



Supercharging DevOps with GitHub Actions



Soham Dasgupta
Partner Solution Architect
Startups & Scaleups
App Innovation & AI
Microsoft



sohamda



iamsoham



dasguptasoham

Agenda

- Github
- Triggers, Jobs, Steps
- Actions
- Secrets, Environments
- Runners
- Sharing Workflow



Where the world builds software

100M+
Developers

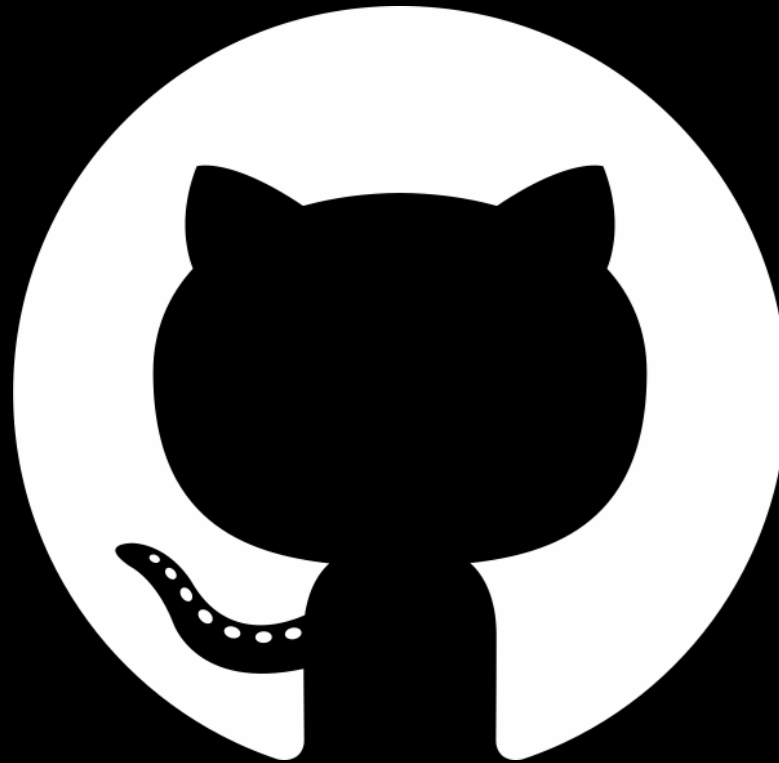
200M+
Repositories

4M+
Organizations

1,000s
Open-Source
Communities

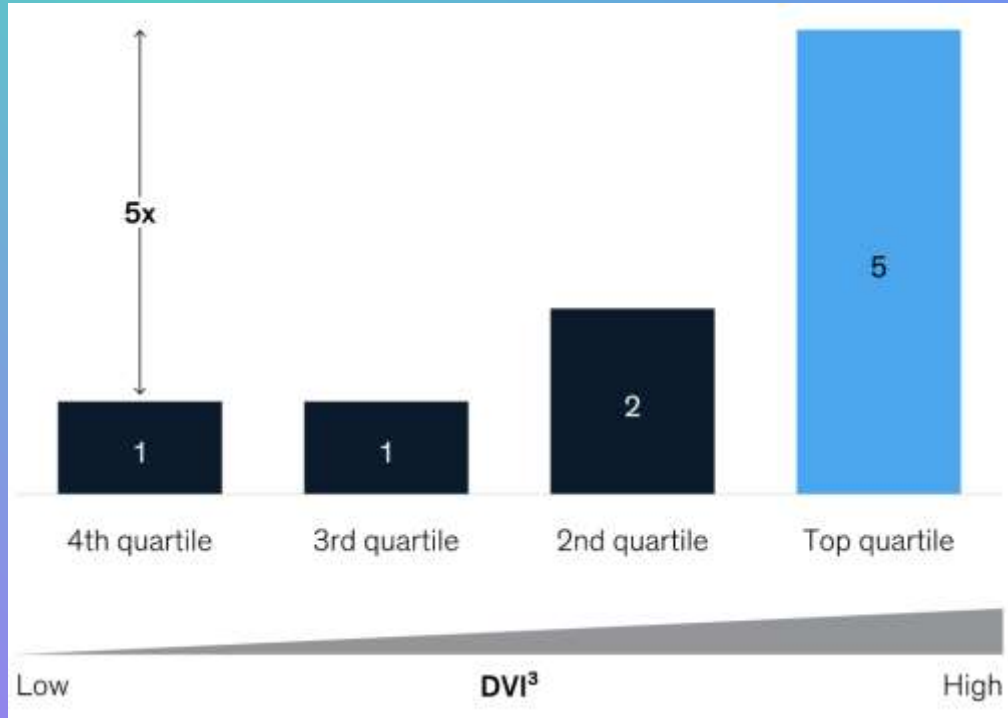
2.6B+
Contributions / Year

84%
Fortune 500
companies



CI/CD adoption feeds developer velocity

Faster developer velocity=faster time to market



Source: Developer Velocity: How software excellence fuels business performance, McKinsey & Company, 2020



Faster time to market




Increased product innovation



Increased customer satisfaction



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

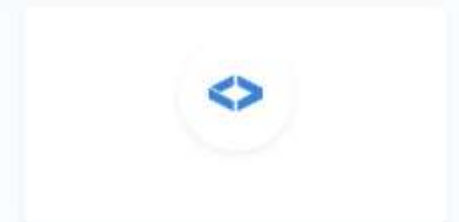
Recently added

Security

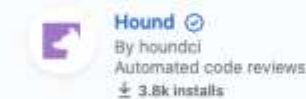
Support

Search for apps and actions

Recommended for you



Trending

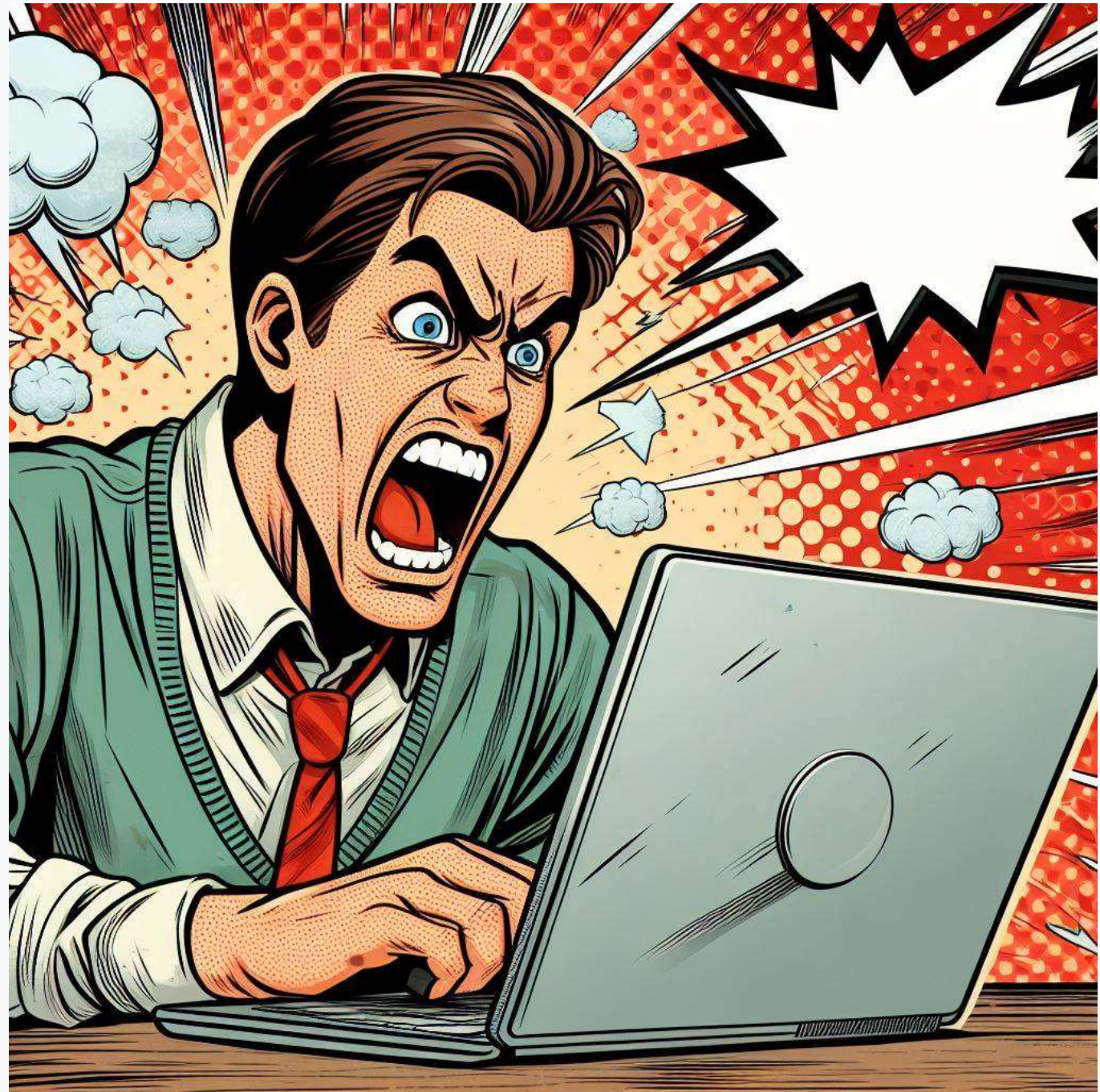


GitHub Actions – More than CI/CD

- 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



YAML (Yelling At My Laptop, again!)



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
```

Use Control + Space to trigger autocomplete in most situations.

Basic syntax

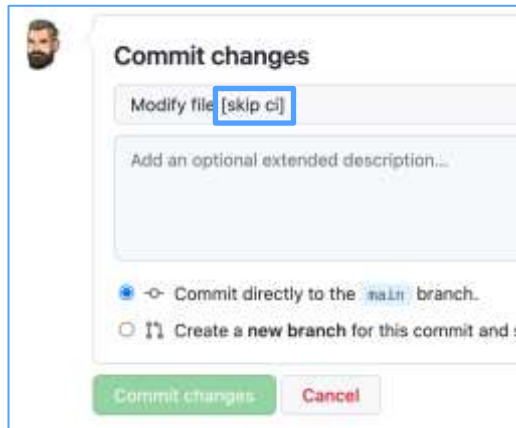
```
./.github/workflows/workflow-file-name.yml
```

events	→	<code>name: Super Linter workflow</code>
		<code>on:</code>
		<code> push:</code>
jobs	→	<code>jobs:</code>
		<code> lint:</code>
		<code> name: Lint Code Base</code>
runner	→	<code>runs-on: ubuntu-latest</code>
steps	→	<code>steps:</code>
		<code> - uses: actions/checkout@v2</code>
actions	→	<code> - uses: github/super-linter@v3</code>
		<code> env:</code>
		<code> GITHUB_TOKEN: \${ secrets.GITHUB_TOKEN }</code>
secrets	→	<code> - run: echo "Hello world"</code>
shell	→	

Workflow triggers

Events that triggers workflows

- Trigger:
 - Webhook events
 - Scheduled events
 - Manual event



Commit changes

Modify file [skip ci]

Add an optional extended description...

Commit directly to the main branch.

Create a new branch for this commit and switch to it.

Commit changes Cancel

```
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  schedule:
14    - cron: '*/*15 * * * *'
15    - cron: '* * 9-17 * * *'
16    - cron: '* * 11 * * 5'
17
18  # Manual events
19  workflow_dispatch:
20    inputs:
21      homedrive:
22        description: 'The home drive on the machine'
23        required: true
24        default: '/home'
```

Events that triggers workflows

Manual events



The screenshot shows a 'Run workflow' dialog box with the following configuration:

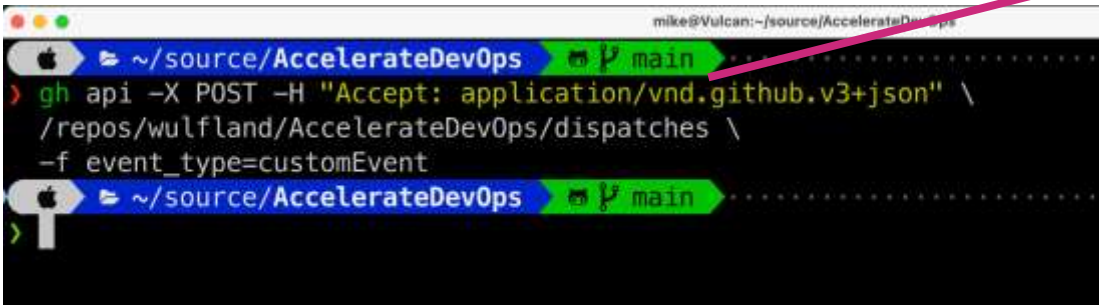
- Use workflow from: Branch: main
- The home drive on the machine *: /home
- Log level *: warning
- True to print to STDOUT
- Environment to run tests against *: Prod
- Run workflow button

A red arrow points from the 'Log level' dropdown menu to the corresponding configuration in the code block on the right.

```
# 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
```

```
# 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 `${github.event.inputs.environment}`."
      - run: echo "📖 log level is `${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.

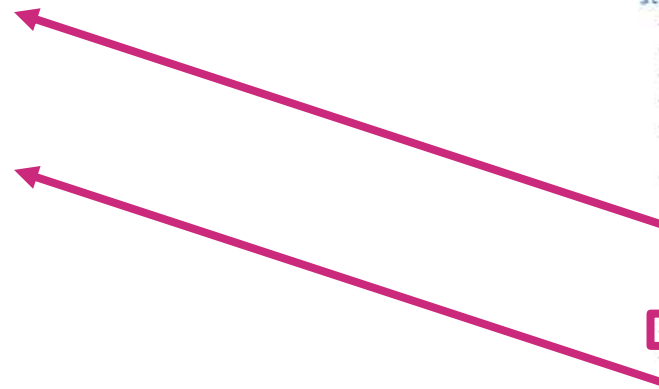
```
✓ Get OS information
  1 ▶ Run import platform
  3 Linux-5.13.0-1021-azure-x86_64-with-glibc2.29
  > Run actions/checkout@v3.0.0
  ✓ Display documentation
    1 ▶ Run tree
    4
    5 |— About.md
    6 |— _config.yml
    7 |— _posts
    8 |   |— 2021-08-10-posting-source-code.md
    9 |   |— 2021-08-12-writing-with-markdown.md
   10 |   |— 2021-08-13-posting-in-jekyll.md
   11 |— about-markdown.md
   12 |— get-started.md
   13 |— index.md
   14
   15 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 `${{ github.event.inputs.environment }}`."
    - run: echo "📊 log level is `${{ 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:

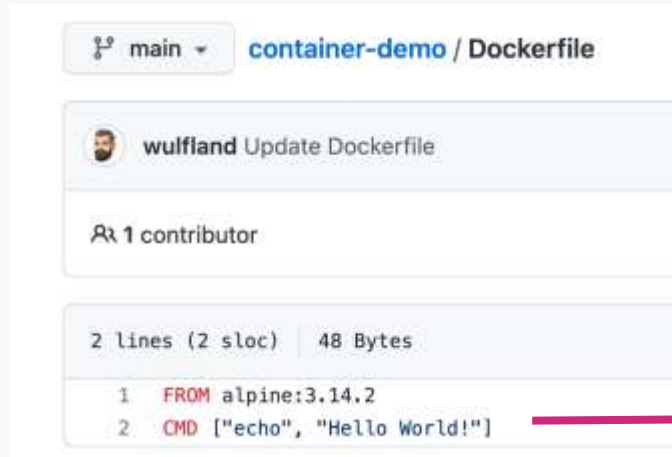
```
~/source/checkout main 15:2
> git remote -v
origin https://github.com/actions/checkout.git (fetch)
origin https://github.com/actions/checkout git (push)
~/source/checkout main 15:2
> git log --oneline --graph --decorate --all -15
* add3486 (HEAD -> main origin/main, origin/HEAD) Patch to fix the dependbot alert. (#744)
* 5126516 Bump minimist from 1.2.5 to 1.2.6 (#741)
* d50f8ea Add v3.0 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@a12a3943b4bdde767164f792f33f40b04645d846
- 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`



The screenshot shows a GitHub repository for 'wulfland/container-demo'. The file 'Dockerfile' is selected, showing two lines of code: 'FROM alpine:3.14.2' and 'CMD ["echo", "Hello World!"]'. A red arrow points from the second line of code to the 'Hello World!' output in the action log.



The screenshot shows a GitHub Actions log for the step 'Run ghcr.io/wulfland/container-demo:latest'. The log contains three lines: 1. 'Run docker://ghcr.io/wulfland/container-demo:latest', 2. the full Docker run command, and 3. the output 'Hello World!'. A red arrow points from the 'Hello World!' output back to the 'CMD ["echo", "Hello World!"]' line in the Dockerfile screenshot.

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

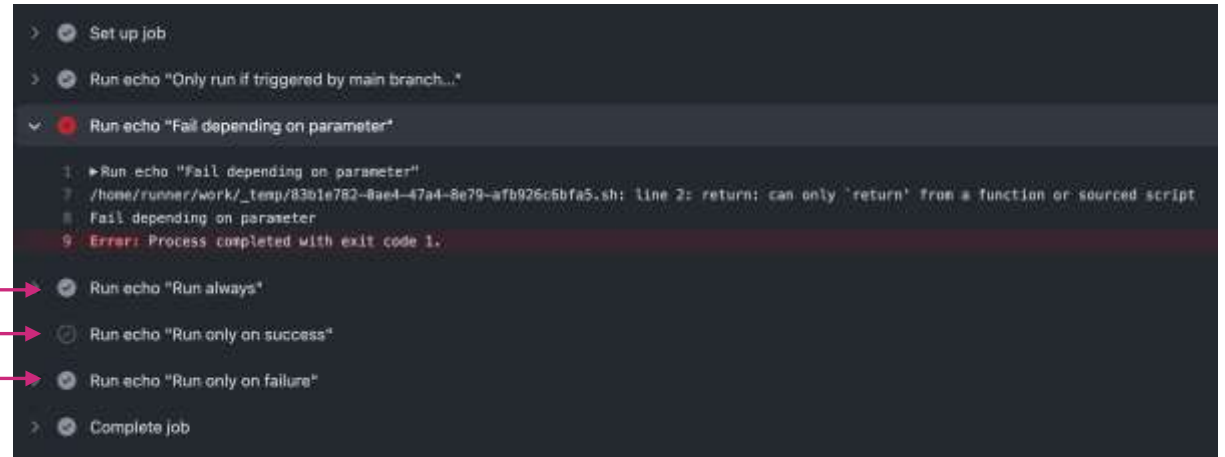
```
steps:  
- name: Dump runner context  
  run: echo '${{ toJSON(runner) }}'  
- name: Dump GitHub context  
  run: echo '${{ toJSON(github) }}'
```

```
✓ Dump runner context  
1 ▶ Run echo '{  
11 {  
12   "os": "Linux",  
13   "arch": "X64",  
14   "name": "GitHub Actions 2",  
15   "tool_cache": "/opt/hostedtoolcache",  
16   "temp": "/home/runner/work/_temp",  
17   "workspace": "/home/runner/work/AccelerateDevOps"  
18 }
```

```
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       "logLevel": "warning",  
203       "run_matrix": "false"
```

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()
```



The screenshot shows the execution of the workflow defined in the code block to the left. The steps are listed with their status and output:

- Set up job
- Run echo "Only run if triggered by main branch..."
- Run echo "Fail depending on parameter" (Failed)
- Run echo "Run always"
- Run echo "Run only on success"
- Run echo "Run only on failure"
- Complete job

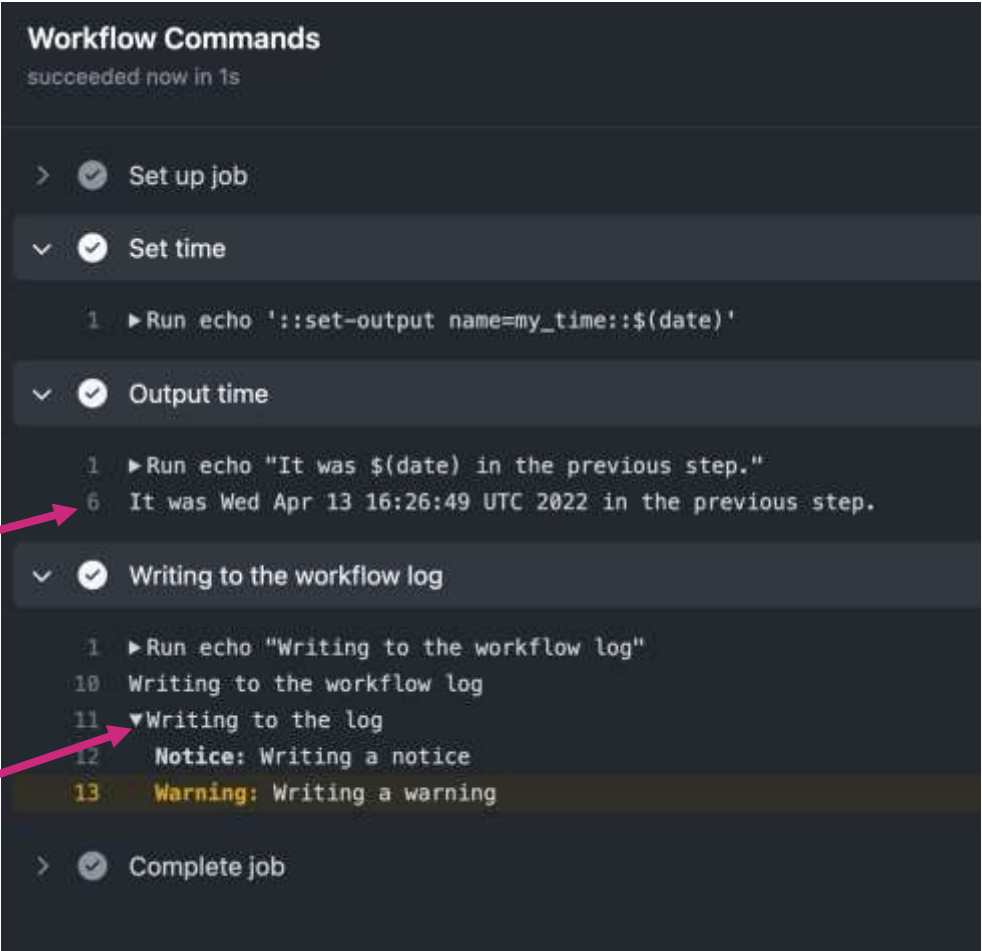
The error message for the failed step is: `1: Run echo "Fail depending on parameter"`, `2: /home/runner/work/_temp/83b1e782-8ae4-47a4-8e79-afb926c5bfa5.sh: line 2: return: can only 'return' from a function or sourced script`, `3: Fail depending on parameter`, `4: Error: Process completed with exit code 1.`

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 'Workflow Commands' interface for a workflow that succeeded in 1s. The steps are listed as follows:

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

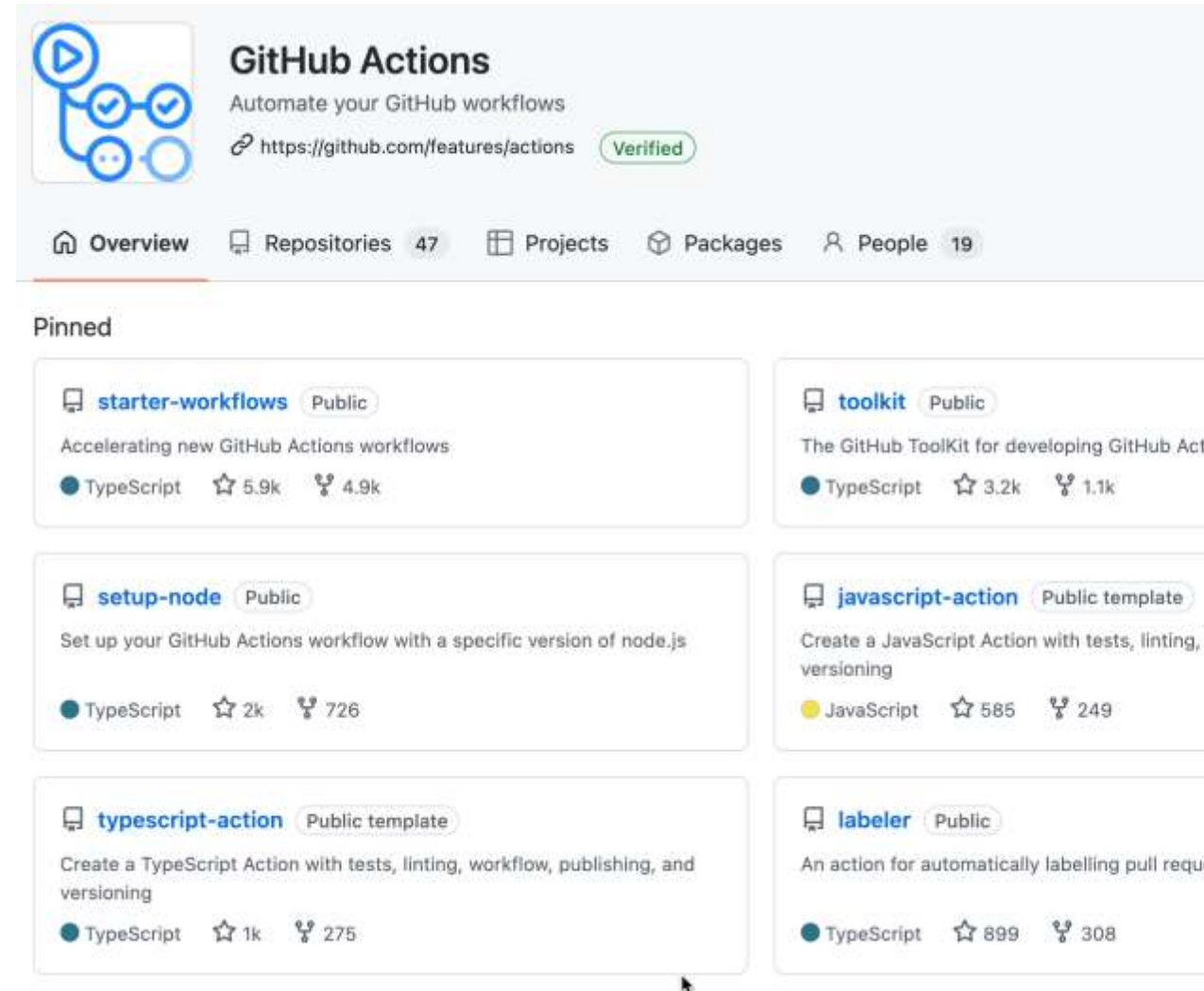
More advanced syntax elements

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

Actions

GitHub Actions

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

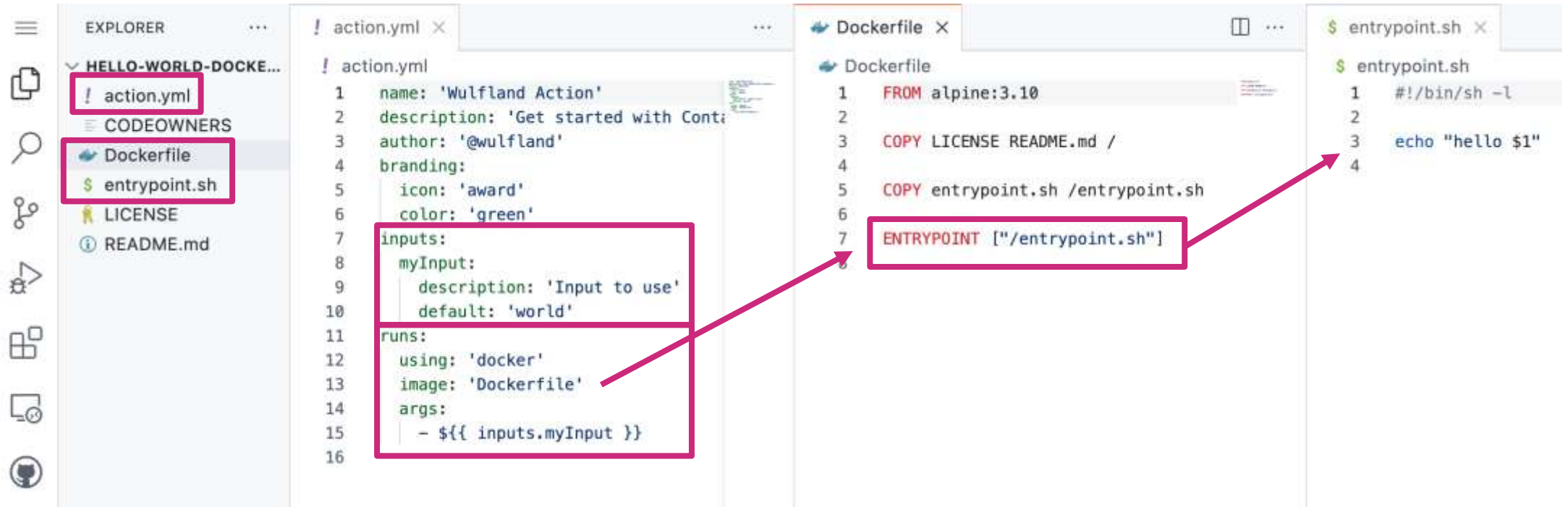
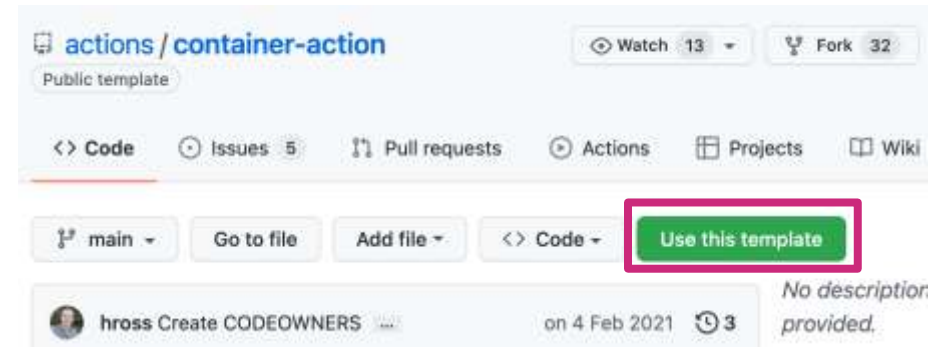


The screenshot displays the GitHub Actions marketplace page. At the top, there is a header for "GitHub Actions" with the tagline "Automate your GitHub workflows" and a verified badge. Below the header, there are navigation tabs for "Overview", "Repositories 47", "Projects", "Packages", and "People 19". The main content area is titled "Pinned" and features a grid of workflow templates. Each template card includes a repository icon, the name of the workflow, its visibility (Public or Public template), a brief description, the programming language used, and the number of stars and forks.

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

Container Actions

- Dockerfile or existing image
- Inputs



Container Actions

- Dockerfile or existing image
- inputs

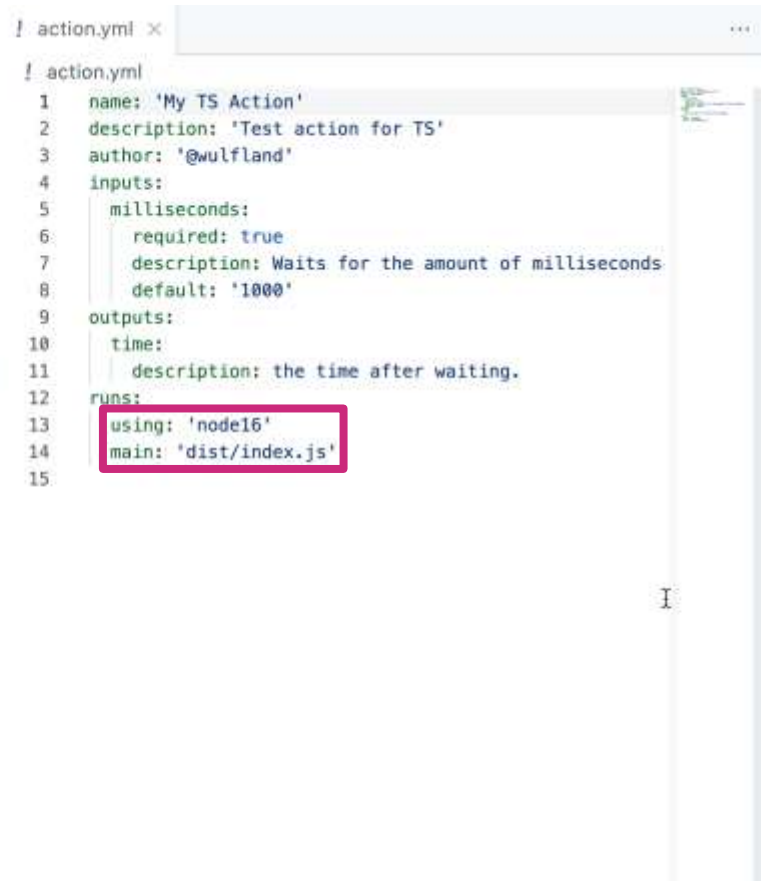
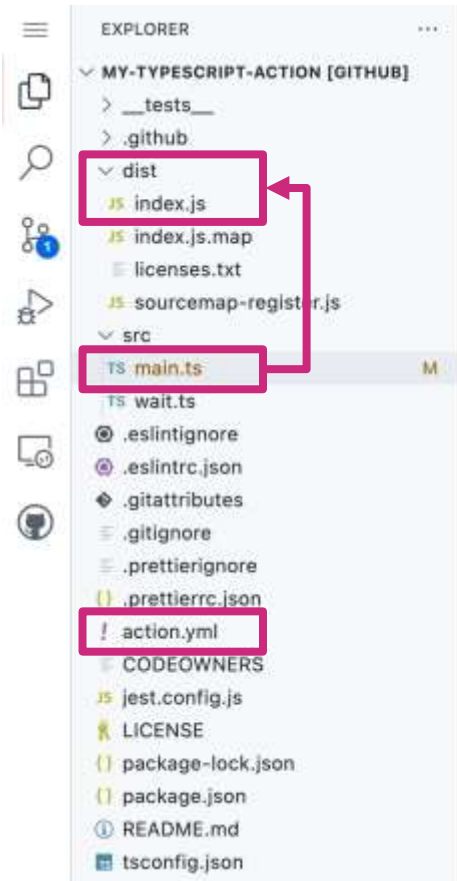
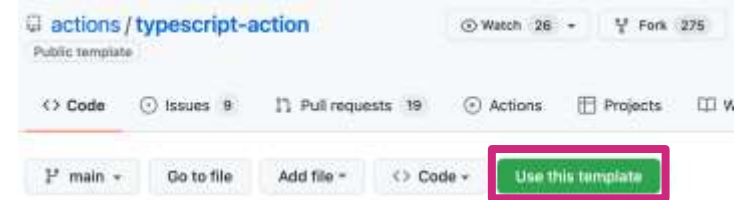
15 lines (13 sloc) | 398 Bytes

```
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 }"
```

```
test
succeeded 17 seconds ago in 5s

> Set up job
v Build wulfland/hello-world-docker-action@v1.2
1 ▶ Build container for action use: '/home/runner/work/_actions/wulfland/hello-world-docker-action/v1.2/Dockerfile'.
v Run my own container action
1 ▶ Run wulfland/hello-world-docker-action@v1.2
4 /usr/bin/docker run --name bcf090f977186e9874b92a188d8a409df5216_6a7e50 --label 2bcf09 --workdir /github/workspace --rm --e INPUT_MYI
GITHUB_REPOSITORY_OWNER --e GITHUB_RUN_ID --e GITHUB_RUN_NUMBER --e GITHUB_RETENTION_DAYS --e GITHUB_RUN_ATTEMPT --e GITHUB_ACTOR --e GITH
GITHUB_SERVER_URL --e GITHUB_API_URL --e GITHUB_GRAPHQL_URL --e GITHUB_REF_NAME --e GITHUB_REF_PROTECTED --e GITHUB_REF_TYPE --e GITH
GITHUB_ACTION_REF --e GITHUB_PATH --e GITHUB_ENV --e GITHUB_STEP_SUMMARY --e RUNNER_OS --e RUNNER_ARCH --e RUNNER_NAME --e RUNNER_TOOL_CACH
ACTIONS_RUNTIME_TOKEN --e ACTIONS_CACHE_URL --e GITHUB_ACTIONS=true --e CI=true --v "/var/run/docker.sock":"/var/run/docker.sock" --v "/h
/home/runner/work/_temp/_github_workflow":"/github/workflow" --v "/home/runner/work/_temp/_runner_file_commands":"/github/file_comma
act100":"/github/workspace" 2bcf09:0f977186e9874b92a188d8a409df5216 @wulfland
5 hello @wulfland
v Output time set in the container
1 ▶ Run echo "The time in the container was Wed Apr 13 18:16:52 UTC 2022"
4 The time in the container was Wed Apr 13 18:16:52 UTC 2022
> Complete job
```

JavaScript Actions



Composite Actions

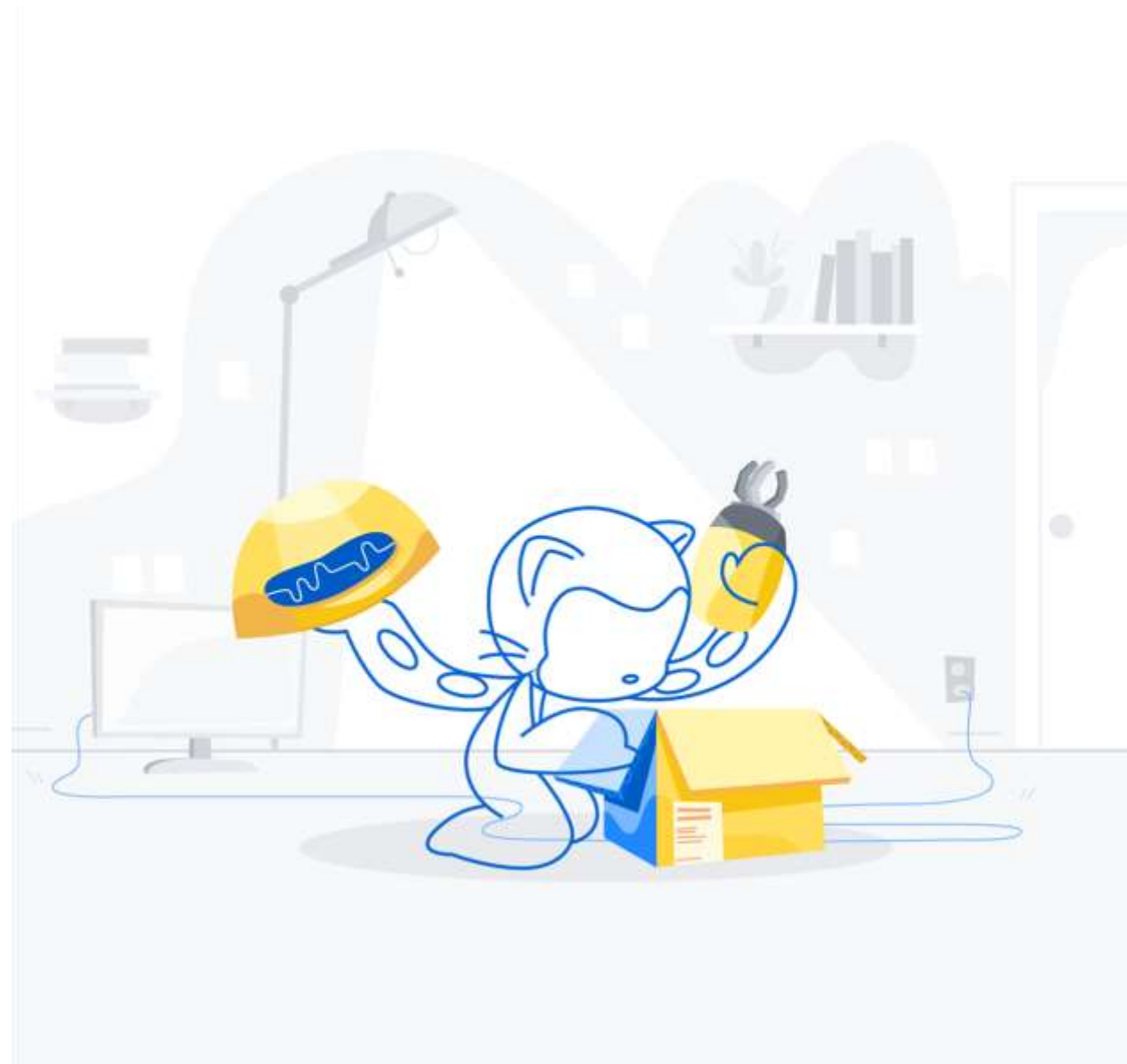
- Just a `action.yml` file
- Inputs
- Outputs
- Runs

```
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 ${{ inputs.who-to-greet }}.
16       shell: bash
17
18     - id: random-number-generator
19       run: echo ":set-output name=random-id:$(echo $RANDOM)"
20       shell: bash
21
22     - run: echo "${{ 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 }}"
```


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



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/
```

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
```

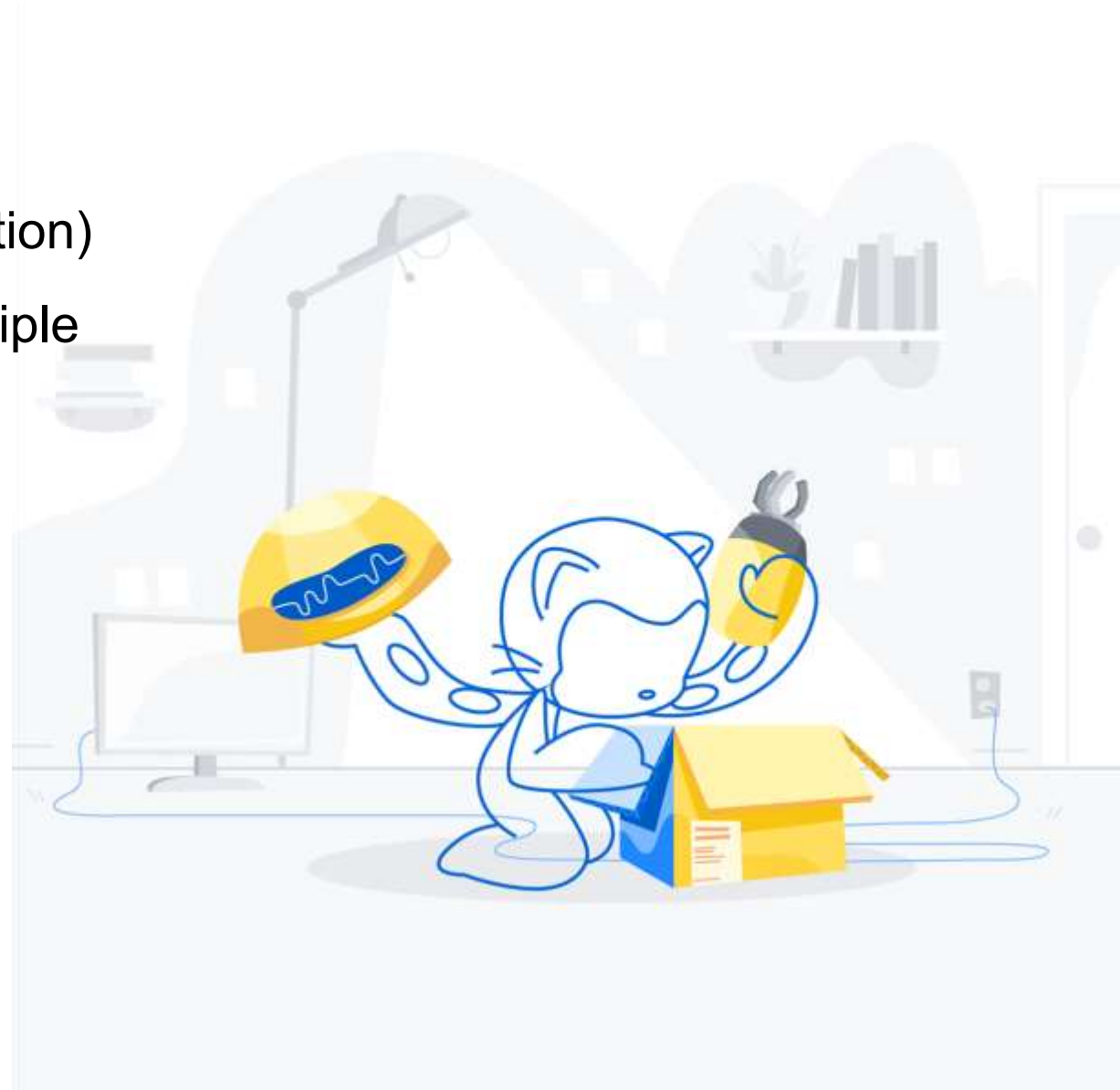
Java - Maven

```
- 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 interface for managing secrets and variables. On the left is a navigation sidebar with categories: General, Access, Code and automation, Security, and Integrations. The 'Secrets and variables' option is selected and expanded to show 'Actions'. The main content area is titled 'Actions secrets and variables' and contains explanatory text about encrypted secrets and plain text variables. Below the text are tabs for 'Secrets' and 'Variables', and a 'New repository secret' button. The 'Secrets' tab is active, displaying three sections: 'Environment secrets' (with a 'Manage environments' button), 'Repository secrets' (with a 'MY_REPO_SECRET' entry updated 2 minutes ago), and 'Organization secrets' (with a 'MY_ORG_SECRET' entry updated on Apr 25).

General

Access

- Collaborators and teams
- Moderation options

Code and automation

- Branches
- Tags
- Rules Beta
- Actions
- Webhooks
- Environments
- Pages

Security

- Code security and analysis
- Deploy keys
- Secrets and variables**
- Actions
- Codespaces
- Dependabot

Integrations

- GitHub Apps
- Email notifications

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	✎ 🗑️
----------------	-----------------------	--------------------------------

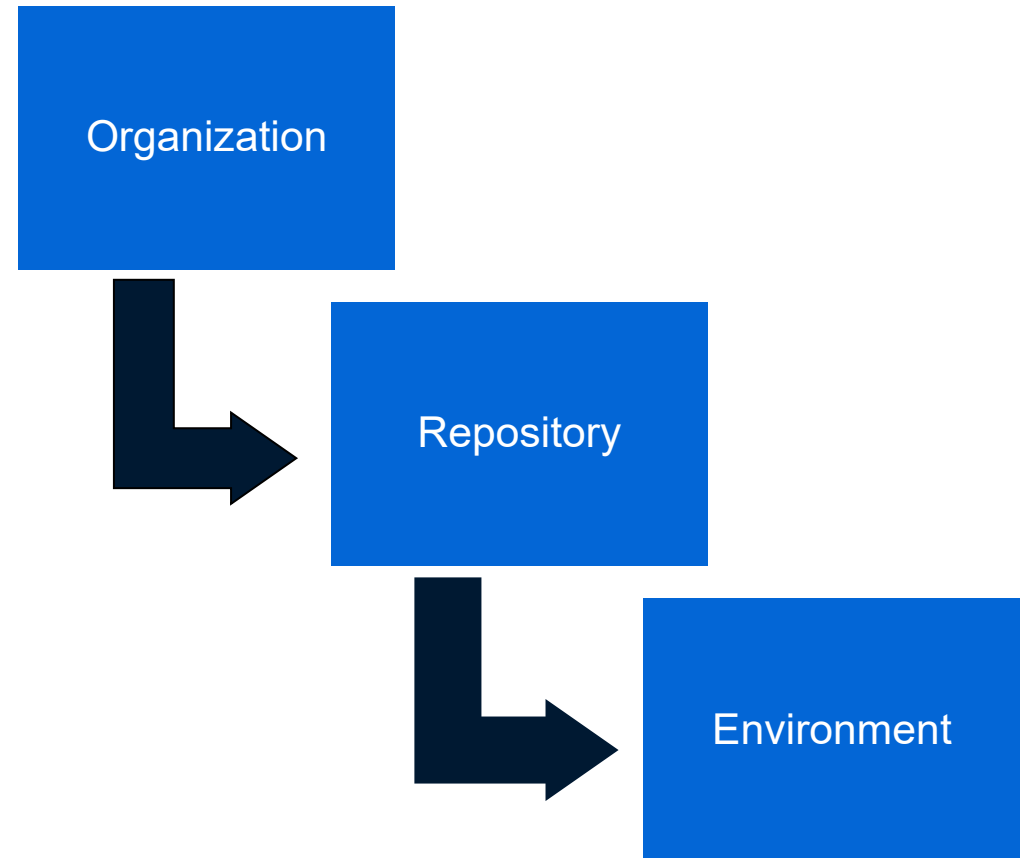
Organization secrets

Manage organization secrets

MY_ORG_SECRET	Updated on Apr 25
---------------	-------------------

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
```

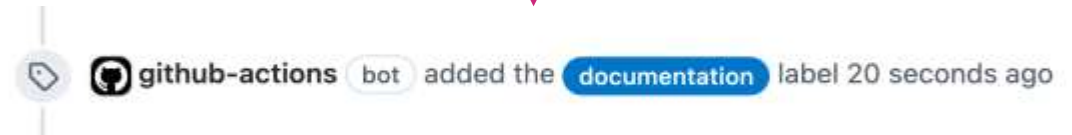
```
permissions: read-all
```

```
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`

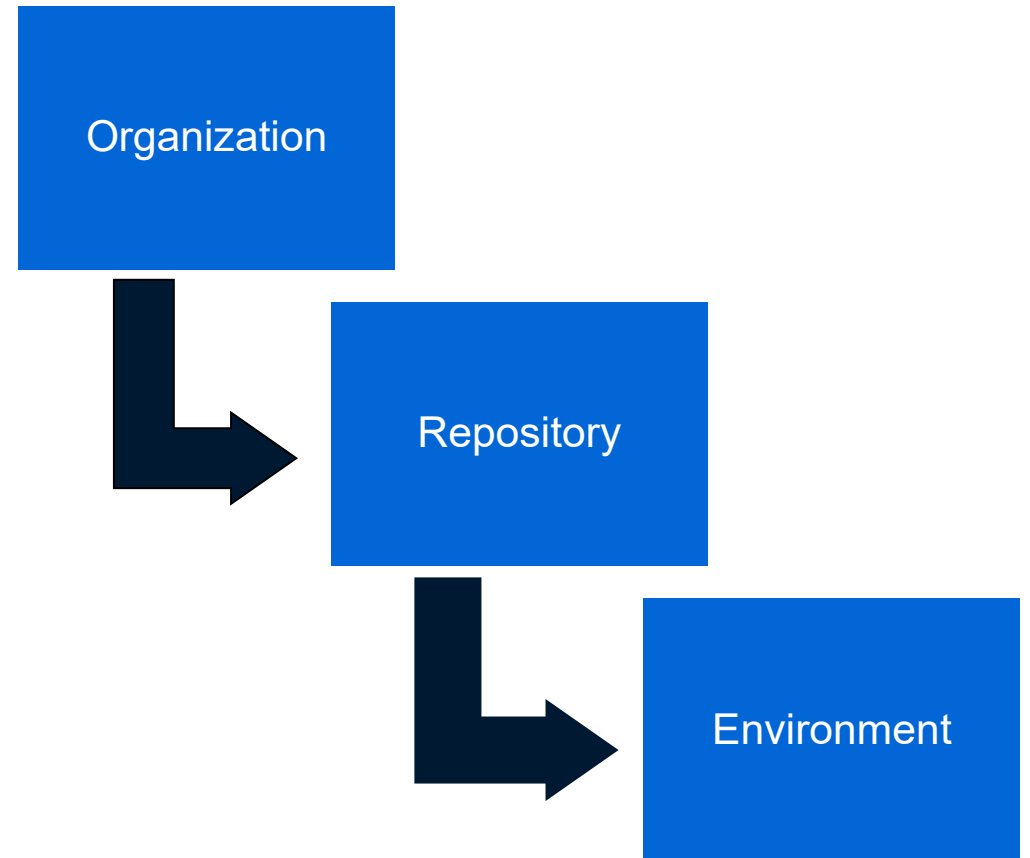
Perform actions as “github-actions”:

```
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 }}
```



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

Wait timer

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

 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

 wulfland ×

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

Staged Deployment Staged Deployment #17

Summary

Jobs

- Build
- Test
- Load-Test
- Staging
- Production

Manually triggered 1 minute ago | Status: **In progress** | Total duration: - | Artifacts: -

Wulfliand id: e7dx135

StagedDeployment.yml
in: workflow_dispatch

```
graph LR; Build[Build 1s] --> Load-Test[Load-Test 10s]; Build --> Test[Test 0s]; Load-Test --> Staging[Staging 0s]; Test --> Staging; Staging --> Production[Production 3s];
```

Staging: <https://pre.wireheabout.net>

Production: Deploying to prod.

Deployment reviews
Learn who reviewed the deployments in this run

Event	Environments	Comment
wulfliand approved now	Prod	Checked
wulfliand approved 41 seconds ago	Test	Looks good

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
<p>Hardware:</p> <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	<p>Hardware:</p> <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)

```
Set up job
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
13 GITHUB_TOKEN Permissions
14 Actions: write
15 Checks: write
16 Contents: write
17 Deployments: write
18 Discussions: write
19 Issues: write
20 Metadata: read
21 Packages: write
22 Pages: write
23 PullRequests: write
24 RepositoryProjects: write
25 SecurityEvents: write
26 Statuses: write
27 Secret source: Actions
28 Prepare workflow directory
29 Prepare all required actions
30 Getting action download info
31 Download action repository 'actions/checkout@a12a3943b4bdde767164f792f33f40b04645d846' (SHA:a12a3943b4bdde)
> Pull alpine:3.8
> Pull ghcr.io/wulfland/container-demo:latest
> Run echo "👉 The job was triggered by a workflow_dispatch event."
```

Larger runners

Runners / Create GitHub-hosted runner

Name

Runner image



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

Repositories 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>

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

macOS Linux Windows

Architecture

x64

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
```


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 "a7c623e013f97db6c73c27288047c1d02ab5964619020ad0e87e69c328e96534
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
ADYVC2CCL5A65VD3S8MJ3SLEW7RD

# 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/*

I

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/*

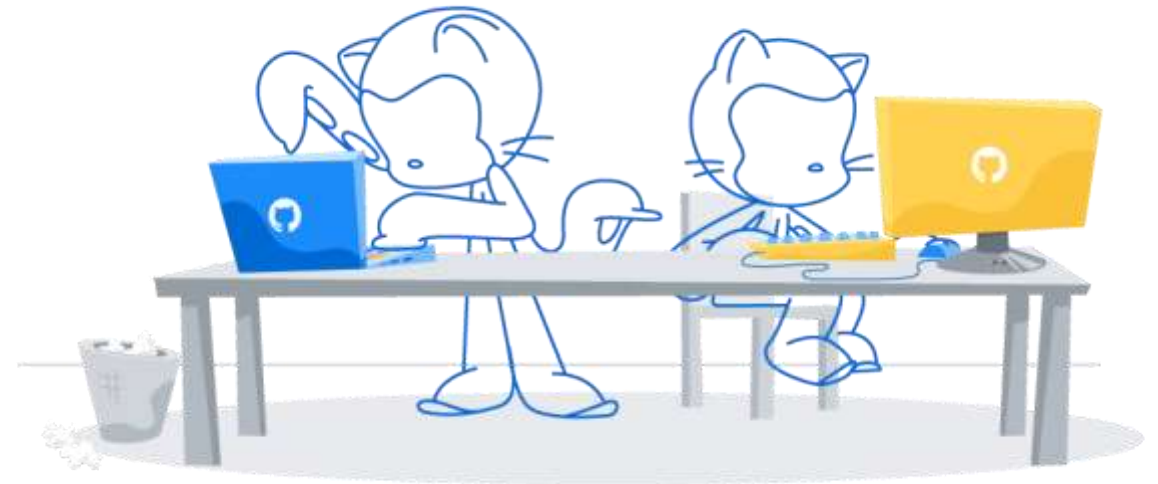
Save

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

My Workflow Template 

By Accelerate DevOps

Description of template workflow

[Configure](#) javascript 


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`

main ▾ **.github / workflow-templates /**

wulfland Update my-template.properties.json

..

my-template.properties.json

my-template.svg

my-template.yml

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      outputs:
12        current-time:
13          description: 'The time when greeting.'
14          value: ${ jobs.reusable-job.outputs.current-time }
15
16 jobs:
17   reusable-job:
18     runs-on: ubuntu-latest
19     outputs:
20       current-time: ${ steps.time.outputs.current-time }
21     steps:
22     - name: Greet someone
23       run: echo "Hello ${ inputs.who-to-greet }"
24     - name: Set time
25       id: time
26       run: echo "::set-output name=current-time::$(date)"
27
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

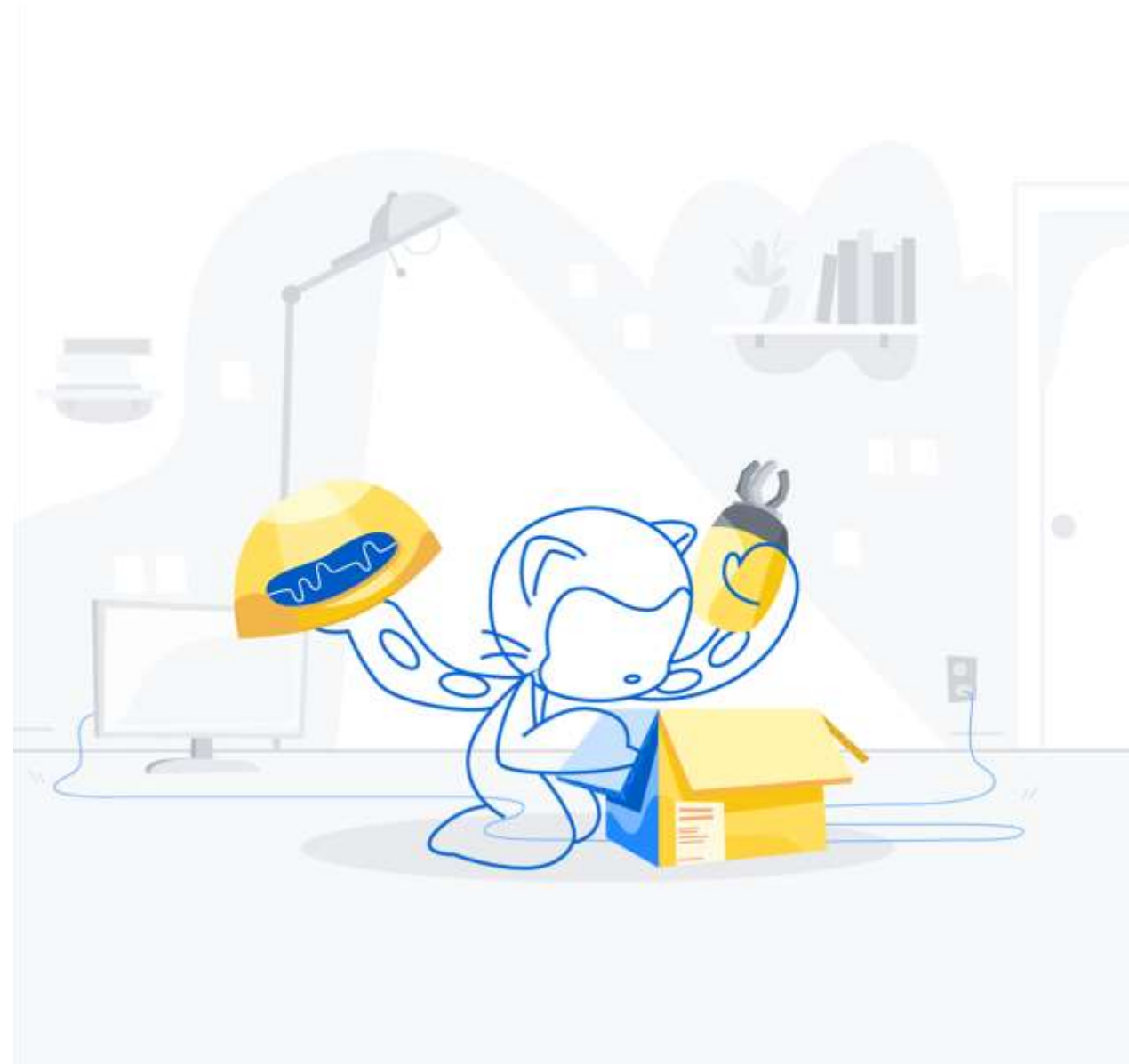
Composite action:

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

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



Thank you

