



Creating a composite action

Create an action

4 of 7 in learning path

Next: Metadata syntax for GitHub Actions

In this article

Introduction

Prerequisites

Creating an action metadata file

Testing out your action in a workflow

Example composite actions on GitHub.com

In this guide, you'll learn how to build a composite action.

Note: GitHub-hosted runners are not currently supported on GitHub Enterprise Server. You can see more information about planned future support on the <u>GitHub public roadmap</u>.

Introduction @

In this guide, you'll learn about the basic components needed to create and use a packaged composite action. To focus this guide on the components needed to package the action, the functionality of the action's code is minimal. The action prints "Hello World" and then "Goodbye", or if you provide a custom name, it prints "Hello [who-to-greet]" and then "Goodbye". The action also maps a random number to the random-number output variable, and runs a script named goodbye.sh.

Once you complete this project, you should understand how to build your own composite action and test it in a workflow.

Warning: When creating workflows and actions, you should always consider whether your code might execute untrusted input from possible attackers. Certain contexts should be treated as untrusted input, as an attacker could insert their own malicious content. For more information, see "Security hardening for GitHub Actions."

Prerequisites @

Before you begin, you'll create a repository on your GitHub Enterprise Server instance.

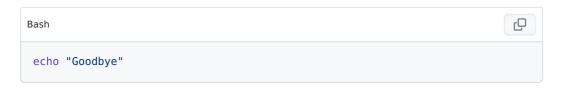
- 1 Create a new public repository on your GitHub Enterprise Server instance. You can choose any repository name, or use the following hello-world-composite-action example. You can add these files after your project has been pushed to GitHub Enterprise Server. For more information, see "Creating a new repository."
- 2 Clone your repository to your computer. For more information, see "Cloning a

repository."

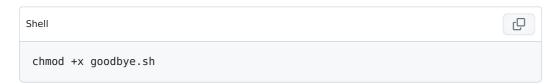
3 From your terminal, change directories into your new repository.



4 In the hello-world-composite-action repository, create a new file called goodbye.sh, and add the following example code:



5 From your terminal, make goodbye.sh executable.



6 From your terminal, check in your goodbye.sh file.

```
git add goodbye.sh
git commit -m "Add goodbye script"
git push
```

Creating an action metadata file &

In the hello-world-composite-action repository, create a new file called action.yml and add the following example code. For more information about this syntax, see "Metadata syntax for GitHub Actions".

action.yml

```
name: 'Hello World'
description: 'Greet someone'
inputs:
who-to-greet: # id of input
description: 'Who to greet'
required: true
default: 'World'
outputs:
random-number:
description: "Random number"
value: ${{ steps.random-number-generator.outputs.random-number }}
runs:
using: "composite"
steps:
```

```
- run: echo Hello ${{ inputs.who-to-greet }}.
    shell: bash
- id: random-number-generator
    run: echo "random-number=$(echo $RANDOM)" >> $GITHUB_OUTPUT
    shell: bash
- run: echo "${{ github.action_path }}" >> $GITHUB_PATH
    shell: bash
- run: goodbye.sh
    shell: bash
```

This file defines the who-to-greet input, maps the random generated number to the random-number output variable, adds the action's path to the runner system path (to locate the goodbye.sh script during execution), and runs the goodbye.sh script.

For more information about managing outputs, see "<u>Metadata syntax for GitHub Actions</u>".

For more information about how to use github.action path, see "Contexts".

2 From your terminal, check in your action.yml file.

```
Shell

git add action.yml
git commit -m "Add action"
git push
```

3 From your terminal, add a tag. This example uses a tag called v1 . For more information, see "About custom actions."

```
Shell

git tag -a -m "Description of this release" v1
git push --follow-tags
```

Testing out your action in a workflow &

The following workflow code uses the completed hello world action that you made in "Creating a composite action".

Copy the workflow code into a .github/workflows/main.yml file in another repository, but replace actions/hello-world-composite-action@v1 with the repository and tag you created. You can also replace the who-to-greet input with your name.

.github/workflows/main.yml

```
on: [push]

jobs:
hello_world_job:
runs-on: ubuntu-latest
name: A job to say hello
steps:
- uses: actions/checkout@v4
- id: foo
```

```
uses: actions/hello-world-composite-action@v1
with:
    who-to-greet: 'Mona the Octocat'
- run: echo random-number ${{ steps.foo.outputs.random-number }}
shell: bash
```

From your repository, click the **Actions** tab, and select the latest workflow run. The output should include: "Hello Mona the Octocat", the result of the "Goodbye" script, and a random number.

Example composite actions on GitHub.com &

You can find many examples of composite actions on GitHub.com.

- microsoft/action-python
- microsoft/gpt-review
- <u>tailscale/github-action</u>

Previous

Creating a JavaScript action

Next

Metadata syntax for GitHub Actions

Legal

© 2023 GitHub, Inc. Terms Privacy Status Pricing Expert services Blog