GitHub ACTIONS Cheat Sheet

@ctapiadev

Fundamentos del Archivo de Flujo de Trabajo (Workflow)

Un flujo de trabajo de GitHub Actions es un archivo con extensión .yml o .yaml ubicado en el directorio .github/workflows de tu repositorio. Este archivo define el proceso automatizado.

name → Especifica el nombre del flujo de trabajo. Este nombre aparece en la pestaña “Actions” de tu repositorio de GitHub.
on → Define el evento que desencadena el flujo de trabajo. Puede ser un push a una rama, una nueva pull request, una ejecución programada (schedule) o un evento manual.

- push → Se activa con un evento push. Puedes especificar ramas o etiquetas.
- pull_request → Se activa con un evento de pull request.
- workflow_dispatch → Permite ejecutar el flujo de trabajo manualmente.
- schedule → Ejecuta el flujo de trabajo a una hora específica usando la sintaxis de cron.

Comandos Esenciales en los Pasos (Steps)

uses → Especifica una acción para ejecutar como un step. Las acciones son unidades de código reutilizables. Puedes usar acciones creadas por la comunidad o crear las tuyas. Un ejemplo común es `actions/checkout@v4`, que clona tu repositorio.
run → Ejecuta instrucciones de la línea comandos usando el shell del sistema operativo.
with → Proporciona una forma de pasar parámetros de entrada a una acción.
env → Establece variables de entorno para un único step.

Contextos y Expresiones

Los contextos son una colección de variables que te permiten acceder a información sobre la ejecución del flujo de trabajo, el evento, el runner, etc. Se acceden usando la sintaxis de expresión: ${{ <contexto>.<propiedad> }}.

github → Contiene información sobre el evento que disparó el workflow. Ej: github.actor (el usuario que inició el evento), github.ref (la rama o tag del evento), github.event_name.
steps → Se usa para acceder a los resultados (outputs) de pasos anteriores dentro del mismo job. Ej: ${{ steps.nombre_del_paso.outputs.un_valor }}.
runner → Contiene información sobre el ejecutar que está corriendo el job. Ej: runner.os (ej. Linux o Windows), runner.temp (ruta a un directorio temporal).
matrix → Contiene las propiedades de la matriz que se están ejecutando en el job actual. Ej. ${{ matrix.node-version }}.

Control de Flujo con Condicionales

Un job puede definir “salidas” (outputs) que pueden ser utilizadas por otros jobs que dependan de él. Es la manera más eficiente de pasar información como un número de versión, el resultado de una prueba, o una ID generada.

outputs → Se define a nivel de job para declarar qué salidas estarán disponibles. Dentro de un step, se establece el valor de esa salida.
needs → Para acceder al output, el job dependiente debe usar el contexto needs. La sintaxis es ${{ needs.<ID_DEL_JOB>.outputs.<NOMBRE_DEL_OUTPUT> }}.

¿Cómo funciona?

- Generar el dato → Un paso dentro del primer job genera un valor y lo asigna a una salida.
- Usar el dato → un segundo job, que tiene al primero en su lista de needs, puede leer ese valor directamente.

Componentes Clave del Flujo de Trabajo

jobs → La ejecución de un flujo de trabajo se compone de uno o más jobs (trabajos). Por defecto, los jobs se ejecutan en paralelo. Para ejecutarlos de forma secuencial, debes definir dependencias.

- runs-on → Especifica el tipo de máquina en la que se ejecutará el job (ej. ubuntu-latest, windows-latest o macos-latest).
- steps → Una secuencia de tareas, llamadas steps (pasos), que se ejecutan como parte de un job. Los steps pueden ejecutar comandos o usar una acción.
- needs → Identifica cualquier job que deba completarse con éxito antes de que este job se ejecute.

Gestión de Datos y Secretos

secrets → Se utiliza para almacenar información sensible, como contraseñas, claves de API o tokens. Los secretos se crean a nivel de repositorio u organización y se puede acceder a ellos en el flujo de trabajo utilizando el contexto secrets (ej. ${{ secrets.TU_SECRETO }}).
github.token → Un secreto preconfigurado que permite autenticarte en la API de GitHub en nombre del flujo de trabajo.

Matrices de Estrategia (Strategy Matrix)

Una matriz te permite ejecutar un mismo job con múltiples configuraciones. Es ideal para probar tu código en diferentes versiones de un lenguaje, sistemas operativos o dependencias.

strategy → Define la estrategia de ejecución para un job.
matrix → Define las diferentes combinaciones de variables que se usarán para crear los jobs.

Ej: Este job se ejecutará 4 veces (2 versiones de Node x 2 sistemas operativos).

```
jobs:
  test:
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-latest, windows-latest]
        node-version: [16.x, 18.x]
    steps:
    - uses: actions/setup_node@v3
      with:
        node-version: ${{ matrix.node-version }}
```

Control de Flujo con Condicionales

Puedes controlar si un job o un paso se ejecuta utilizando la palabra clave if. La condición debe ser verdadera para que se ejecute.

if → Ejecuta el paso o job solo si la condición se cumple. Utiliza contextos para crear las condicionales.

Ej: Solo se ejecuta en la rama main.

```
steps:
  - name: Deploy to production
  if: github.ref == ‘refs/heads/main’
  run: echo “Deploying!”
```

@ctapiadev

¡No olvides seguirme en mis redes sociales!