

# **API Online**

Afondamento nas Competencias Profesionais

Cristian Fernández

23 de enero de 2026

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Herramientas empleadas</b>	<b>3</b>
2.1. VirtualBox . . . . .	3
2.2. Ngrok . . . . .	4
2.3. Coolify . . . . .	5
<b>3. Instalaciones</b>	<b>6</b>
<b>4. Configuraciones y despliegues</b>	<b>7</b>
4.1. Ngrok local . . . . .	7
4.2. Coolify (repositorio y base de datos) . . . . .	8
4.3. Ngrok expuesto en Internet . . . . .	16
4.4. Coolify (Webhook) . . . . .	18

# 1. Introducción

El objetivo de esta tarea es detallar el proceso de configuración y despliegue automatizado de un proyecto (contenedor *Docker API Express.js + MariaDB*) alojado en un repositorio remoto (*GitHub*), garantizando que cualquier cambio en el código fuente se refleje de manera eficiente en el entorno de ejecución.

También buscaremos la Integración Continua mediante la conexión directa entre el repositorio remoto y *Coolify*. Este enfoque permitirá al servidor detectar automáticamente nuevas actualizaciones en la rama principal, iniciar la construcción de las imágenes de *Docker* y desplegar los contenedores sin intervención manual, mejorando sustancialmente la calidad de vida del trabajo consiguiendo, entre otra cosas, reducir errores humanos y optimizar los tiempos de entrega.

Para ello seguiremos los siguiente pasos:

- Crearemos una **máquina virtual Ubuntu** (versión 24.04.2 LTS) que actuará como nodo principal de despliegue.
- Instalaremos y usaremos **Ngrrok** para crear túneles seguros desde nuestra máquina virtual hacia internet, permitiendo exponer servidores web (HTTP/HTTPS), sitios en desarrollo, webhooks y servicios TCP/SSH a una URL pública temporal.
- Instalaremos y usaremos **Coolify** para gestionar el ciclo de vida de la aplicación, desde la lectura del repositorio hasta el monitoreo de los contenedores.

## 2. Herramientas empleadas

### 2.1. VirtualBox

*VirtualBox* es un hipervisor de tipo 2 gratuito y de código abierto (GPLv2), ideal para virtualización en equipos de escritorio. Sus características clave incluyen la capacidad de ejecutar múltiples sistemas operativos invitados (Windows, Linux, macOS, etc.) de forma simultánea, alta portabilidad entre plataformas, creación de "snapshots" (instantáneas) para revertir fallos, soporte para dispositivos USB y redes virtuales.

Las características principales de *VirtualBox* son las siguientes:

- **Multiplataforma y Soporte de SOs:** Puede instalarse en Windows, macOS, Linux y Solaris (anfitrión), y virtualizar casi cualquier otro sistema operativo (invitado) de 32 o 64 bits.
- **Guest Additions:** Conjunto de controladores y aplicaciones que optimizan el rendimiento, mejoran la comunicación (carpetas compartidas, portapapeles compartido) y ajustan la resolución de pantalla.
- **Amplias Opciones de Red:** Soporta modos NAT, puente (bridged), red interna, y red exclusiva del anfitrión (host-only) para adaptar la conectividad de la VM.
- **Soporte de Hardware Virtual:** Emula hardware moderno, incluyendo soporte para USB 2.0/3.0, hasta 32 CPUs virtuales, y aceleración 3D.

## 2.2. Ngrok

*Ngrok* es una herramienta de desarrollo que crea túneles seguros desde una red pública a servidores locales, permitiendo exponer proyectos localhost mediante URLs HTTPS sin configurar firewalls o redes. Ofrece inspección de tráfico en tiempo real, soporte para múltiples protocolos (HTTP, TCP), y es ideal para pruebas rápidas, webhooks y demostraciones a clientes.

Las características principales de *Ngrok* son las siguientes:

- **Exposición de Localhost:** Permite que servicios locales sean accesibles desde internet a través de una URL pública.
- **Inspección de Tráfico:** Proporciona una interfaz web para analizar y depurar solicitudes HTTP/HTTPS entrantes.
- **Túneles Seguros y Rápidos:** Crea conexiones seguras y cifradas (HTTPS/TLS) hacia muestra máquina local de forma instantánea.
- **Soporte de Webhooks:** Facilita el desarrollo al permitir recibir notificaciones de servicios externos directamente en el entorno local.

### 2.3. Coolify

*Coolify* es una plataforma de autoalojamiento (self-hosting) de código abierto y sin límites, diseñada como una alternativa gratuita a Vercel o Heroku. Permite desplegar y gestionar aplicaciones web, bases de datos (PostgreSQL, MySQL, Redis, MongoDB) y servicios en cualquier servidor (VPS, Raspberry Pi) mediante Docker. Se caracteriza por la automatización Git-Ops, SSL gratuito, backups y una interfaz intuitiva.

Las características principales de *Coolify* son las siguientes:

- **Soporte Multi-tecnología:** Compatible con cualquier aplicación dockerizable, incluyendo Node.js, Python, PHP, Ruby on Rails, Rust y sitios estáticos.
- **Gestión Total de Bases de Datos:** Permite crear, gestionar y hacer copias de seguridad de bases de datos como PostgreSQL, MySQL, Redis y MongoDB, así como herramientas como WordPress o N8N.
- **Despliegue Automatizado (Git-Ops):** Conexión con GitHub, GitLab o Bitbucket para realizar despliegues automáticos cada vez que se actualiza el código.
- **Interfaz de Usuario Sencilla:** Panel de control moderno y fácil de usar, ideal tanto para principiantes como para desarrolladores experimentados.

### 3. Instalaciones

En la máquina virtual, abrimos un terminal y pegamos los códigos que se muestran en las siguientes capturas (es necesario crearse una cuenta en cada aplicación):

The screenshot shows the Coolify website with a dark theme. The main heading is "Why should you self-host?". Below it is a subtext "You will be cool." and a "Get Started" button. A code block contains the command: `# The self-hosted version is ready on your server in under a minute - always free, with all functionalities.  
> curl -fsSL https://cdn.coollabs.io/coolify/install.sh | sudo bash`. A red box highlights this command with the text "COPIAMOS EN TERMINAL Y PULSAMOS ENTER (LA INSTALACIÓN PUEDE TARDAR UN RATO)". Below the command is the heading "Self-hosting in general." and the subtext "There are plenty of benefits."

The screenshot shows the ngrok dashboard with a sidebar menu. The main area is titled "Setup - ngrok" and shows the "Apt" tab selected. It contains instructions to install ngrok via Apt with the command: `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`. A red box highlights this command with the text "COPIAMOS EN TERMINAL Y PULSAMOS ENTER". Below the command, there are sections for adding an auth token and deploying an app online. The "Deploy your app online" section includes the command `ngrok http 80`, which is highlighted with a red arrow and the text "EN ESTA PRÁCTICA USAREMOS EL PUERTO 8000 EN VEZ DEL 80".

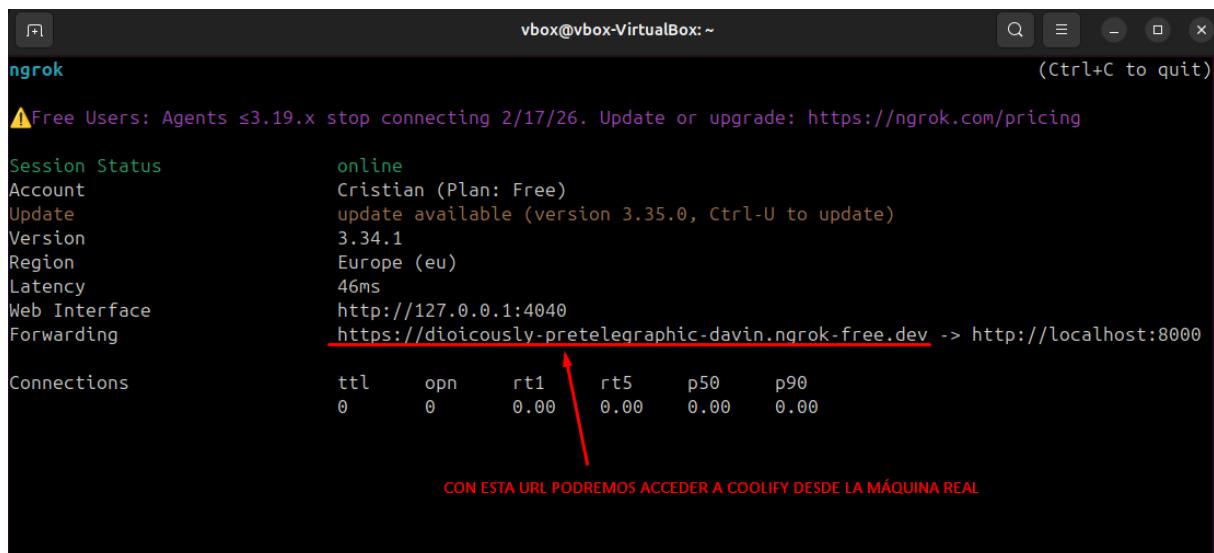
## 4. Configuraciones y despliegues

### 4.1. Ngrok local

En las siguientes capturas se muestra cómo levantar ngrok desde la terminal:



```
vbox@vbox-VirtualBox:~$ ngrok http 8000
LAS PRUEBAS EN ENTORNO LOCAL LAS HACEMOS EN PUERTO 8000
```



```
vbox@vbox-VirtualBox:~$ ngrok
(Ctrl+C to quit)

⚠️Free Users: Agents ≤3.19.x stop connecting 2/17/26. Update or upgrade: https://ngrok.com/pricing

Session Status      online
Account            Cristian (Plan: Free)
Update             update available (version 3.35.0, Ctrl-U to update)
Version            3.34.1
Region             Europe (eu)
Latency            46ms
Web Interface     http://127.0.0.1:4040
Forwarding         https://dioicously-pretelegraphic-davin.ngrok-free.dev -> http://localhost:8000
Connections        ttl     opn      rt1     rt5      p50      p90
                    0       0       0.00    0.00    0.00    0.00

CON ESTA URL PODREMOS ACCEDER A COOLIFY DESDE LA MÁQUINA REAL
```

## 4.2. Coolify (repositorio y base de datos)

Una vez levantado Ngrok en la máquina virtual podremos acceder a Coolify en la máquina real abriendo un navegador y entrando en la url. Miraremos lo siguiente:

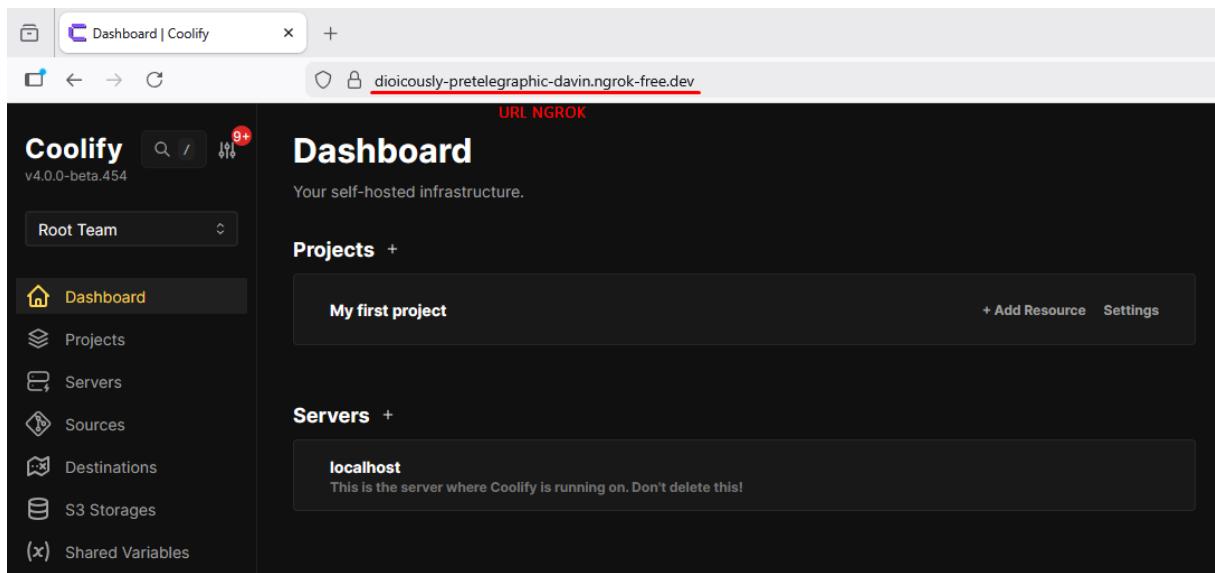


Figura 1: Página principal de Coolify

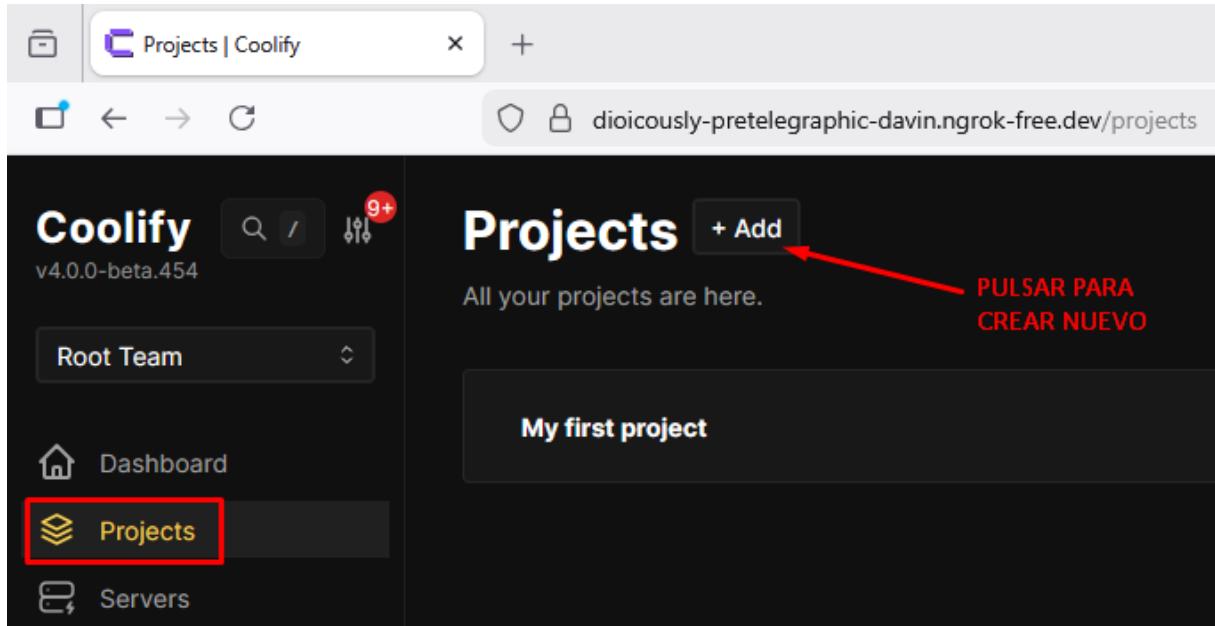


Figura 2: Crearemos un nuevo proyecto

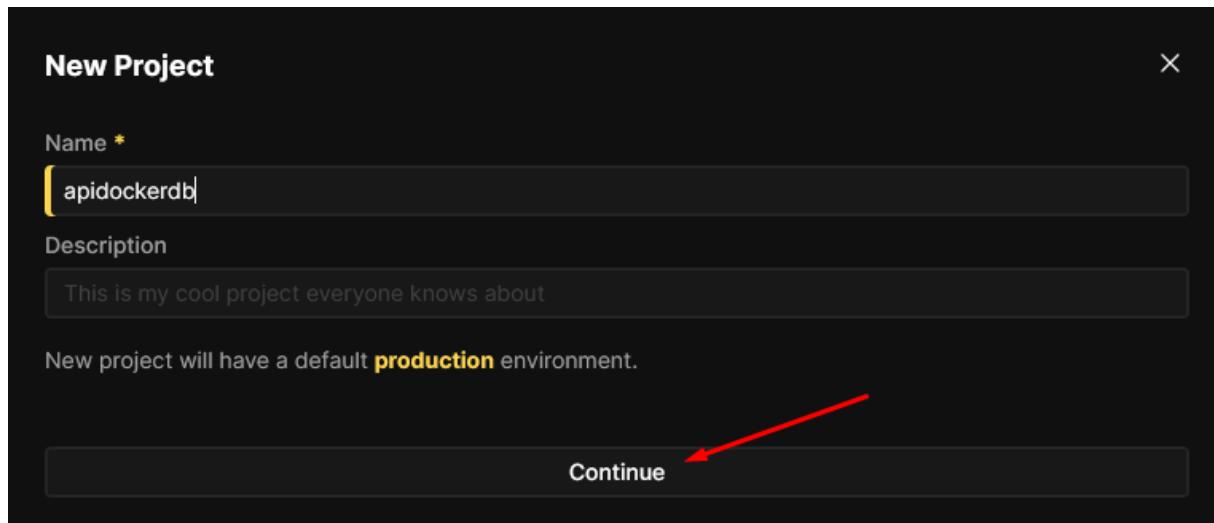


Figura 3: Nos aparecerá el siguiente modal e introduciremos un nombre



Figura 4: Pulsamos el botón de añadir recurso

**Coolify** v4.0.0-beta.454

**New Resource** Environment: production

Deploy resources, like Applications, Databases, Services...

Type / to search... Filter by category

**Applications** **AÑADIREMOS EL REPOSITORIO DE NUESTRA API**

**Git Based**

- Public Repository** You can deploy any kind of public repositories from the supported git providers.
- GitHub** **Private Repository (with GitHub App)** You can deploy public & private repositories through your GitHub Apps.
- Private Repository (with Deploy Key)** You can deploy private repositories with a deploy key.

**Docker Based**

- Dockerfile** You can deploy a simple Dockerfile, without Git.
- Docker Compose Empty** You can deploy complex application easily with Docker Compose, without Git.
- Docker Image** You can deploy an existing Docker Image from any Registry, without Git.

**Databases** **CREAREMOS UNA BASE DE DATOS**

- PostgreSQL** PostgreSQL is an object-relational database known for its robustness, advanced features, and strong standards compliance.
- MySQL** MySQL is an open-source relational database management system.
- MariaDB** MariaDB is a community-developed, commercially supported fork of the MySQL relational database management system, intended to remain free and open-source.

Figura 5: Deberemos crear 2 recursos: Uno que apunte al repositorio de nuestra API y una base de datos compatible con la de nuestra API. En este caso empezaremos por crear el recurso del repositorio público, aunque el orden es indiferente

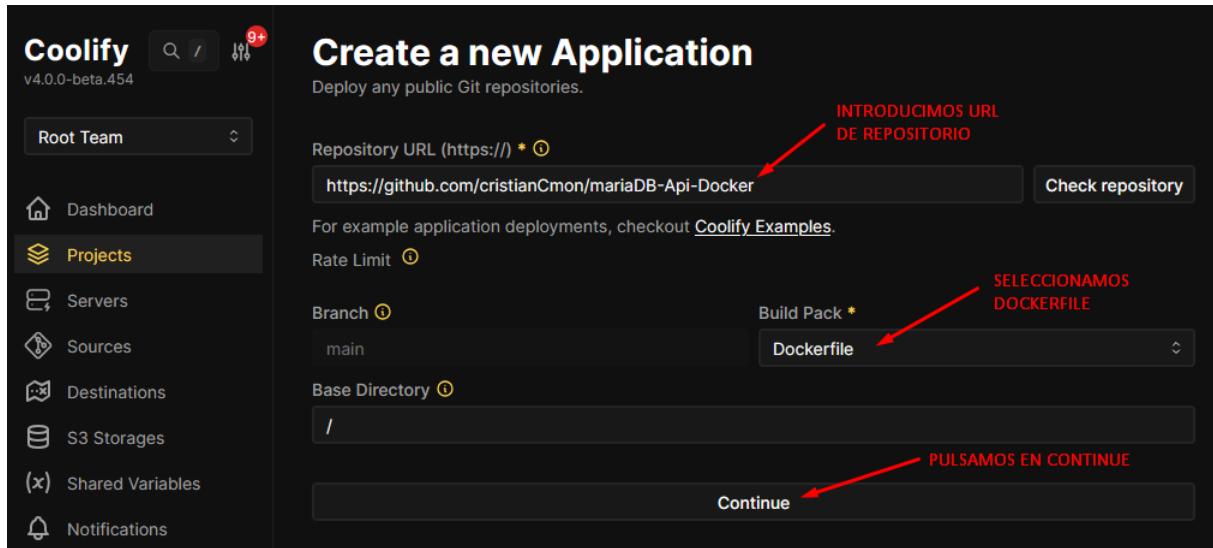


Figura 6: Configuramos siguiendo las indicaciones de esta captura

```
vbox@vbox-VirtualBox:~$ ifconfig
br-db1b4a5fd155: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.1.1 netmask 255.255.255.0 broadcast 10.0.1.255
        inet6 fd97:44f5:65cb::1 prefixlen 64 scopeid 0x0<global>
        inet6 fe80::42:d2ff:fe2b:2d1f prefixlen 64 scopeid 0x20<link>
ether 02:42:d2:2b:2d:1f txqueuelen 0 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.255.255.0 broadcast 10.0.0.255
        inet6 fe80::42:a5ff:fe39:87c prefixlen 64 scopeid 0x20<link>
ether 02:42:a5:39:08:7c txqueuelen 0 (Ethernet)
RX packets 4329 bytes 5789410 (5.7 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 3506 bytes 350366 (350.3 KB)
TX errors 0 dropped 21 overruns 0 carrier 0 collisions 0
                                         IP MÁQUINA VIRTUAL

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.115 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe86:2973 prefixlen 64 scopeid 0x20<link>
```

Figura 7: Paralelamente necesitamos conocer la ip de la máquina virtual

**General** Save

General configuration for your application.

Name \* cristian-cmon/maria-d-b--api--docker:main-bw8s0wscwo0kcs080w4kwkgk

Build Pack \* Dockerfile

Domains ⓘ http://kwkkg4cwko0gco08s80000k4.192.168.0.115.sslip.io  
ESTA DEBE SER LA IP DE LA MÁQUINA VIRTUAL

Direction \* ⓘ Allow www & non-www.

**Docker Registry** ⓘ

Docker Image ⓘ Empty means it won't push the image to a docker registry.

**Build**

Base Directory ⓘ / Dockerfile Location /Dockerfile

Custom Docker Options ⓘ --cap-add SYS\_ADMIN --device=/dev/fuse --security-opt apparmor:unconfined -

Use a Build Server? ⓘ

**Network**

Ports Exposes \* ⓘ 3000 Ports Mappings 3000:3000

ESTE PUERTO DEBE COINCIDIR CON EL DE LA API

Figura 8: Seguimos las indicaciones de la captura. Posteriormente volveremos a este punto a configurar las variables de entorno

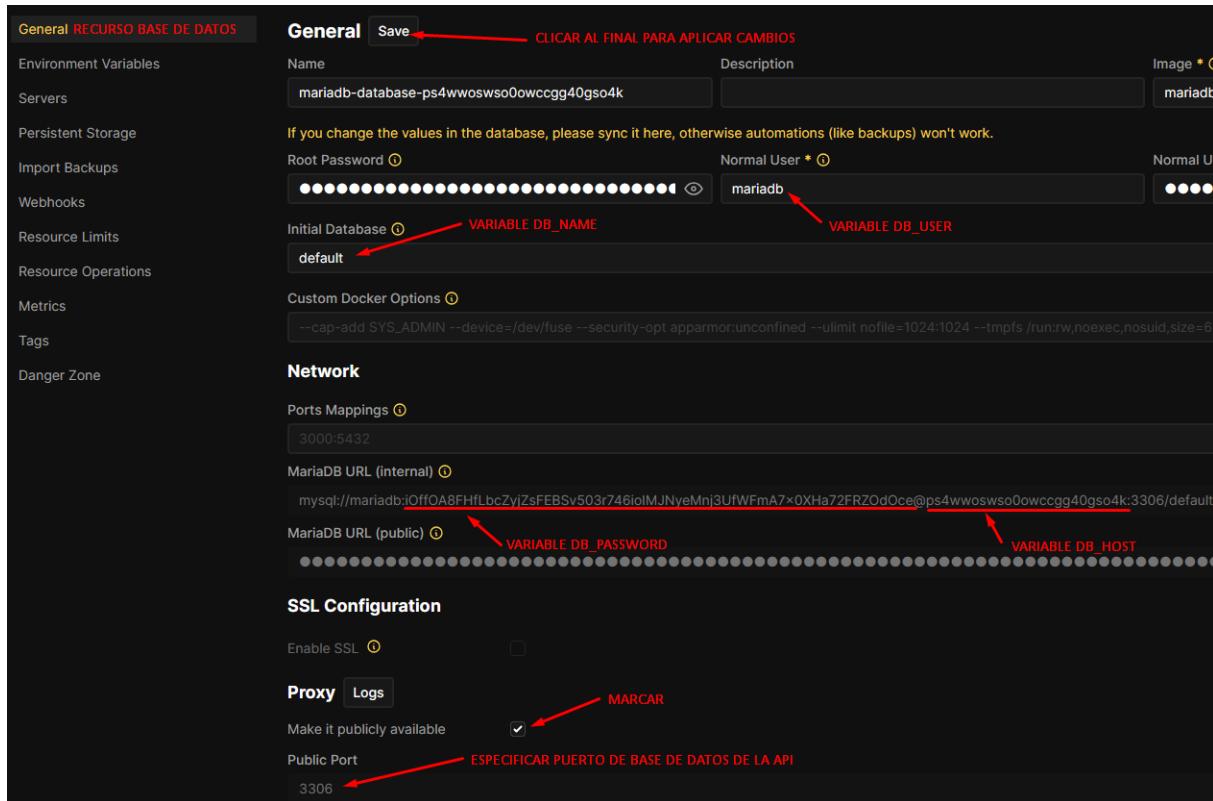


Figura 9: Creamos el recurso de base de datos y la configururamos según la captura

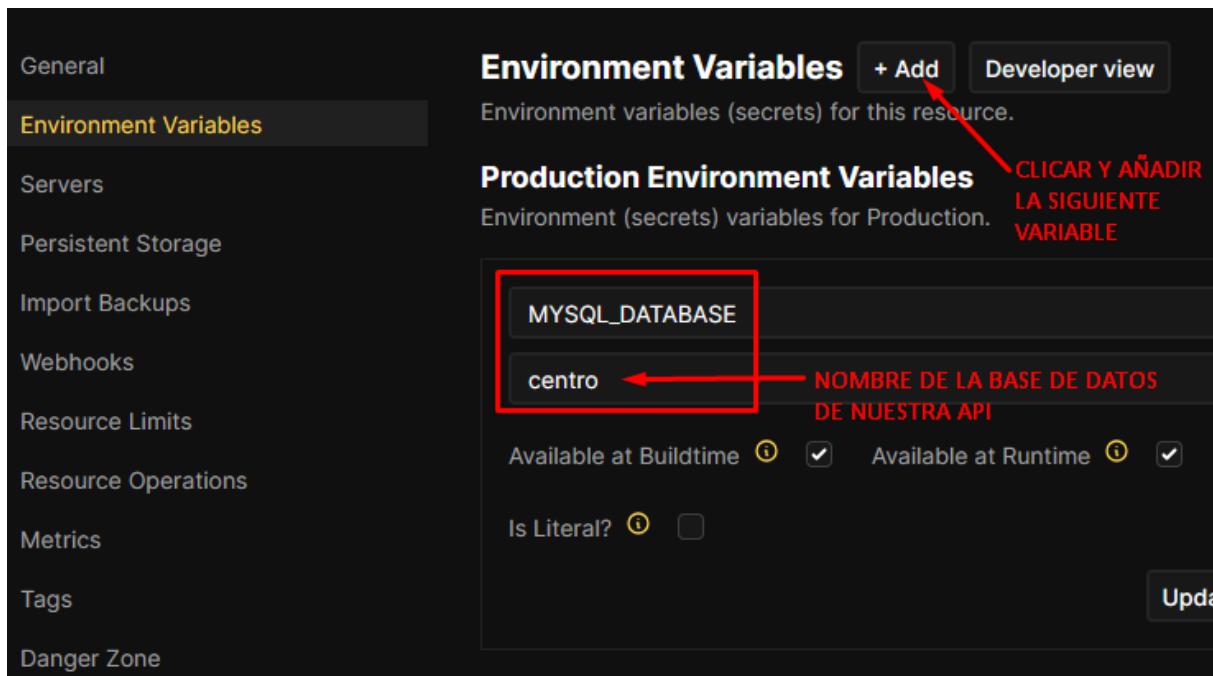


Figura 10: En el mismo recurso configururamos la siguiente variable

General

Advanced

**Environment Variables**

Persistent Storage

Git Source

Servers

Scheduled Tasks

Webhooks

Preview Deployments

Healthcheck

Rollback

Resource Limits

Resource Operations

Metrics

Tags

Danger Zone

**Environment Variables** **+ Add** **Developer view**

Environment variables (secrets) for this resource.

Sort alphabetically

Use Docker Build Secrets

SE AÑADEN DESDE AQUÍ

**Production Environment Variables**

Environment (secrets) variables for Production.

**DB\_HOST** → **ps4wwoswso0owccgg40gso4k**

Available at Buildtime  Available at Runtime  Is Multiline?  Is Literal?

**DB\_NAME** → **default**

Available at Buildtime  Available at Runtime  Is Multiline?  Is Literal?

**DB\_PASSWORD** → **iOffOA8FHfLbcZyjZsFEBSv503r746ioIMJ...**

Available at Buildtime  Available at Runtime  Is Multiline?  Is Literal?

**DB\_USER** → **mariadb**

Available at Buildtime  Available at Runtime  Is Multiline?  Is Literal?

**Update** **Lock** **Delete**

**Update** **Lock** **Delete**

**Update** **Lock** **Delete**

**Update** **Lock** **Delete**

Figura 11: Volvemos al recurso repositorio y añadimos las variables de entorno de la base de datos tal como se muestra en la captura

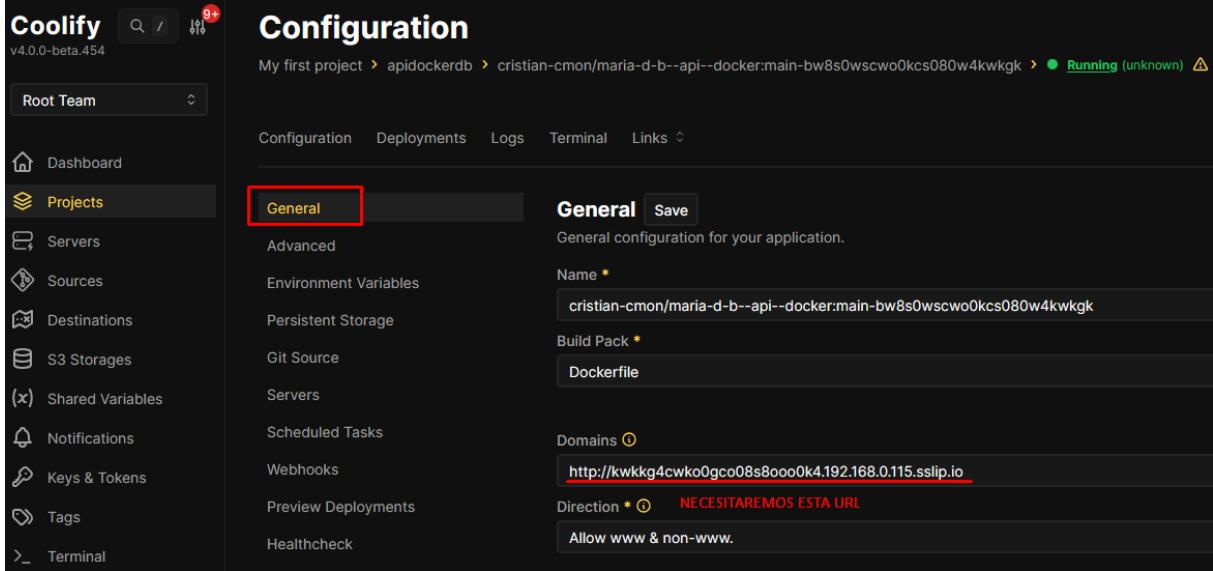
Ahora sólo falta desplegar ambos recursos y comprobar si se pueden hacer peticiones a la API. Para ello introduciremos la url de la Figura 8 (en la que insertamos la ip de la máquina virtual). El resultado es el siguiente:

The screenshot shows a browser window displaying a JSON response. The URL in the address bar is highlighted with a red box and labeled "URL COOLIFY". To the right of the URL, another red box highlights the word "usuarios" and is labeled "ENDPOINT EN EL QUE TENGAMOS REGISTROS ALMACENADOS".

	id	nombre	apellidos	sexo	edad	telefono
0:	1	"Pepe"	"Fernandez"	"Masculino"	"30"	"886112234"
1:	2	"Estefanía"	"Pérez"	"Femenino"	"26"	"886546712"
2:	3	"Mario"	"López"	"Masculino"	"46"	"888712543"

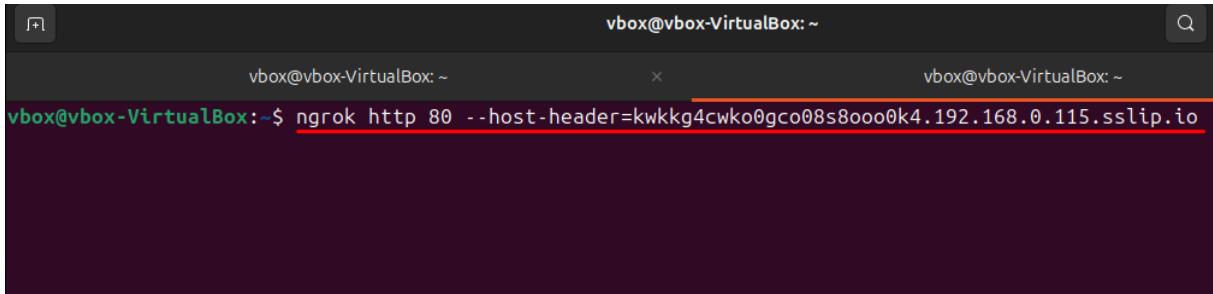
Figura 12: Si todo ha ido bien deberíamos visualizar correctamente la llamada a la API

### 4.3. Ngrok expuesto en Internet



The screenshot shows the Coolify web interface. On the left, there's a sidebar with 'Root Team' selected. Under 'Projects', 'General' is highlighted with a red box. The main area is titled 'Configuration' and shows a project path: 'My first project > apidockerdb > cristian-cmon/maria-d-b--api--docker:main-bw8s0wscwo0kcs080w4kwkgk'. It indicates the application is 'Running (unknown)'. Below this, tabs for 'Configuration', 'Deployments', 'Logs', 'Terminal', and 'Links' are visible. The 'General' tab is active. It contains fields for 'Name' (set to 'cristian-cmon/maria-d-b--api--docker:main-bw8s0wscwo0kcs080w4kwkgk'), 'Build Pack' (set to 'Dockerfile'), and 'Domains' (set to 'http://kwkk4cwko0gco08s8000k4.192.168.0.115.sslip.io'). A note says 'NECESITAREMOS ESTA URL'.

Figura 13: Desde Coolify copiamos la url del dominio enlazado al repositorio



The screenshot shows a terminal window on a VirtualBox machine. The prompt is 'vbox@vbox-VirtualBox: ~'. The user has typed the command 'ngrok http 80 --host-header=kwkk4cwko0gco08s8000k4.192.168.0.115.sslip.io'. The command is highlighted with a red box.

Figura 14: Desde un terminal levantamos Ngrok en el puerto 80 con la url anterior pasada al parámetro *-host-header*

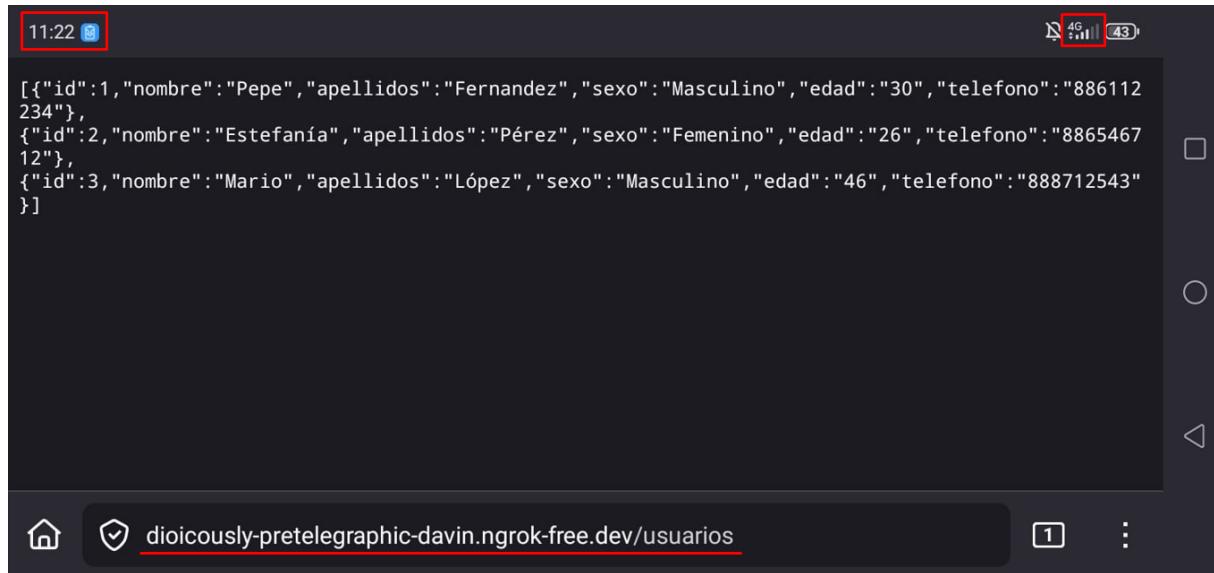


Figura 15: Entramos a la url proporcionada por Ngrok desde el teléfono móvil

```
ngrok
[ One gateway for every AI model. Available in early access *now*: https://ngrok.com/r/ai

Session Status          online
Account                 Cristian (Plan: Free)
Update                  update available (version 3.35.0, Ctrl-U to update)
Version                3.34.1
Region                 Europe (eu)
Latency                45ms
Web Interface          http://127.0.0.1:4040
Forwarding             https://dioicously-pretelegraphic-davin.ngrok-free.dev -> http://localhost:80

Connections            ttl     opn     rt1     rt5     p50     p90
                      0       1      0.00    0.00    0.00    0.00

HTTP Requests
-----
11:22:03.250 CET GET /usuarios                         200 OK ←
```

Figura 16: Conexiones satisfactorias en captura actual y anterior

## 4.4. Coolify (Webhook)

Los Webhooks nos permiten, entre otras cosas, desplegar automáticamente una nueva versión de la aplicación cada vez que enviamos cambios (push) a una rama específica de nuestro repositorio remoto.

La configuración del lado de Coolify quedaría así:

The screenshot shows the Coolify configuration interface for a project named 'My first project'. The 'Webhooks' section is highlighted with a red box. Under 'Manual Git Webhooks', there is a GitHub entry with the URL `http://109.227.134.36:8000/webhooks/source/github/events/manual`. To its right, under 'GitHub Webhook Secret', is a placeholder field containing the value `27a728f38f358e83659b92fb131d220c`, also highlighted with a red box. A red arrow points from the text 'NECESARIO GUARDAR CAMBIOS' at the bottom right towards the 'Save' button, which is also highlighted with a red box. The 'Webhooks' tab in the sidebar is also highlighted with a red box.

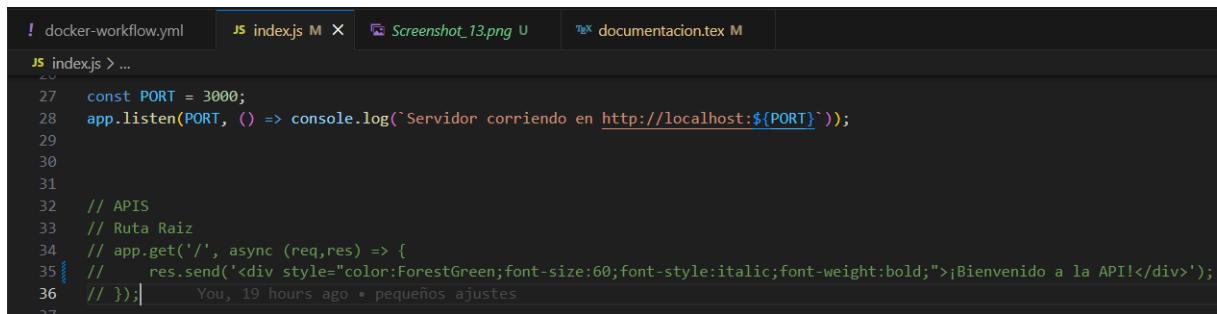
Figura 17: Necesitaremos estos datos para enlazarlos con GitHub

La configuración del lado de nuestro proyecto de GitHub quedaría así:

Figura 18: Importante seleccionar el formato de datos json

Figura 19: Una vez añadido el Webhook se hará un ping de comprobación

Ahora comprobaremos el correcto funcionamiento del Webhook recién creado realizando un push a la rama principal de nuestro repositorio:



```
! docker-workflow.yml   JS index.js M X  Screenshot_13.png U  documentacion.tex M
JS index.js > ...
27 const PORT = 3000;
28 app.listen(PORT, () => console.log(`Servidor corriendo en http://localhost:${PORT}`));
29
30
31
32 // APIs
33 // Ruta Raiz
34 // app.get('/', async (req,res) => {
35 //   res.send('<div style="color:ForestGreen;font-size:60;font-style:italic;font-weight:bold;">¡Bienvenido a la API!</div>');
36 // });
37 You, 19 hours ago * pequeños ajustes
```

Figura 20: El endpoint raíz de nuestra API...

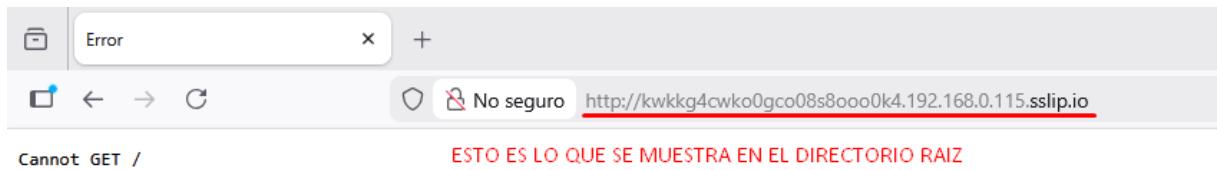
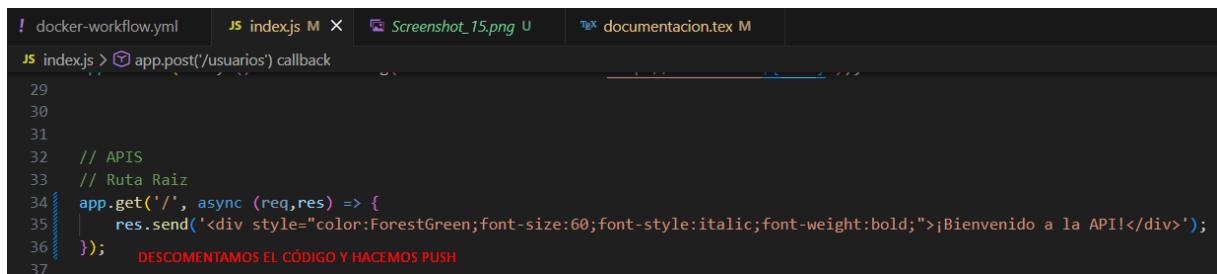


Figura 21: ...muestra lo siguiente



```
! docker-workflow.yml   JS index.js M X  Screenshot_15.png U  documentacion.tex M
JS index.js > ⓘ app.post('/usuarios') callback
29
30
31
32 // APIs
33 // Ruta Raiz
34 app.get('/', async (req,res) => {
35   res.send('<div style="color:ForestGreen;font-size:60;font-style:italic;font-weight:bold;">¡Bienvenido a la API!</div>');
36 });
37 DESCOMENTAMOS EL CÓDIGO Y HACEMOS PUSH
```

Figura 22: Si lo modificamos...

## Webhooks / Manage webhook

Settings	Recent Deliveries		
✓	e7126bf0-f850-11f0-9b9d-3cb71d0dba3a push	SE REALIZÓ CORRECTAMENTE	2026-01-23 12:44:41 ...
✓	0db0e27a-f84f-11f0-8e97-5681cfb28fe3 ping		2026-01-23 12:31:27 ...

Figura 23: ...y pusheamos el cambio...



Figura 24: ...observaremos que el cambio se ha realizado automáticamente