Contextos de flujo de trabajo

En GitHub Actions, un contexto es un conjunto de objetos o variables predefinidos que contienen información relevante sobre el entorno, los eventos u otros datos asociados con la ejecución de un flujo de trabajo.

Puede usar contextos para acceder a información sobre pasos, ejecuciones de flujos de trabajo, trabajos y entornos de ejecución. Cada vez que desee acceder a un contexto desde un archivo de flujo de trabajo, debe usar una sintaxis similar a `${{ <context> }}`. Por ejemplo:

```
name: Simple Contexts Example

on: push

jobs:
  print-info:
    runs-on: ubuntu-latest
    steps:
      - name: Set custom environment variable
        run: echo "CUSTOM_VARIABLE=Hello, World!" >> $GITHUB_ENV

      - name: Print commit author and custom environment variable
        run: |
          echo "Commit author: ${{ github.actor }}"
          echo "Custom variable: ${{ env.CUSTOM_VARIABLE }}"
```

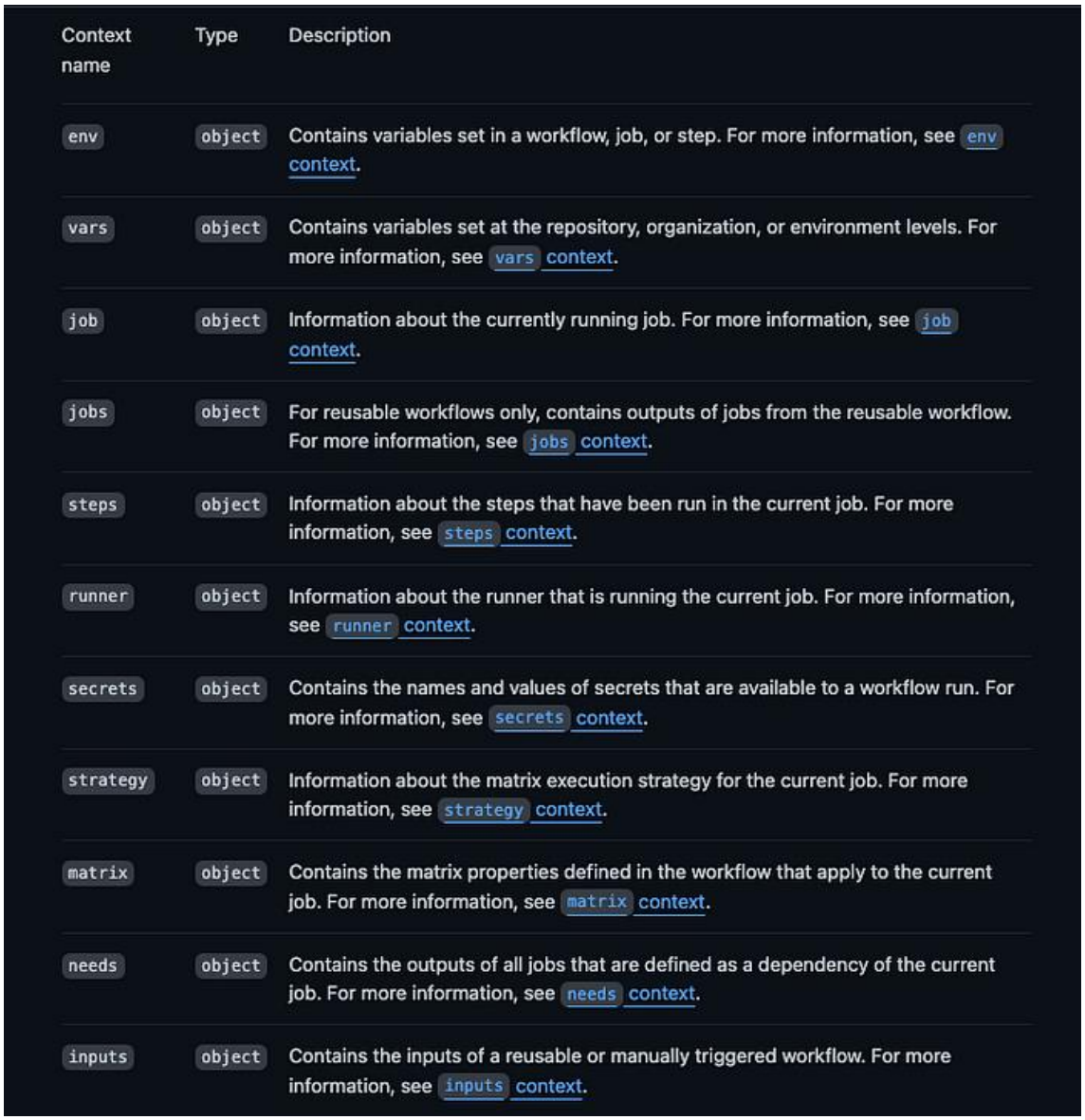
En este ejemplo, el `github.actor`contexto proporciona el nombre de usuario de la persona que activó el evento push y el `env.CUSTOM_VARIABLE`contexto hace referencia a la variable de entorno personalizada establecida en el paso anterior. El flujo de trabajo imprime ambos valores en la consola.

Los contextos se pueden usar principalmente en cualquier parte de su archivo de flujo de trabajo. A menudo se usan con expresiones para verificar condiciones específicas. El siguiente ejemplo usa la instrucción **if** para validar el contexto **de github** . En este caso, el trabajo solo se ejecutará si se **aprueba** el resultado de la expresión :

```
if: github.event_name == 'pull_request_review' &&
github.event.review.state == 'approved'
```

La sintaxis para acceder a un contexto es simple: puede usar la sintaxis de índice **`github['event_name']`** o la sintaxis de de referencia de propiedad **`github.event_name`**.

Las acciones de Github admiten los siguientes contextos en este momento:



Context name	Type	Description
`env`	`object`	Contains variables set in a workflow, job, or step. For more information, see `env` context.
`vars`	`object`	Contains variables set at the repository, organization, or environment levels. For more information, see `vars` context.
`job`	`object`	Information about the currently running job. For more information, see `job` context.
`jobs`	`object`	For reusable workflows only, contains outputs of jobs from the reusable workflow. For more information, see `jobs` context.
`steps`	`object`	Information about the steps that have been run in the current job. For more information, see `steps` context.
`runner`	`object`	Information about the runner that is running the current job. For more information, see `runner` context.
`secrets`	`object`	Contains the names and values of secrets that are available to a workflow run. For more information, see `secrets` context.
`strategy`	`object`	Information about the matrix execution strategy for the current job. For more information, see `strategy` context.
`matrix`	`object`	Contains the matrix properties defined in the workflow that apply to the current job. For more information, see `matrix` context.
`needs`	`object`	Contains the outputs of all jobs that are defined as a dependency of the current job. For more information, see `needs` context.
`inputs`	`object`	Contains the inputs of a reusable or manually triggered workflow. For more information, see `inputs` context.

Imagen de documentos de github

Expresiones

En Github Actions, las expresiones se pueden usar para establecer variables en un archivo de flujo de trabajo y acceder a contextos. Una expresión también puede usar una combinación de literales, funciones, contextos y operadores.

Expresiones Literales

Los literales se representan mediante tipos de datos como los siguientes:

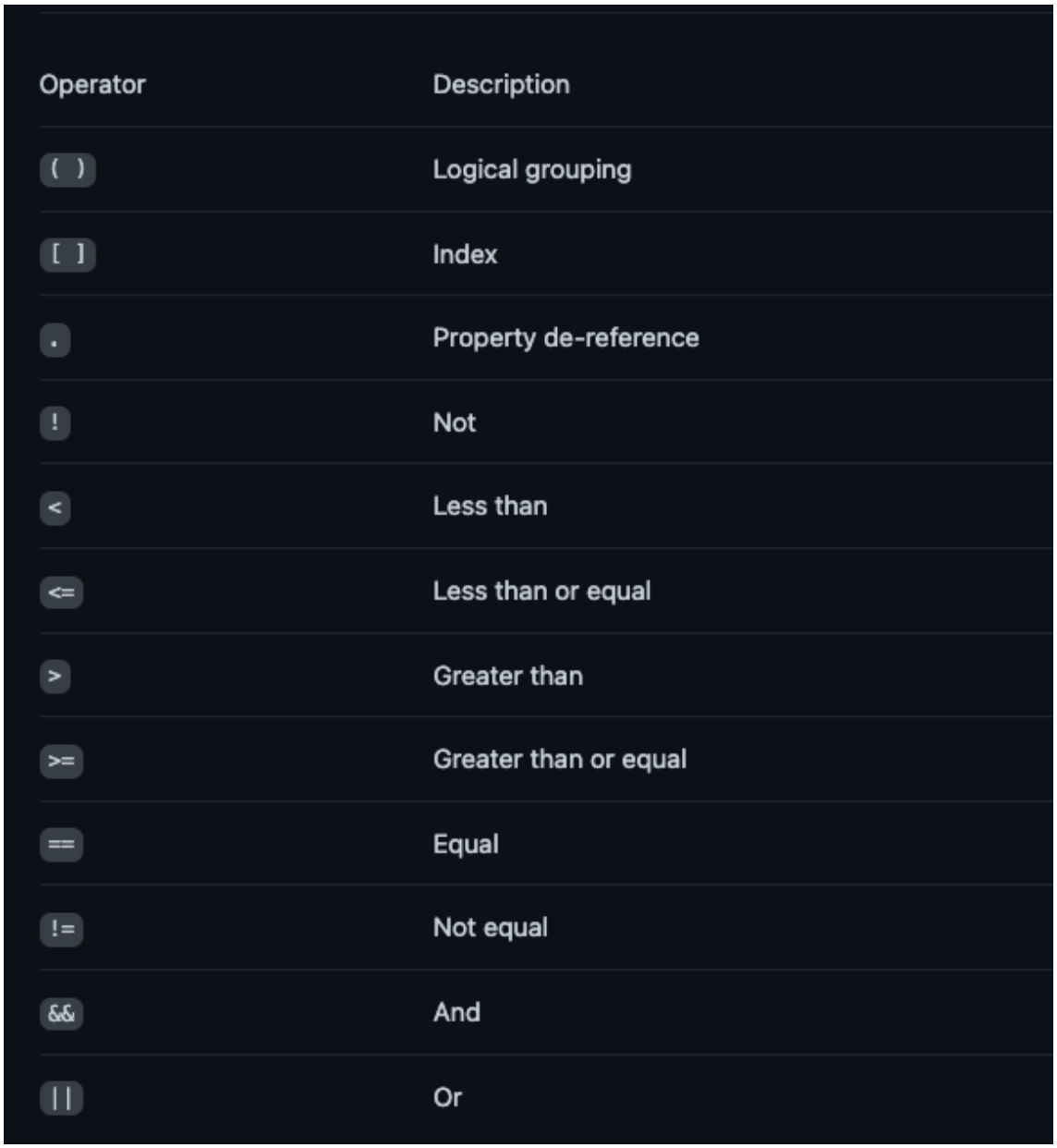
- **booleano** : **verdadero** o **falso** , no distingue entre mayúsculas y minúsculas.
- **nulo**
- **número** : cualquier formato de número que sea compatible con JSON.
- **string** : Las comillas simples deben usarse con cadenas.

Por ejemplo:

```
env:
  myNull:  ${{  null  }}
  myBoolean:  ${{  false  }}
  myIntegerNumber:  ${{  711  }}
  myFloatNumber:  ${{  -9.2  }}
  myHexNumber:  ${{  0xff  }}
  myExponentialNumber:  ${{  -2.99 e-2  }}
  myString:  Mona  the  Octocat
  myStringInBraces:  ${{  '¡Es código abierto!'  }}
```

Operadores

Los operadores se utilizan dentro de las expresiones para realizar diversas operaciones, como operaciones aritméticas, de comparación o lógicas. Las expresiones se envuelven entre llaves dobles `{{ }}`. Los operadores admitidos en las expresiones de flujo de trabajo de GitHub Actions son:



Operator	Description
( )	Logical grouping
[ ]	Index
.	Property de-reference
!	Not
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
==	Equal
!=	Not equal
&&	And
||	Or

Imagen de documentos de github

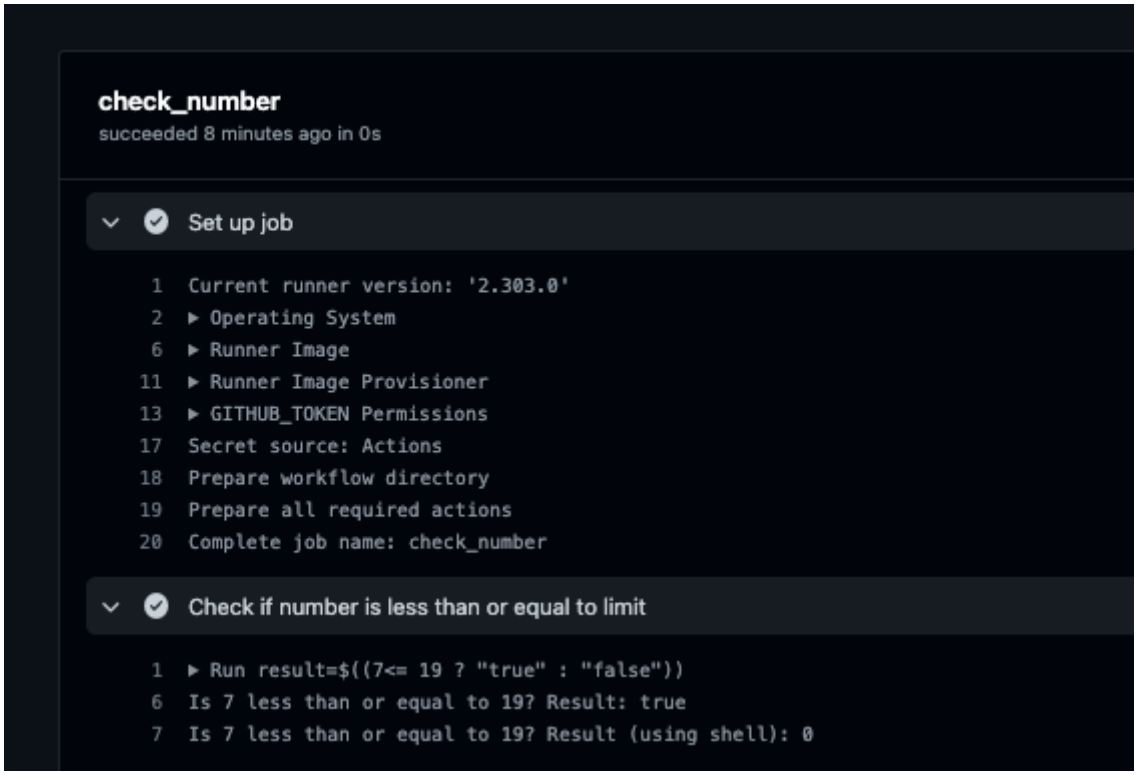
Ejemplo de archivo de flujo de trabajo:

```
name: Less Than or Equal Operator Example

on: push

jobs:
  check_number:
```

```
    runs-on: ubuntu-latest
    steps:
      - name: Check if number is less than or equal to limit
        run: |
          result=$((7<= 19 ? "true" : "false"))
          echo "Is 7 less than or equal to 19? Result: ${{ 7 <= 19 }}"
          echo "Is 7 less than or equal to 19? Result (using shell): $result"
```



Funciones

Las funciones son métodos predefinidos o incorporados que se pueden usar dentro de las expresiones para realizar varias operaciones o manipular datos. Las funciones se pueden usar dentro de las llaves dobles `{{ }}`que denotan una expresión.

- **empieza con** y **termina con** : **empieza con('cadena')** , **termina con('cadena')** .
- **toJSON** : devuelve una representación JSON impresa del valor que se ha pasado. Un ejemplo es **toJSON(value)** .
- **succes** : esta función de verificación del estado del trabajo devuelve verdadero cuando ninguno de los pasos anteriores ha fallado o se ha cancelado. Un ejemplo es **if: ${{ succes() }}** .
- **always** : esta función de verificación del estado del trabajo devuelve verdadero incluso cuando se cancela. Un ejemplo es **if: ${{ always() }}** .
- **cancelled** : esta función de verificación del estado del trabajo devuelve verdadero si se canceló el flujo de trabajo. Un ejemplo es **if: ${{ cancelled() }}** .
- **failure** : esta función de verificación del estado del trabajo devuelve verdadero cuando falla cualquier paso anterior de un trabajo. Un ejemplo es **if: ${{ failure() }}** .

Por ejemplo:

```
name: build

on: push
```

```
jobs:
  job1:
    runs-on: ubuntu-latest
    outputs:
      matrix: ${{ steps.set-matrix.outputs.matrix }}
    steps:
      - id: set-matrix
        run: echo
"matrix={\"include\":[{\"project\":\"foo\",\"config\":\"Debug\"},{\"project\":\"bar\",\"co
nfig\":\"Release\"}]}" >> $GITHUB_OUTPUT
  job2:
    needs: job1
    runs-on: ubuntu-latest
    strategy:
      matrix: ${{ fromJSON(needs.job1.outputs.matrix) }}
    steps:
      - run: build
```

El flujo de trabajo anterior establece una matriz JSON en un trabajo y la pasa al siguiente trabajo mediante un archivo de salida y `fromJSON`.