platform.sh

upsun
Powered by Platform.sh

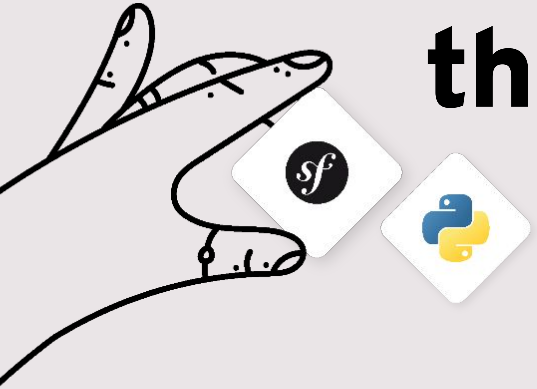
Introduction to GitHub Actions:

Understanding key terms and building your first GitHub action

Paul Gilzow
He/Him/His
Developer Relations Engineer

platform.sh

The cloud platform that helps web teams innovate faster



Introduction to GitHub Actions:

Understanding key terms and building your first GitHub Action

Paul Gilzow
He/Him/His
Developer Relations Engineer

Warning!

Outline

- What is GitHub Actions?
- Components of GitHub Actions
- Components of a workflow
- Build our first workflow
- Components of a custom action
- Build our first custom action
- Caveats and gotchas
- Putting it all together

What is GitHub Actions?

GitHub Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline.

Components of the GitHub Actions platform

GitHub Actions platform

Components

- Workflows
- Events
- Runners
- Jobs
- Steps

GitHub Actions platform

Components

- Workflows
- Events
- Runners
- Jobs
- Steps
- Github actions

GitHub Actions platform

Components

```
* aka custom actions
aka actions
```

- Workflows
- Events
- Runners
- Jobs
- Steps
- Github actions*

Warning!

GitHub Actions platform

Warning

The same base word is reused but means a different thing depending on the context

Status vs Checks vs Status Checks

GitHub Actions platform

Components

- Workflows
- Events
- Runners
- Jobs
- Steps
- actions

You configure a GitHub Actions ***workflow*** to be triggered when an ***event*** occurs that then runs a series of ***jobs*** on a ***runner*** that include one or more ***steps*** that call a script(s) or ***action***(s)

The components

GitHub Actions platform

Components

Workflow

+ A configurable automated process that will run one or more *jobs*
+ Defined in the `.github/workflows` directory of a repository
+ A repository can have multiple workflows, each of which can perform a different set of tasks

GitHub Actions platform

Components

Event

- Specific activity that occurs to/in the repository that can be used to trigger a workflow
- Sometimes referred to as "workflow triggers"
- Events that trigger workflows docs

GitHub Actions platform

Components

Runner

+ A virtual server instance that runs the jobs in your workflow when it is triggered
+ GitHub-provided and self-hosted
+ Windows, Ubuntu, and MacOS

GitHub Actions platform

Components

Job

+ Series of steps in a workflow which are executed on the same runner
+ Each job runs in a runner environment specified by its `runs-on` property
+ Jobs run in parallel by default
+ A workflow can run an unlimited* number of jobs

GitHub Actions platform

Components

Steps

+ Either a shell script that will be executed, or an *action* that will be run
+ Executed in order and are dependent on each other
+ All steps in a job are executed on the same *runner*

GitHub Actions platform

Components

action

+ a self-contained, modular, sharable script or piece of automation that performs a specific task
+ Can be chained together in a workflow to perform complex automation tasks

Components of a workflow

Workflow components

A workflow file must contain:

+ One more *events* that will trigger the workflow
+ One or more *jobs*, each of which will execute on a *runner* machine
+ Each *job* must contain one or more *steps*.
+ Each *step* must either run a script OR use an *action*

Our first workflow

Our First Workflow

Must contain:

- Event
- Job
- Runner
- Step

first.yaml

```yaml
on:
  workflow_dispatch:

jobs:
  say_hello:
    runs-on: ubuntu-latest
    steps:
      - run: echo "Hello there!"
```

Digging deeper into workflow components

Workflow components

`A workflow must contain:jobs`

jobs.<job-id>

+ Is a string
+ Must be unique inside the workflow
+ Must start with **a letter** or `_` and can only contain **alphanumeric characters**, `-`, or `_`

Workflow components

`A workflow job must contain one or more steps:`

- An array of tasks
- Each step runs in its own process in the runner environment
- Has access to the workspace and file system
- Must include either `uses` or `run`
  - `uses` - sets the step to use a github action
  - `run` - Runs command-line programs using the operating system's shell

Our second workflow

second.yaml

```yaml
name: Welcome to the party (second)
on:
  workflow_dispatch:
jobs:
 say_hello:
    name: "Let's greet the user!"
    runs-on: ubuntu-latest
    steps:
      - run: echo "Hello there!"
      - uses: actions/checkout@v3
      - run: cat ./github/workflows/list.txt
        name: display contents of list.txt
      - run: echo "I am step 4!"
```

Let's talk about... inputs

Inputs

- Only applicable when using the `workflow_dispatch` or `workflow_call` event in a workflow… or in an action
- Defines the inputs the workflow/action can accept
- Each input id has to be unique, **alphanumeric**, **`-`**, or **`_`**

Inputs

Properties

+ `inputs.<input_id>.description` - what will be displayed to the user. **Required**
+ `inputs.<input_id>.required` - whether or not an entry is required. *Optional*
+ `inputs.<input_id>.default` - a default value to use. *Optional*
+ Access an input's value via the `inputs` context

Our third workflow

third.yaml

```yaml
name: Welcome to the party (third)
on:
  workflow_dispatch:
    inputs:
      thename:
        description: 'What is your name?'
        required: true
jobs:
  say_hello:
    name: "Let's greet the user!"
    runs-on: ubuntu-latest
    steps:
      - run: echo "Hello there, ${{ inputs.thename }}!"
```

...and Contexts

Contexts

+ Provides access to information about workflow runs, variables, runner environments, jobs, and steps
+ Exposed as an object that contains properties, which can be strings or other objects
+ Twelve types of contexts
+ The types of contexts available will depend on the context in which they are being accessed

Contexts

The syntax to access a context inside of workflows and composite actions:

```
${{ <context>.<property> }}
```

Example:

```
${{ steps.get-target-url.outputs.target_url }}
${{ inputs.thename }}
```

Inside of JavaScript action:

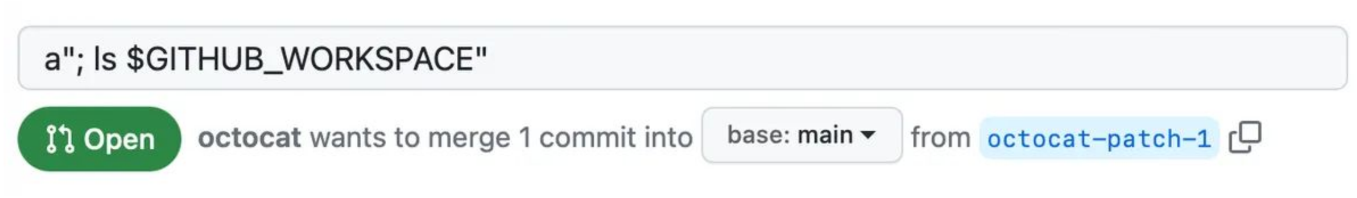
```
const core = require('@actions/core');
const nameToGreet = core.getInput('thename');
```

Warning!

Contexts

- **Not all context values are escaped for use in shell commands**
- **Many context values are potential vectors for script injections**

```
steps:
  - run: |
    Title="${{ github.event.pull_request.title }}"
    echo "PR was created: ${title}"
```



a"; ls $GITHUB_WORKSPACE"

...and outputs

Outputs

- **Data that an action or step will return upon completion**
- **Actions/steps that run later in a workflow can use the output data set in previously run action/step**

Outputs

Set an output:

```
echo "<output name>=<value>" >> $GITHUB_OUTPUT
```

or

```
const core = require('@actions/core');

core.setOutput("<output name>", <value>);
```

Use an output:

```
${{ steps.<step-id>.outputs.<output-name> }}
```

Outputs example

Outputs

outputs.yaml

```yaml
jobs:
  say_hello:
    name: "Let's greet the user!"
    runs-on: ubuntu-latest
    steps:
      - id: get-time
        name: Grab the current time
        run: |
          curTime=$(date)
          echo "currenttime=${curTime}" >> $GITHUB_OUTPUT
      - run: echo "The time from the early step is
          ${{ steps.get-time.outputs.currenttime }}"
```

Custom Actions

Custom actions

Custom actions

Three types:

+ Docker
+ Javascript
+ Composite

Custom actions

Three types:

+ Docker
+ Javascript
+ Composite

Stored:

+ In a repository of its own
+ In a directory of its own in the repository

Components of an action

Components of an action

Required: Metadata file

- **File must be named action.yaml or action.yml**
- **The metadata file must contain:**
  - Name
  - Description
  - Runs* and Runs:using
- **The metadata file can also define `inputs`, and `outputs`, for the action**

Components of an action

Required: Metadata file

`* Depending on which type of action you've selected, there may be additional required properties`

- **File must be named action.yaml or action.yml**
- **The metadata file must contain:**
  + Name
  + Description
  + Runs* and Runs:using
- **The metadata file can also define `inputs`, and `outputs`, for the action**

Components of an action

Must contain:

- Name: `Greets a User`
- Description: `Greets a User to the GitHub…`
- runs:using: `node20`

```
action.yaml
```

```yaml
name: Greets a User
description: Greets a user to the GitHub Actions platform
runs:
  using: 'node20'
  main: 'main.js'
```

Warning!

Gotchas with custom actions

```
runs:
  using: composite
```

VS

```
runs:
  using: node20
```

- For Docker and Composite actions you use those words for `runs:using`; for JavaScript you use the *node* version
- GitHub Actions only supports version~~s 16 and~~ 20 for Node
- Must commit your `node_modules` to the repository, or use something like `vercel/ncc` to package everything into a single file
- Binaries/programs in a runner may be out-of-date

Our first custom action

First Custom action

```
.github/actions/greet-user/action.yaml
```

```yaml
name: Welcome to the party (action)
description: Welcomes someone to our party
inputs:
  who_to_greet:
  description: 'What is your name?'
  type: string
  required: true
runs:
  using: composite
  steps:
     - name: Truly greet the user
     id: greet-user
     shell: bash
     run: |
echo "::notice::Hello, ${{ inputs.who_to_greet }} from
an action!!!"
```

Using our first custom action

```
.github/workflows/fifth.yaml
```

```yaml
name: Welcome a user via our First Action
on:
  workflow_dispatch:
    inputs:
      thename:
        description: "Who should we greet?"
        type: string
        required: true
jobs:
  say_hello:
  steps:
     - uses: actions/checkout@v3
     - uses: ./.github/actions/greet-user
       with:
         who_to_greet: ${{ inputs.thename }}
```

What we accomplished

GitHub Actions

- What is GitHub Actions?
- Components of GitHub Actions
- Components of a workflow
- Built our first workflow
- Components of a custom action
- Built our first custom action
- Caveats and gotchas

Putting it all together

Putting it all together

The goal:

+ Create an action to run a visual regression test using a baseline URL to test against a development URL

Questions?

Resources

- Workflows and action built during the presentation: https://github.com/gilzow/github-actions-presentation
- Custom action shown during the "Putting it all together" section: https://github.com/gilzow/github-actions-presentation-vrt
- Demo site workflow to use the custom action above
- GitHub documentation on Events
- GitHub documentation on contexts
- BackstopJS Visual Regression Testing

Thank you!

SCAN ME

Paul Gilzow

He/Him/His

Developer Relations Engineer, Platform.sh

Email: paul.gilzow@platform.sh

Social: https://linktr.ee/gilzow

Powered by Platform.sh