

Documentación Técnica: Instalación y uso de Coolify con Ngrok

Autor: Pedro José Meixús Belsol

23 de enero de 2026

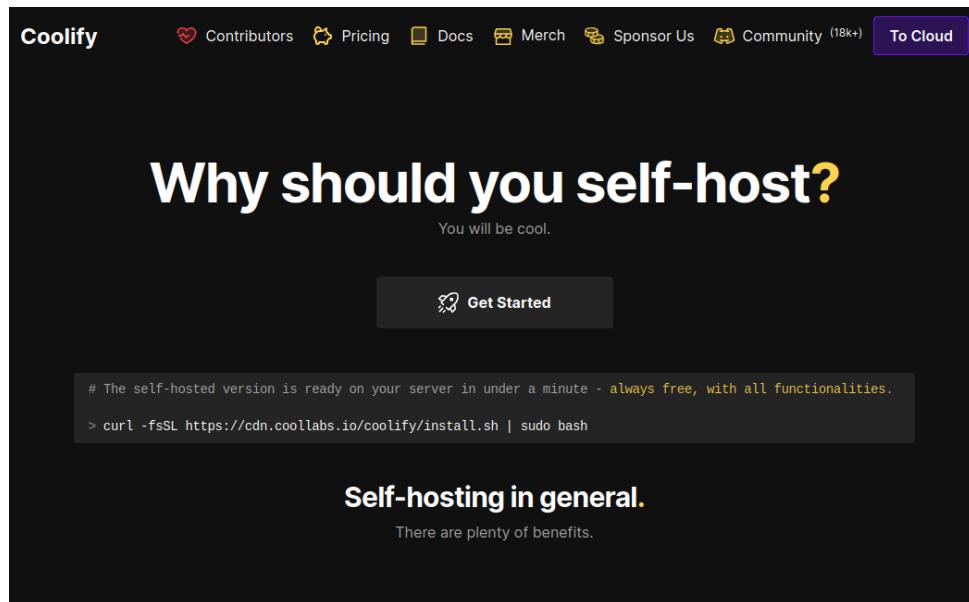
Índice

| | |
|------------------------------|---|
| 1. Instalación y preparación | 2 |
| 2. Configuración Coolify | 3 |
| 3. Pruebas de conexión | 6 |
| 4. Webhook | 8 |

1. Instalación y preparación

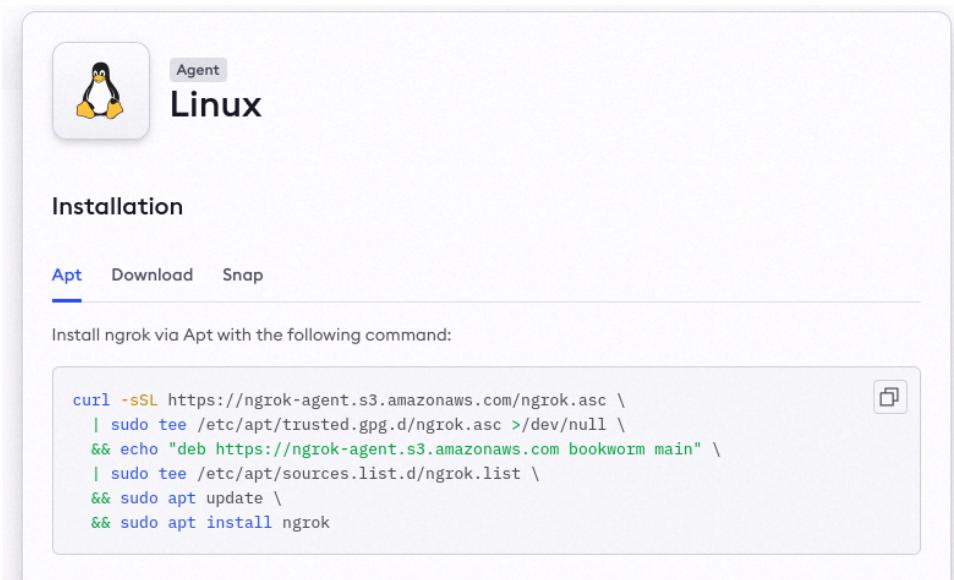
Para empezar, instalaremos una máquina virtual Linux (Ubuntu).

Ya con ella instalada, vamos a descargar Coolify, vamos a su página y utilizamos el comando que nos indican para la instalación:



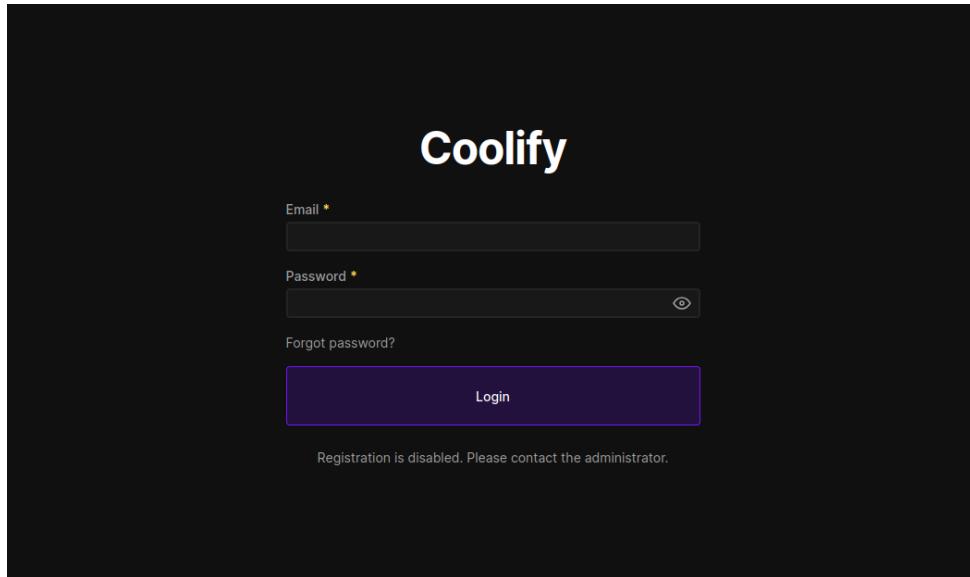
Se instalará y una vez termine accederemos a la IP que nos indica y crearemos una cuenta de Coolify.

Hecho esto vamos a instalar Ngrok, igual que con Coolify, accederemos a su página y usaremos el comando que nos indican:



2. Configuración Coolify

Una vez hecho esto, accederemos al 'localhost:8000' desde nuestro host, donde deberá aparecer la interfaz de Coolify, en esta podremos logearnos.



Una vez logeados crearemos nuestro propio proyecto. En el crearemos una nueva base de datos (en nuestro caso será Mongo):

New Resource Environment: production
Deploy resources, like Applications, Databases, Services...
Type / to search...

Databases

- PostgreSQL**
PostgreSQL is an object-relational database known for its robustness, advanced features, and strong standards compliance.
- Redis**
Redis is a source-available, in-memory storage, used as a distributed, in-memory key-value database, cache and message broker, with optional...
- MongoDB**
MongoDB is a source-available, cross-platform, document-oriented database program.

Podremos acceder a su configuración y veremos que nos proporciona una URI. Le daremos a 'Deploy' y si arranca correctamente mostrará 'Running'

Ahora con la BBDD preparada, creamos la API, que obtiene desde nuestro repositorio de github:

Para una futura comprobación le añadire algo para mostrar, por ejemplo la palabra 'contenido'

```
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
  res.send('contenido')
})
```

Una vez indicado el repositorio y aceptado, se desplegará y cuando termine nos dejará en una ventana como esta, donde debemos indicar el dockerfile, puerto, etc.

The screenshot shows the Coolify application configuration interface. On the left is a sidebar with navigation links: Root Team, Dashboard, Projects (selected), Servers, Sources, Destinations, S3 Storages, Shared Variables, Notifications, Keys & Tokens, Tags, Terminal, Profile, Teams, and Settings. Below these are links for Upgrade, Sponsor us, and Feedback.

The main area is titled "General" and contains the following fields:

- Name**: st4rill/-mongo-docker:main-two844s8oggwgck400c80g
- Description**: (empty)
- Build Pack**: Dockerfile
- Domains**: http://l0kk0s4okw4gssc8k880cog192.168.0.74.sslip.io
- Direction**: Allow www & non-www. (Set Direction button)
- Docker Registry**: Docker Image: Empty means it won't push the image to a docker registry. Docker Image Tag: Empty means only push commit sha tag.
- Build**:
 - Base Directory: /
 - Dockerfile Location: /dockerfile
 - Docker Build Stage Target: (empty)
 - Custom Docker Options: --cap-add SYS_ADMIN --device=/dev/fuse --security-opt apparmor:unconfined --ulimit nofile=1024:1024 --tmpfs /run:rw,noexec,nosuid,size=65536k --hostname=myapp
 - Use a Build Server? (checkbox)
- Network**:
 - Ports Exposes: 3000
 - Ports Mappings: 3000:3000
 - Network Aliases: (empty)

3. Pruebas de conexión

Con la IP que nos genera en el apartado 'Domains' podemos cambiar la dirección IP que sale ahí por la de nuestra máquina y redesplegar la API, así podremos acceder desde nuestra máquina virtual.

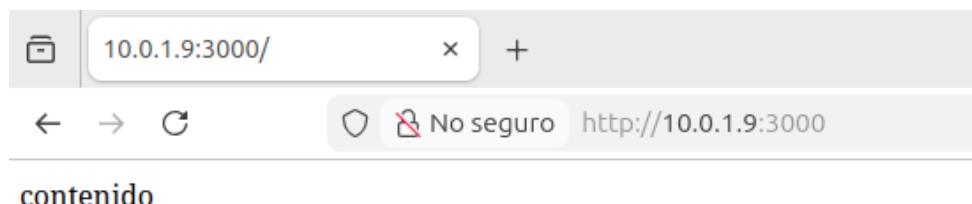


Ahora que tenemos la conexión funcional, vamos a buscar la IP del contenedor Docker donde esté nuestra API, para eso usaremos el comando 'inspect' junto con los parámetros necesarios y el ID del contenedor.

```
ubuntu@ubuntu-VirtualBox:~$ sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' abd681a4d4ae
10.0.1.9
```

The terminal window shows the command "sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' abd681a4d4ae" being run, followed by the output "10.0.1.9".

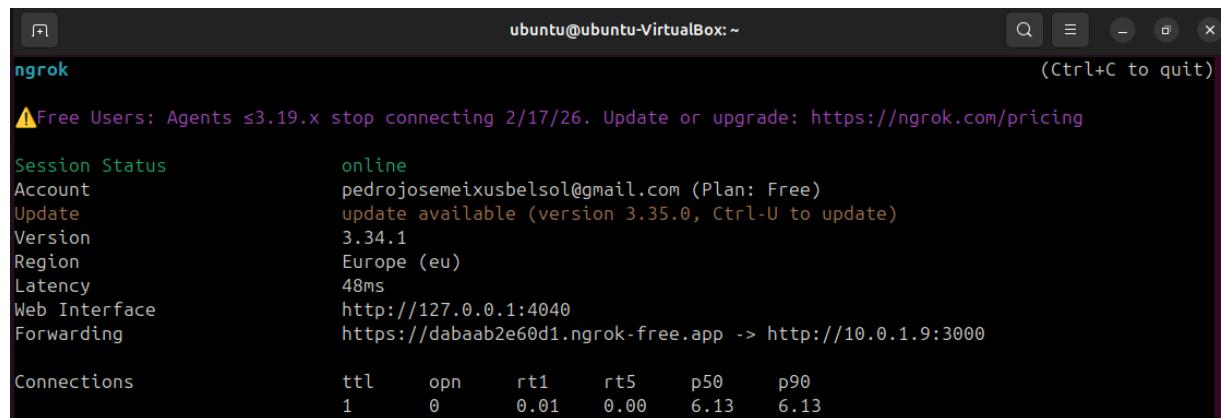
Si introducimos esa IP que nos ha dado junto con el puerto usado (3000) nos deberá llevar al mismo lugar.



Si lo anterior ha funcionado, entonces podemos utilizar ngrok para exponer nuestra API, con el siguiente comando:

```
ubuntu@ubuntu-VirtualBox:~$ sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' abd681a4d4ae
10.0.1.9
ubuntu@ubuntu-VirtualBox:~$ ngrok http 10.0.1.9:3000
```

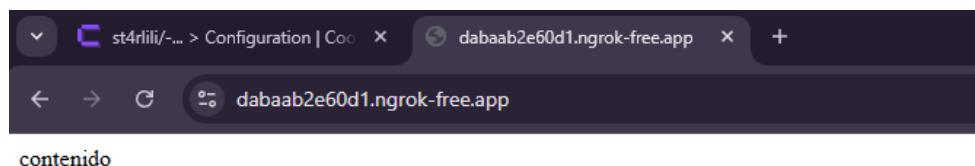
Nos llevará a esta ventana (la cual dejaremos abierta), que es el túnel que nos da la URL (la del apartado 'Forwarding') con la que podremos acceder desde nuestra máquina al contenido de la API.



The screenshot shows a terminal window titled "ubuntu@ubuntu-VirtualBox:~". It displays the output of the "ngrok" command. The output includes a warning about free users needing to update, session status (online), account information (pedrojosemeixusbel@gmail.com, Plan: Free), version (3.34.1), region (Europe (eu)), latency (48ms), web interface (http://127.0.0.1:4040), and forwarding (https://dabaab2e60d1.ngrok-free.app -> http://10.0.1.9:3000). It also shows connection statistics (ttl, opn, rt1, rt5, p50, p90).

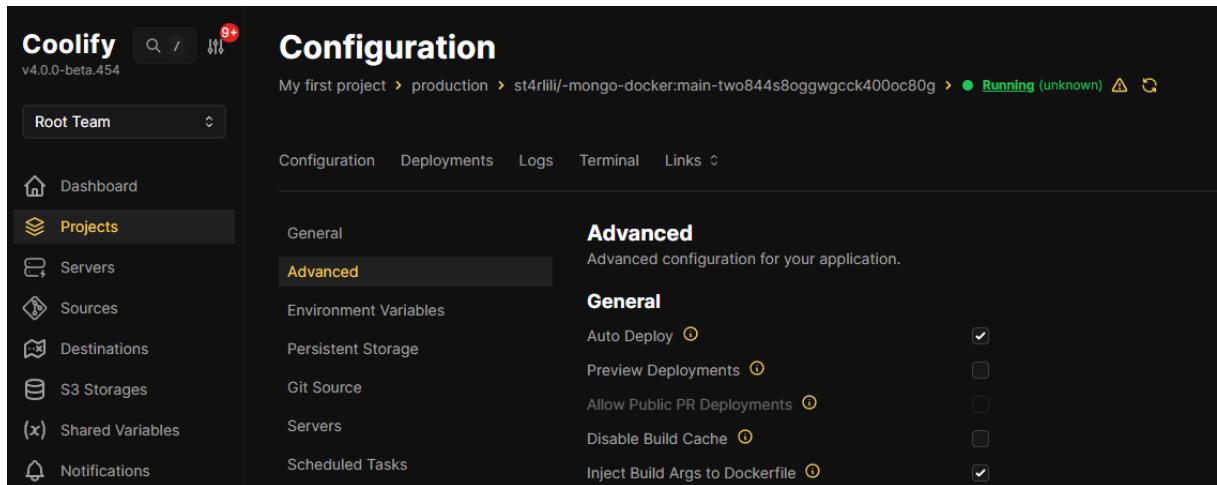
```
ubuntu@ubuntu-VirtualBox:~$ ngrok http 10.0.1.9:3000
⚠️Free Users: Agents ≤3.19.x stop connecting 2/17/26. Update or upgrade: https://ngrok.com/pricing
Session Status      online
Account            pedrojosemeixusbel@gmail.com (Plan: Free)
Update             update available (version 3.35.0, Ctrl-U to update)
Version            3.34.1
Region             Europe (eu)
Latency            48ms
Web Interface     http://127.0.0.1:4040
Forwarding         https://dabaab2e60d1.ngrok-free.app -> http://10.0.1.9:3000
Connections        ttl     opn      rt1      rt5      p50      p90
                   1       0       0.01    0.00    6.13    6.13
```

Accediendo a esa URL nos deberá aparecer el contenido correspondiente, al igual que en las anteriores pruebas.



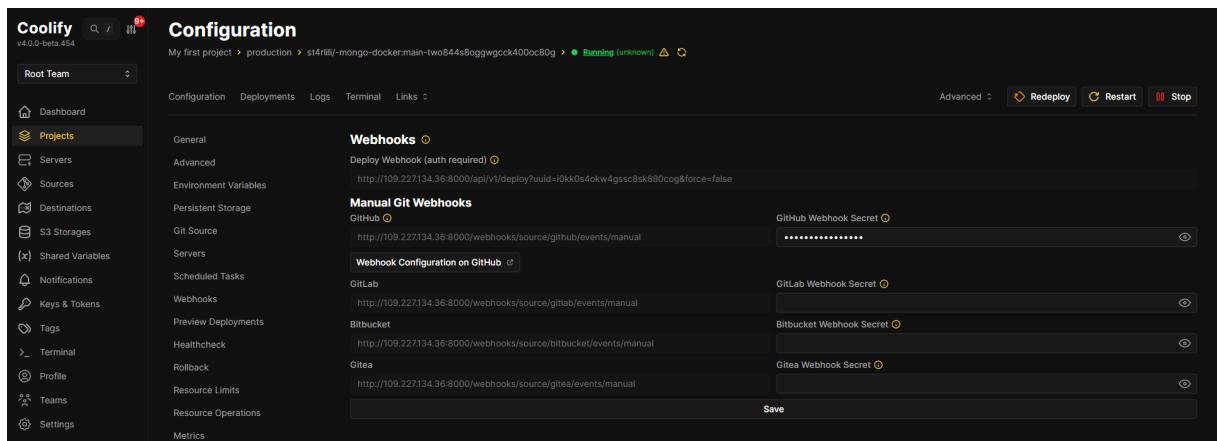
4. Webhook

El último paso será configurar el webhook. Primero tendremos que ir a la configuración de nuestra API y activar el 'Auto Deploy'.



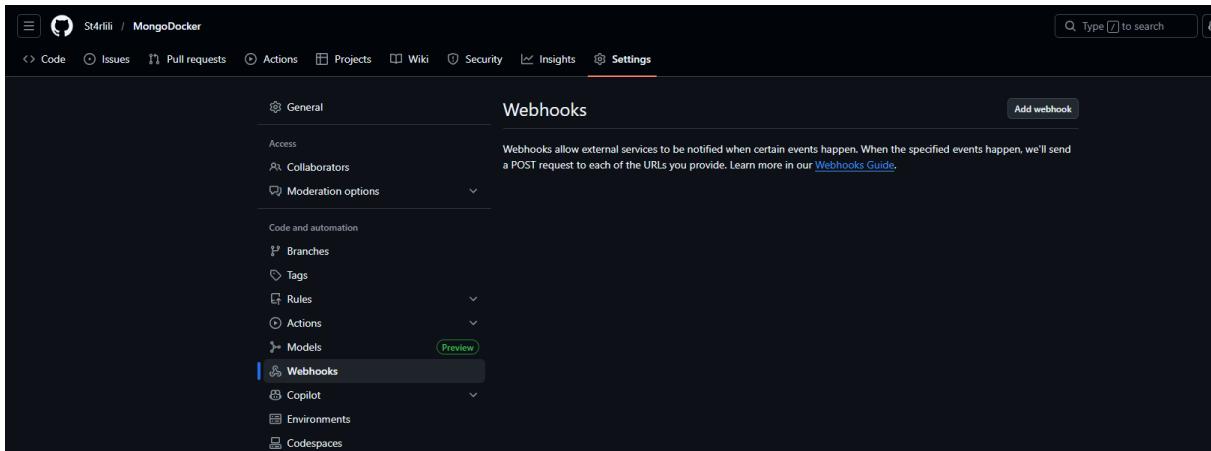
The screenshot shows the Coolify configuration interface for a project named 'st4rlili/-mongo-docker:main-two844s8oggwgcc400oc80g'. The 'Advanced' tab is selected under the 'General' section. The 'Auto Deploy' option is checked. Other options like 'Preview Deployments', 'Allow Public PR Deployments', 'Disable Build Cache', and 'Inject Build Args to Dockerfile' are also listed with their respective checkboxes.

En el apartado de Webhook de la configuración de la API, añadiremos un 'secret' que será la 'contraseña' de nuestro webhook.

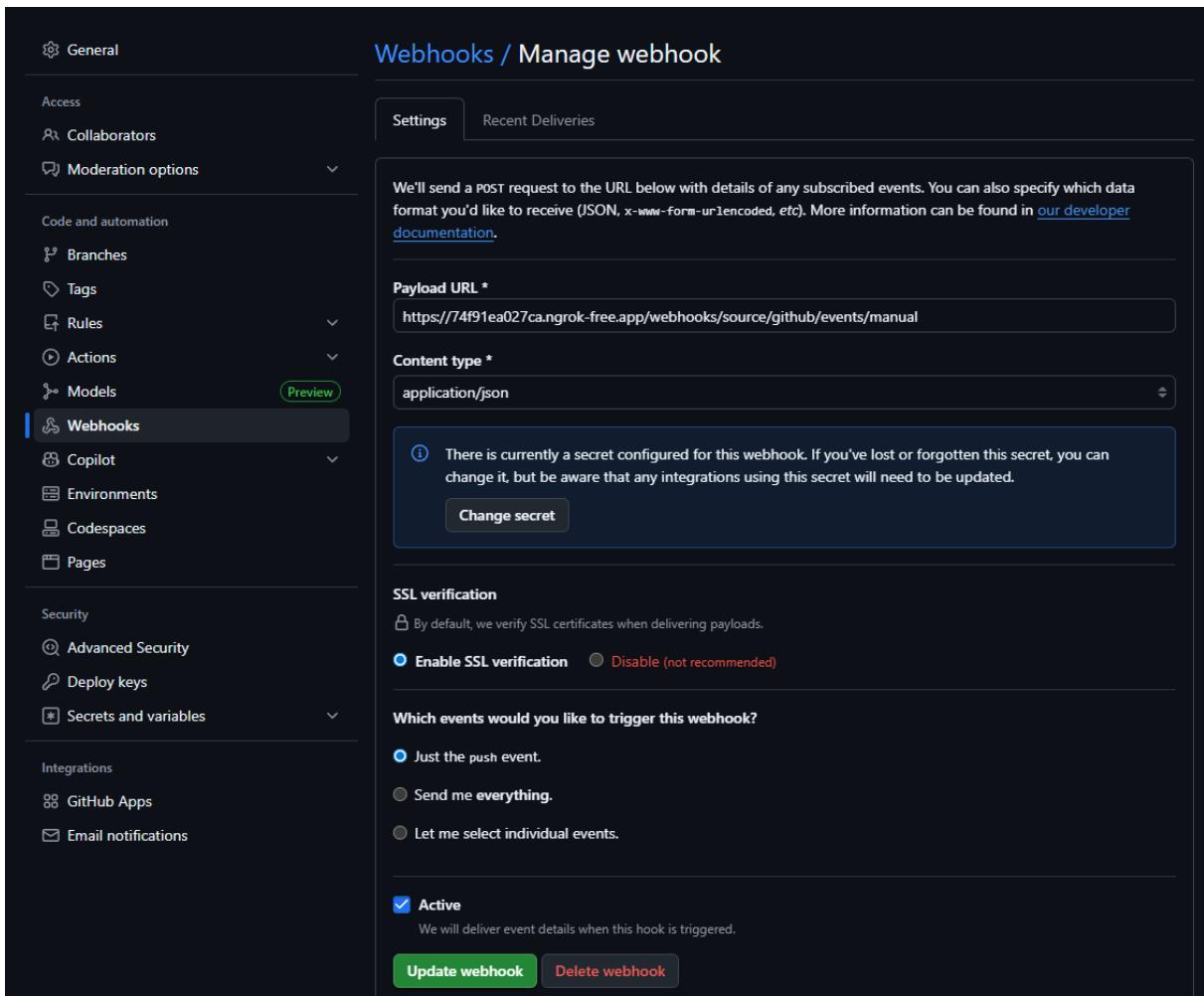


The screenshot shows the 'Webhooks' section of the Coolify configuration. It lists four entries: GitHub, GitLab, Bitbucket, and Gitea, each with a corresponding 'Webhook Secret' input field. The GitHub secret is filled with a series of asterisks. Buttons for 'Redeploy', 'Restart', and 'Stop' are visible at the top right of the configuration panel.

Ahora iremos al repositorio de github, al apartado 'settings' y en 'Webhooks' añadiremos uno nuevo.



Aquí solo deberemos introducir el 'Payload URL' que será la URL que nos da ngrok junto con la ruta de los webhooks de coolify, además de el secret que indicamos en coolify. El resto quedará predeterminado.



Con eso hecho el webhook estaría hecho y debería funcionar correctamente. En mi

caso tuve un error y coolify no recibía el mensaje del webhook, estuve probando por si podia ser el puerto, la ruta de ngrok o algo similar, pero no hago que funcione.

The screenshot shows a web-based interface titled "Webhooks / Manage webhook". At the top, there are two tabs: "Settings" and "Recent Deliveries", with "Recent Deliveries" being the active tab. Below the tabs, a single delivery record is listed. The record includes a warning icon (red triangle), a small icon of a cube, a unique identifier "6904dd9c-f86e-11f0-91c9-d534604cacce", a "ping" status indicator, and a "redelivery" status indicator. To the right of the delivery details, the timestamp "2026-01-23 17:39:04" and a three-dot ellipsis menu are visible.