



GitHub Actions

Fundamentals

Presented by GitHub Professional Services

Introduction

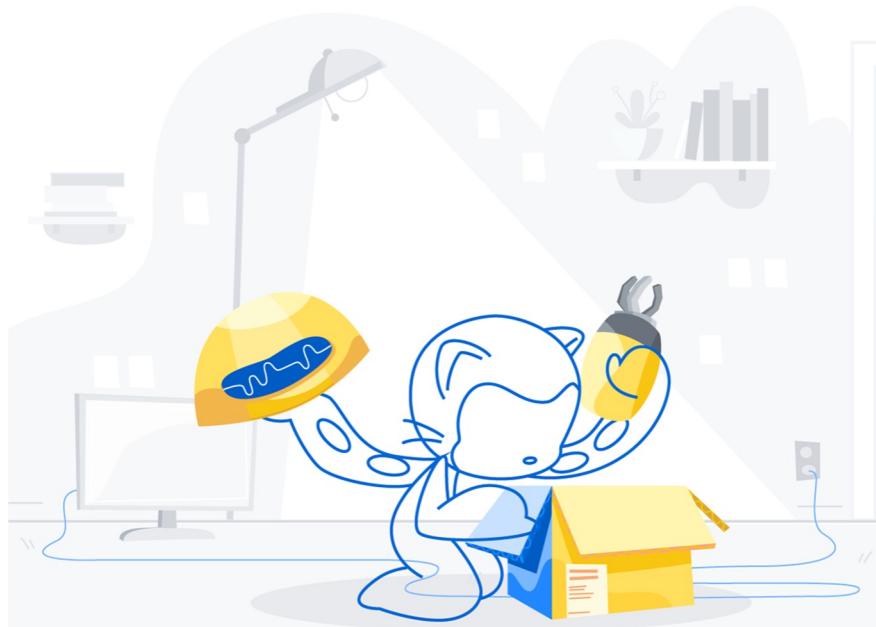
Objectives



- Understand the **basic components** and vocabulary of GitHub Actions
- Understand the **YAML** syntax
- Understand the **workflow syntax** and how to write workflows using intellisense
- Understand events that can **trigger** workflows
- Learn the **context and expression syntax** as well as workflow commands
- Know the different types of **actions** and how to create/publish them
- Understand the different hosting options for **runners**
- Use **Secrets, Variables** and **Environments** for staged deployments
- Workflow templates and **reusable workflows**

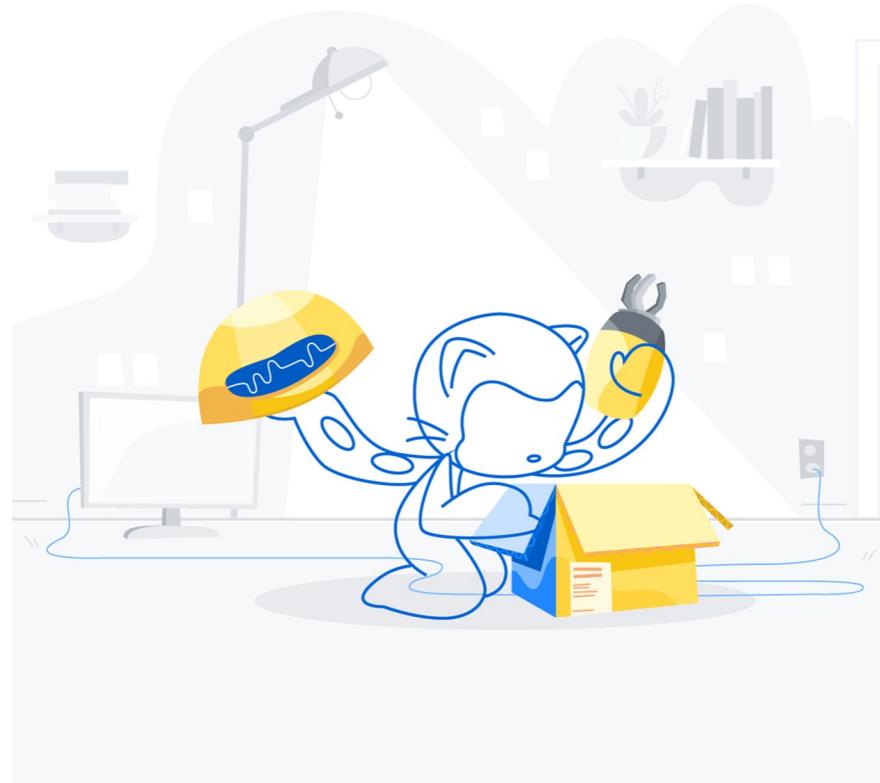
Agenda – day 1

- Workflow fundamentals
- YAML and workflow syntax
- Writing workflows
- GitHub Actions
- Authoring Actions



Agenda – day 2

- CI/CD with GitHub Actions
- Secrets, Variables and Environments
(staged deployments)
- Running workflows (runners)
- Action policies
- Sharing workflows (reusable workflows)



Icebreaker

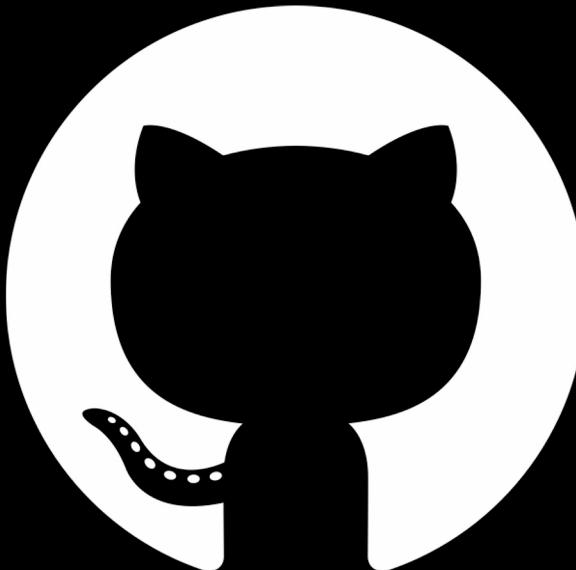
GitHub Actions Fundamentals

Where the world builds software

100M+
Developers

4M+
Organizations

2.6B+
Contributions / Year



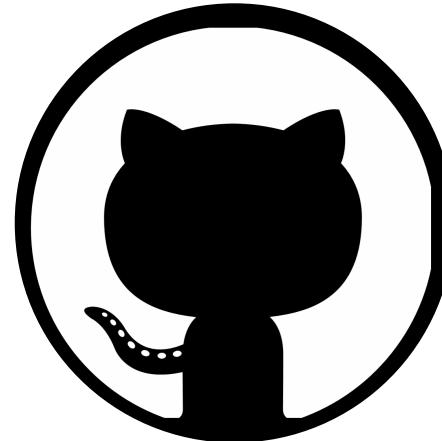
200M+
Repositories

1,000s
Open-Source
Communities

84%
Fortune 500
companies

A different approach...

- Leverage the power of the open-source community
- Platform
- Flexible



GitHub Marketplace

- Discover open-source Actions across multiple domains
- ~15,000 Actions (and counting...)
- Verified creators 
(Publisher domain and email verified)
- Reference these Actions directly in your workflow
- Integrated into the GitHub editor

Extend GitHub

Add tools to help you build and grow

[Explore apps](#)



Types

Apps

Actions

Categories

API management

Chat

Code quality

Code review

Continuous integration

Dependency management

Deployment

IDEs

Learning

Localization

Mobile

Monitoring

Project management

Publishing

Recently added

Security

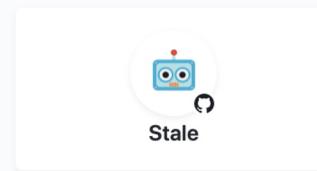
Support

Search for apps and actions

Recommended for you



Codecov | Code Coverage



Stale



Imgbot

Trending



Hound

By houndci

Automated code reviews

↓ 3.8k installs



Moesif API Insights

By Moesif

Understand API usage and take action with user-centric API observability

↓ 698 installs



Octobox

By octobox

Spend less time managing your GitHub notifications

↓ 4.4k installs



WhiteSource Bolt

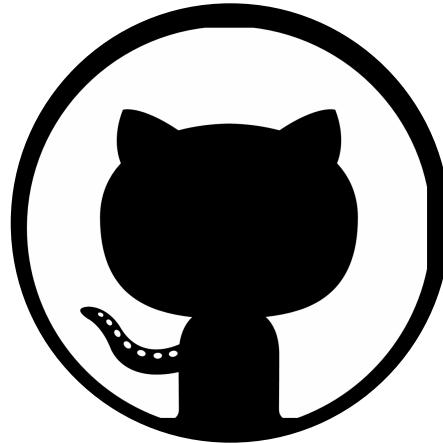
By whitesource

Detect open source vulnerabilities in real time with suggested fixes for quick remediation

↓ 3.6k installs

GitHub Actions – More than CI/CD

- Generic workflow engine
- Automate everything with workflows
- 35 events can trigger a workflow
- GitHub Token and Workflow Permissions
- Community-powered workflows
- Any platform, any language, any cloud



Workflow Fundamentals

- A text file in your repository (.github/workflows)
- YAML Ain't Markup Language (**YAML**)
- Events trigger workflows (`on:`)
- One or multiple jobs
- Executed on a runner
- Contains steps
- A reusable step is action

```
<> Edit file Preview changes Spaces 2 No wrap

1 name: PR-Validation
2
3 on: [pull_request]
4
5 jobs:
6   Build:
7     runs-on: ubuntu-latest
8
9   steps:
10    - name: 'Checkout Github Action'
11      uses: actions/checkout@master
12
13    - name: Set up .NET Core
14      uses: actions/setup-dotnet@v1
15      with:
16        dotnet-version: '5.0.x'
17
18    - name: Setup Node
19      uses: actions/setup-node@v2.5.1
20      with:
21        node-version: 10.16.3
22
23    - name: Install dependencies in client app
24      working-directory: src/Tailwind.Traders.Web/ClientApp
25      run: npm install
26
27    - name: Build and publish with dotnet
28      working-directory: src/Tailwind.Traders.Web
29      run: |
30        dotnet build --configuration Release
31
```

YAML

YAML basics

- Extension: `.yml` or `.yaml`
- A strict superset of **JSON**
- Contains syntactically relevant **newlines** and **indentation** instead of braces
- Data-serialization language writable and readable by humans



YAML basics

```
1 # This is a comment in yaml
2
3 # Scalar types:
4 key: value
5
6 # Data types:
7 integer: 42
8 float: 42.0
9 string: a text value
10 boolean: true
11 null value: null
12 datetime: 1999-12-31T23:59:43.1Z
13
14 # Keys and values can contain spaces and do not need quotation.
15 # You can quote both with single or double quotes:
16 'single quotes': 'have ''one quote'' as the escape pattern'
17 "double quotes": "have the \"backslash \\\" escape pattern"
18
19 # Literal blocks:
20 literal_block: |
21     Text blocks use 4 spaces as indentation. The entire
22     block is assigned to the key 'literal_block' and keeps
23     line breaks and empty lines.
24
25     The block continuous until the next element.
26
27 # Collection types
28
29 # Maps
30 # Maps use 2 spaces of indentation:
31 nested_type:
32     key1: value1
33     key2: value2
34     another_nested_type:
35         key1: value1
36
37 #JSON syntax:
38 map: {key: value}
39
40 # Sequence
41 # Uses a dash before each item:
42 sequence:
43     - item1
44     - item2
45
46 #JSON syntax:
47 sequence: [item1, item2, item3]
```

Basic workflow syntax

Basic syntax

```
./.github/workflows/workflow-file-name.yml
```

```
name: Super Linter workflow
```

events → `on:`
 `push:`

jobs → `jobs:`
 `lint:`
 `name: Lint Code Base`

runner → `runs-on: ubuntu-latest`

steps → `steps:`
 `- uses: actions/checkout@v2`

actions → `- uses: github/super-linter@v3`
 `env:`

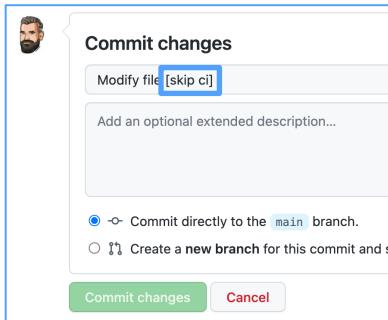
secrets → `GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}`
shell → `- run: echo "Hello world"`

Workflow triggers

Events that triggers workflows

- Trigger:

- Webhook events
- Scheduled events
- Manual event



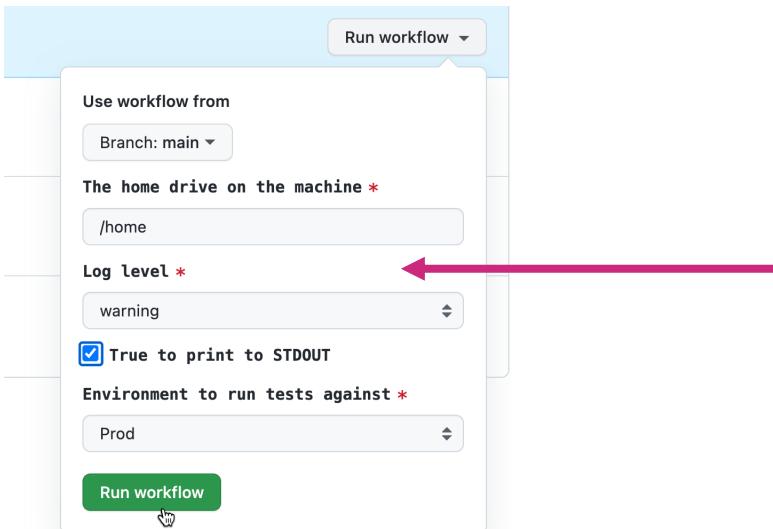
AccelerateDevOps / .github / workflows / starter.yml in main

```
<> Edit file ⌂ Preview changes

1   name: Starter Workflow
2
3   on:
4     # Webhook events
5     push:
6       branches:
7         - main
8     issues:
9       types: [opened, edited, milestoned]
10
11    # Scheduled events
12
13
14
15      schedule: Runs every 15 minutes.
16      Actions schedules run at most every 5 minutes. Learn more
16      - cron: '*/15 * * * *'
17      - cron: '0 9-17 * * *'
18      - cron: '11 11 * * 5'
19
20    # Manual events
21    workflow_dispatch:
22      inputs:
23        homedrive:
24          description: 'The home drive on the machine'
25          required: true
26          default: '/home'
27
```

Events that triggers workflows

Manual events



```
# Manual events
workflow_dispatch:
  inputs:
    homedrive:
      description: 'The home drive on the machine'
      required: true
      default: '/home'
    logLevel:
      description: 'Log level'
      required: true
      default: 'warning'
      type: choice
      options:
        - info
        - warning
        - debug
    print_tags:
      description: 'True to print to STDOUT'
      required: true
      type: boolean |
environment:
  description: 'Environment to run tests against'
  type: environment
  required: true
```

Events that triggers workflows

Manual events: trigger using the API
(*curl*, *octokit*, *Github CLI*)



```
mike@Vulcan:~/source/AccelerateDevOps$ gh api -X POST -H "Accept: application/vnd.github.v3+json" \
/repos/wulfland/AccelerateDevOps/dispatches \
-f event_type=customEvent
mike@Vulcan:~/source/AccelerateDevOps$
```

```
# Trigger using the API
repository_dispatch:
  types: [customEvent]

# Call for example using GitHub CLI:
# $ gh api -X POST -H "Accept: application/vnd.github.v3+json" \
# /repos/wulfland/AccelerateDevOps/dispatches \
# -f event_type=customEvent
```

Jobs and steps

Workflow jobs

- Map – run in parallel by default
- Can be chained using **needs** keyword
- Runs on a **runner** in one process
- Contains a sequence of steps
- Steps can be a shell command (run)
or an action (uses)

```
jobs:  
  job_1:  
    runs-on: ubuntu-latest  
  
    steps:  
      - run: echo "⚡ The job was triggered by a ${{ github.event_name }} event."  
      - run: echo "💽 drive is `${{ github.event.inputs.homedrive }}`."  
      - run: echo "🌐 environment is `${{ github.event.inputs.environment }}`."  
      - run: echo "LogLevel is `${{ github.event.inputs.logLevel }}`."  
      - run: echo "♾ Run the matrix? `${{ github.event.inputs.run_matrix }}`."  
  
  job_2:  
    runs-on: ubuntu-latest  
    needs: job_1  
    steps:  
      - run: echo "Status ${{ job.status }}"  
  
  job_3:  
    runs-on: ubuntu-latest  
    needs: job_1  
    steps:  
      - run: echo "Services ${{ job.services }}"  
  
  job_4:  
    runs-on: ubuntu-latest  
    needs: [job_2, job_3]  
    steps:  
      - run: echo "Status ${{ job.status }}"
```

Workflow steps

- Sequence in a job
- Runs in the same process / same directory
- Runs in a shell

Parameter	Description
bash	Bash shell. The default shell on all non-Windows platforms with a fallback to sh. When specified on Windows, the bash shell included with Git is used.
pwsh	PowerShell Core. Default on the Windows platform.
python	The python shell. Allows you to run python scripts
cmd	Windows only! The windows command prompt.
powershell	Windows only! The classical Windows PowerShell.

```
Get OS information
Run import platform
Linux-5.13.0-1021-azure-x86_64-with-glibc2.29

Run actions/checkout@v3.0.0

Display documentation
Run tree
.
├── About.md
├── _config.yml
└── _posts
    ├── 2021-08-10-posting-source-code.md
    ├── 2021-08-12-writing-with-markdown.md
    └── 2021-08-13-posting-in-jekyll.md
    └── about-markdown.md
    └── get-started.md
    └── index.md
1 directory, 8 files
```

```
job_1:
  runs-on: ubuntu-latest

  steps:
    - run: echo "🎉 The job was triggered by a ${{ github.event_name }} event."
    - run: echo "📍 drive is '${{ github.event.inputs.homedrive }}'."
    - run: echo "🌐 environment is '${{ github.event.inputs.environment }}'."
    - run: echo "LogLevel is '${{ github.event.inputs.logLevel }}'."
    - run: echo "💡 Run the matrix? '${{ github.event.inputs.run_matrix }}'."

    - name: Get OS information
      run:
        import platform
        print(platform.platform())
      shell: python

    - uses: actions/checkout@v3.0.0
    - name: Display documentation
      run: tree
      working-directory: docs|
```

Actions

- A reusable step
 - Lives in a git repo
 - Syntax
 - `{owner}/{repo}@{ref}`
 - `{owner}/{repo}/{path}@{ref}`
 - `./github/actions/my-action`
 - References
 - SHA / Tag / Branch
 - Pass variables to Action
 - `with:`
 - `env:`

```
❯ git remote -v
origin https://github.com/actions/checkout.git (fetch)
origin https://github.com/actions/checkout.git (push)
❯ git log --oneline --graph --decorate --all -15
* add3486 (HEAD --> main origin/main, origin/HEAD) Patch to fix the dependbot alert. (#744
* 5126516 Bump minimist from 1.2.5 to 1.2.6 (#741)
* d50f8ea Add v3.0.0 release information to changelog (#740)
* 2d1c119 update test workflows to checkout v3 (#709)
* a12a394 (tag: v3.0.0 tag: v3) update readme for v3 (#708)
* 8f9e05e Update to node 16 (#689)
* 230611d (origin/releases/v2) Change secret name for PAT to not start with GITHUB_ (#623)
* ec3a7ce (tag: v2.4.0, tag: v2) set insteadOf url for org-id (#621)
* fd47087 codeql should analyze lib not dist (#620)
* 3d677ac script to generate license info (#614)
* 826ba42 npm audit fix (#612)
* eb8a193 update dev dependencies and react to new linting rules (#611)
* c49af7c Create codeql-analysis.yml (#602)
* 1e204e9 (tag: v2.3.5) update licensed check (#606)
* 0299a0d update dist (#605)
❯
```

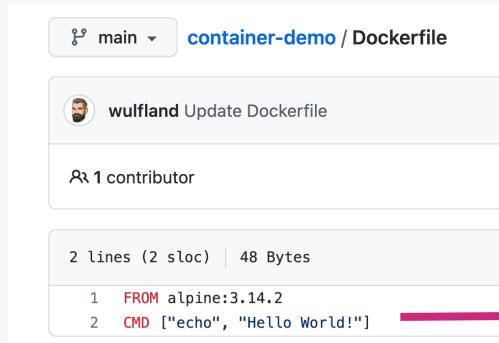
- **uses:** actions/checkout@v3.0.0
- **uses:** actions/checkout@v3
- **uses:** actions/checkout@main

Actions

User docker images as actions

```
- name: Run a docker containers as an action
  uses: docker://alpine:3.8

- uses: docker://ghcr.io/wulfland/container-demo:latest
```



A screenshot of a GitHub Actions workflow log titled 'Run ghcr.io/wulfland/container-demo:latest'. The log shows the following steps:

- ▶ Run docker://ghcr.io/wulfland/container-demo:latest
- /usr/bin/docker run --name ghcriowulflandcontainerdemolate e GITHUB_RUN_ID -e GITHUB_RUN_NUMBER -e GITHUB_RETENTION_D GITHUB_API_URL -e GITHUB_GRAPHQL_URL -e GITHUB_REF_NAME -e GITHUB_PATH -e GITHUB_ENV -e GITHUB_STEP_SUMMARY -e RUNNER ACTIONS_CACHE_URL -e GITHUB_ACTIONS=true -e CI=true -v "/v "/home/runner/work/_temp/_github_workflow":"/github/workfl ghcr.io/wulfland/container-demo:latest
- Hello World!



Demo

Hands-on: My first workflow

Contexts and expression syntax

Contexts and expressions syntax

- `$({ <expression> })`
- Context syntax
 - `context['key']` (if key starts with number or contains special characters)
 - `context.key`
- Context
 - `matrix`
 - `github`
 - `env`
 - `runner`

```
steps:  
- name: Dump runner context  
  run: echo '${{ toJSON(runner) }}'  
- name: Dump GitHub context  
  run: echo '${{ toJSON(github) }}'
```

```
  ✓ Dump runner context  
  
  1 ► Run echo '{  
  2   "  
  3     "os": "Linux",  
  4     "arch": "X64",  
  5     "name": "GitHub Actions 2",  
  6     "tool_cache": "/opt/hostedtoolcache",  
  7     "temp": "/home/runner/work/_temp",  
  8     "workspace": "/home/runner/work/AccelerateDevOps"  
  9   }  
10  
11 }
```

Context

succeeded 1 minute ago in 3s

> ✓ Set up job

✓ Dump GitHub context

```
1 ► Run echo '{  
177   "  
178     "token": "***",  
179     "job": "context_job",  
180     "ref": "refs/heads/main",  
181     "sha": "c610cff739f85138a175c892651d204e71cedb43",  
182     "repository": "wulfland/AccelerateDevOps",  
183     "repository_owner": "wulfland",  
184     "repository_owner_id": "5276337",  
185     "repositoryUrl": "git://github.com/wulfland/AccelerateDevOps.git",  
186     "run_id": "2161816664",  
187     "run_number": "32",  
188     "retention_days": "90",  
189     "run_attempt": "1",  
190     "artifact_cache_size_limit": "10",  
191     "repository_id": "383720539",  
192     "actor_id": "5276337",  
193     "actor": "wulfland",  
194     "workflow": "Starter Workflow",  
195     "head_ref": "",  
196     "base_ref": "",  
197     "event_name": "workflow_dispatch",  
198     "event": {  
199       "inputs": {  
200         "environment": "github-pages",  
201         "homedrive": "/home",  
202         "toplevel": "warning",  
203         "run_matrix": "false"  
204       }  
205     }  
206   }  
207 }
```

Contexts and expressions syntax

```
expression_job:  
  runs-on: ubuntu-latest  
  name: Expressions  
  if: ${{ github.ref == 'refs/heads/main' && github.event.inputs.logLevel == 'debug' }}  
  
  steps:  
    - run: echo "Only run if triggered by main branch..."  
      if: contains(github.ref, 'main')  
  
    - run: |  
        echo "Fail depending on parameter"  
        return 1  
      if: github.event.inputs.run_matrix == 'true'  
  
    - run: echo "Run always"  
      if: always()  
    - run: echo "Run only on success"  
      if: success()  
    - run: echo "Run only on failure"  
      if: failure()
```

```
>  Set up job  
>  Run echo "Only run if triggered by main branch..."  
↳  Run echo "Fail depending on parameter"  
  1 ► Run echo "Fail depending on parameter"  
  7 /home/runner/work/_temp/83b1e782-0ae4-47a4-8e79-afb926c6bfa5.sh: line 2: return: can only 'return' from a function or sourced script  
  8 Fail depending on parameter  
  9 Error: Process completed with exit code 1.  
  
>  Run echo "Run always"  
>  Run echo "Run only on success"  
  ↳  Run echo "Run only on failure"  
>  Run echo "Run only on failure"  
>  Complete job
```

Contexts and expressions syntax

Function	Description
success()	Returns true if none of the previous steps have failed or been cancelled.
always()	Returns true even if a previous step was cancelled and causes the step to always get executed anyway.
cancelled()	Returns only true if the workflow was canceled.
failure()	Returns true if a previous step of the job had failed.

Operator	Description
()	Logical group
!	Not
< , <=	Less than, less than or equal
> , >=	Greater than, greater than or equal
==	Equal
!=	Not equal
&&	And
	Or

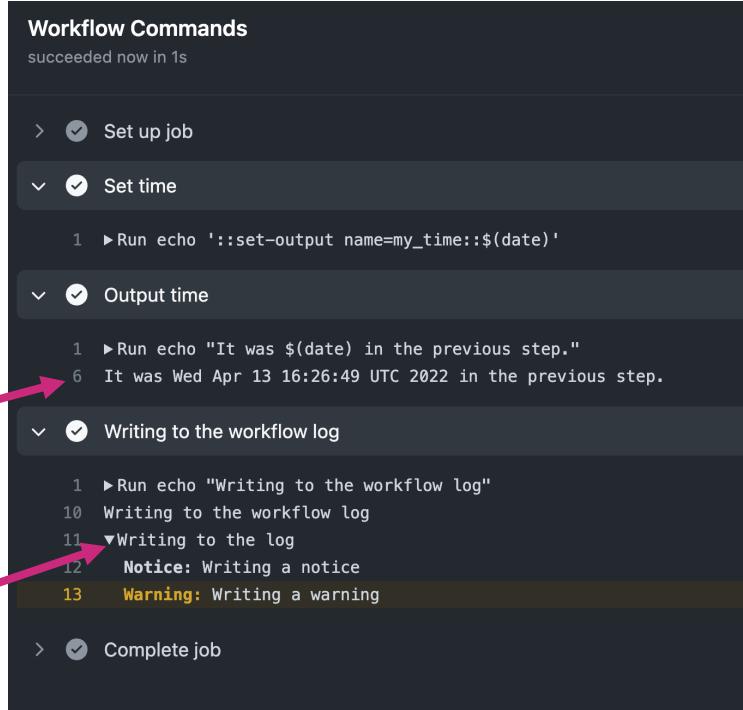
Function	Description
contains(search, item)	Returns true if search contains item.
startsWith(search, item)	Returns true if search start with item.
endsWith(search, item)	Returns true if search ends with item
format(`{0}`, item)	Replaces placeholders in a string.
join(array, separator)	All values in array are concatenated into a string.
toJSON(value)	Returns a pretty-print JSON representation of value.
fromJSON(value)	Returns a JSON object or JSON data type for value.

Workflow commands

Workflow commands

- Interact with the workflow from within your steps
- Write command to output (normally using echo)
- Examples
 - set-output
 - error

```
steps:  
  - name: Set time  
    run: echo '::set-output name=my_time::$(date)'  
    id: set-time  
  
  - name: Output time  
    run: echo "It was ${{ steps.set-time.outputs.my_time }} in the previous step."  
  
  - name: Writing to the workflow log  
    run: |  
      echo "Writing to the workflow log"  
      echo "::group::Writing to the log"  
      echo "::notice file=starter.yml,line=169,col=19,endColumn=22::Writing a notice"  
      echo "::warning file=starter.yml,line=171,col=19,endColumn=22::Writing a warning"  
      echo "::endgroup::"
```



The screenshot shows the GitHub Actions Workflow Logs for a workflow named "Workflow Commands". The logs indicate the workflow succeeded in 1 second. The steps are listed as follows:

- Set up job (passed)
- Set time (passed)
 - Run echo '::set-output name=my_time::\$(date)'
- Output time (passed)
 - Run echo "It was \$(date) in the previous step."
 - It was Wed Apr 13 16:26:49 UTC 2022 in the previous step.
- Writing to the workflow log (passed)
 - Run echo "Writing to the workflow log"
 - Writing to the workflow log
 - Writing to the log
 - Notice: Writing a notice
 - Warning: Writing a warning
- Complete job (passed)

Two pink arrows point from the highlighted code in the YAML to the corresponding log entries in the workflow logs. One arrow points from the 'set-time' step in the YAML to the 'Set time' step in the logs. Another arrow points from the 'writing-to-log' section in the YAML to the 'Writing to the workflow log' step in the logs.

Workflow commands

Toolkit function	Equivalent workflow command
<code>core.addPath</code>	Accessible using environment file <code>GITHUB_PATH</code>
<code>core.debug</code>	<code>debug</code>
<code>core.notice</code>	<code>notice</code>
<code>core.error</code>	<code>error</code>
<code>core.endGroup</code>	<code>endgroup</code>
<code>core.exportVariable</code>	Accessible using environment file <code>GITHUB_ENV</code>
<code>core.getInput</code>	Accessible using environment variable <code>INPUT_{NAME}</code>
<code>core.getState</code>	Accessible using environment variable <code>STATE_{NAME}</code>

Toolkit function	Equivalent workflow command
<code>core.isDebugEnabled</code>	Accessible using environment variable <code>RUNNER_DEBUG</code>
<code>core.saveState</code>	<code>save-state</code> Deprecated
<code>core.setCommandEcho</code>	<code>echo</code>
<code>core.setFailed</code>	Used as a shortcut for <code>::error</code> and <code>exit 1</code>
<code>core.setOutput</code>	<code>set-output</code> Deprecated
<code>core.setSecret</code>	<code>add-mask</code>
<code>core.startGroup</code>	<code>group</code>
<code>core.warning</code>	<code>warning</code>

Workflow commands

Examples

A workflow using `save-state` or `set-output` like the following

```
- name: Save state
  run: echo "::save-state name={name}::{value}"

- name: Set output
  run: echo "::set-output name={name}::{value}"
```

should be updated to write to the new `GITHUB_STATE` and `GITHUB_OUTPUT` environment files:

```
- name: Save state
  run: echo "{name}={value}" >> $GITHUB_STATE

- name: Set output
  run: echo "{name}={value}" >> $GITHUB_OUTPUT
```

More advanced syntax elements

Syntax element	Description
permissions	Set workflow permissions for GITHUB_TOKEN
env	Set environment variable for all run steps
defaults	Set the shell and working directory for the run
concurrency	Manage workflows running concurrently
needs	Make job dependent of each other. Share outputs
if	Check whether a job should run based on variables. Options are: success() always() cancelled() failure()
timeout	Limit runtime
continue-on-error	Handle termination of workflows
container	Use a container for the steps execution
services	Use a container for the steps execution



Actions

GitHub Actions

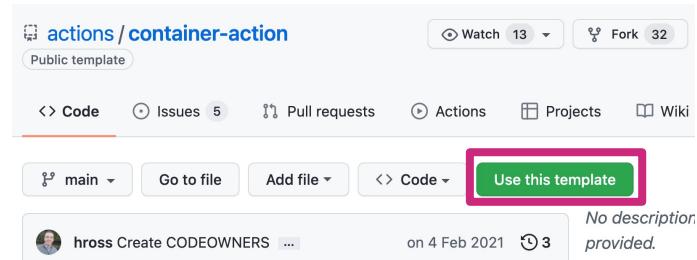
- Actions are reusable
- 3 kinds of Actions:
 - Container
 - JavaScript / Typescript
 - Composite Actions

The screenshot shows the GitHub Actions overview page. At the top, there's a logo of three blue circles connected by lines, followed by the text "GitHub Actions" and "Automate your GitHub workflows". Below that is a link to "https://github.com/features/actions" and a green "Verified" badge. The main navigation bar includes "Overview" (which is underlined in red), "Repositories 47", "Projects", "Packages", "People 19", and other icons. Below the navigation is a section titled "Pinned" which lists several actions:

- starter-workflows** (Public) - Accelerating new GitHub Actions workflows. Created by TypeScript, 5.9k stars, 4.9k forks.
- toolkit** (Public) - The GitHub ToolKit for developing GitHub Actions. Created by TypeScript, 3.2k stars, 1.1k forks.
- setup-node** (Public) - Set up your GitHub Actions workflow with a specific version of node.js. Created by TypeScript, 2k stars, 726 forks.
- javascript-action** (Public template) - Create a JavaScript Action with tests, linting, versioning. Created by JavaScript, 585 stars, 249 forks.
- typescript-action** (Public template) - Create a TypeScript Action with tests, linting, workflow, publishing, and versioning. Created by TypeScript, 1k stars, 275 forks.
- labeler** (Public) - An action for automatically labelling pull requests. Created by TypeScript, 899 stars, 308 forks.

Container Actions

- Dockerfile or existing image
- Inputs



EXPLORER HELLO-WORLD-DOCKE... action.yml CODEOWNERS Dockerfile entrypoint.sh LICENSE README.md

action.yml

```
1 name: 'Wulfland Action'
2 description: 'Get started with Container Actions'
3 author: '@wulfland'
4 branding:
5   icon: 'award'
6   color: 'green'
7 inputs:
8   myInput:
9     description: 'Input to use'
10    default: 'world'
11 runs:
12   using: 'docker'
13   image: 'Dockerfile'
14   args:
15     - ${{ inputs.myInput }}
```

Dockerfile

```
1 FROM alpine:3.10
2
3 COPY LICENSE README.md /
4
5 COPY entrypoint.sh /entrypoint.sh
6
7 ENTRYPOINT ["/entrypoint.sh"]
```

entrypoint.sh

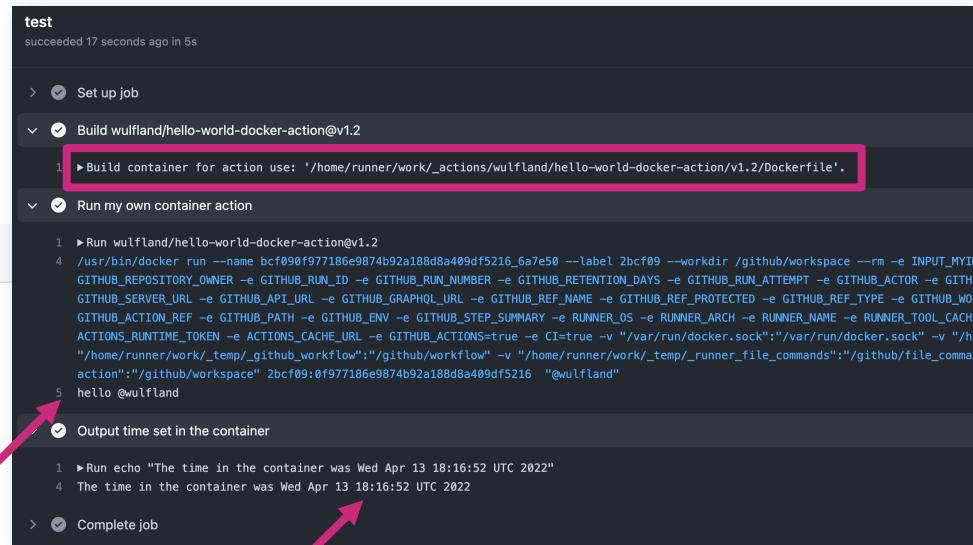
```
$! /bin/sh -l
1 #!/bin/sh -l
2
3 echo "hello $1"
4
```

Container Actions

- Dockerfile or existing image
- inputs

```
15 lines (13 sloc) | 398 Bytes

1 name: Test Action
2
3 on: [workflow_dispatch]
4
5 jobs:
6   test:
7     runs-on: ubuntu-latest
8     steps:
9       - name: Run my own container action
10         id: hello-action
11         uses: wulfland/hello-world-docker-action@v1.2
12         with:
13           myInput: '@wulfland'
14       - name: Output time set in the container
15         run: echo "The time in the container was ${{ steps.hello-action.outputs.time }}"
```



JavaScript Actions

The screenshot illustrates the process of creating a new GitHub Action using a template. The repository structure is as follows:

- MY-TYPESCRIPT-ACTION [GITHUB]**:
 - _tests_
 - .github
 - dist:
 - index.js
 - index.js.map
 - licenses.txt
 - sourcemap-register.js
 - src:
 - main.ts
 - wait.ts
 - .eslintrc.json
 - .gitignore
 - .gitignore
 - .prettierignore
 - .prettierrc.json
 - CODEOWNERS
 - jest.config.js
 - LICENSE
 - package-lock.json
 - package.json
 - README.md
 - tsconfig.json

The `action.yml` file contains the configuration for the action:

```
name: 'My TS Action'
description: 'Test action for TS'
author: '@wulfland'

inputs:
  milliseconds:
    required: true
    description: Waits for the amount of milliseconds
    default: '1000'

outputs:
  time:
    description: the time after waiting.

runs:
  using: 'node16'
  main: 'dist/index.js'
```

The `main.ts` file contains the logic for the action:

```
import * as core from '@actions/core'
import {waitFor} from './Wait'

async function run(): Promise<void> {
  try {
    const ms: string = core.getInput('milliseconds')
    core.debug(`Waiting ${ms} milliseconds ...`)
    core.debug(new Date().toTimeString())
    await waitFor(parseInt(ms, 10))
    core.debug(new Date().toTimeString())

    core.setOutput('time', new Date().toTimeString())
  } catch (error) {
    if (error instanceof Error) core.setFailed(error.message)
  }
}

run()
```

A pink box highlights the "Use this template" button in the GitHub interface, indicating the source of the template used for this repository.

Composite Actions

- Just a `action.yml` file
- Inputs
- Outputs
- Runs

28 lines (25 sloc) | 689 Bytes

```
1 name: 'Hello World'
2 description: 'Greet someone'
3 inputs:
4   who-to-greet:
5     description: 'Who to greet'
6     required: true
7     default: 'World'
8 outputs:
9   random-number:
10    description: "Random number"
11    value: ${{ steps.random-number-generator.outputs.random-id }}
12 runs:
13   using: "composite"
14   steps:
15     - run: echo Hello ${inputs.who-to-greet}
16       shell: bash
17
18     - id: random-number-generator
19       run: echo "$RANDOM" | sed -e 's/\([0-9]\{1\}\)/\1/g' | head -n 1 | awk '{print "0x" $0}'
20       shell: bash
21
22     - run: echo "${github.action_path}" >> $GITHUB_PATH
23       shell: bash
24
25     - run: echo "Goodbye $YOU"
26       shell: bash
27       env:
28         YOU: ${inputs.who-to-greet}
```



Share your action in the marketplace

- Really easy to share
- Draft a release
- Unique name
- Check for README, Icon Color

```
name: 'Wulfland Action'  
description: 'Get started with Container actions'  
author: '@wulfland'  
branding:  
  icon: 'award'  
  color: 'green'
```

The screenshot shows the GitHub Marketplace release draft interface. At the top, there are tabs for 'Releases' (which is selected) and 'Tags'. Below the tabs, the title 'Release Action' is displayed, followed by a checked checkbox for 'Publish this Action to the GitHub Marketplace'. A green success message indicates that 'Everything looks good! You have all the required information.' The configuration details are listed in a table:

action.yml	<input type="button" value="Edit"/>
✓ Name	Wulfland Action
✓ Description	Get started with Container actions
✓ Icon	award
✓ Color	green

Below this, another section for 'README' is shown, indicating that 'A README exists.' At the bottom, there are dropdown menus for 'Primary Category' and 'Another Category — optional', both currently set to 'Choose an option'.



Demo

Hands-on: My first Action

The GitHub API and Octokit

Using the GitHub API

- REST API (v3)
 - Libraries available for most languages
 - Octokit
- GraphQL (v4)
 - The future of the GitHub API
 - A query language allowing granular control of request and response

The screenshot shows a browser window displaying the GitHub API documentation for the `octokit.rest.repos.createRelease` endpoint. The URL in the address bar is `octokit.github.io/test/v3/repos#create-release`. The left sidebar has a tree view with 'Create a release' selected under the 'Repos' category. The main content area is titled 'Create a release' and describes it as an endpoint for users with push access to create a release. It includes notes about notifications and abuse rate limits. Below this is a code snippet in JavaScript:

```
octokit.rest.repos.createRelease({  
  owner,  
  repo,  
  tag_name,  
});
```

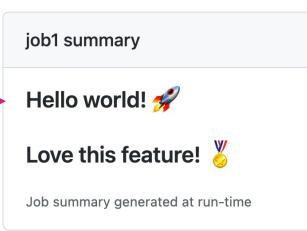
Underneath the code, there is a table for 'Parameters' with the following columns: name, required, and description.

name	required	description
owner	yes	
repo	yes	
tag_name	yes	The name of the tag.
target_commitish	no	Specifies the commitish value that determines where the Git tag is created from. Can be any branch or commit SHA. Unused if the Git tag already exists. Default: the repository's default branch (usually <code>master</code>).

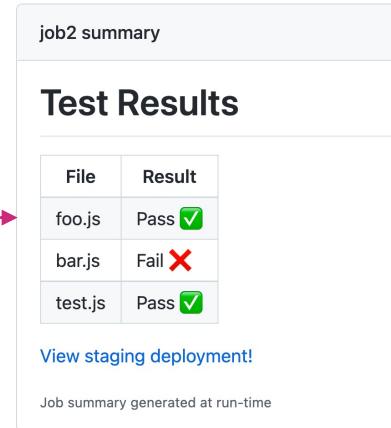
Job Summaries

```
jobs:  
  job1:  
    runs-on: ubuntu-latest
```

```
steps:  
  - run: echo '## Hello world! :rocket:' >> $GITHUB_STEP_SUMMARY  
  - run: echo '## Love this feature! :medal_sports:' >> $GITHUB_STEP_SUMMARY
```



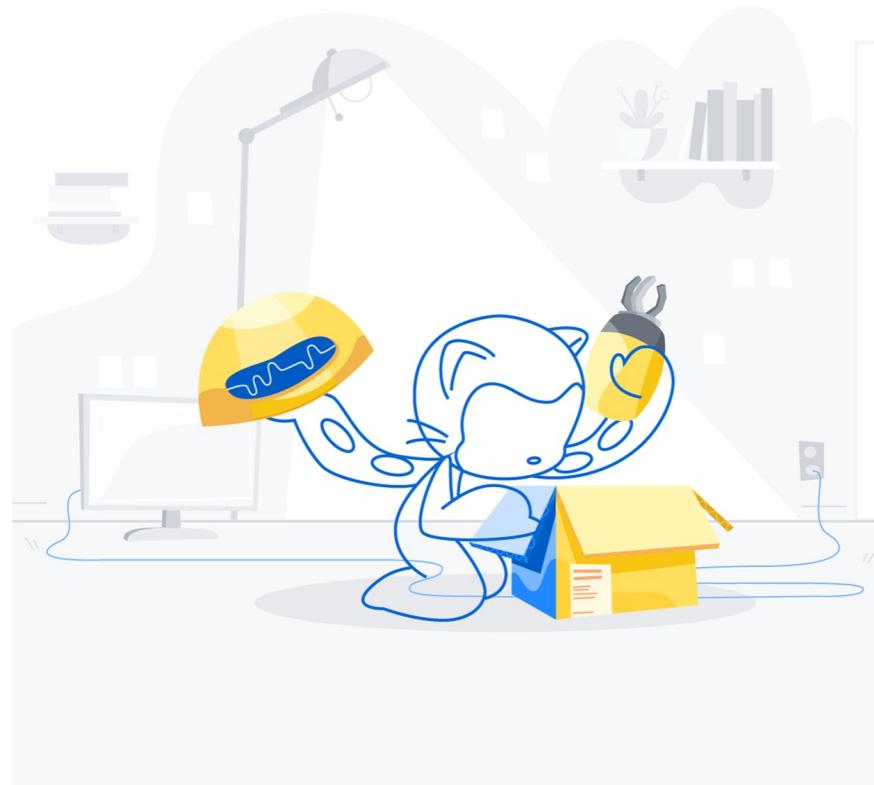
```
steps:  
  - name: Write Summary from Action  
    uses: actions/github-script@v6.1.0  
    with:  
      script: |  
        await core.summary  
        .addHeading('Test Results')  
        .addTable([  
          {data: 'File', header: true}, {data: 'Result', header: true}],  
          ['foo.js', 'Pass ✅'],  
          ['bar.js', 'Fail ❌'],  
          ['test.js', 'Pass ✅'])  
        .addLink('View staging deployment!', 'https://github.com')  
        .write()
```



Writing Actions

Best Practices

- Design for reusability
- Small and focused (Single Responsibility Principle)
- Write tests and a test workflow
- Semantic versioning
- Documentation
- Proper `action.yml` metadata
- github.com/actions/toolkit
- Publish the Action to the marketplace

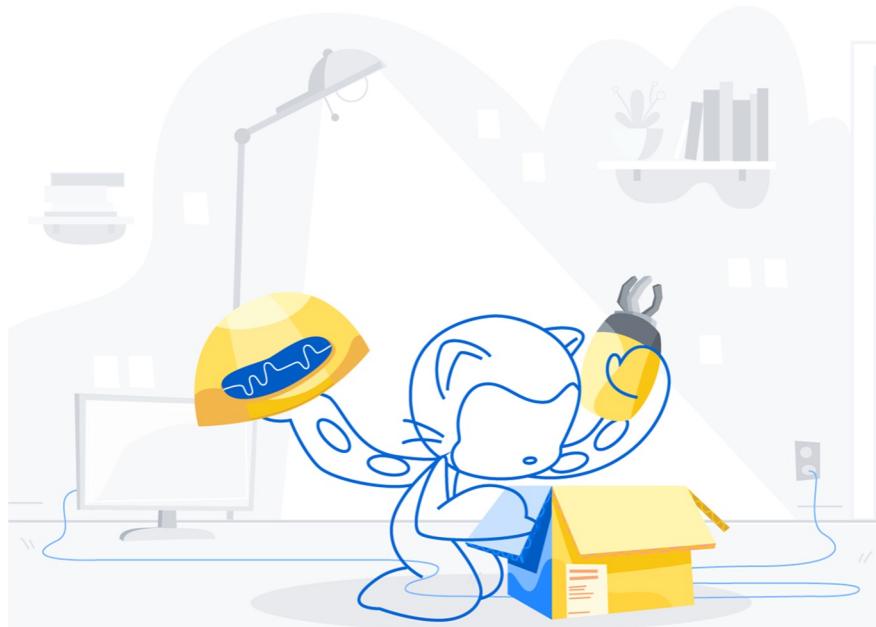


Retro Day 1

Introduction Day 2

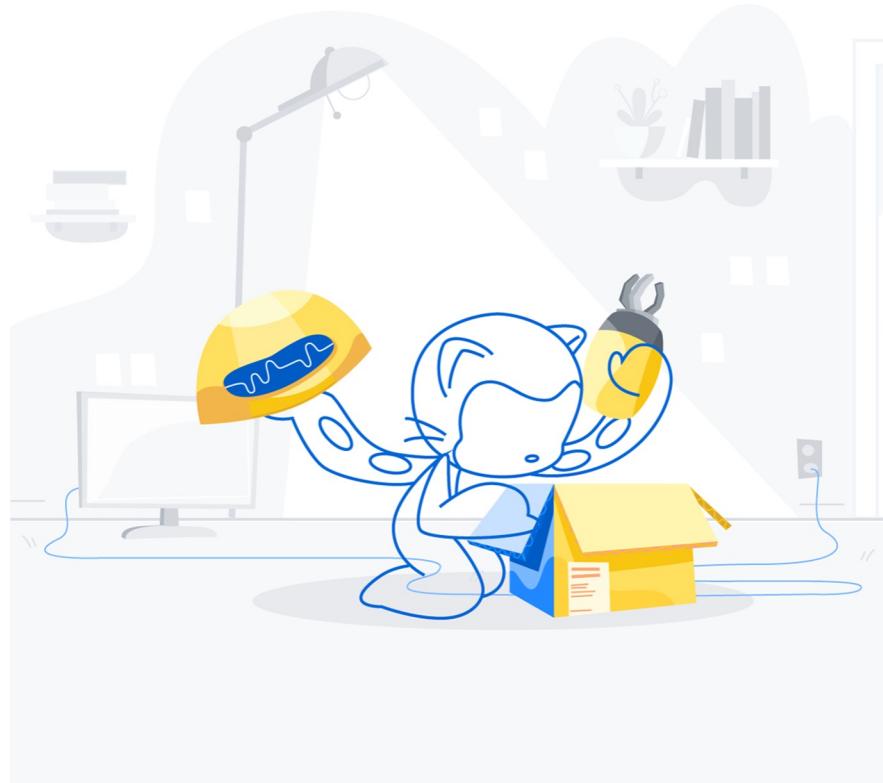
Agenda – day 1

- Workflow fundamentals
- YAML and workflow syntax
- Writing workflows
- GitHub Actions
- Authoring Actions



Agenda – day 2

- CI/CD with GitHub Actions
- Secrets, Variables and Environments
(staged deployments)
- Running workflows (runners)
- Action policies
- Sharing workflows (reusable workflows)



Icebreaker

Actions for CI / CD

Strategy

- For-loop – array
- Nested for-loops:
multidimensional array
- Runs for all combinations in all dimensions
- Fail-fast (yes/no)
- Max 256 parallel jobs

```
matrix_job:  
  name: matrix-job  
  runs-on: ${{ matrix.runner }}  
  if: github.event.inputs.run_matrix  
  
  strategy:  
    matrix:  
      runner: [ubuntu-18.04, ubuntu-20.04]  
      node: [10,12]  
  
    steps:  
      - run: echo "${{ matrix.runner }}"  
      - run: echo "${{ matrix.node }}"
```

✓ Starter Workflow Starter Workflow #17

Summary

Jobs

✓ job_1

✓ matrix-job (ubuntu-18.04, 10)

✓ matrix-job (ubuntu-18.04, 12)

✓ matrix-job (ubuntu-20.04, 10)

✓ matrix-job (ubuntu-20.04, 12)

matrix-job (ubuntu-20.04, 12)
succeeded now in 0s

> ✓ Set up job

> ✓ Run echo "ubuntu-20.04"

> ✓ Run echo "12"

> ✓ Complete job

Basic CI workflow

- Uses a build matrix across multiple node versions
- Runs on the VM
 - Ubuntu in this case
- Actions are composable
- Checkout is separate
- Setup for most languages in github.com/actions
- npm run by shell
- Artifact upload is a separate action

```
name: Node CI

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [10.x, 12.x]

    steps:
      - uses: actions/checkout@v2
      - name: Use Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v2
        with:
          node-version: ${{ matrix.node-version }}
      - name: Install and test
        run: |
          npm ci
          npm run build --if-present
          npm test
      - uses: actions/upload-artifact@v2
        with:
          name: artifact
          path: dist/
```

Linting

- Linting as part of CI runs
- See e.g. the super-linter
 - github.com/github/super-linter
 - Supports ~45 different languages
- Easily added as a new step to an existing workflow

```
name: Lint Code Base

on:
  push:
    branches-ignore: [main]
  pull_request:
    branches: [main]

jobs:
  build:
    name: Lint Code Base
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v2
        with:
          fetch-depth: 0

      - name: Lint Code Base
        uses: github/super-linter@v4
        env:
          VALIDATE_ALL_CODEBASE: false
          DEFAULT_BRANCH: main
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Caching

Optimizing your workflow performance with caching:

- Temporarily save files between workflow runs
- 10GB max cache size per repo
- 7 days retention
- Scoped to key and branch
- Never cache sensitive data



[Caching dependencies](#) to speed up workflows

Caching can help with speeding up workflows when you need to install dependencies. NPM, Python, Ruby, etc... these are simple examples of applications that require dependencies to be built. But there are more complex scenarios, such as Java, C/C++ and modularized microservices that often require downstream artifacts. Caching can speed up your builds when your dependencies have not changed

Caching

```
steps:  
- uses: actions/checkout@v3  
  
- name: Cache Primes  
  id: cache-primes  
  uses: actions/cache@v3  
  with:  
    path: primes  
    key: ${{ runner.os }}-primes  
  
- name: Generate Prime Numbers  
  if: steps.cache-primes.outputs.cache-hit != 'true'  
  run: |  
    sleep 60  
    echo "1 2 3..." > primes  
  
- name: Use Prime Numbers  
  run: cat primes
```

Java - Maven

```
- name: Cache local Maven repository  
  uses: actions/cache@v3  
  with:  
    path: ~/.m2/repository  
    key: ${{ runner.os }}-maven-${{ hashFiles('**/pom.xml') }}  
    restore-keys: |  
      ${{ runner.os }}-maven-
```

CI with Actions

Best Practices

- Always use `setup` actions
- Implement caching if (only) needed (`cache` action)
- Use the `matrix` strategy to build and test multiple versions
- Use `upload-artifact`
- Use the super-linter:
 - `github/super-linter:v4`
 - `github/super-linter:slim-v4`
- Use tests and job summaries to display results
- Require status checks for pull requests



Secrets & variables

GitHub Secret store

- Built-in secret store
- Encrypted
 - LibSodium sealed box
- Use directly from your workflow
- Redacted in workflow logs
- API support
- Organization / repository / environment level secrets
- Do not use structured data!

The screenshot shows the GitHub Secrets and Variables interface. On the left, a sidebar lists various repository settings: General, Access, Collaborators and teams, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Pages), Security, Code security and analysis, Deploy keys, Secrets and variables (selected), Actions, Codespaces, Dependabot, Integrations, GitHub Apps, and Email notifications. A 'Beta' badge is visible next to the Actions section.

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for non-sensitive data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets **Variables** **New repository secret**

Environment secrets **Manage environments**

MY_ENV_SECRET Demo Updated 3 minutes ago

Repository secrets

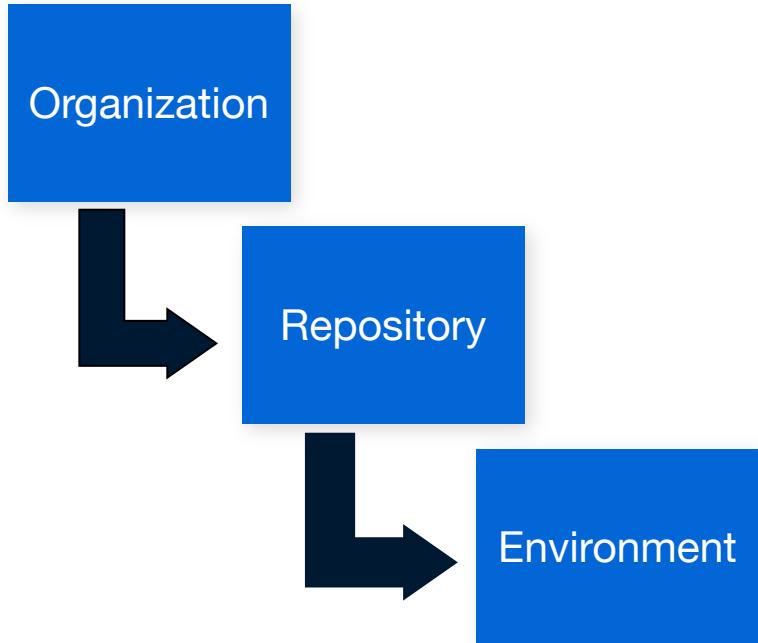
MY_REPO_SECRET Updated 2 minutes ago [Edit](#) [Delete](#)

Organization secrets **Manage organization secrets**

MY_ORG_SECRET Updated on Apr 25 [Edit](#) [Delete](#)

Secrets

- Defined on org, repo, or environment level
- Secret context
 - `$(secrets.MY_SECRET)`
 - Set as input (with:) or environment (env:) for actions
- Set in UI or CLI
 - `$ gh secret set MY_SECRET --body "$value"`
 - `$ gh secret set MY_SECRET --env Prod`
 - `$ gh secret set MY_SECRET --org my-org`
- Masked in log



The GITHUB_TOKEN

- `${{ secrets.GITHUB_TOKEN }}` or `${{ github.token }}`
- Authenticate to GitHub to perform automation inside the workflow's repo
- Default permission read/write for all scopes (old default) or set to read

```
permissions:  
  contents: read  
  pull-requests: write
```

```
permissions: read-all
```

```
permissions:  
  actions: read|write|none  
  checks: read|write|none  
  contents: read|write|none  
  deployments: read|write|none  
  issues: read|write|none  
  packages: read|write|none  
  pull-requests: read|write|none  
  repository-projects: read|write|none  
  security-events: read|write|none  
  statuses: read|write|none
```

The GITHUB_TOKEN

Perform actions as “github-actions”:

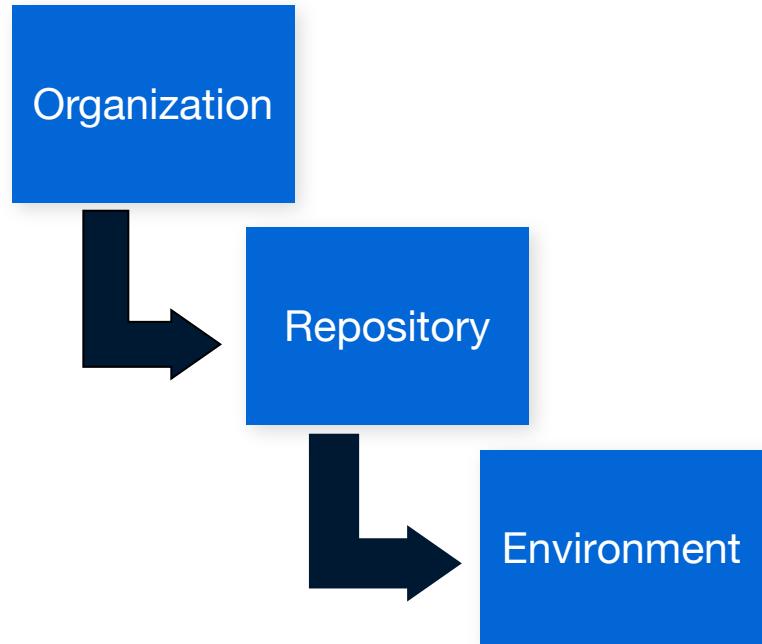
```
permissions:  
  contents: read  
  issues: write  
  
label_issues:  
  runs-on: ubuntu-latest  
  if: github.event_name == 'issues'  
  
steps:  
  - uses: andymckay/labeler@e6c4322d0397f3240f0e7e30a33b5c5df2d39e90  
    with:  
      add-labels: documentation  
      repo-token: ${{ secrets.GITHUB_TOKEN }}
```



github-actions (bot) added the documentation label 20 seconds ago

Variables

- Same setup as secrets, but no redacting
- Defined on org, repo, or environment level
- vars context
 - \${ { vars.MY_VAR } }
 - Set as input (with:) or environment (env:) for actions
- **Not** masked in log



Environments

Environments

- Control deployments
- Add gated deployments with approvals
- Control secrets
- Review all deployments to an env
- Navigate directly to urls for deployments
- Fully integrated with the checks API (previously called deployment API)
- Supports matrix for gated deployments

Environments / Configure Development

Environment protection rules

Can be used to configure manual approvals and timeouts.

Required reviewers

Specify people or teams that may approve workflow runs when they access this environment.

Add up to 6 more reviewers

Search for people or teams...

Wait timer

Set an amount of time to wait before allowing deployments to proceed.

15 minutes

[Save protection rules](#)

Deployment branches

Can be used to limit what branches can deploy to this environment using branch name patterns.

All branches ▾

Environment secrets

Secrets are encrypted environment variables. They are accessible only by GitHub Actions in the context of this environment.

[⊕ Add Secret](#)

Environments

- Environments

- Reviewers / Approvers
- Wait timer (until 30 days)
- Branches (→ branch protection!)
- Deployment branches
- Secrets

```
Test:  
  runs-on: ubuntu-latest  
  environment: Test  
  needs: Build  
  steps:  
    - name: Test app  
      run: echo "Testing..."
```

```
Production:  
  runs-on: ubuntu-latest  
  environment:  
    name: prod  
    url: https://writeabout.net  
  needs: Staging  
  steps:  
    - name: Deploy app  
      run: echo "Deploying..."
```

Environments / Configure Test

Environment protection rules

Can be used to configure manual approvals and timeouts.

Required reviewers

Specify people or teams that may approve workflow runs when they access this environment.

Add up to 5 more reviewers

Search for people or teams...



wulfand

x

Wait timer

Set an amount of time to wait before allowing deployments to proceed.

Save protection rules

Deployment branches

Can be used to limit what branches can deploy to this environment using branch name patterns.

All branches ▾

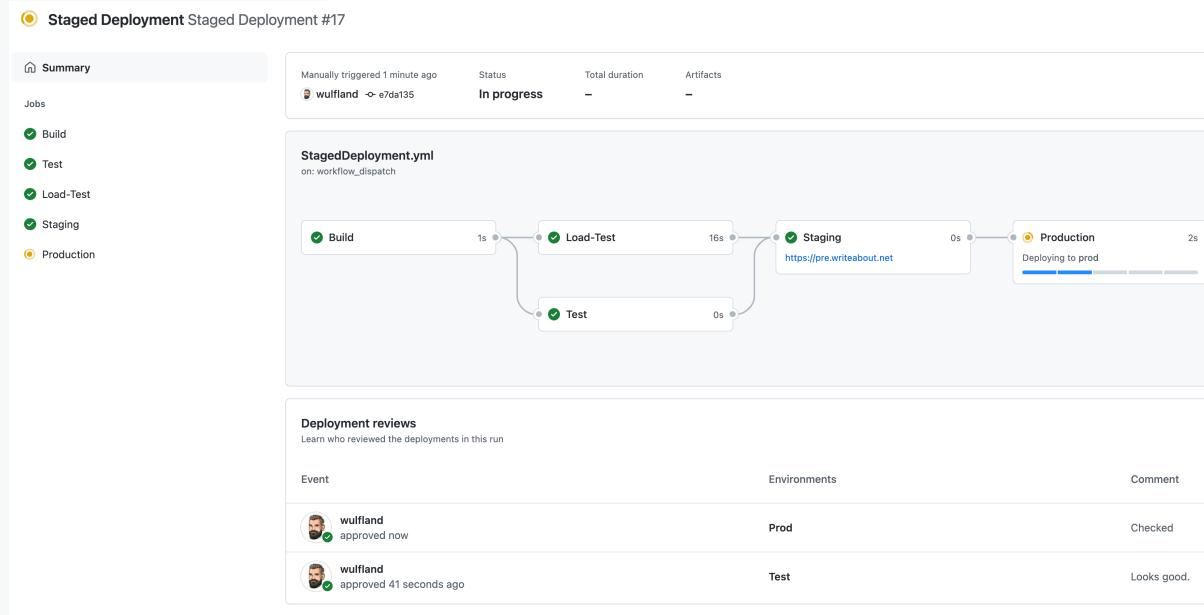
Environment secrets

Secrets are encrypted environment variables. They are accessible only by GitHub Actions in the context of this environment.

+ Add Secret

Environments

- Approvals
- Secrets after approval
- Set URL from output of other job/step
- Progress





Demo

Hands-on: Staged deployments

Basic CD workflow

- Starter workflows available for most cloud providers
- Store the images in GitHub
- Jobs run on different envs
 - Uses the Docker image
 - Deploys the container image to Azure
- Use `login` actions
- Security hardening with **OpenID Connect**

```
35 Build-Docker-Image:  
36   runs-on: ubuntu-latest  
37   needs: build  
38   name: Build image and store in GitHub Packages  
39   steps:  
40     - name: Checkout  
41       uses: actions/checkout@v1  
42  
43     - name: Download built artifact  
44       uses: actions/download-artifact@master  
45       with:  
46         name: webpack artifacts  
47         path: public  
48  
49     - name: create image and store in Packages  
50       uses: mattdavis0351/actions/docker-gpr@1.3.0  
51       with:  
52         repo-token: ${{secrets.GITHUB_TOKEN}}  
53         image-name: ${{env.DOCKER_IMAGE_NAME}}  
54  
55 Deploy-to-Azure:  
56   runs-on: ubuntu-latest  
57   needs: Build-Docker-Image  
58   name: Deploy app container to Azure  
59   steps:  
60     - name: "Login via Azure CLI"  
61       uses: azure/login@v1  
62       with:  
63         creds: ${ secrets.AZURE_CREDENTIALS }  
64  
65     - uses: azure/docker-login@v1  
66       with:  
67         login-server: ${env.IMAGE_REGISTRY_URL}  
68         username: ${github.actor}  
69         password: ${ secrets.GITHUB_TOKEN }  
70  
71     - name: Deploy web app container  
72       uses: azure/webapps-container-deploy@v1  
73       with:  
74         app-name: ${env.AZURE_WEBAPP_NAME}  
75         images: ${env.IMAGE_REGISTRY_URL}/${github.repository}/${env.DOCKER_IMAGE_NAME}:${{ github.sha }}  
76  
77     - name: Azure logout  
78       run: |  
79         az logout
```

Concurrency

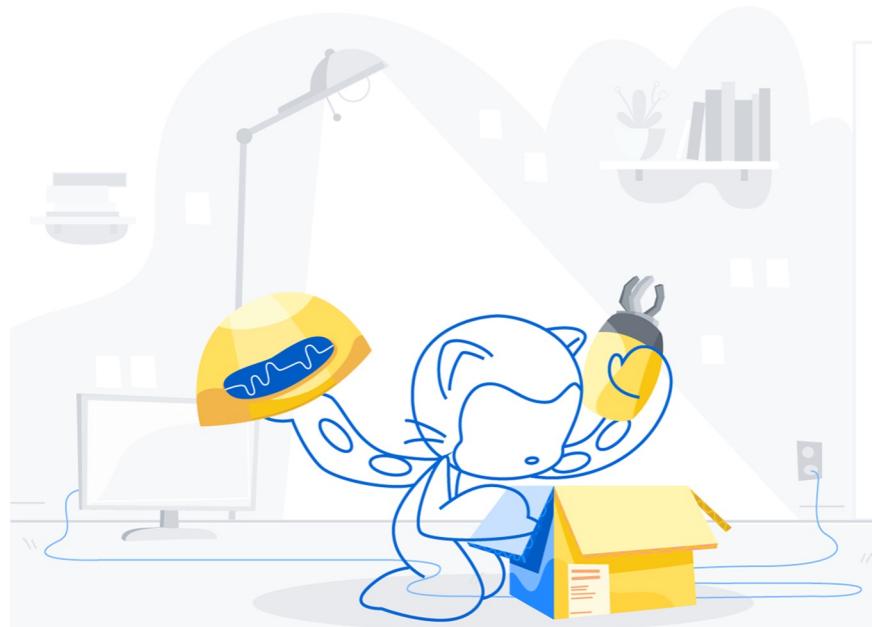
- Workflow or job
- Optional: cancel in-progress jobs
- Use cases:
 - Wait job/workflow until deployment completed
 - Cancel deployment and deploy newer version instead

```
1 name: Concurrency
2
3 on: [workflow_dispatch]
4
5 jobs:
6   job1:
7     concurrency: group1
8     runs-on: ubuntu-latest
9     steps:
10      - run: sleep 60
11      - run: echo "Hello World! $(date)"
12
13   job2:
14     concurrency: group1
15     runs-on: ubuntu-latest
16     steps:
17      - run: sleep 60
18      - run: echo "Hello World! $(date)"
19
20   job3:
21     concurrency:
22       group: group2
23       cancel-in-progress: true
24
25     runs-on: ubuntu-latest
26     steps:
27      - run: sleep 60
28      - run: echo "Hello World! $(date)"
29
30   job4:
31     concurrency:
32       group: group2
33       cancel-in-progress: true
34
35     runs-on: ubuntu-latest
36     steps:
37      - run: sleep 60
38      - run: echo "Hello World! $(date)"
```

CD with Actions

Best Practices

- Use `download-artifact`
- Use `concurrency` groups
- Use staged deployments
(Environments)
- Pull request: require deployments to succeed before merging
- Use container registry
- Use login actions
- Use OpenID Connect (**OIDC**)





Action Policies

Actions policies

- Configure Actions policies on enterprise / organization / repository level
 - Which Actions are allowed
 - Artifact retention period
 - Running workflows from fork PRs
 - Permissions of GITHUB_TOKEN

The screenshot shows the GitHub Actions policies configuration interface. It includes sections for General settings, Actions permissions, artifact retention, fork pull request workflows, and workflow permissions.

General

- Access
- Collaborators
- Moderation options
- Code and automation
 - Branches
 - Tags
 - Rules
 - Actions** (selected)
 - General
 - Runners
 - Webhooks
 - Environments
 - Codespaces
 - Pages
- Security
 - Code security and analysis
 - Deploy keys
 - Secrets and variables
- Integrations
 - GitHub Apps
 - Email notifications

Actions permissions

- Allow all actions and reusable workflows
- Disable actions
- Allow aatmmr actions and reusable workflows
- Allow aatmmr, and select non-aatmmr, actions and reusable workflows

Any action or reusable workflow can be used, regardless of who authored it or where it is defined.
The Actions tab is hidden and no workflows can run.
Any action or reusable workflow defined in a repository within aatmmr can be used.
Any action or reusable workflow that matches the specified criteria, plus those defined in a repository within aatmmr, can be used. [Learn more about allowing specific actions and reusable workflows to run](#).

Save

Artifact and log retention

Choose the repository settings for artifacts and logs.

Artifact and log retention

90 days **Save**

Your organization has set a maximum limit of 90 days. [Learn more](#).

Fork pull request workflows from outside collaborators

Choose which subset of outside collaborators will require approval to run workflows on their pull requests. [Learn more about approving workflow runs from public forks](#).

- Require approval for first-time contributors who are new to GitHub
- Require approval for first-time contributors
- Only first-time contributors who recently created a GitHub account will require approval to run workflows.
- Require approval for all outside collaborators

Save

Workflow permissions

Choose the default permissions granted to the GITHUB_TOKEN when running workflows in this repository. You can specify more granular permissions in the workflow using YAML. [Learn more](#).

- Read and write permissions
- Read repository contents and packages permissions
- Workflows have read permissions in the repository for the contents and packages scopes only.

Choose whether GitHub Actions can create pull requests or submit approving pull request reviews.

Allow GitHub Actions to create and approve pull requests

Save



Demo

Running your workflows

Runners

GitHub-hosted

- Receive automatic updates for the operating system, pre-installed packages and tools, and the self-hosted runner application.
- Are managed and maintained by GitHub.
- Provide a clean instance for every job execution.
- Use free minutes on your GitHub plan, with per-minute rates applied after surpassing the free minutes.

Self-hosted

- Receive automatic updates for the self-hosted runner application only. You are responsible updating the operating system and all other software.
- Can use cloud services or local machines that you already pay for.
- Are customizable to your hardware, operating system, software, and security requirements.
- Don't need to have a clean instance for every job execution.
- Are free to use with GitHub Actions, but you are responsible for the cost of maintaining your runner machines.

GitHub hosted runners

Linux Windows	MacOS
Hardware: <ul style="list-style-type: none">• Standard_DS2_v2 virtual machines in Microsoft Azure• 2-core CPU• 7 GB of RAM• 14 GB of SSD disk space	Hardware: <ul style="list-style-type: none">• 3-core CPU• 14 GB of RAM• 14 GB of SSD disk space
Passwordless sudo / UAC disabled	Passwordless sudo

Runner Images

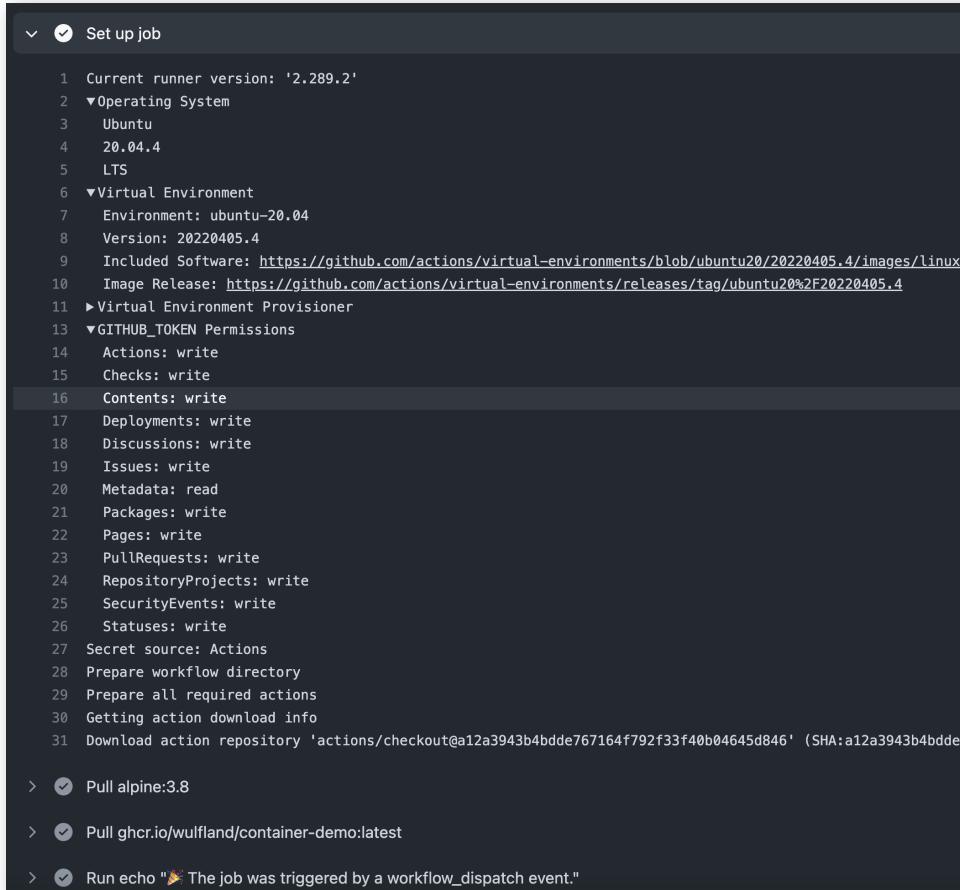
Environment	YAML label	Included Software
Ubuntu 22.04	ubuntu-latest or ubuntu-22.04	<u>ubuntu-22.04</u>
Ubuntu 20.04	ubuntu-20.04	<u>ubuntu-20.04</u>
macOS 11	macos-latest or macos-11	<u>macOS-11</u>
macOS 10.15	macos-10.15	<u>macOS-10.15</u>
Windows Server 2022	windows-latest or windows-2022	<u>windows-2022</u>
Windows Server 2019	windows-2019	<u>windows-2019</u>
Windows Server 2016	windows-2016	<u>windows-2016</u>

<https://github.com/actions/runner-images>

GitHub hosted runners pricing

- Build minutes
 - On Linux \$0.008
 - On Windows \$0.016 = x2
 - On macOS \$0.080 = x10

GitHub edition	Storage	Minutes	Max concurrent jobs
GitHub Free	500 MB	2,000	20 (5 for macOS)
GitHub Pro	1 GB	3,000	40 (5 for macOS)
GitHub Free for organizations	500 MB	2,000	20 (5 for macOS)
GitHub Team	2 GB	3,000	60 (5 for macOS)
GitHub Enterprise Cloud	50 GB	50,000	180 (50 for macOS)



The screenshot shows the "Set up job" configuration in GitHub Actions. It includes environment variables like runner version, operating system (Ubuntu 20.04 LTS), virtual environment (ubuntu-20.04), and GITHUB_TOKEN permissions. Below this, a workflow script is shown with three steps: pulling alpine:3.8, pulling ghcr.io/wulfland/container-demo:latest, and running an echo command.

```
1 Current runner version: '2.289.2'
2 ▼Operating System
3   Ubuntu
4   20.04.4
5   LTS
6 ▼Virtual Environment
7   Environment: ubuntu-20.04
8   Version: 20220405.4
9   Included Software: https://github.com/actions/virtual-environments/blob/ubuntu20/20220405.4/images/linux
10  Image Release: https://github.com/actions/virtual-environments/releases/tag/ubuntu20%2F20220405.4
11  ▶ Virtual Environment Provisioner
12  ▼GITHUB_TOKEN Permissions
13    Actions: write
14    Checks: write
15    Contents: write
16    Deployments: write
17    Discussions: write
18    Issues: write
19    Metadata: read
20    Packages: write
21    Pages: write
22    PullRequests: write
23    RepositoryProjects: write
24    SecurityEvents: write
25    Statuses: write
26    Secret source: Actions
27    Prepare workflow directory
28    Prepare all required actions
29    Getting action download info
30    Download action repository 'actions/checkout@v2' (SHA:a12a3943b4bdde767164f792f33f40b04645d846' (SHA:a12a3943b4bdde767164f792f33f40b04645d846)
31
> ✓ Pull alpine:3.8
> ✓ Pull ghcr.io/wulfland/container-demo:latest
> ✓ Run echo "💡 The job was triggered by a workflow_dispatch event."
```

Larger runners (beta)

Runners / Create GitHub-hosted runner

Name

Runner image

 Ubuntu  Windows Server

Ubuntu version

GitHub images are kept up to date and secure, containing all the tools you need to get started building and testing your applications. [Learn more about images](#).

"Latest" tag matches with standard GitHub-hosted runners latest tag for the images. [Learn more about latest tags](#).

Latest (20.04)

Runner size

4-cores · 16 GB RAM · 150 GB HDD

4-cores
16 GB RAM · 150 GB HDD

8-cores
32 GB RAM · 300 GB HDD

16-cores
64 GB RAM · 600 GB HDD

32-cores
128 GB RAM · 1200 GB HDD

64-cores
256 GB RAM · 2040 GB HDD

Any repository can use the runner. [Learn more about runner groups](#).

Networking

Assign a unique & static public IP address range for this runner

All instances of this GitHub-hosted runner will be assigned a static IP from a range unique to this runner. [Learn more about networking for runners](#).

You have used 0 out of 10 static public IP addresses available on your account.

Runners /  test-16 Shutdown

Runner group: BIG

Size: 16-cores · 64 GB RAM · 600 GB HDD

Image: Ubuntu Latest (20.04)

Public IP range: 20.237.78.80/28

<https://docs.github.com/en/actions/using-github-hosted-runners/using-larger-runners>



Demo

Self-hosted runners

- **Free**
- **Any platform** (x64: Linux, macOS, Windows. ARM64 and ARM32 on Linux)
- **HTTPS long polling** port 443 – 50 seconds
- Can be used to **deploy to local resources**
- Can be added at **Enterprise, Organization, and Repository level**

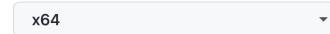
Runners / Create self-hosted runner

Adding a self-hosted runner requires that you download, configure, and execute the GitHub Actions Runner. By downloading and configuring the GitHub Actions Runner, you agree to the [GitHub Terms of Service](#) or [GitHub Corporate Terms of Service](#), as applicable.

Runner image



Architecture



Download

```
# Create a folder
$ mkdir actions-runner && cd actions-runner
# Download the latest runner package
$ curl -o actions-runner-linux-x64-2.289.2.tar.gz -L
https://github.com/actions/runner/releases/download/v2.289.2/actions-runner-linux-x64-
2.289.2.tar.gz
# Optional: Validate the hash
$ echo "7ba89bb75397896a76e98197633c087a9499d4c1db7603f21910e135b0d0a238" actions-runner-linux-x64-
2.289.2.tar.gz" | shasum -a 256 -c
# Extract the installer
$ tar xzf ./actions-runner-linux-x64-2.289.2.tar.gz
```

Self-hosted runners

- Access: Runner Groups
- A runner can only be in 1 group
- Can be moved to another group
- Apply labels
 - `$./config.sh --labels self-hosted, x64, linux, matlab`
 - `runs-on: [self-hosted, linux, X64, matlab]`

Runner Groups / New Runner Group

Group name

Repository access

Selected repositories ▾ 1 selected repository ⓘ

Allow public repositories

Runners can be used by public repositories. Allowing self-hosted runners on public repositories and allowing workflows on public forks introduces a significant security risk. [Learn more](#)

Workflow access

Control how these runners are used by restricting them to specific workflows. [Learn more](#)

Selected workflows ▾ 0 selected workflows ⓘ

[Create group](#)

Adding self-hosted runners

- Configure on enterprise / organization / repository level
- Download and extract the scripts
- Configure and authenticate the runner with the token
- Start listening for jobs
- For GHES: Blob storage must be provided (Azure Blob storage, Amazon S3, MinIO)

[Runners / Create self-hosted runner](#)

Adding a self-hosted runner requires that you download, configure, and execute the GitHub Actions Runner. By downloading and configuring the GitHub Actions Runner, you agree to the [GitHub Terms of Service](#) or [GitHub Corporate Terms of Service](#), as applicable.

Runner image

macOS Linux Windows

Architecture

x64

Download

```
# Create a folder
$ mkdir actions-runner && cd actions-runner

# Download the latest runner package
$ curl -o actions-runner-osx-x64-2.305.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.305.0/actions-runner-osx-x64-2.305.0.tar.gz

# Optional: Validate the hash
$ echo "a7c623e013f97db6c73c27288047c1d02ab6964519020ad0e87e69c328e96534
actions-runner-osx-x64-2.305.0.tar.gz" | shasum -a 256 -c

# Extract the installer
$ tar xzf ./actions-runner-osx-x64-2.305.0.tar.gz
```

Configure

```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/xpirit-training/training-manual --token
ADYVCZCCL5A65VD3SBWUJSLEVW7R0

# Last step, run it!
$ ./run.sh
```

Using your self-hosted runner

```
# Use this YAML in your workflow file for each job
runs-on: self-hosted
```

Self-hosted runners

Gotchas

- Runners are not ephemeral per default – you have to clean up after a build yourself
 - `$./config.sh --ephemeral`
- Use web hooks to auto scale
(github.com/jonico/awesome-runners)
- Do **not** allow public repositories!
- Limit Actions and use SHA or fork
- Create a company marketplace
(github.com/rajbos/actions-marketplace)

General actions permissions

Policies

Choose which repositories are permitted to use GitHub Actions.

All repositories ▾

Allow all actions and reusable workflows

Any action or reusable workflow can be used, regardless of who authored it or where it is defined.

Allow accelerate-devops actions and reusable workflows

Any action or reusable workflow defined in a repository within the accelerate-devops organization can be used.

Allow accelerate-devops, and select non-accelerate-devops, actions and reusable workflows

Any action or reusable workflow that matches the specified criteria, plus those defined in a repository within the accelerate-devops organization, can be used.

[Learn more about allowing specific actions and reusable workflows to run.](#)

Allow actions created by GitHub

Allow actions by Marketplace **verified creators**

Allow specified actions and reusable workflows

microsoft/*

my-org/*

Wildcards, tags, and SHAs are allowed.

Action examples: octo-org/octo-repo@*, octo-org/octo-repo@v2

Reusable workflow examples: octo-org/octo-repo/.github/workflows/build.yml@main

Entire organisation or repository examples: octo-org/*@*, octo-org/octo-repo/*

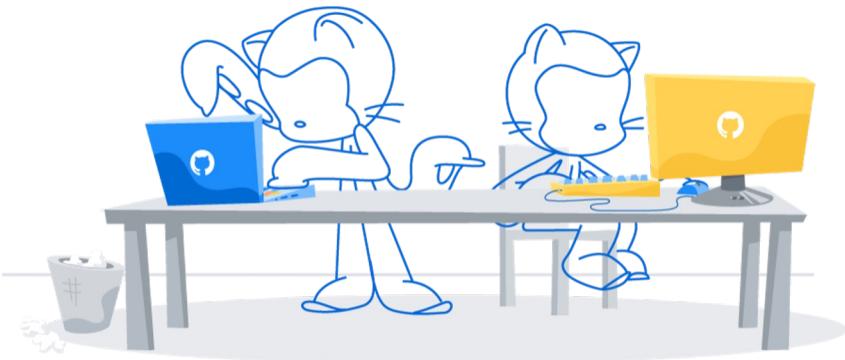
Save

Security with self-hosted runners



Public repositories with self-hosted runners pose potential risks:

- Malicious programs running on the machine
- Escaping the machine's runner sandbox
- Exposing access to the machine's network
- Persisting unwanted or dangerous data on the machine



Self-hosted runners and Security

Forked repositories will contain the same Actions configuration as the parent repository, including the self-hosted runners. Creates the potential for a fork to run malicious code on a runner inside your network. For this reason, it is highly recommended to use self-hosted runners only with **private** repositories.

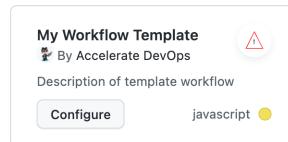
Sharing Workflows

Workflow templates

Workflow templates

- Available in Actions / New workflow
- Get copied one time
- Starter workflows

By Accelerate DevOps



Deployment

[View all](#)

Deploy Node.js to Azure Web App

By Microsoft Azure

Build a Node.js project and deploy it to an Azure Web App.

[Configure](#) [Deployment](#) ●

Deploy to Amazon ECS

By Amazon Web Services

Deploy a container to an Amazon ECS service powered by AWS Fargate or Amazon EC2.

[Configure](#) [Deployment](#) ●

Build and Deploy to GKE

By Google Cloud

Build a docker container, publish it to Google Container Registry, and deploy to GKE.

[Configure](#) [Deployment](#) ●

Terraform

By HashiCorp

Set up Terraform CLI in your GitHub Actions workflow.

[Configure](#) [Deployment](#) ●

Deploy to Alibaba Cloud ACK

By Alibaba Cloud

Deploy a container to Alibaba Cloud Container Service for Kubernetes (ACK).

[Configure](#) [Deployment](#) ●

Deploy to IBM Cloud Kubernetes Service

By IBM

Build a docker container, publish it to IBM Cloud Container Registry, and deploy to IBM Cloud Kubernetes Service.

[Configure](#) [Deployment](#) ●

Tencent Kubernetes Engine

By Tencent Cloud

This workflow will build a docker container, publish and deploy it to Tencent Kubernetes Engine (TKE).

[Configure](#) [Deployment](#) ●

OpenShift

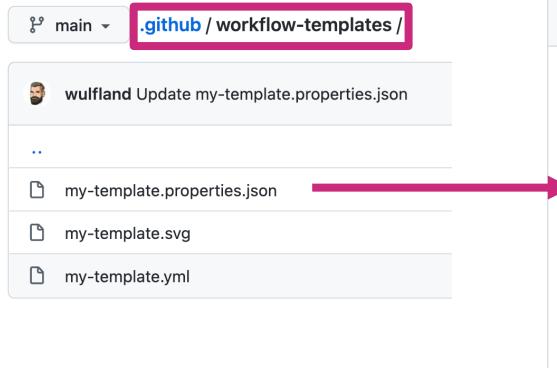
By Red Hat

Build a Docker-based project and deploy it to OpenShift.

[Configure](#) [Deployment](#) ●

Workflow templates

- <org>/.github/workflow-templates



The screenshot shows a GitHub repository interface. In the top navigation bar, the 'main' branch is selected, and the path '.github / workflow-templates /' is highlighted. The sidebar lists several files: 'my-template.properties.json', 'my-template.svg', and 'my-template.yml'. A red arrow points from the 'my-template.properties.json' entry in the sidebar to its content in the main pane. The main pane displays the JSON content of 'my-template.properties.json'.

```
13 lines (13 sloc) | 269 Bytes
```

```
1 {  
2   "name": "My Workflow Template",  
3   "description": "Description of template workflow",  
4   "iconName": "my-template",  
5   "categories": [  
6     "javascript"  
7   ],  
8   "filePatterns": [  
9     "package.json$",  
10    "^Dockerfile",  
11    ".*\\".md$"  
12  ]  
13 }
```

```
15 lines (11 sloc) | 232 Bytes
```

```
1 name: My templated workflow  
2  
3 on:  
4 push:  
5 branches: [ $default-branch ]  
6  
7 jobs:  
8 build:  
9 runs-on: ubuntu-latest  
10  
11 steps:  
12 - uses: actions/checkout@v2  
13  
14 - name: Run a one-line script  
15 run: echo Hello World!
```

Reusable Workflows

Reusable workflows

```
1 name: Reusable workflow
2
3 on:
4   workflow_call:
5     inputs:
6       who-to-greet:
7         description: 'The person to greet'
8         type: string
9         required: true
10        default: World
11
12      outputs:
13        current-time:
14          description: 'The time when greeting.'
15          value: ${{ jobs.reusable-job.outputs.current-time }}
16
17    jobs:
18      reusable-job:
19        runs-on: ubuntu-latest
20        outputs:
21          current-time: ${{ steps.time.outputs.current-time }}
22        steps:
23          - name: Greet someone
24            run: echo "Hello ${{ inputs.who-to-greet }}"
25          - name: Set time
26            id: time
27            run: echo "::set-output name=current-time::$(date)"
28
```

```
1 name: Reuse other workflow
2
3 on: [workflow_dispatch]
4
5 jobs:
6   call-workflow:
7     uses: ./github/workflows/reusable.yml
8     with:
9       who-to-greet: '@wulfland'
10
11 use-output:
12   runs-on: ubuntu-latest
13   needs: [call-workflow]
14   steps:
15     - run: echo "Time was ${{ needs.call-workflow.outputs.current-time }}"
16
```

Reusable workflow vs Composite action

Reusable workflow:

- Defines the entire job
- Can enforce runner labels
- No option to do something before and after the steps

Composite action:

- Defines the list of steps
- Full flexibility to do something before and after the steps in the composite action

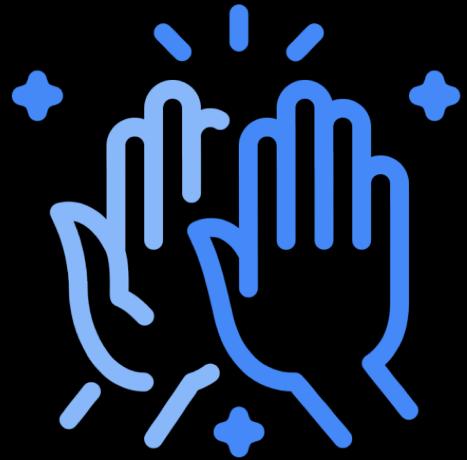
Hands-on: Reusable workflows

Sharing workflows

Best Practices

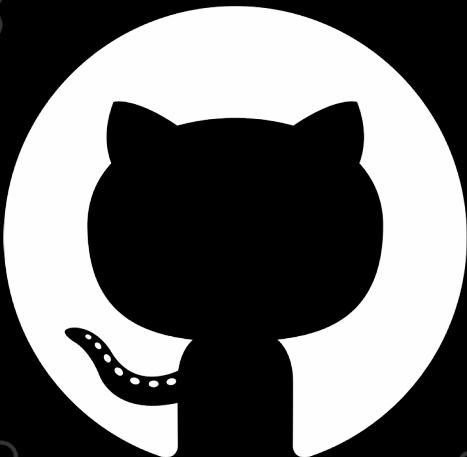
- Use **actions** and composite actions as building blocks
- Use **workflow templates** and **template repositories** for discoverability
- Use **reusable workflows** for complex scenarios
- Share actions and reusable workflows in internal repositories





Q&A

Retro Day 2



Thank you