



Francisco Daines [Follow](#)

May 11, 2021 · 5 min read

Utilizando Github Actions para desplegar un servicio sobre Heroku

Este artículo tiene como objetivo desplegar un servicio (desarrollado en Go) sobre Heroku, utilizando un flujo configurado mediante Github Actions.

Analizando nuestro servicio

El servicio que utilizaremos para este tutorial es bastante sencillo, su responsabilidad es retornar datos de países en Latinoamérica, para esto los datos se obtienen desde la API del banco mundial ([referencia](#)) y se realizan los filtros y transformaciones necesarios para presentar los datos de forma simplificada a las aplicaciones cliente.



Servicio de información de países.

En términos de estructura, nuestro servicio distribuye sus archivos en los siguientes directorios,

```
github-actions-example
model/
repository/
translator/
usecase/
.gitignore
go.mod
go.sum
LICENSE
main.go
README
```

Preparando el despliegue del servicio

Heroku nos ofrece varias formas para desplegar nuestro servicio en esa plataforma, pero para efectos de este tutorial vamos a optar por crear un pipeline que se ejecute cada vez que hacemos push a nuestro repositorio en *Github*.



Flujo de despliegue de un servicio desde Github hacia Heroku.

El proceso representado en la figura anterior es bastante sencillo de comprender, inicia cuando un desarrollador envía sus cambios (push) al repositorio en *Github*, lo que inicia la ejecución de uno o varios workflows configurados en *Github Actions*. Uno de esos flujos contiene un paso que se conectará a nuestra cuenta en *Heroku* y desplegará nuestro servicio, el

cual quedará publicado para que otras aplicaciones puedan consumirlo.

Para efectos de simplicidad en este artículo vamos a considerar que nuestro workflow de *Github Actions* sólo ejecuta el despliegue hacia *Heroku*, pero en una aplicación real debiésemos considerar otros pasos como:

- Pruebas unitarias, de integración, mutación, carga, PenTests, otros.
- Análisis estático de código.
- Análisis de seguridad (módulos/packages/librerías de las cuales dependemos, entre otros).
- Otras verificaciones o pruebas que estimemos necesarias.

Obtener tu token desde Heroku

Para poder desplegar sobre Heroku nuestro servicio, de forma automatizada, necesitamos contar con una API Key que nos permita interactuar de forma remota con nuestra cuenta de Heroku. Para poder visualizar tu API Key debes ir a la sección *API Key* de la opción *Account Settings* en tu menú de usuario.

API Key

641a8257-f01a-4c8a-b218-f88d28bc204e

Hide

Regenerate API Key...

API Key de tu cuenta en Heroku

- **Nota:** Nunca compartas tu API Key con otras personas, para efectos de este tutorial, la API Key de la imagen fue generada únicamente para efectos del ejemplo, por lo que fue eliminada inmediatamente.

Configurando nuestro Workflow en Github Actions

Para configurar un workflow sobre *Git Hub Actions*, es necesario crear un archivo *YAML* con la especificación de dicho flujo dentro de nuestro repositorio, en el directorio *.github/workflows*. Para este ejemplo, vamos a crear el archivo *.github/workflows/workflow.yml*.

Para desplegar nuestro servicio a Heroku podemos utilizar el template Heroku Deploy (<https://github.com/AkhileshNS/heroku-deploy>) que nos permite realizar el despliegue a partir de 3 datos:

- **heroku_api_key:** API_KEY de nuestra cuenta en Heroku.
- **heroku_app_name:** Nombre de la aplicación en Heroku que se actualizará con nuestro servicio. Una vez que el servicio esté desplegado, quedará publicado en [https://\[heroku_app_name\].herokuapp.com/](https://[heroku_app_name].herokuapp.com/)
- **heroku_email:** Email al que se notificará el resultado del despliegue

Dado lo anterior, el contenido de nuestro archivo *workflow.yml* debería ser lo siguiente:

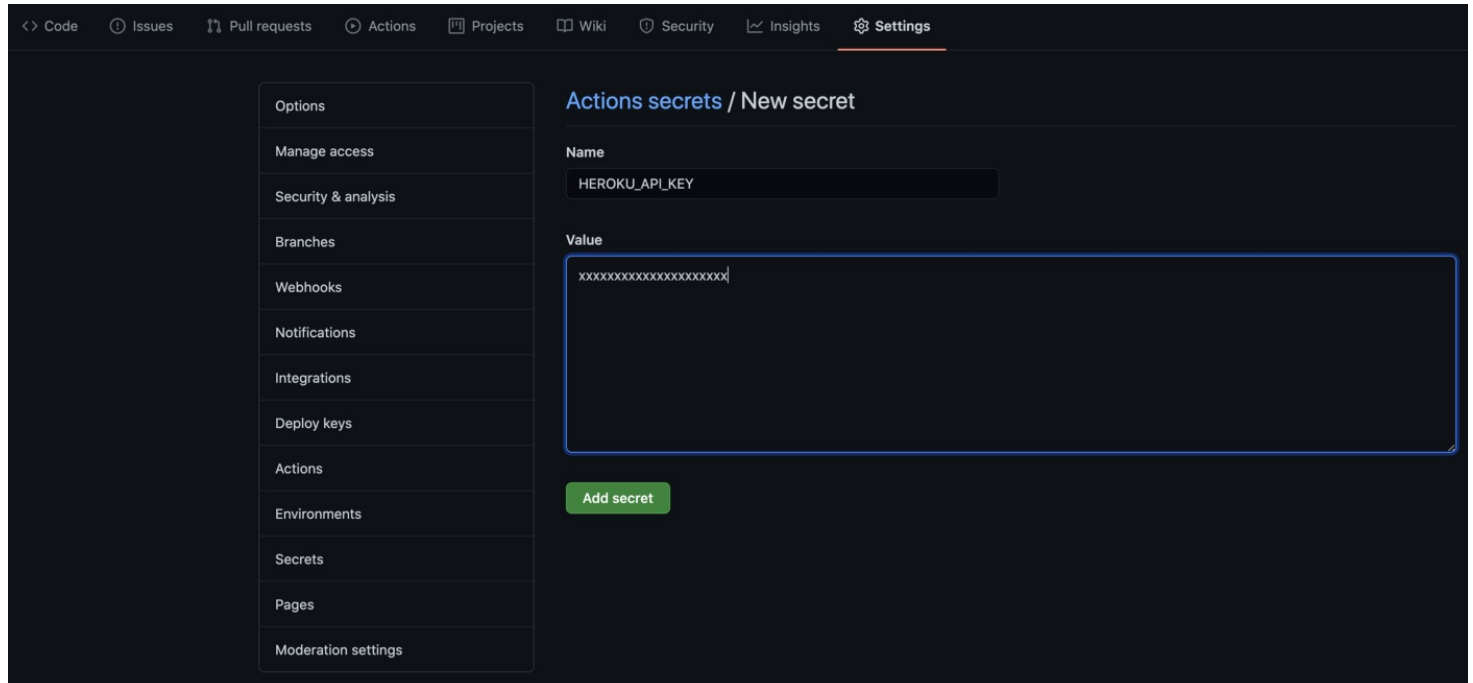
```
name: Heroku Deployment
on: push

jobs:
  heroku-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Deploy to Heroku
        uses: akhileshns/heroku-deploy@v3.12.12
        with:
          heroku_api_key: ${ secrets.HEROKU_API_KEY }
          heroku_app_name: ${ secrets.HEROKU_APP_NAME }
```

```
heroku_email: ${ secrets.HEROKU_EMAIL }}
```

Como puedes notar, podríamos asignar directamente el valor de estas variables en el archivo YAML, pero dado que estos valores son información sensible que puede ser mal utilizada, lo recomendable es obtenerlos mediante secretos, que se pueden configurar a nivel de repositorio en GitHub.

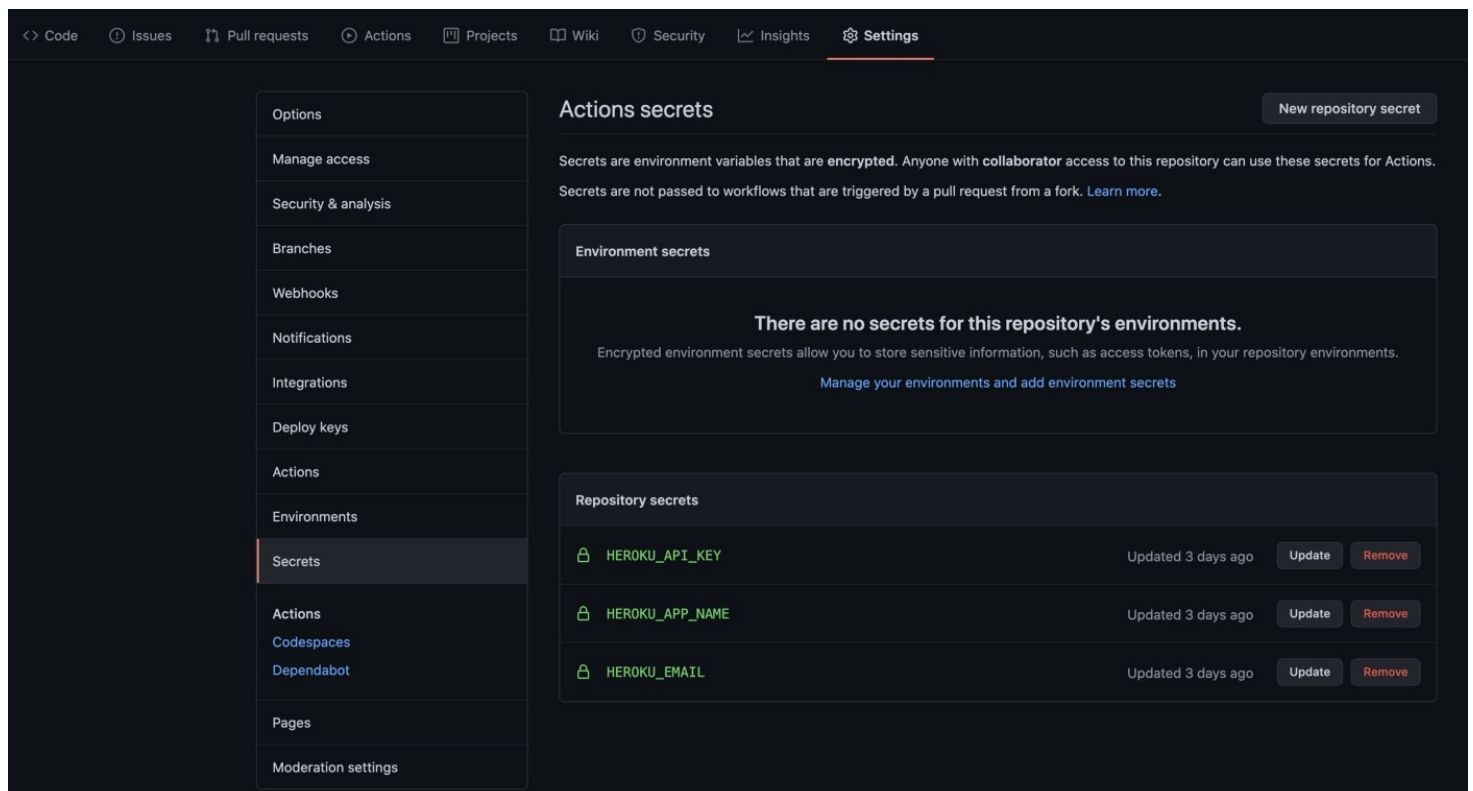
Los secretos los podemos configurar en la sección *Settings/secrets*. La siguiente figura es un ejemplo de cómo podemos agregar el secreto *HEROKU_API_KEY*.



The screenshot shows the GitHub 'Actions secrets / New secret' page. On the left is a sidebar with navigation links: Options, Manage access, Security & analysis, Branches, Webhooks, Notifications, Integrations, Deploy keys, Actions, Environments, Secrets, Pages, and Moderation settings. The 'Secrets' link is highlighted. The main content area is titled 'Actions secrets / New secret'. It contains a 'Name' field with the value 'HEROKU_API_KEY' and a 'Value' text area with a placeholder 'xxxxxxxxxxxxxxxxxxxxxx'. Below the text area is a green 'Add secret' button.

La API_KEY de Heroku debemos ingresarla como secreto a nuestro repositorio.

Una vez que configurar los 3 secretos requeridos por *heroku-deploy*, la sección *Secrets* debería ser similar a la siguiente figura.

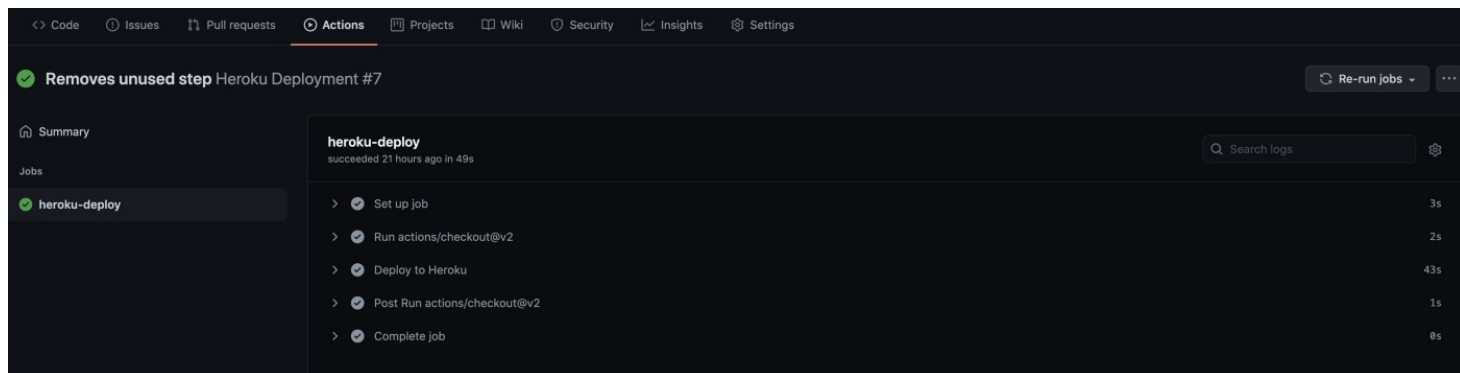


The screenshot shows the GitHub 'Actions secrets' page. The left sidebar is the same as in the previous image, with 'Secrets' highlighted. The main content area is titled 'Actions secrets' and has a 'New repository secret' button in the top right. Below the title is a description: 'Secrets are environment variables that are encrypted. Anyone with collaborator access to this repository can use these secrets for Actions. Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.' There are two sections: 'Environment secrets' which says 'There are no secrets for this repository's environments.' and 'Repository secrets' which lists three secrets: 'HEROKU_API_KEY', 'HEROKU_APP_NAME', and 'HEROKU_EMAIL'. Each secret is shown with a lock icon, its name, the update time 'Updated 3 days ago', and 'Update' and 'Remove' buttons.

Secretos configurados a nivel de repositorio

Pipeline de despliegue en Github

Ahora que configuramos el workflow de despliegue en Github Actions, cada vez que hagamos push a la rama master, se ejecutará el flujo de forma automática, y tendremos un resultado similar a la siguiente figura.

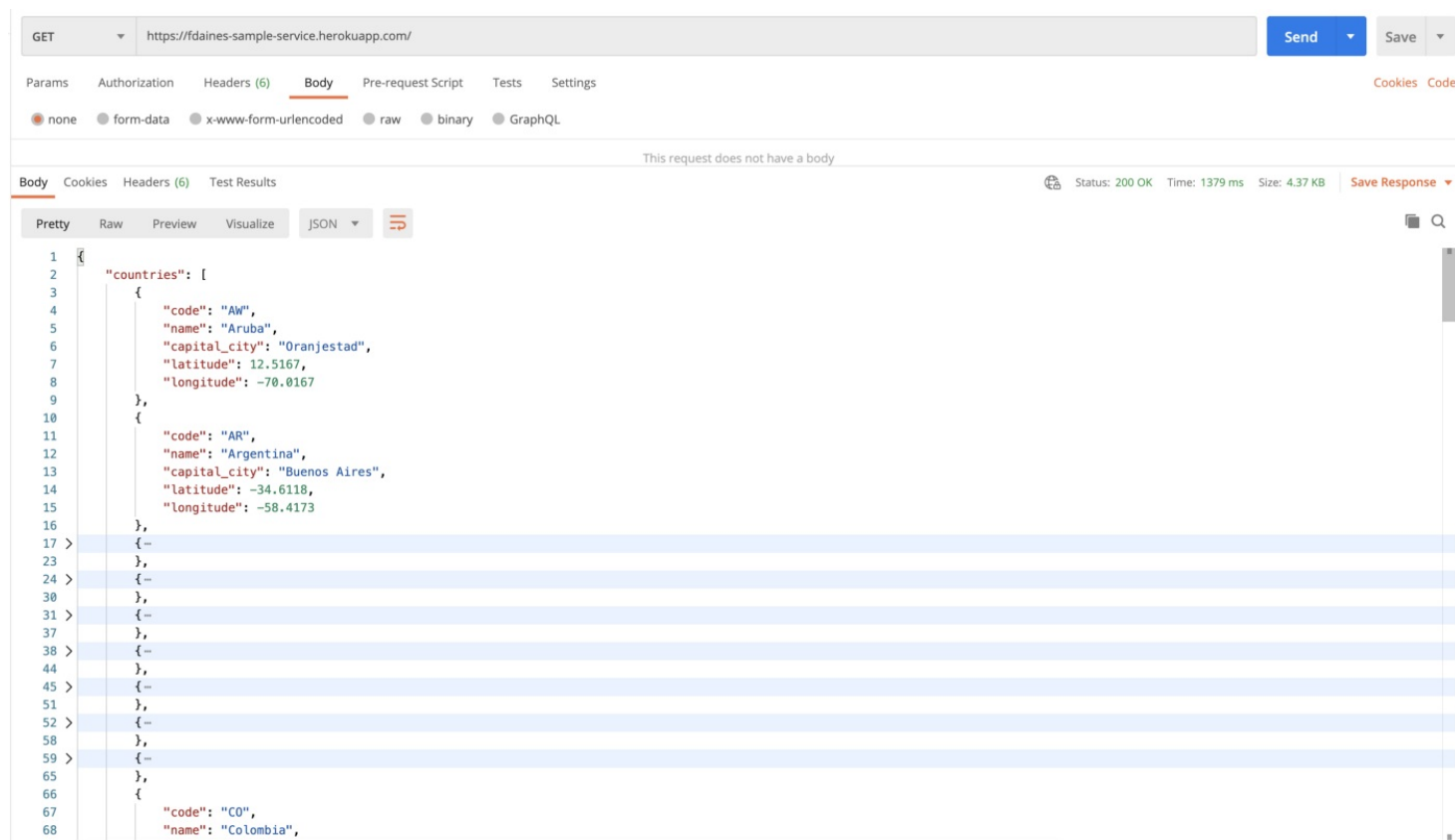


Vista de la ejecución del workflow en Github Actions.

Dado que el workflow fue ejecutado de forma exitosa, ahora podemos verificar que el servicio fue desplegado y está disponible para ser consumido desde las aplicaciones cliente.

Consumiendo nuestro servicio desde Heroku

Podemos probar que el servicio está disponible utilizando cualquier cliente HTTP como puede ser *curl* o bien, como en el siguiente ejemplo, *Postman*. Recuerda que la URL de publicación está relacionada a la variable que definimos como *HEROKU_APP_NAME* en nuestros secretos (en este caso <https://fdaines-sample-service.herokuapp.com/>)



Acceso a nuestro servicio desde Postman.

Conclusiones

Una opción sencilla y que requiere bajo esfuerzo de implementación, cuando queremos desplegar aplicaciones o servicios en *Heroku*, es construir un pipeline que se encargue de la construcción, verificación y despliegue.

Github Actions es la herramienta provista por *Github* para construir workflows que se inicien de forma automática a partir de nuestro repositorio, en estos flujos podemos especificar las actividades a ejecutar en cada etapa, y una de ellas puede ser

el desplegar nuestra aplicación/servicio sobre *Heroku*.

El código utilizado para este artículo lo puedes revisar en el siguiente repositorio: <https://github.com/fdaines/github-actions-example>.

