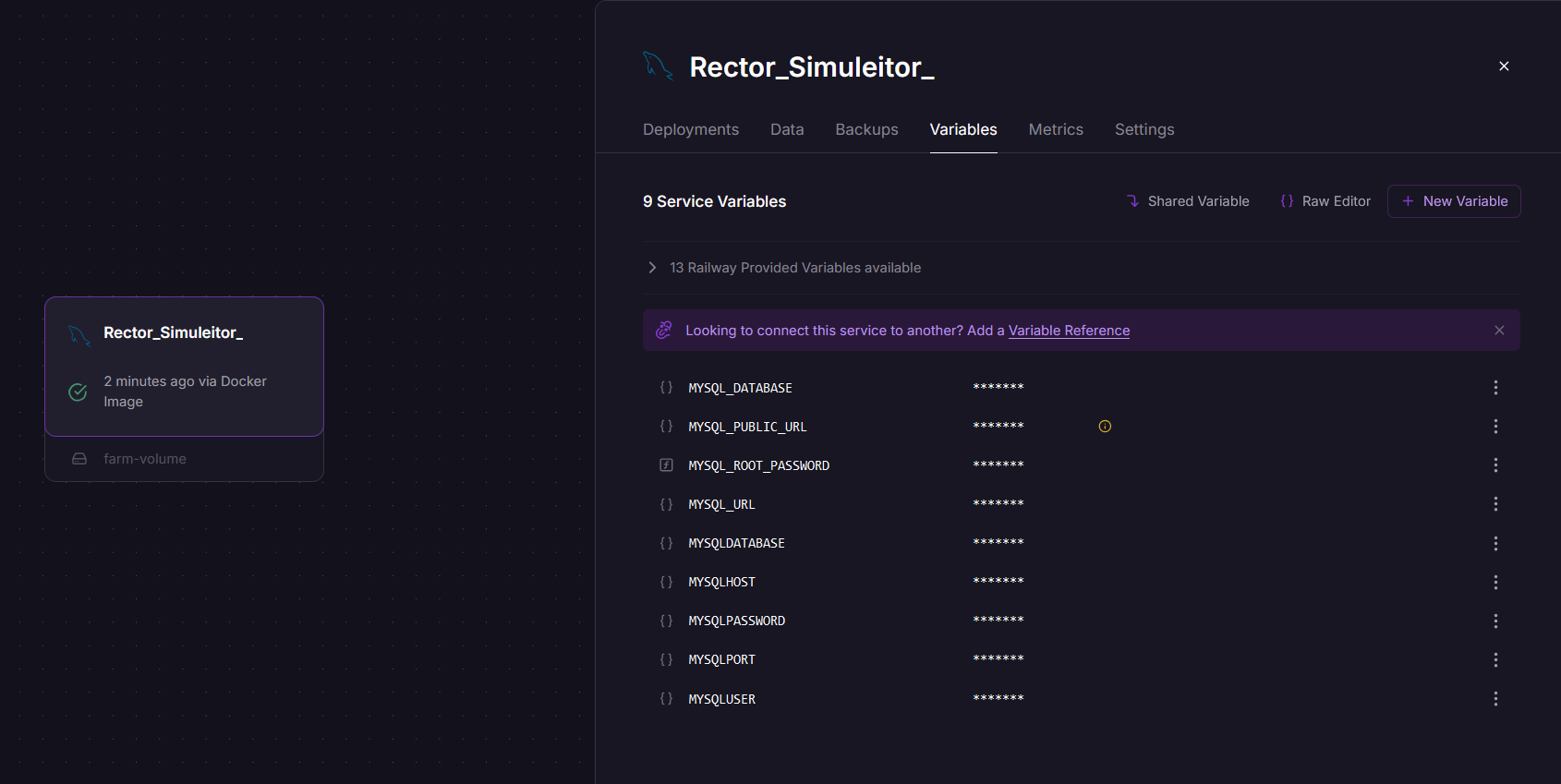
En el caso del backend, el procedimiento es igual al del frontend pero con el paso extra agregar la API Key de OpenAI como variable de entorno.



# Despliegue de Base de datos

5.1 Creamos un nuevo proyecto en RailWay, del tipo deploy MySQL

5.2 Automáticamente se genera un contenedor para nuestro servidor de MySQL donde luego de esperar unos cuantos minutos obtenemos acceso a toda la informacion necesaria para conectarnos atrevés de un cliente como MySQL Workbench



5.3 Para poder conectarnos con nuestra aplicacion Django debemos configurar en el archivo settings.py el diccionario de credenciales de esta manera, donde después de configurar nuestros modelos de datos ejecutamos make migrations y posteriormente migrate.







5.4 Finalmente, para tener un mayor control y un manejo más natural sobre operaciones complejas con la BDD nos conectamos al cliente de Workbench

