DESPLIEGUE CON COOLIFY

Configuración de Coolify para el despliegue de proyectos

Eder Martínez Castro

Desarrollo de Aplicaciones Multiplataforma

23 de enero de 2026

Tabla de contenidos

1 OBJETIVO

El objetivo de está práctica es desplegar **Coolify** en una máquina virtual **Ubuntu**, configurándola como un entorno `self-hosted`. Además, utilizaremos **Ngrok** para poder exponer nuestro servicio de forma externa y poder permitir el acceso a la interfaz web de Coolify, sin necesidad de tener un IP pública fija.

RECORDATORIO:

Antes de empezar la prática debemos instalar un **máquina virtual Linux** (recomendación Ubuntu),para ello hay que tener la `iso` de nuestra distro elegida y usaremos **Oracle VirtualBox** para crear la máquina virtual.

Configuraciones de nuestra máquina virtual:

- **Sistema operativo**: Ubuntu (64-bit).
- **Memoria base (RAM)**: 8 GB.
- **Procesadores**: 4 CPUs.
- **Almacenamiento (Disco Duro)**: 60 GB.
- **Red**: Adaptador puente.

Página para descargar **Oracle VirtualBox**: https://www.oracle.com/es/virtualization/technologies/vm/downloads/virtualbox-downloads.html

Página para descargar el `.iso` de **Ubuntu**: https://ubuntu.com/download/desktop

Cuando tengamos nuestra máquina virtual lista con Ubuntu ya instalado, lo primero que vamos a hacer es abrir la terminal y **instalar el paquete** `curl` para instalar tanto Coolify y Ngrok.

```
sudo apt install curl
```

2 COOLIFY

Coolify es una **plataforma de despliegue y gestión de aplicaciones**, que nos permite alojar y administrar nuestros proyectos en servidores propios, esto nos ofrece mayor control sobre la infraestructura.

2.1 Instalación de Coolify

En este apartado vamos a empezar con la instalación de coolify en nuestra máquina virtual Ubuntu.

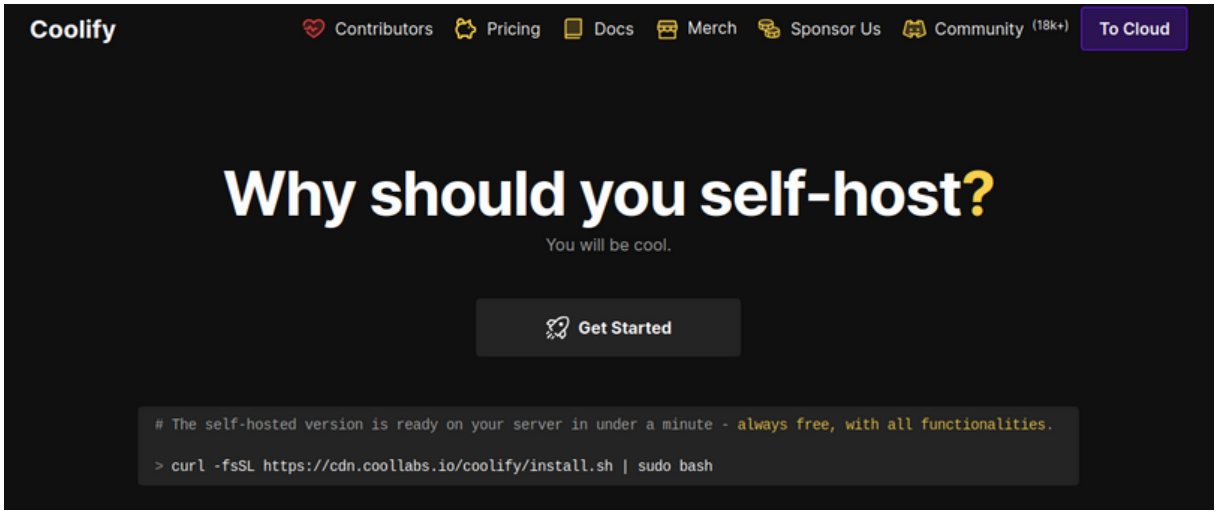

Figura 1. Web oficial de coolify para obtener el comando para instalarlo en `self-host`.

Para poder obtener el **comando de instalación de Coolify**, nos iremos a la página oficial de Coolify en la sección de despliegue `self-hosted`. En ella encontraremos el comando oficial que nos proporciona la plataforma para realizar la **instalación automática** de Coolify **en nuestro servidor propio** (máquina virtual).

Este comando descarga y ejecuta un script que instala todas las dependencias necesarias como **Docker** y **Docker Compose**, además de **configurar el entorno inicial** de Coolify:

```
curl -fsSL https://cdn.coollabs.io/coolify/install.sh | sudo bash
```

Página oficial de Coolify: https://coolify.io/self-hosted/

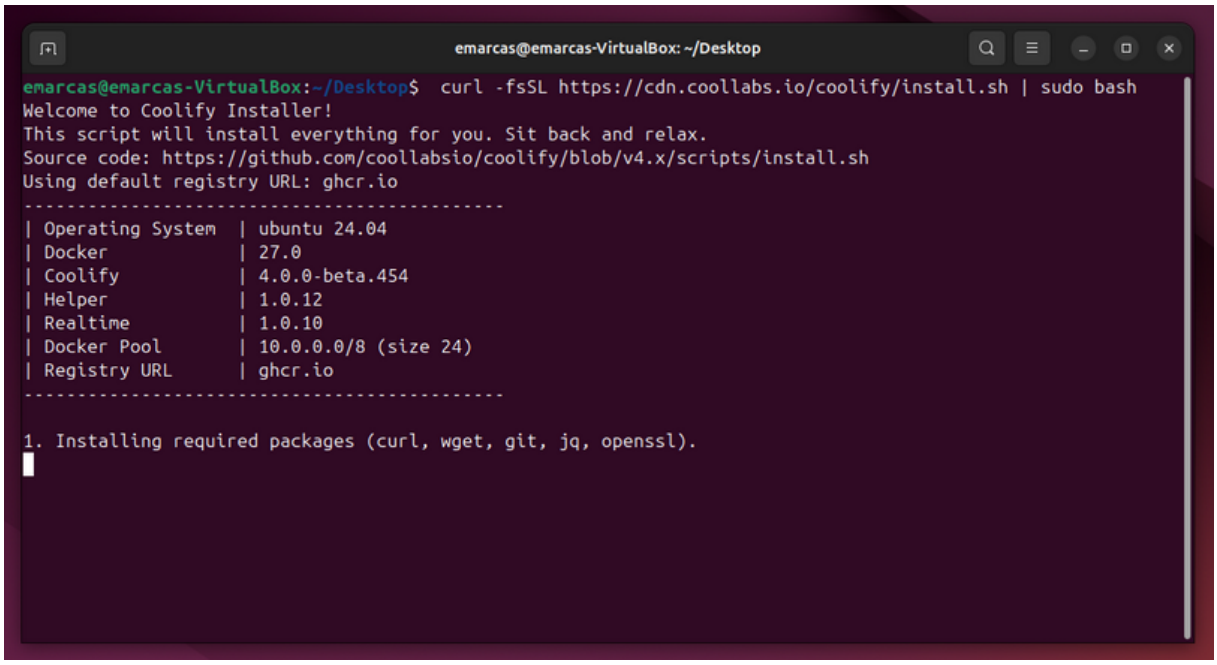

Figura 2. Ejecutamos el comando para instalar el self-host de coolify.

Ahora ejecutaremos el comando previamente copiado, en nuestra **terminal** de la máquina virtual.

Figura 3. Captura de la terminal cuando ya termino de instalar el self-host de coolify en nuestro equipo.

Como podemos ver ya termino la **instalación de Coolify** correctamente, con esto ya tendríamos la plataforma lista para acceder desde el navegador web de nuestro equipo mediante la `IP` de nuestra máquina y el puerto `8000` como nos indica en la captura.

2.2 Obtener la IP pública

Para poder conectarnos desde nuestro equipo a la **máquina virtual ubuntu** donde tenemos instalado **coolify**, debemos saber nuestra `dirección IP pública`, para así conectarnos y ver el incio de sesión de coolify.

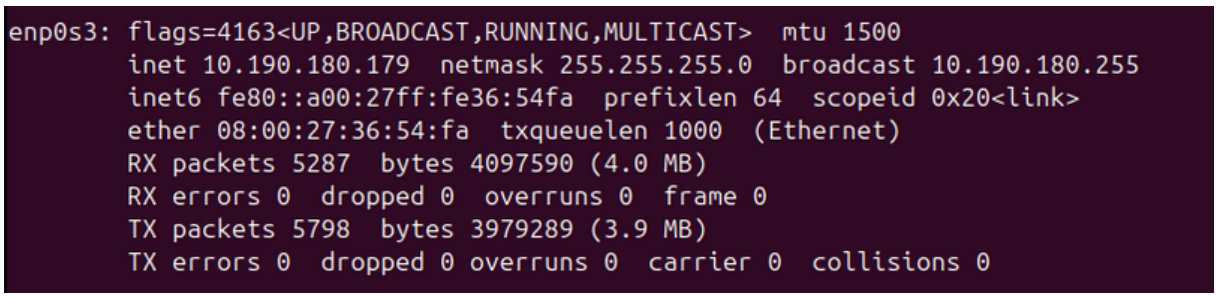

Figura 4. Resultado de ejecutar `ifconfig` para saber nuestra dirección IP pública.

Como podemos apreciar en la imagen en la segunda línea donde pone **inet** tenemos nuestra dirección IP pública que es `10.190.180.179`.

Para poder obtener la dirección IP teneís que hacer lo siguiente:

Instalamos el paquete `net-tools`:

```
sudo apt install net-tools
```

Para poder consultar las **interfaces de red** y ver nustra IP pública:

```
ifconfig
```

2.3 Verificar el acceso a Coolify

Ahora vamos a verificar que podemos acceder a la interfaz de coolify desde nuestro equipo.

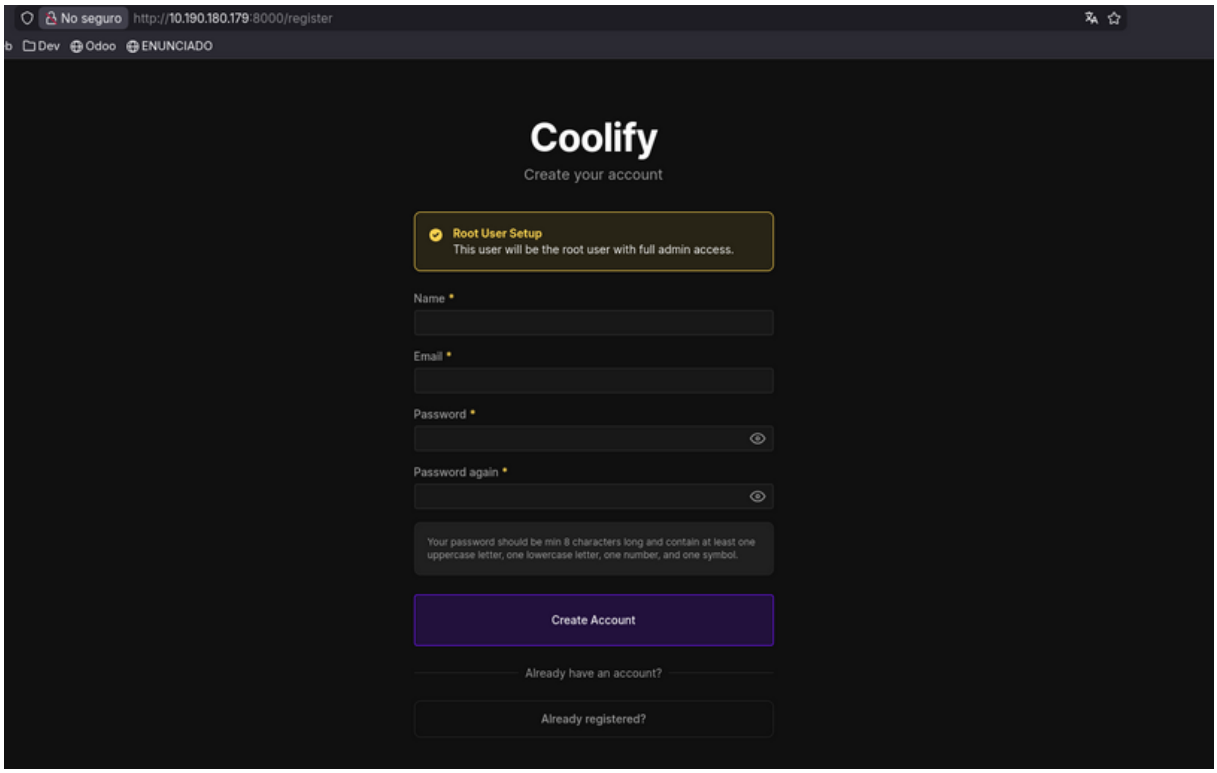

Figura 5. Interfaz web de Coolify con la página de crear una cuenta.

Para conectarnos y ver la pantalla de registro de coolify, tenemos que irnos a nuestro equipo y en nuestro navegador buscamos lo siguiente `"http://<IP_MÁQUINA>:8000"` y debería salirnos lo mismo que sale en la captura.

2.4 Configuración de coolify

Para terminar con la preparación de coolify, solo nos falta realizar unas configuraciones rápidas.

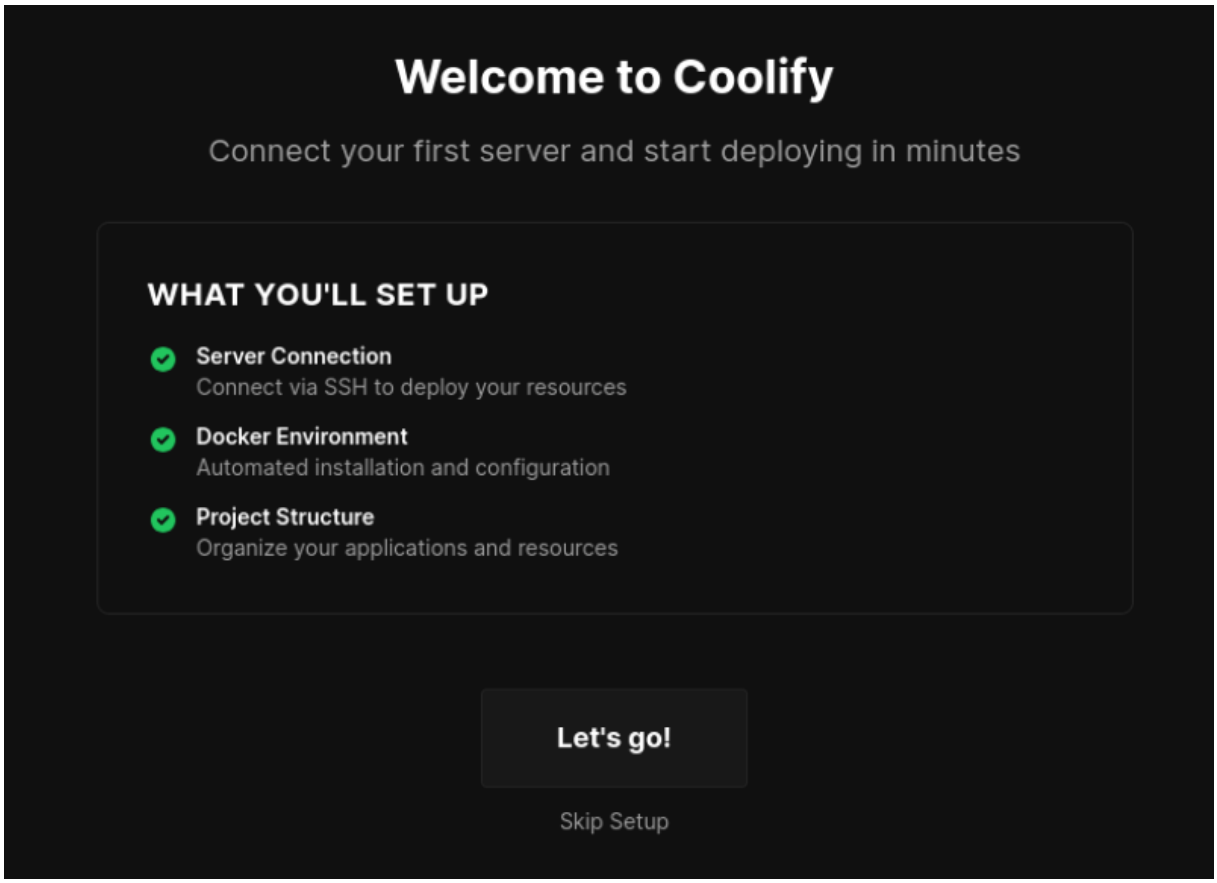

Figura 6. Pantalla con un mensaje de bienvenida después de haber creado una cuenta en coolify.

Tras crear la cuenta, Coolify nos mostrara la **pantalla de bienvenida**, donde se confirma que el registro se ha realizado correctamente. Aquí nos dice que vamos a configurar la **conexión del servidor** mediante `SSH`, la **instalación del entorno Docker** y la **organización de los proyectos**.

Ahora pulsaremos `"Let's go!"` esto nos iniciara el asistente de configuración para realizar las configuraciones necesarias, la opción `"Skip Setup"` omitira la configuración y accederemos directamente a la plataforma.

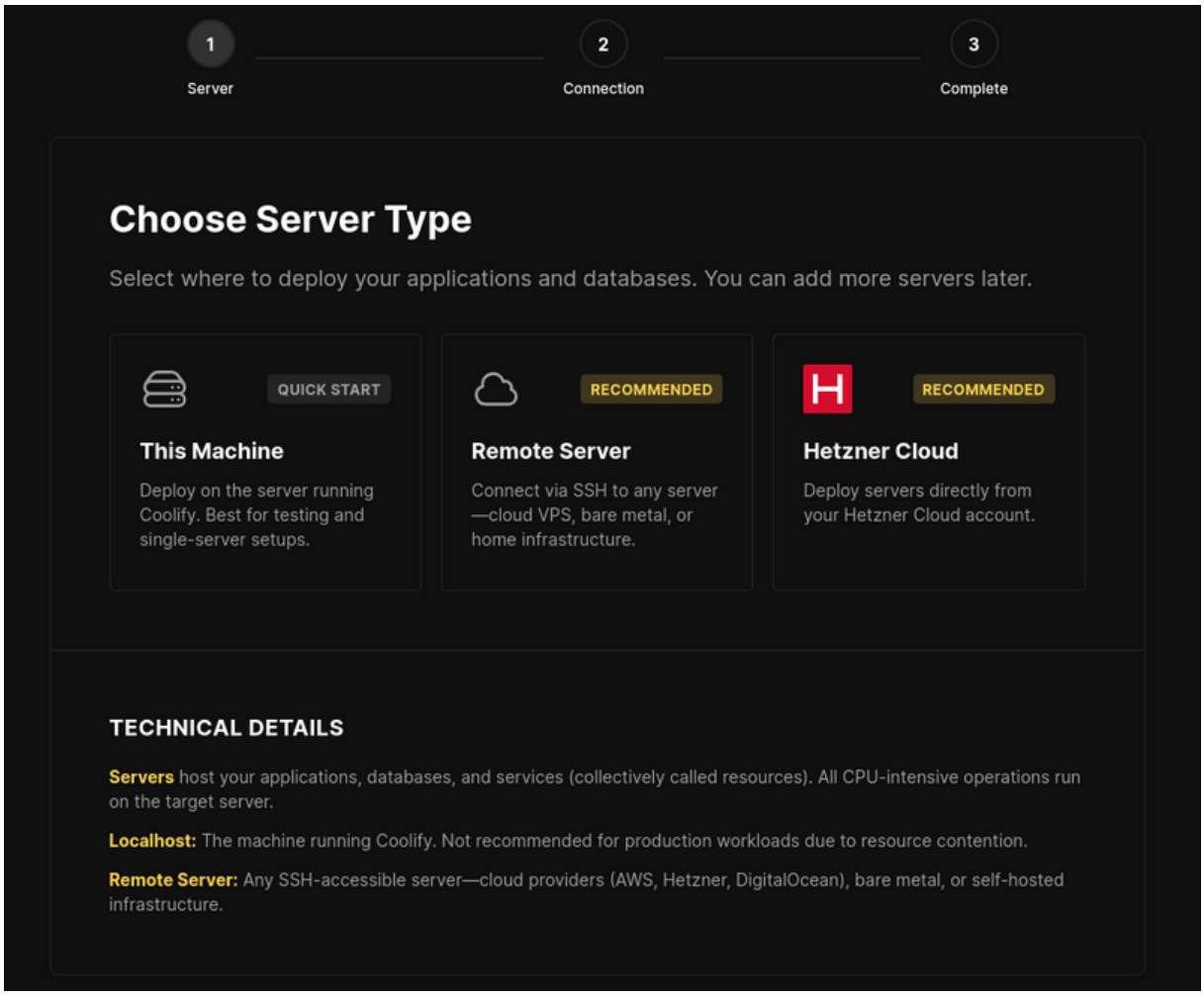

Figura 7. Pantalla para elegir el tipo de servidor que queremos

Como podemos ver ahora nos tacará **elegir el tipo de servidor** que necesitamos, para está prática elegiremos la opción de `"This Machine"` para utilizar nuestro ordenenador como servidor.

Cuando lo seleccionaresmos podremos crear nuestro primer proyecto o ir a la aplicación derectamente.

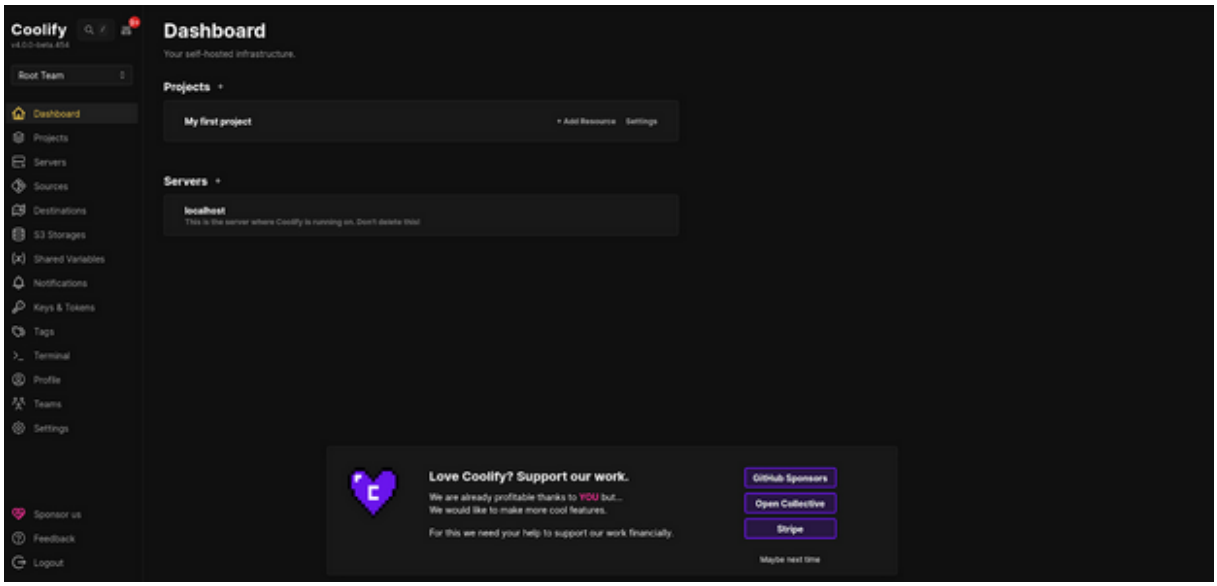

Figura 8. Página del dashboard de Coolify

Como podemos ver ya terminamos de configurar y ya tenemos nuestro **coolify listo** para poder trabajar con él.

3 NGROK

Ngrok es una **herramienta** que permite **exponer servicios locales** a internet mediante túneles. Esto es especialmente útil en entornos de desarrollo y pruebas, ya que nos quita de configuraciones complejas de red.

En este caso, usaremos Ngrok para poder **hacer accesible el panel web de Coolify** desde cualquier navegador.

3.1 Instalación de Ngrok

En este apartado vamos a realizar la instalación de Ngrok en nuestra máquina virtual Ubuntu.

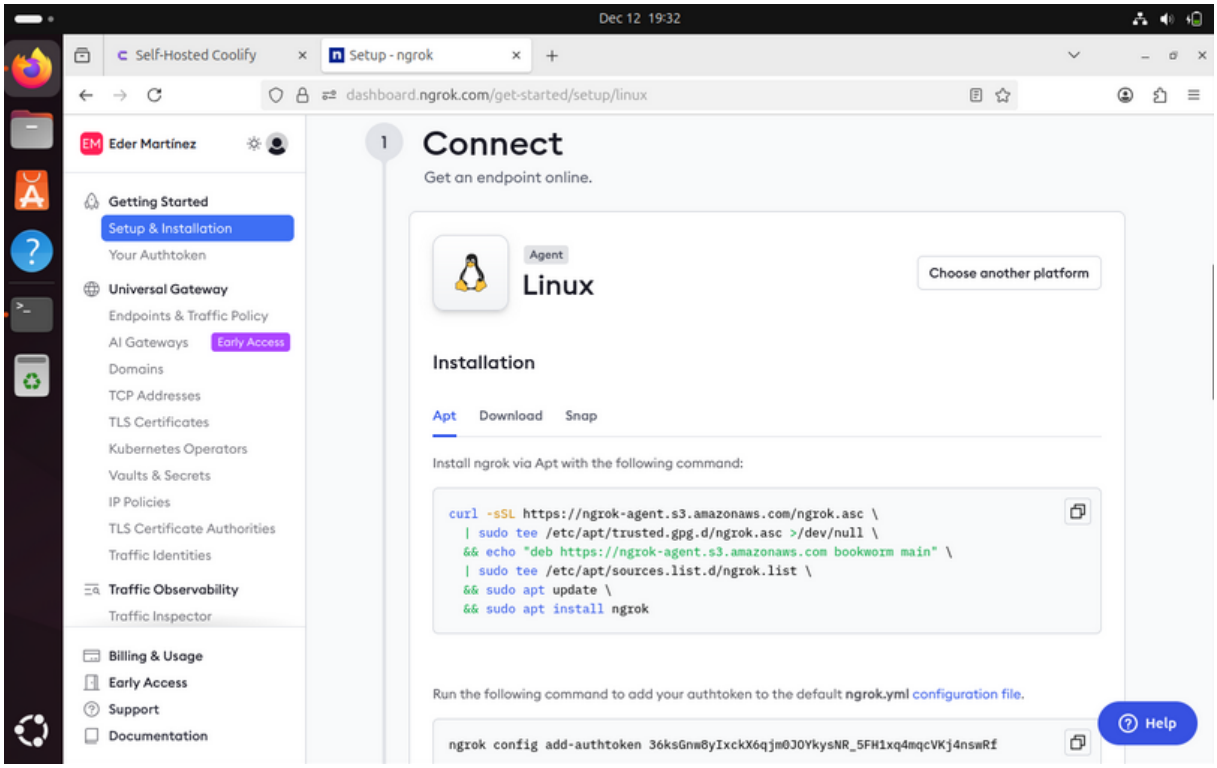

Figura 9. Web oficial de ngrok para obtener el comando para instalarlo.

Para poder obtener el **comando de instalación de Ngrok**, nos iremos a la página oficial de Ngrok y allí nos creamos una cuenta y iniciaremos sesión, cuando ya estemos autentificados en Ngrok estaremos en esta pantalla y bajaremos hasta llegar a la parte de **connect** y copiaremos el comando.

Este comando añade el **repositorio oficial de Ngrok** y permite su instalación mediante el gestor de paquetes apt:

```
curl -sSL https://ngrok-agent.s3.amazonaws.com/ngrok.asc \
| sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null \
&& echo "deb https://ngrok-agent.s3.amazonaws.com bookworm main"
   \
| sudo tee /etc/apt/sources.list.d/ngrok.list \
&& sudo apt update \
&& sudo apt install ngrok
```

Página oficial de Ngrok: https://ngrok.com/

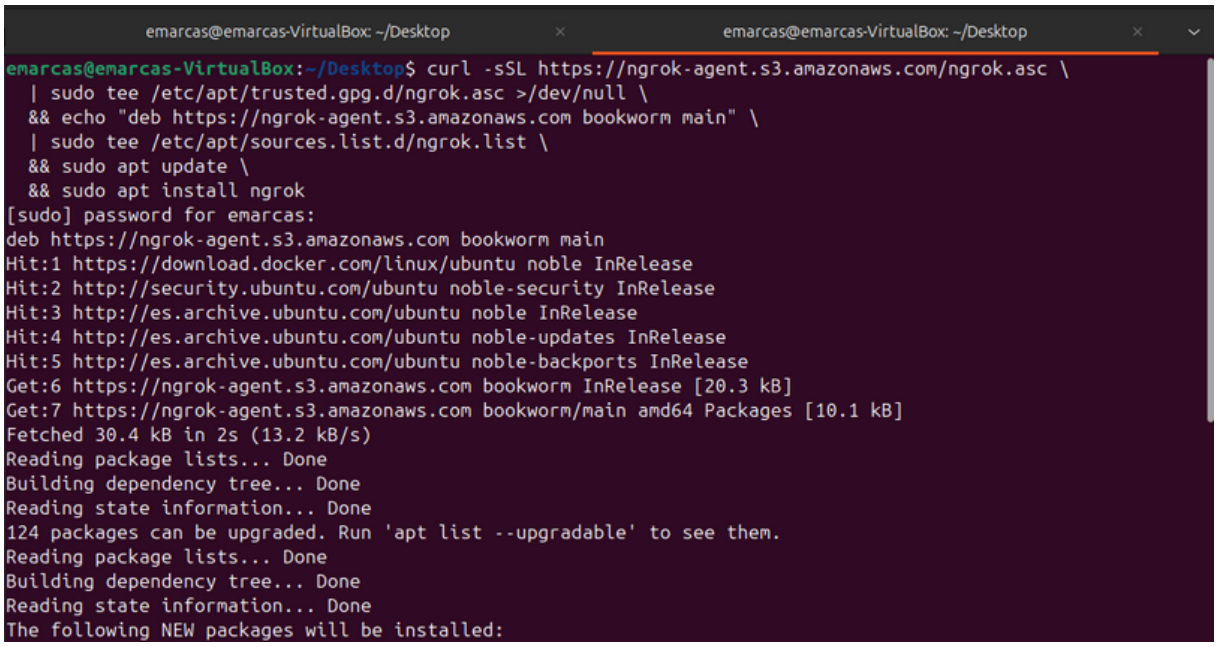

Figura 10. Ejecutamos el comando para instalar ngrok.

Ahora vamos a ejecutar el comando que copiamos anteriormente, en nuetra **terminal** de la máquina virtual.

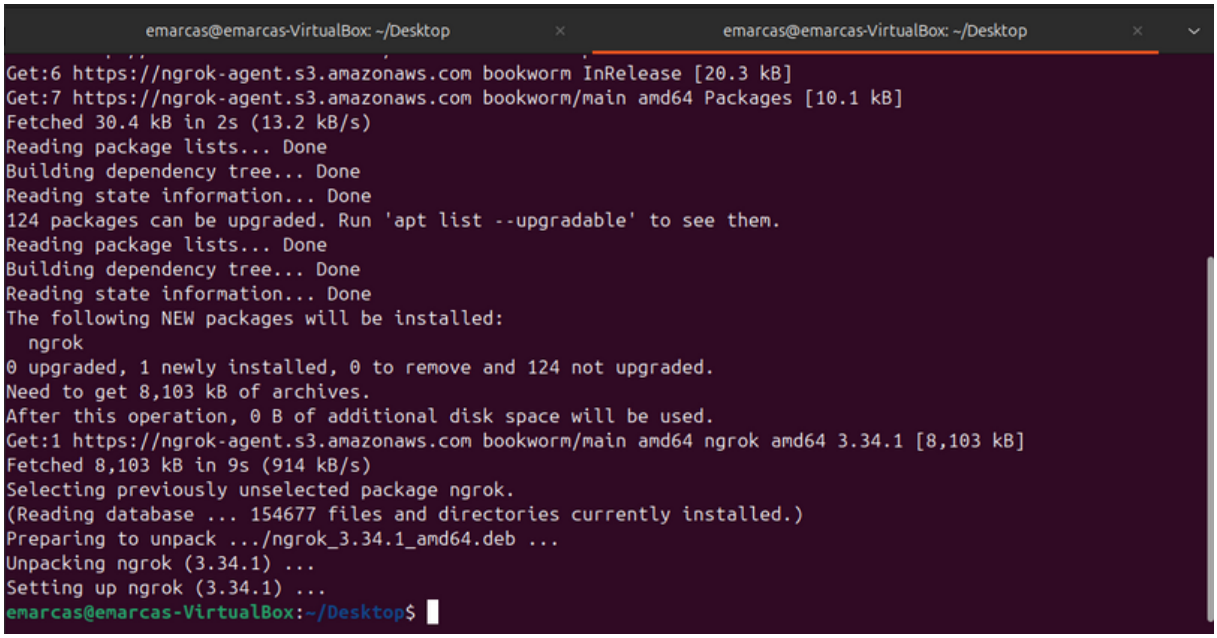

Figura 11. Captura de la terminal cuando ya termino de instalar ngrok en nuestro equipo.

Como podemos ver ya termino la **instalación de Ngrok** correctamente, ahora ya podemos crear `túneles seguros` que expongan nuestros servicios locales a `Internet`.

3.2 Verificación conexión con Ngrok

Ahora vamos a verificar que podemos acceder desde nuestra equipo a nuestro Coolify en la máquina virtual a través de **Ngrok**.

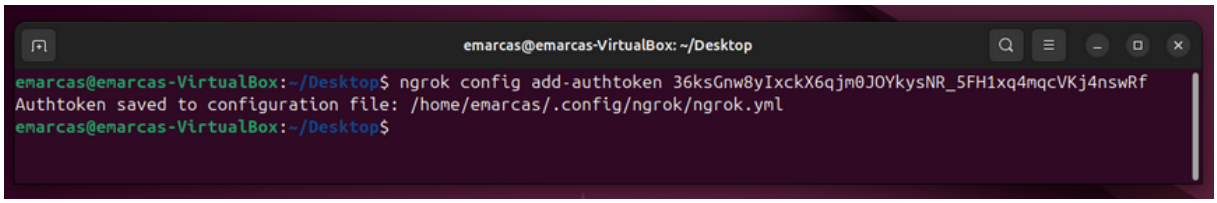

Figura 12. Ejecución del comando `ngrok config add-authtoken` para autenticar ngrok.

Tenemos que ejecutar este comando para poder **autenticar el cliente de ngrok** que instalamos en nuestra máquina virtual con nuestra cuenta de Ngrok. Esto es necesario para poder **crear túneles seguros y estables** entre el servicio local y el exterior.

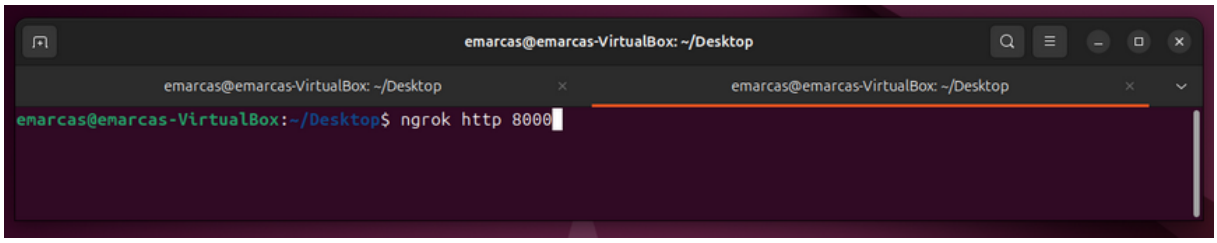

Figura 13. Ejecución del comando `ngrok http 8000` para exponer el servicio a través de un túnel público.

Al ejecutar este comando **creamos un túnel público seguro** que expone nuetro servicio que se esta ejecutando en el `puerto 8000` de nuestra máquina virtual que es nuestro **Coolify** a través de internet usando Ngrok.

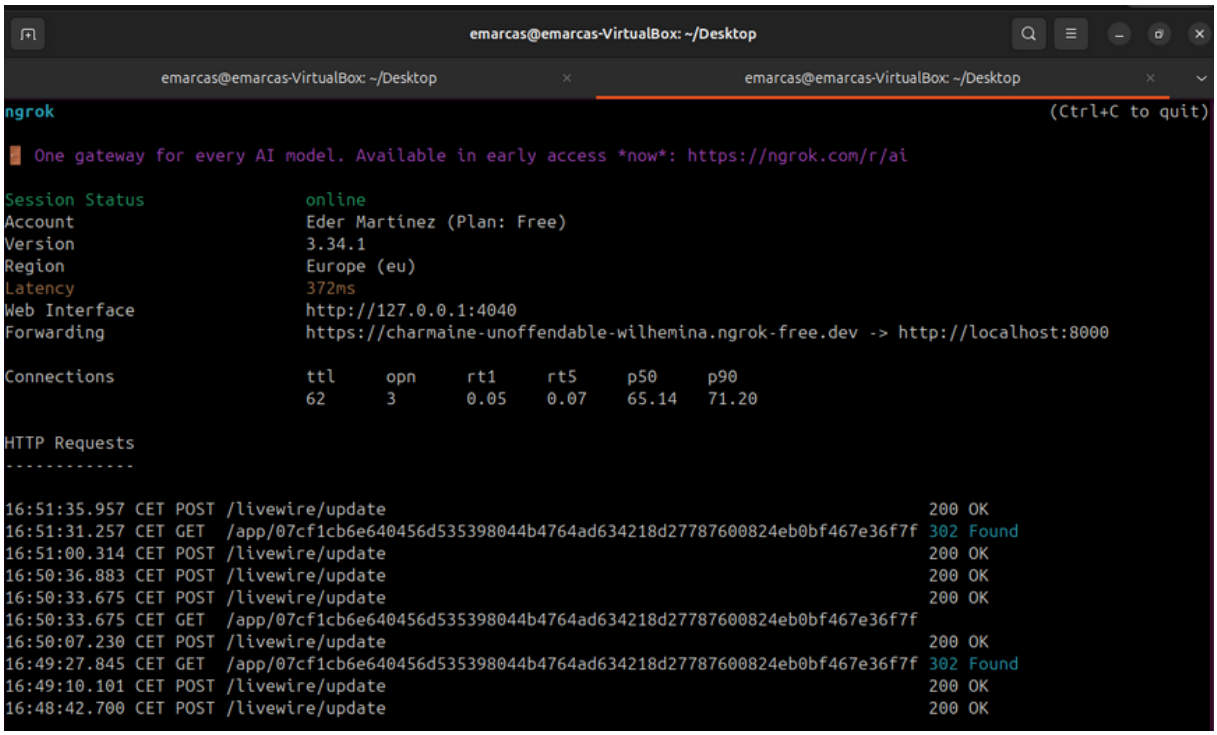

Figura 14. Estado del túnel ngrok activo y la URL pública generada para acceder al servicio.

Después de ejecutar el comando anterior para crear el túnel, en nuestra terminal se nos **mostrará el estado del túnel**, y nos mostrará la **URL pública https para poder acceder a nuestro servicio** de Coolify que está corriendo en el `puerto 8000`.

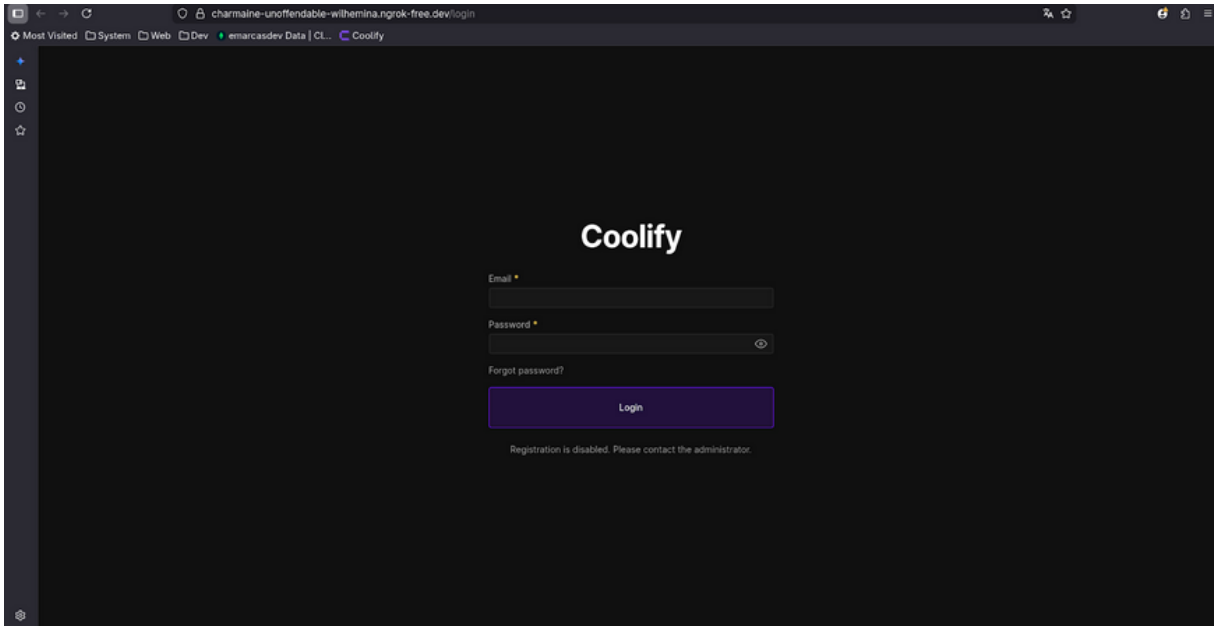

Figura 15. Inicio de sesión de Coolify accesible por la URL pública generada por Ngrok.

Como vemos en la imagen, hemos podido acceder a **nuestro Coolify** a través de túnel público creado con Ngrok, verificando que configuramos correctamente Ngrok.

4 DESPLEGAR NUESTRO PRIMER PROYECTO

Ahora vamos a realizar el despliegue de nuestra API realizada en **Express.js** y con base de datos **Maria DB** utilizando **Coolify**. Esta API la tenemos subida en nuestro GitHub: https://github.com/emarcasdev/LEARN_API-Users-Groups.

4.1 Crear nuevo proyecto

Ahora vamos a ver que necesitamos para crear un nuevo proyecto en Coolify

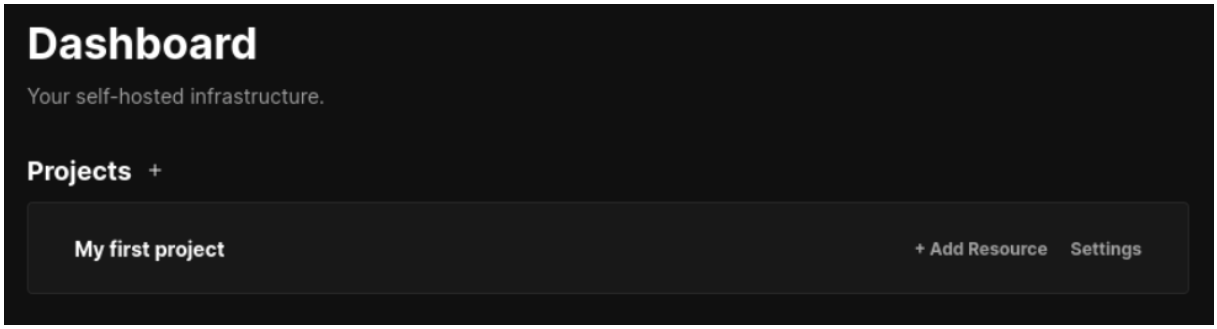

Figura 16. Botón para crear el nuevo proyecto en Coolify en nuestro caso es el primero.

Para crear nuestro primer proyecto solo tendremos que clicar en **"My first proyect"** y luego tendremos que agregar los siguientes recursos:

- Recurso para la **base de datos** en nuestro caso Maria DB.
- Recurso para la **aplicación** la API que tenemos subida en GitHub.

4.2 Creación de la base de datos (Maria DB)

A continuación veremos la creación y configuración del recurso de base de datos en Coolify para poder realizar el despliegue.

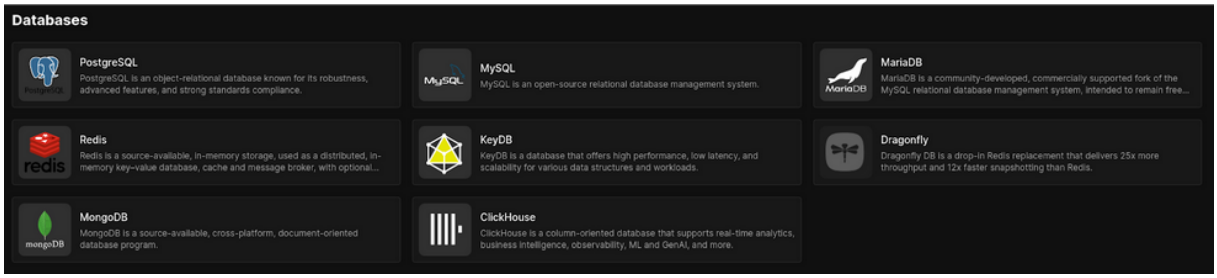

Figura 17. Aquí vemos todas las bases de datos que podemos crear para nuestro despliegue.

Para desplegar nuestro proyecto, al haber utilizado **MariaDB** en nuestra API eligiremos está opción para poder crear la base de datos para el despliegue.

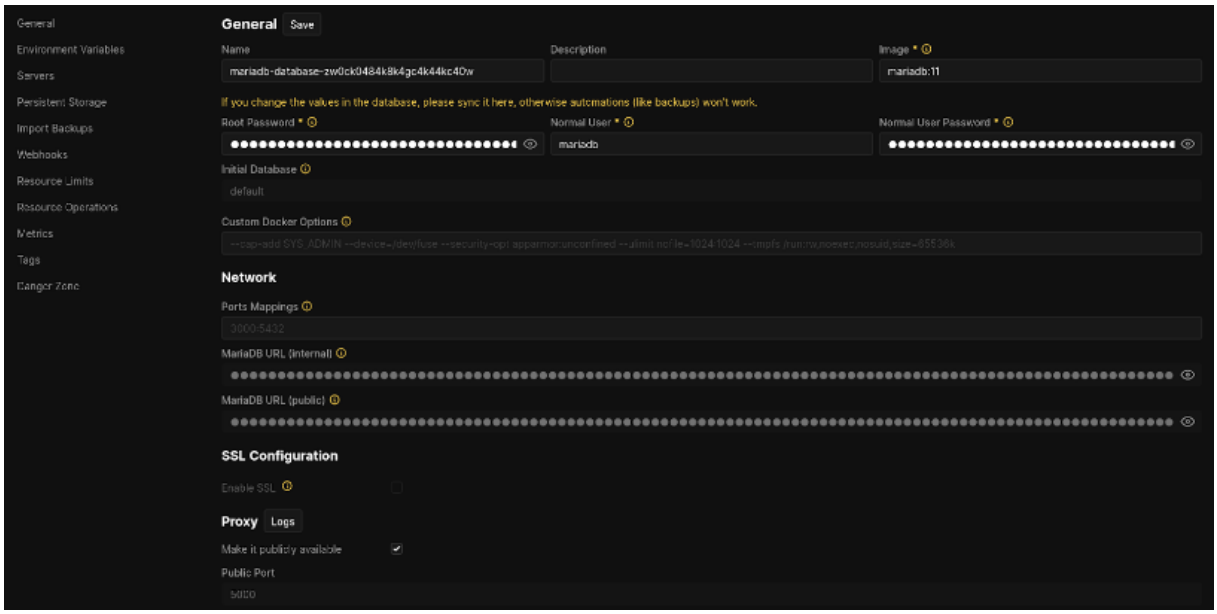

Figura 18. Configuración básica de nuestra base de datos para el despliegue.

En el panel de configuración de nuestra base de datos **MariaDB**, podemos ver que tenemos las credenciales de acceso, la base de datos y la imagen de **Docker** utilizada. Nosotros solo modificaremos los siguiente el puerto público en nuestro caso el 5000 y marcaremos la visibilidad del servicio como pública. Con estos ajustes hacemos que la base de datos funcione correctamente y que sea accesible por las aplicaciones desplegadas.

Figura 19. Verificación de que la base de datos **MariaDB** está corriendo

Después de haber terminado con la configuración del recurso de **MariaDB**, lo iniciaremos y esperaremos se inicialice y esté activa, eso lo podemos ver por el texto en verde que nos pone que esta corriendo correctamente como se aprecía en la imagen.

4.3 Creación de la aplicación (API Express.js)

A continuación veremos la creación y configuración del recurso de aplicación en Coolify para poder realizar el despliegue.

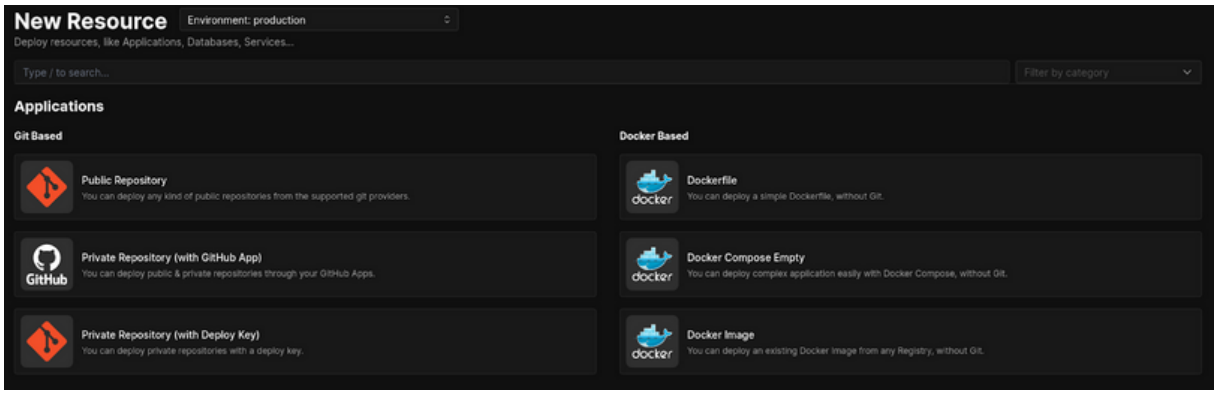

Figura 20. Como podemos ver tenemos varias formas de subir nuestra aplicacion a coolify

Para desplegar nuestro proyecto, al haber subido el código de nuestra API en github, eligiremos está opción de repositorio público para el despliegue.

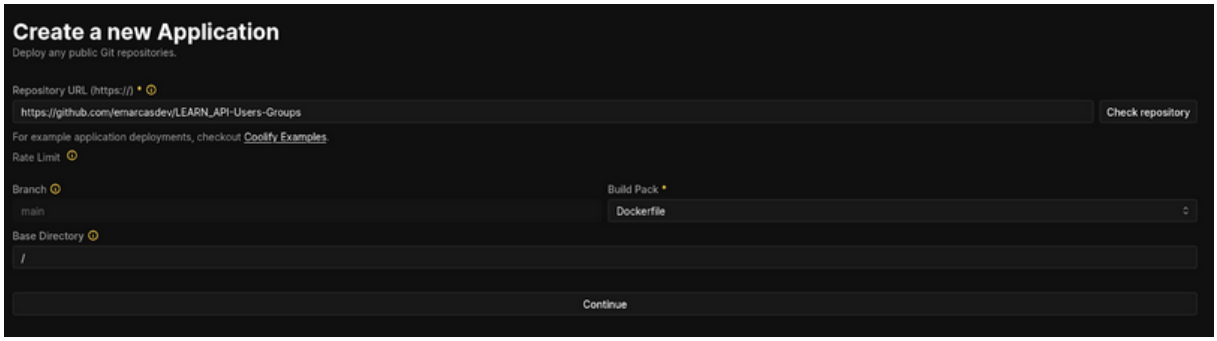

Figura 21. Captura donde creamos la aplicación basada en nuestro repositorio público

Para crear la nueva aplicación debemos escribir lo siguiente la URL de nuestro **repositorio público**, el Build Pack usaremos **Dockerfile** y en mi caso el directorio base sera el raiz.

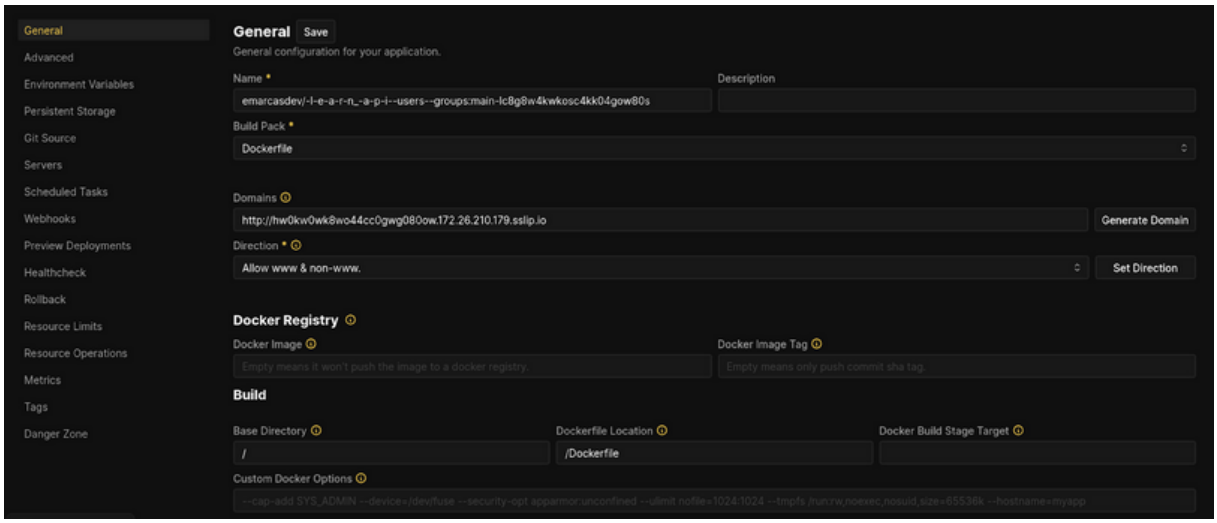

Figura 22. Configuración básica de nuestra aplicación para el despliegue

En el panel de configuración de la aplicación a desplegar, definimos que el método de construcción sea el `Dockerfile`, el dominio con el cuál accederemos y el puerto en el que expondremos la aplicación será el `6060`. Con estos parámetros tenemos lo necesario para el correcto despliegue y acceso a la aplicación.

NOTA:

Importante para que funcione el dominio que nos dío Coolify tenemos que **poner la IP de la máquina virtual** en vez de la IP que pone por defecto porque puede estar mal.

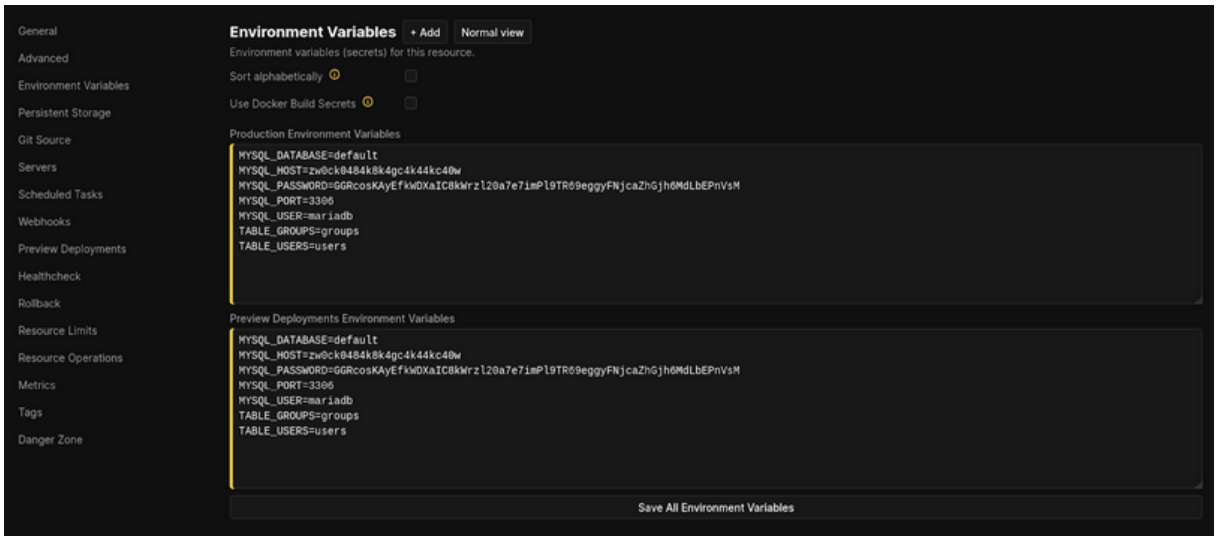

Figura 23. Captura donde definimos las variables de entorno para la aplicación.

En las variables de entorno nos pondremos en el modo desarrollador para facilitar el poder escribirlas, donde definimos las partes de la uri de **MariaDB** y los nombres de la tablas.

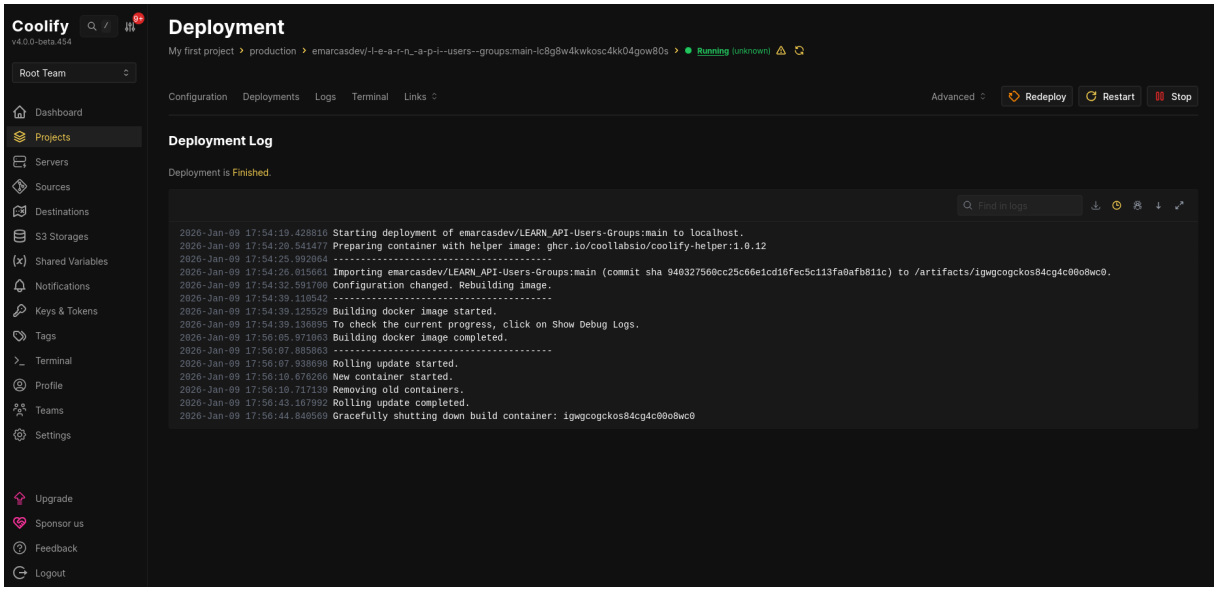

Figura 24. Captura del despliegue de nuestra aplicación.

Aquí podemos ver el **log del despliegue de nuestra aplicación**, podemos ver como se construyo la imagen de Docker, y vemos como el proceso finalizo correctamente y la aplición queda lista y accesible.

4.4 Verifición del despliegue

Ahora vamos a verificar que nuestra **API** está desplegada en la URL que nos había generado **Coolify**, para ver si podemos acceder.

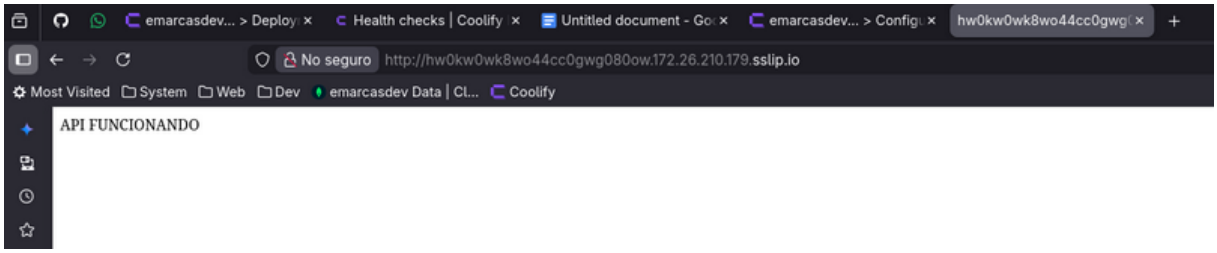

Figura 25. Mensaje de que la API está funcionando

Como podemos ver al poner la URL en nuestra navegador nos muestra nuestra API desplegada a través de Coolify, así que podemos decir que el despliegue fue éxitoso.

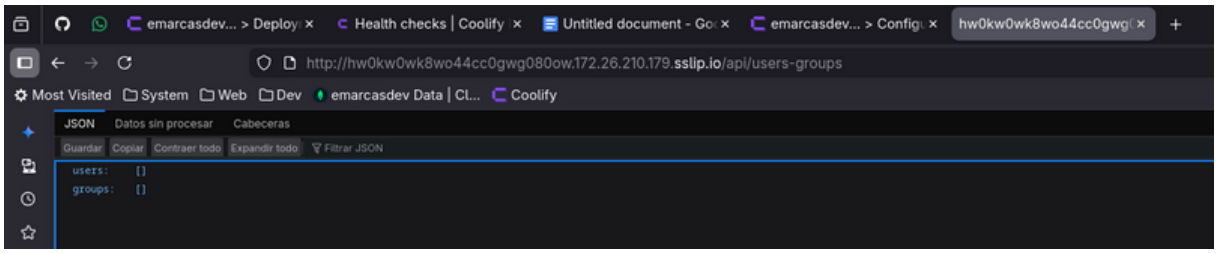

Figura 26. Prueba de recuperar los usarios y grupos de nuestra base de datos

Y para finalizar vamos a probar nuestra petición `GET` para recuperar los usuarios y grupos de nuestra base de datos, y como se puede ver nos muestra los usuarios y grupos vácios porque todavía no hay, pero esto nos confirma que la **API funciona correctamente**.

4.5 Creación de un puente para exponer la API

Ahora vamos a crear un puente para exponer nuestra API pero de forma que podamos **acceder** a ella **desde fuera de la red**. Esto nos generará una URL pública HTTPS que redirigirá las solicitudes hacia nuestro servicio.

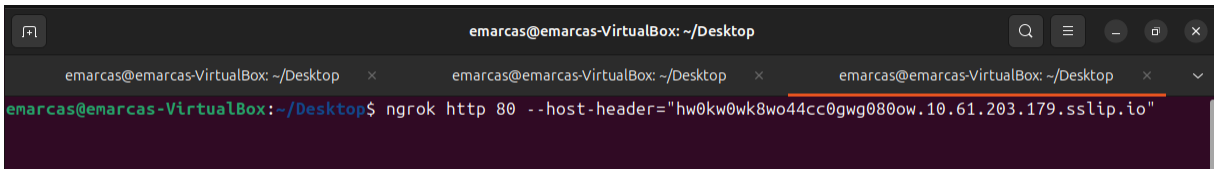

Figura 27. Ejecución del comando `ngrok http 80 -host-header=URL` para exponer el servicio a través de un túnel público.

Al ejecutar este comando, creamos un túnel que expone el servicio que está corriendo en la **máquina virtual**. De esta forma, las solicitudes que lleguen a la URL pública de ngrok se redirigen al **puerto configurado** del servidor, permitiendo el acceso **desde fuera de la red local**.

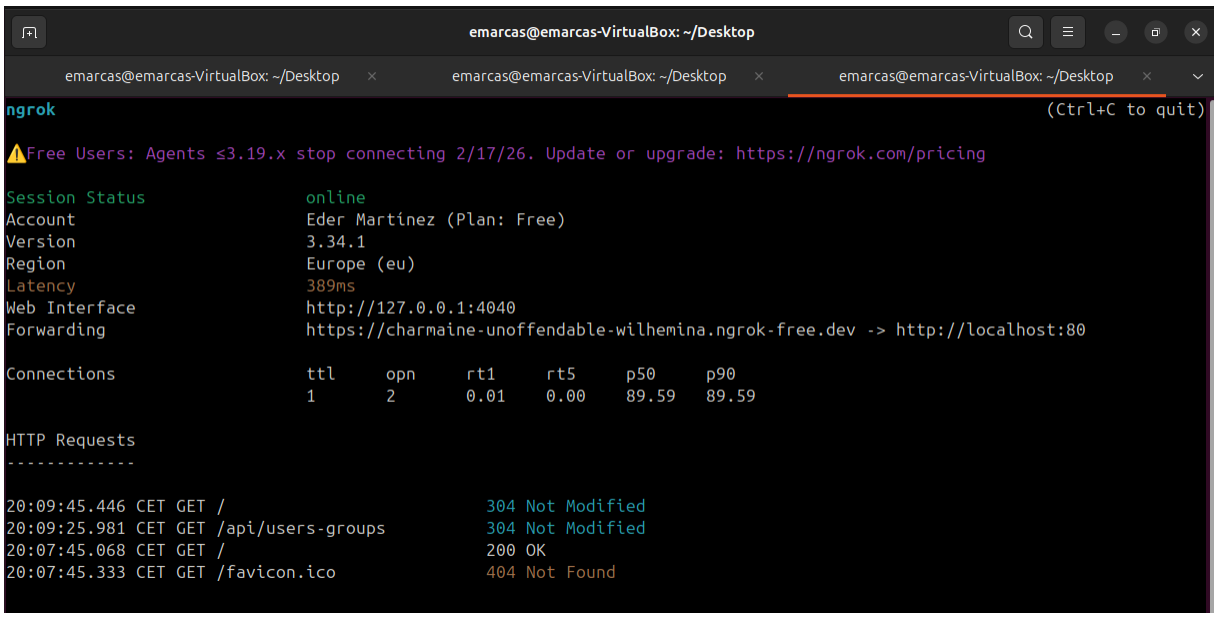

Figura 28. Estado del túnel ngrok activo y la URL pública generada para acceder al servicio.

Una vez creado el túnel, la terminal muestra el **estado de la sesión** y la **URL pública HTTPS** generada por ngrok. Este enlace nos permitira acceder al servicio, ya que cualquier petición enviada a esa URL será encaminada automáticamente hacia el servicio que se está ejecutando en el `puerto 80`.

Para concluir, vamos a mostrar las capturas donde demostramos que nos accedemos a la API tanto como en el ordenador como en móvil correctamente.

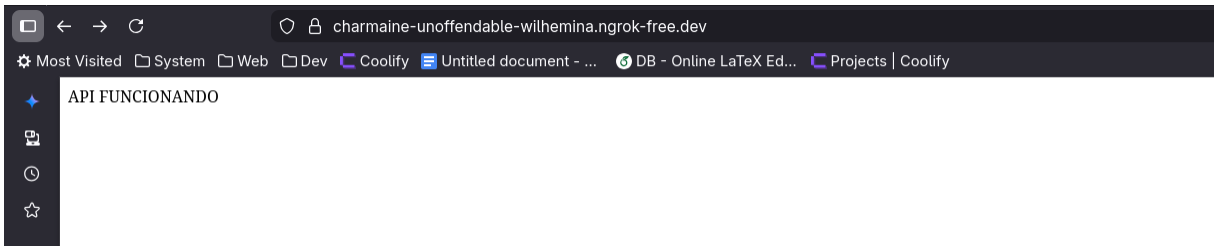

Figura 29. Comprobación de que podemos acceder a la API directamente con el enlace del puente de Ngrok (Ordenador)

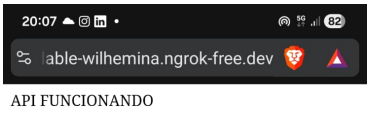

Figura 30. Comprobación de que podemos acceder a la API directamente con el enlace del puente de Ngrok (Móvil)

4.6 Agregar CI/CD

Ahora para finalizar esta práctica vamos a agregar CI/CD a nuestro proyecto desplegado en Coolify.

4.6.1 Creación y configuración del webhook

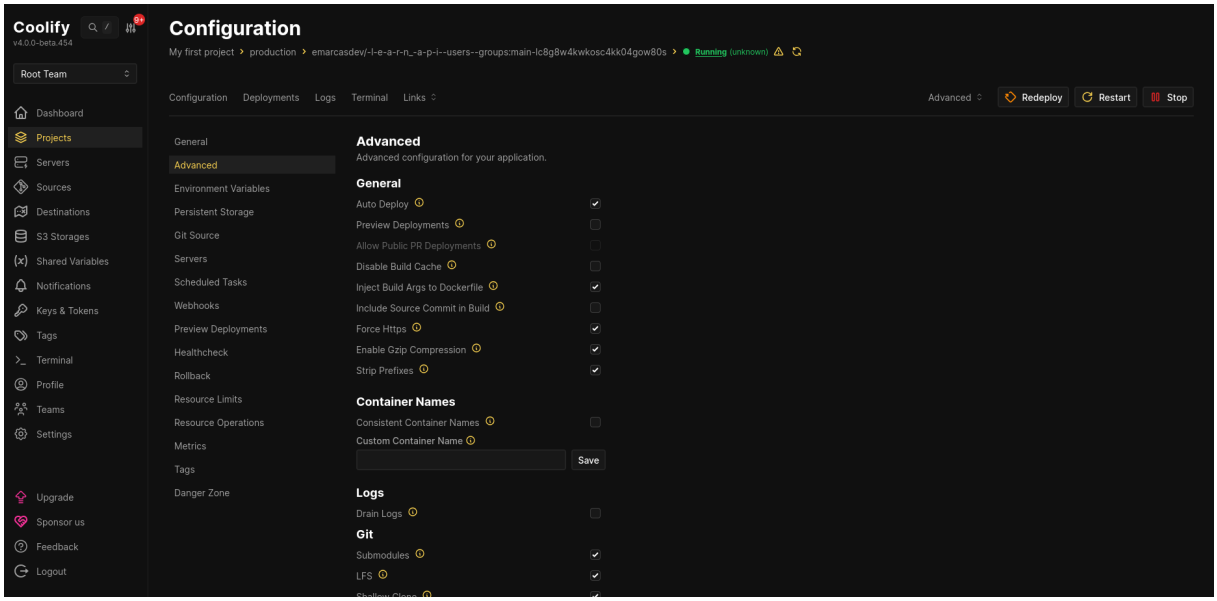

Figura 31. Opciones avanzadas de nuestro proyecto en coolify

Nos dirigiremos al apartado de webhook, para **verificar el auto-deploy** para saber si la tenemos activa o activarla en el caso en que no estuviera, para así poder hacer que el proyecto se auto despliegue cada vez hagamos un push a nuestro repositorio.

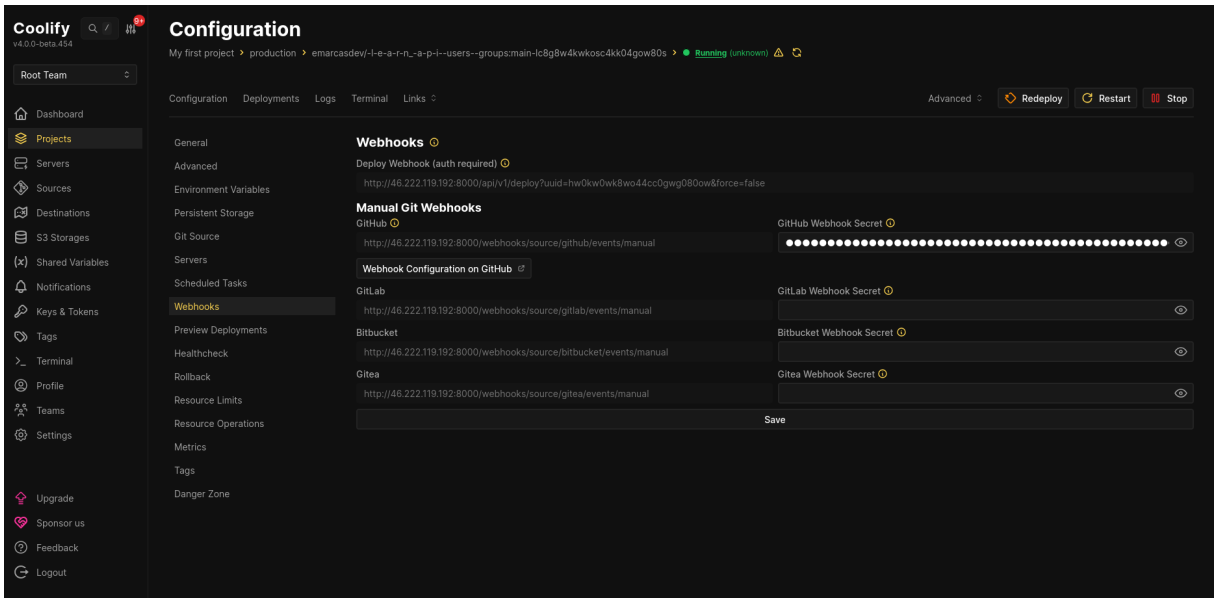

Figura 32. Obtener la ruta para el `webhoook` y crear el `secret`

Ahora iremos al apartado de webhook y ahí nos interesa el enlace que nos sale en la sección de github pero **sustituyendo la dirección ip y el puerto** por nuesta **ruta generada por ngrok** y luego en la sección de GitHub Webhook Secret **crearemos un secret manualmente**.

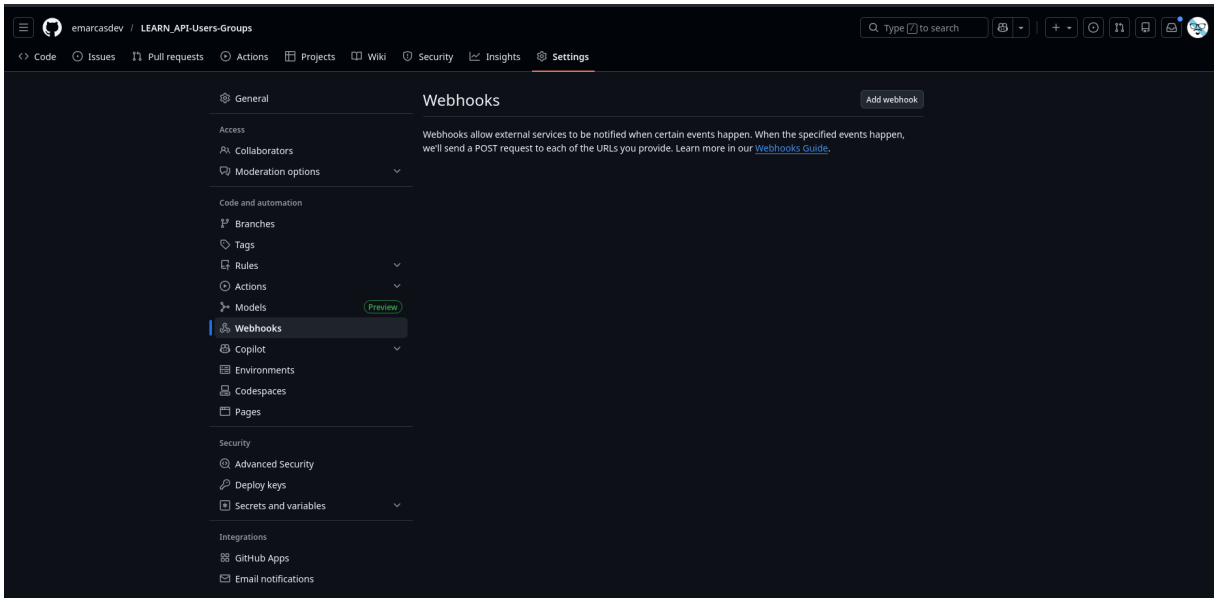

Figura 33. Crear webhook en github

Para crear el webhook en github nos dirigiremos al apartado de configuración y en la sección de Code and automation entraremos en la opción de Webhooks, y como vemos en la captura le daremos al botón de add para **crear nuestro webhook**.

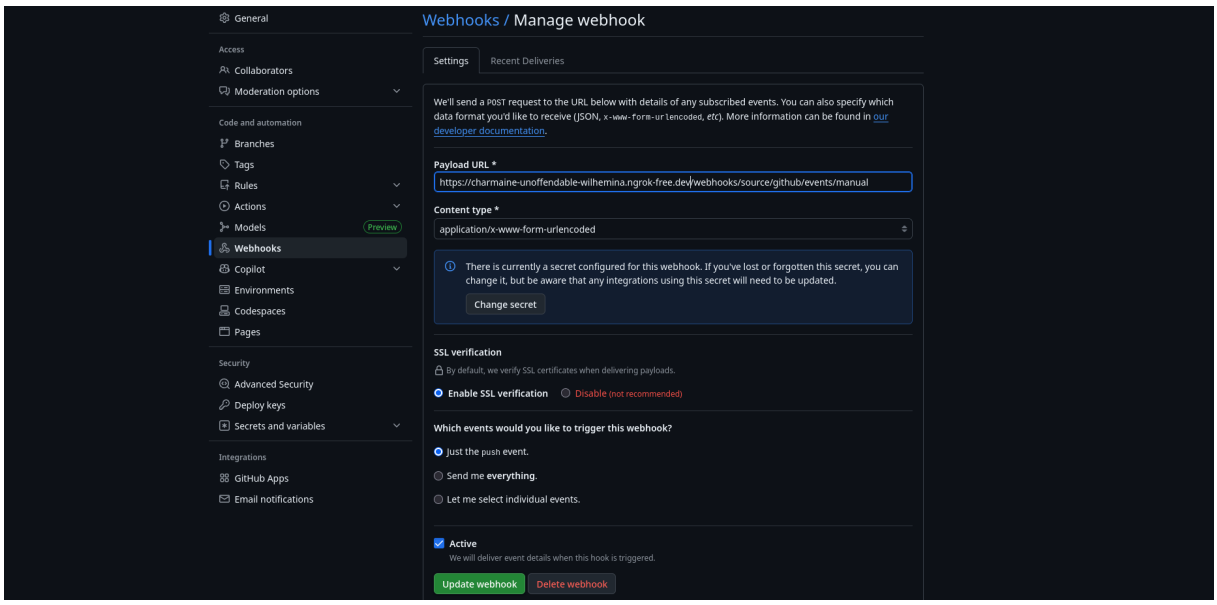

Figura 34. Configuración del webhook en github

Ahora en la configuración debemos rellenar el **Payload URL** con https://charmaine-unoffendable-wilhemina.ngrok-free.dev/webhooks/source/github/events/manual y

el **secret** ponemos el que creamos manualmente anteriormente, lo demás no hace falta modificarlo.

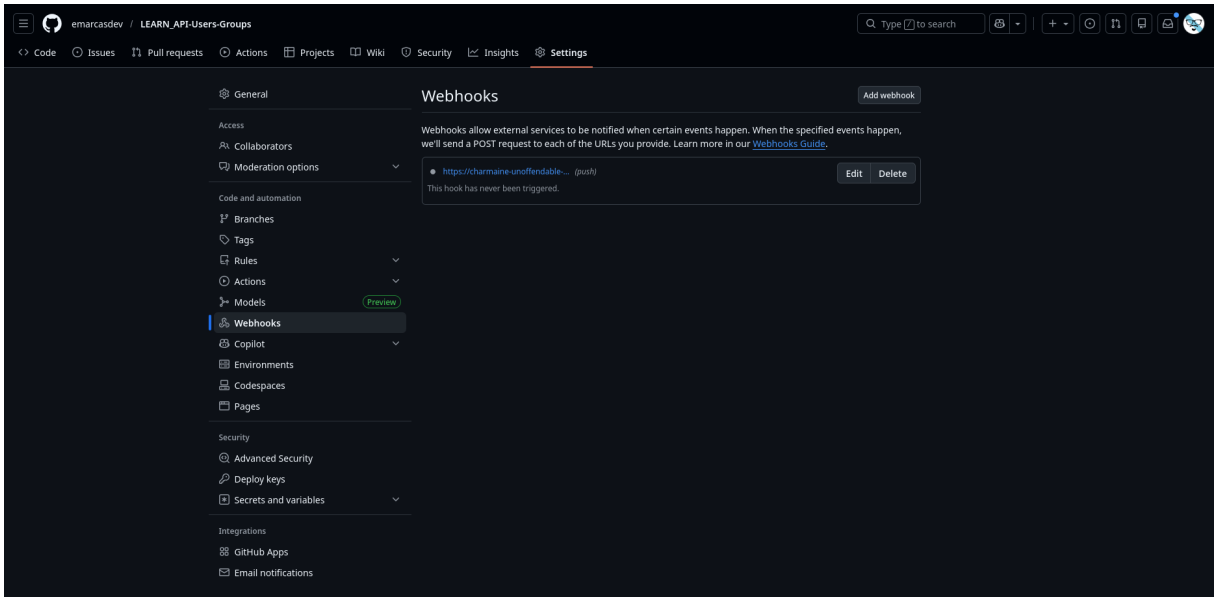

Figura 35. Webhook creado en github

Como podemos ver ahora **se creo el webhook correctamente**, nos sale en gris porque todavía no se ejecuto el hook al no haber hecho ningún push.

4.6.2 Comprobación del funcionando

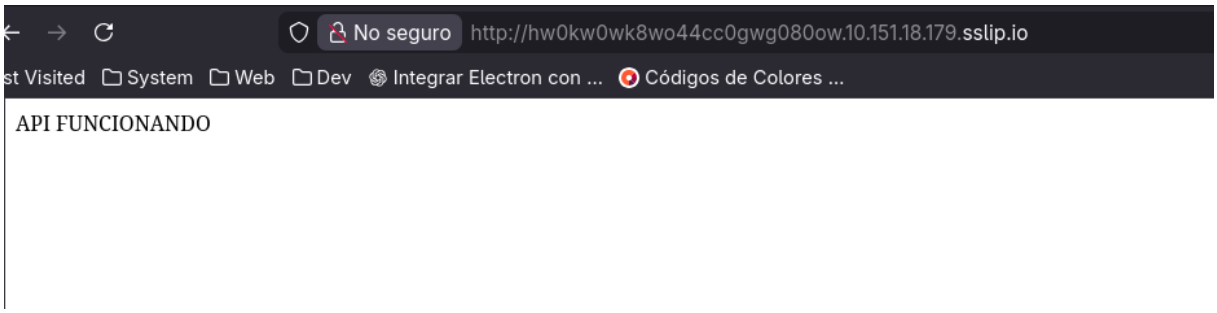

Figura 36. Estado actual de nuestro proyecto sin haber subido cambios al repositorio

Como vemos en la pantalla vemos el **estado actual** antes de hacer el commit y el push, modificando así la API, el cambio va a ser simple y es **modificar la salida del endpoint default** de nuestra API.

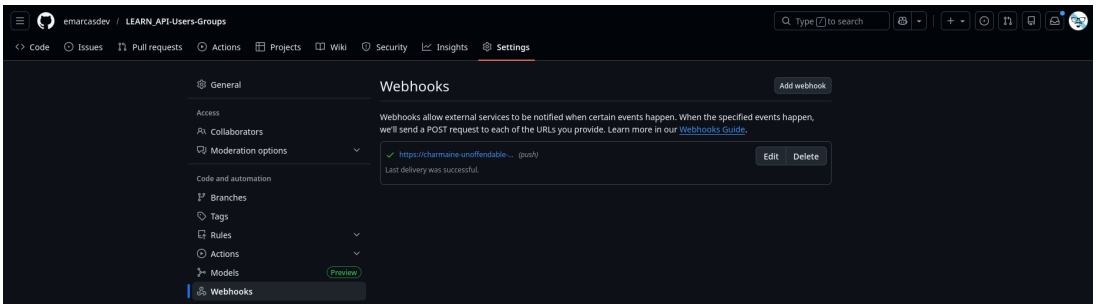

Figura 37. Webhook lanzado correctamente, ya que detecto el push correctamente

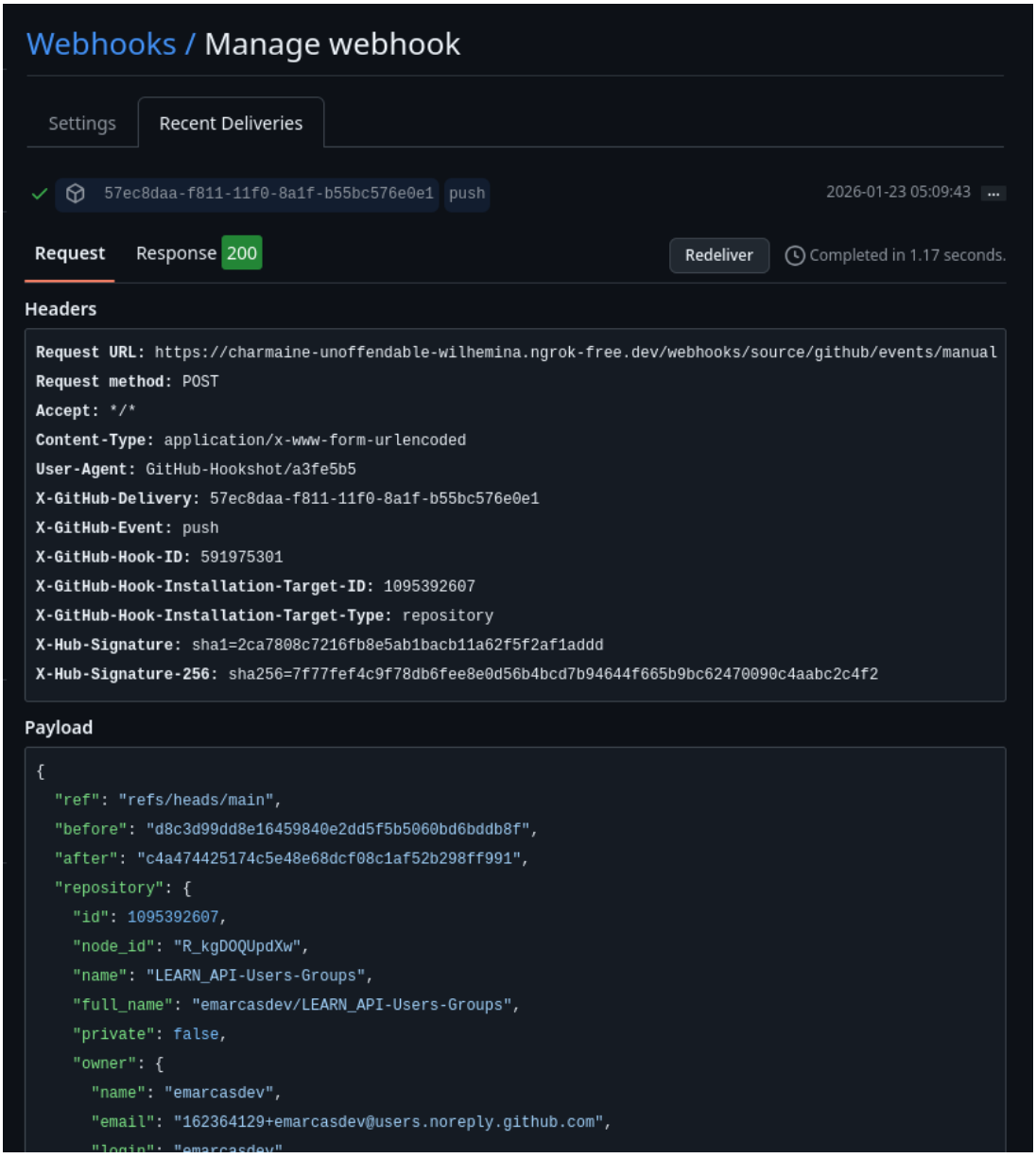

Figura 38. Detalles de que el webhook se lanzo correctamente

Ahora vemos que el **hook se ejecuto correctamente**, indicando que detecto el push correctamente y que se auto desplego correctamente en Coolify.

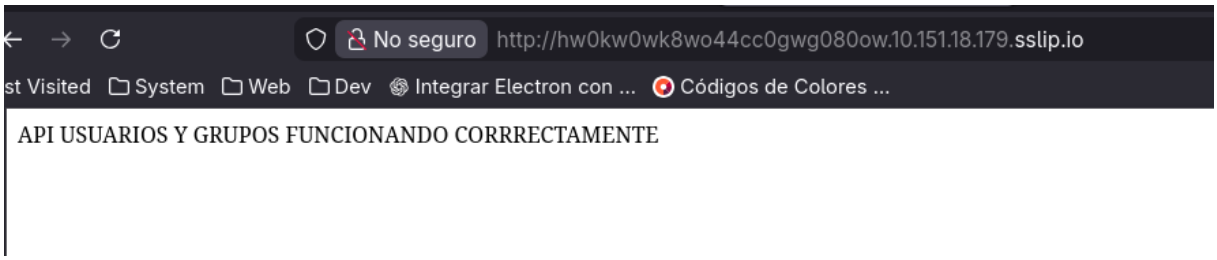

Figura 39. Comprobación de la API actulizada despues del auto despliegue

Ahora **verificamos que se auto desplego correctamente**, comprobando que se aplicaron el cambio que hicimos anteriormente.