

Desplegar un sitio web - GitHub Actions

Usar GitHub Actions para desplegar un sitio web en cualquier servidor al hacer un push en el repositorio.

El repositorio de este demo de GitHub Actions lo tienes en GitHub:

<https://github.com/Daniel-MundosE/github-actions-demo>

En este artículo vamos a ver una pequeña práctica con **GitHub Actions**, que nos permite **desplegar un sitio web de manera automatizada**, cada vez que se actualice una rama de un repositorio en GitHub.

Para acometer esta acción desde GitHub Actions crearemos un workflow sencillo que se ejecute al hacer un push y que mande los archivos al servidor por SCP. Por supuesto, necesitaremos un acceso ssh al servidor para que esta acción sea posible, con nuestro usuario y la correspondiente llave ssh para poder realizar el proceso de autenticación sobre el servidor.

Qué son GitHub Actions

Antes que nada y por si alguien no lo sabe todavía, GitHub Actions es uno de los servicios recientes de GitHub que permite **crear todo tipo de flujos de trabajo para la automatización de tareas** cuando ocurran cosas en el repositorio.

Por ejemplo, cuando se realiza un push a una determinada rama del repositorio se pueden correr las pruebas o validar si el código tiene los parámetros de calidad deseados. Las acciones pueden ser sencillas o muy complejas, como correr las pruebas, compilar un software, crear un servidor nuevo para desplegar una aplicación en preproducción y en general cualquier flujo de trabajo totalmente personalizado que requiera un proyecto.

GitHub Actions es un servicio gratuito para cualquier usuario, aunque solamente por una cuota de uso limitada. Si necesitamos más minutos de procesamiento de GitHub Actions entonces sería necesario contar con una cuenta de pago.

Crear un workflow en GitHub Actions

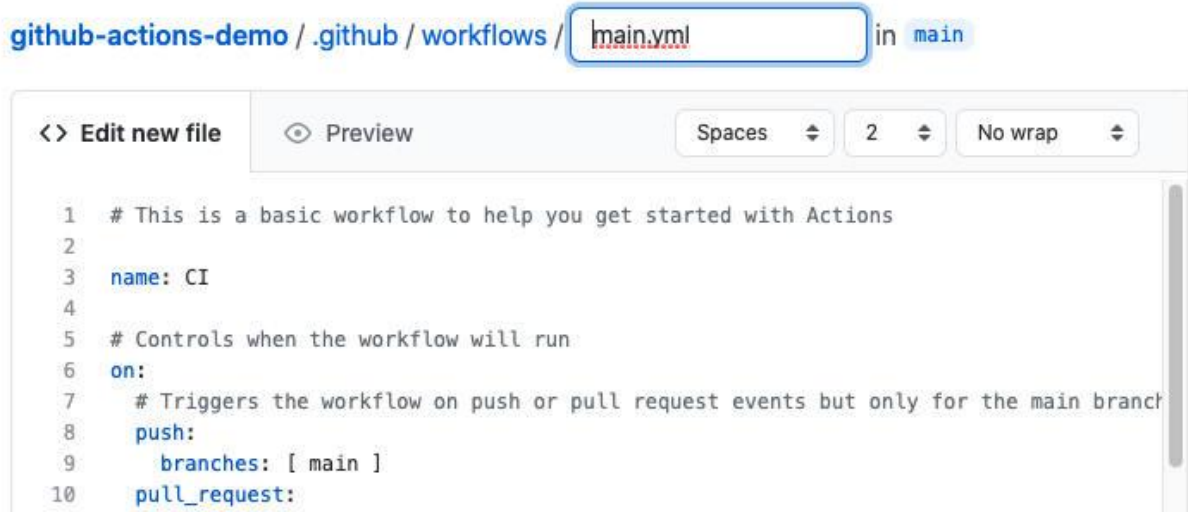
En un repositorio GitHub encontramos una sección llamada "Actions". Desde aquí podemos crear nuevas acciones fácilmente. Si pulsamos el enlace "Actions" y no hemos creado ninguna acción todavía nos llevará a una página donde se explicarán algunas de las posibilidades de esta herramienta.

Podemos empezar pulsando el enlace que dice "set up a workflow yourself". Esto nos llevará a otra página desde donde podemos editar un código de la acción o "workflow".

Archivos yml para definir los workflows

Los flujos de trabajo en GitHub Actions se definen en archivos de código con extensión "yml". Son archivos en un formato llamado **Yaml** que quizás ya conozcas. Si no es así, puedes entenderlos como un JSON pero más ligero, ya que no requieren el uso de llaves. En vez de colocar llaves de inicio y de cierre en Yaml lo que se hace es indentar el código.

Al crear tu primer workflow ya te aportan un archivo llamado "main.yml" de ejemplo, que puedes observar para aprender algunas cosillas. El archivo viene con muchos comentarios, por lo que es sencillo entender qué se hace en cada punto.



The screenshot shows the GitHub Actions workflow editor interface. At the top, the breadcrumb navigation indicates the file path: `github-actions-demo / .github / workflows / main.yml` in the `main` branch. Below this, there are tabs for `Edit new file` and `Preview`. To the right of the `Preview` tab, there are settings for `Spaces` (set to 2) and `No wrap`. The main area displays the content of `main.yml` with line numbers 1 through 10. The code is a basic workflow configuration:

```
1 # This is a basic workflow to help you get started with Actions
2
3 name: CI
4
5 # Controls when the workflow will run
6 on:
7   # Triggers the workflow on push or pull request events but only for the main branch
8   push:
9     branches: [ main ]
10  pull_request:
```

Ese código del `main.yml` de ejemplo lo podemos borrar para crear nuestra propia acción personalizada. Y por supuesto también podemos cambiar el nombre al archivo, para llamarlo algo como `"deploy.yml"` en vez de `"main.yml"`.

Enseguida veremos el código de nuestra acción personalizada, pero antes vamos a explicar cómo conseguir fácilmente workflows para hacer todo tipo de procedimientos imaginables.

Marketplace de GitHub Actions

GitHub tiene un marketplace de actions desde donde puedes obtener trabajos ya listos para realizar **todo tipo de workflows**. Gracias a estos códigos puedes ahorrarte mucho tiempo de aprendizaje e investigación, yendo directamente al grano.

<https://github.com/marketplace?type=actions>

Desde aquí podemos hacer una búsqueda por "scp" y encontraremos varias acciones para hacer cosas que tengan que ver con este comando de Linux.

The screenshot shows the GitHub Marketplace interface. At the top, there's a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. A search bar contains the text 'scp'. Below the navigation bar, the 'Marketplace / Search results' breadcrumb is visible. On the left sidebar, the 'Types' section has 'Actions' selected. The 'Categories' section lists various categories like API management, Chat, Code quality, etc. The main content area shows 'Actions' with the description 'An entirely new way to automate your development workflow.' It indicates '19 results for "scp" filtered by Actions'. Six actions are displayed in a grid:

- SCP Files** by appleboy: Copy files and artifacts via SSH. 417 stars.
- SCP Upload** by betanzos: Upload files using SCP.
- scp-deployer** by siva1024: Deploy using SSH and SCP.
- scp-pipeline** by cross-the-world: Pipeline: scp.
- scp action** by tcidemo: Action to send dist files to a remote server with scp command(fork nogsantos/ssh-scp-deploy).
- SCP action wrapper** by lBeJluk: SCP action wrapper. 3 stars.

Nosotros vamos a ir a lo fácil, escogiendo una de las acciones que tienen más estrellitas en GitHub. La acción que vamos a usar se llama



"SCP Files"



<https://github.com/marketplace/actions/scp-files>

En la propia página de la acción encontramos ya una serie de ejemplos que nos sirven para entender cómo se usa. Desde aquí podemos fácilmente encontrar código de base para nuestro workflow en el archivo Yaml.

Creando el Workflow personalizado

Ahora que sabemos de dónde tirar, vamos a crear ya nuestro propio Workflow de GitHub Actions.

El código sería como este:

```
name: Deploy

on: [push]

jobs:
  deploy:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@master
      - name: Copiar el contenido del repositorio con scp
        uses: appleboy/scp-action@master
        env:
          HOST: ${ secrets.HOST }
          USERNAME: ${ secrets.USERNAME }
          PORT: ${ secrets.PORT }
          KEY: ${ secrets.SSHKEY }
        with:
          source: "site/*"
          target: "/var/www/html"
```

Significado de cada tag

- name: Deploy nos sirve para indicar cómo se llama esta action
- on: [push] quiere decir que se ejecutará siempre que hagamos push al repositorio
- uses: actions/checkout@master sirve para enviar el contenido de nuestro repositorio a el entorno de trabajo de GitHub, de modo que el workflow pueda acceder a nuestros archivos.
- name: Copiar el contenido del repositorio con scp es el nombre de nuestra acción
- uses: appleboy/scp-action@master indica el código de la acción que hemos encontrado en el marketplace, que hace el trabajo duro para enviar los archivos por scp.
- env: indica toda la serie de datos que requiere esta acción, como el host, el nombre de usuario, etc. Luego te explicamos cómo definir esos datos.
- with: sirve para indicar la configuración extra, en la que definimos dónde están los archivos de origen, que se van a llevar a producción y la carpeta del servidor donde los vamos a colocar.

Si guardamos el archivo con el botón verde "Start commit" habremos hecho todo lo necesario para que las acciones se ejecuten, porque ese botón realiza el commit y luego el push al repositorio. Tal como hemos configurado la acción, al procesarse el push se ejecutará.

Sin embargo, si accedemos a la pestaña "Actions" de nuestro repositorio observaremos que la acción habrá fallado. Si está en ejecución todavía veremos que fallará en unos instantes.



Cuando están en ejecución las actions nos aparecerán en naranja. En esos momentos podemos hacer clic y ver todo el proceso de ejecución actual, examinando cómo van cada una de las tareas planificadas. También podremos ver el log de las actions una vez acaben tanto si fallan como si van bien.

No te preocupes, es normal que haya fallado, dado que todavía no hemos definido los secretos para poder saber a qué servidor deben enviarse los archivos y las credenciales de acceso.

Una vez creado el código del workflow puedes hacer el pull en tu repositorio local para editarlo en local con tu propio editor y subirlo con el correspondiente commit y luego el push. También podrías editarlo directamente desde el sitio de GitHub accediendo a la carpeta y pulsando el botón para editar el archivo.

Secrets en GitHub

Todos los archivos yaml de las acciones se colocan en el repositorio, en una carpeta que se ha creado dentro de ".github/workflows". **Ese código estará visible para todas las personas que tengan acceso al repositorio**, por ejemplo todos los desarrolladores, o si el repositorio es público, todas las personas que accedan al repo.

Dado que muchas personas pueden tener acceso al código yaml de las GitHub Actions **es importante no publicar en ese archivo ninguna información sensible, como claves de los servidores**, especialmente cuando el repositorio es público.

Por supuesto, GitHub tiene una zona segura donde colocar los denominados secretos, es decir, todo lo que son las llaves ssh, usuarios y contraseñas. Esta zona está en la parte de "settings" y luego "secrets".

Aquí pulsamos el botón "New repository secret" y nos llevará a una sección donde podemos crear los secretos que vamos a necesitar. Cada uno de ellos tiene un nombre y un valor. Por ejemplo "HOST" y el valor será la IP del servidor que queramos usar para desplegar el proyecto.

Actions secrets / New secret

Name

HOST

Value





0.1.2.3

Add secret

Para nuestro workflow `deploy.yml` tenemos que crear los siguientes secretos:

- HOST: La IP del servidor
- USERNAME: el nombre de usuario con el que accedemos al servidor
- PORT: el puerto de conexiones ssh, generalmente 22
- SSHKEY: la llave ssh que usamos para conectar

Una vez creados estos secretos los podremos ver en la página de "secrets" y los podremos editar si fuera necesario, en caso que nos hayamos equivocado con sus valores o nombres.

Repository secrets		
 HOST	Updated 2 minutes ago	<button>Update</button> <button>Remove</button>
 PORT	Updated 10 minutes ago	<button>Update</button> <button>Remove</button>
 SSHKEY	Updated now	<button>Update</button> <button>Remove</button>
 USERNAME	Updated 2 minutes ago	<button>Update</button> <button>Remove</button>

Ejecutar de nuevo las acciones

Cada vez que hagamos un push al repositorio se ejecutarán de nuevo las acciones. De modo que para poder correrlas de nuevo simplemente actualizamos el repositorio y hacemos un push.

Ahora, si hemos colocado correctamente los secretos, veremos que la acción se ejecuta correctamente.

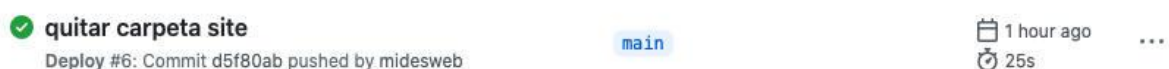
Durante la ejecución es posible ver el progreso haciendo clic en la acción que se está ejecutando.

El despliegue podría fallar por diferentes razones, algunas de las más típicas podrían ser:

- No hemos colocado correctamente los secrets, ya sea porque hemos escrito mal los nombres de los secrets o sus valores
- No hemos colocado bien la ruta donde están los archivos que se van a desplegar (source)
- No hemos colocado bien la ruta del servidor donde colocar los archivos (target)
- No hemos indentado correctamente el archivo .yml
- No hemos escrito bien los nombres de las actions que estamos usando.

Si ha ido mal, aparecerá en rojo la acción y podremos ver, si hacemos clic, el motivo del fallo.

Si ha ido bien, la acción aparecerá en verde y podremos comprobar si los archivos han subido al servidor.



En el caso que haya ido correctamente también podremos hacer una consulta de la salida de la acción haciendo click en ella.

Cambiar la carpeta de destino

Quizás habrás observado que, al subir los archivos al servidor se ha creado una carpeta llamada "site", que es donde están los archivos que queremos llevar a producción, en la estructura de carpetas del repo original.

Quizás los archivos los queríamos situar en la carpeta raíz de publicación del servidor y no en la carpeta "site" como están en el repositorio. Esto se puede configurar añadiendo la línea "strip_components: 1" en la parte del "with".

El código quedará así:

```
name: Deploy

on: [push]

jobs:
  deploy:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@master
      - name: Copiar el contenido del repositorio con scp
        uses: appleboy/scp-action@master
      env:
```

```
HOST: ${ secrets.HOST }
USERNAME: ${ secrets.USERNAME }
PORT: ${ secrets.PORT }
KEY: ${ secrets.SSHKEY }
with:
  source: "site/*"
  target: "/var/www/html"
  strip_components: 1
```

Ahora los archivos de la carpeta "site" se colocarán en el servidor directamente sobre /var/www/html.