05_elementos_de_un_workflow

May 16, 2023

1 Elementos de un *workflow*.

Un repositorio de *Github* puede contener más de un *workflow* y cada *workflows* tiene una estructura básica de ejecución.

https://docs.github.com/es/actions/using-workflows/about-workflows

1.1 El nombre de un *workflow*.

El primer elemento de un *workflow* es el nombre que se le asignará a éste y es definido mediante la clave `name`.

```
name: <nombre>
```

Ejemplo:

En el caso del archivo `blank.yml` el *workflow* es nombrado `CI`.

```
name: CI
```

1.2 Eventos y detonantes (*triggers*).

Por lo general, un *workflow* se ejecuta en función de uno o varios eventos predefinidos, los cuales activan un *trigger*.

Un *workflow* puede tener uno o más *triggers* los cuales son anidados bajo la clave `on`.

```
on:
  <trigger 1>:
    ...
  <trigger 2>:
    ...
  ...
  <trigger n>
   ...
```

Donde:

- `<trigger x>` es un *trigger* con diversos parámetros.

Los repositorios de *Github Actions* son sensibles a diversos eventos, para los que se ha definido un conjunto de detonates (*triggers*) que inician una tarea (*job*).

https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows

Para este curso se utilizarán los *triggers*:

- `pull_request`, el cual se activa cuando se hace un *pull request* en ramas determinadas.
- `push`, el cual se activa cuando se hace un *push* en ramas determinadas.
- `repository_dispatch`, el cual se activa de forma manual o mendiante un agente externo.

Ejemplo:

- En el caso del archivo `blank.yml`, se definen los *triggers* para:
  - El caso de que se ejecute un `push` en la rama `main`.
  - El caso de que se realice un `pull request` en la rama `main`.
  - El caso en el que se realice una activación manual o externa mediante el *trigger* `workflow_dispatch`.

```yaml
on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]
  workflow_dispatch:
```

1.3 Trabajos (*jobs*).

Un *workflow* puede estar conformado por al menos un *job* y cada uno de los *jobs* ejecuta al menos un paso (*step*).

Los *jobs* están anidados bajo la clave `jobs`.

```
jobs:
  <job 1>:
    ...
  <job 2>:
    ...
  ...
  <job n>:
  ...
```

Donde:

- `<job x>` es el identificador que se le asigna al *job*.

Ejemplo:

- En el caso del archivo `blank.yml` se define sólo un *job* con identificador `build`.

```yaml
jobs:
  build:
    ...
```

1.3.1 Definición del *runner*.

Cada *job* se ejecuta dentro de un *runner*. Y el tipo de runner se define como un valor para la clave: `runs-on`.

`runs-on: <tipo>`

Ejemplo:

- En el caso del archivo `blank.yml` el runner del *job* `build` es `ubuntu-latest`.

```
jobs:
  build:
    runs-on: ubuntu-latest
  ...
```

1.3.2 Definición de pasos.

Un *job* consta de una lista de pasos consecutivos, anidada bajo la clave `steps`.

```
steps:
  - <paso 1>
  - <paso 2>
  ...
  - <paso n>
```

Cada paso puede ser:

- Una acción (*action*).
- Un *script* de una línea.
- Un *script* de múltiples líneas.

Ejemplo:

- En el caso del archivo `blank.yml` se definen 3 pasos para el *job* `build`.

```
jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3

      - name: Run a one-line script
        run: echo Hello, world!

      - name: Run a multi-line script
        run: |
          echo Add other actions to build,
          echo test, and deploy your project.
```

1.3.3 *Scripts.*

Un *script* consiste en una expresión que puede ser interpretada por el *shell* del runner.

- Un *script* de una sola línea se define de la siguiente forma:

```
- name: <nombre del script>
  run: <expresión>
```

Un *script* de varias líneas se define de la siguiente forma:

```
- name: <nombre del script>
  run: |
    <expresión 1>
    <expresión 2>
    ...
    <expresión n>
```

Ejemplo:

- En el caso del archivo `blank.yaml` se definen:
  - Un *script* de una línea llamando `Run a one-line script`
  - Un *script* de varias líneas llamado `Run a multi-line script`.

```yaml
jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3

      - name: Run a one-line script
        run: echo Hello, world!

      - name: Run a multi-line script
        run: |
          echo Add other actions to build,
          echo test, and deploy your project.
```

1.3.4 Acciones (*actions*).

Una acción (*action*) es una aplicación creada para la plataforma de *Github Actions* que realiza una tarea compleja pero usada frecuentemente.

Las *actions* permiten extender a *Github Actions* para acceder a recursos propios y de otros proveedores de soluciones.

```
-uses: <ruta>
```

Donde: * `<ruta>` es la ruta donde se encuenta la *action*, así como su versión.

Las *actions* pueden definirse en:

- El mismo repositorio que tu archivo de flujo de trabajo.
- Cualquier repositorio público de *GitHub*.
- Una imagen del contenedor *Docker* publicada en *Docker Hub*.
- *GitHub Marketplace*, la cual es una ubicación central para *actions* creadas por la comunidad de *GitHub*.

El siguiente enlace contiene las instrucciones para crear *actions* propias.

- https://docs.github.com/es/actions/creating-actions

El siguiente enlace contiene instrucciones de como encontrar y personalizar las *actions*.

- https://docs.github.com/en/actions/learn-github-actions/finding-and-customizing-actions

Ejemplo:

- En el caso del archivo `blank.yml`, se utiliza la *action* `actions/chekcout@v3`, la cual permite acceder al repositorio de *GitHub*.

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      ...
```

1.4 La sintaxis de *YAML* de *Github Actions*.

La sintaxis de los documentos *YAML* de *Github Actions* fue diseñada para definir un lenguaje de dominio específico (*DSL*).

La documentación completa de la sintaxis de los documentos *YAML* de *GitHub Actions* puede ser consultada desde:

- https://docs.github.com/es/actions/using-workflows/workflow-syntax-for-github-actions