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

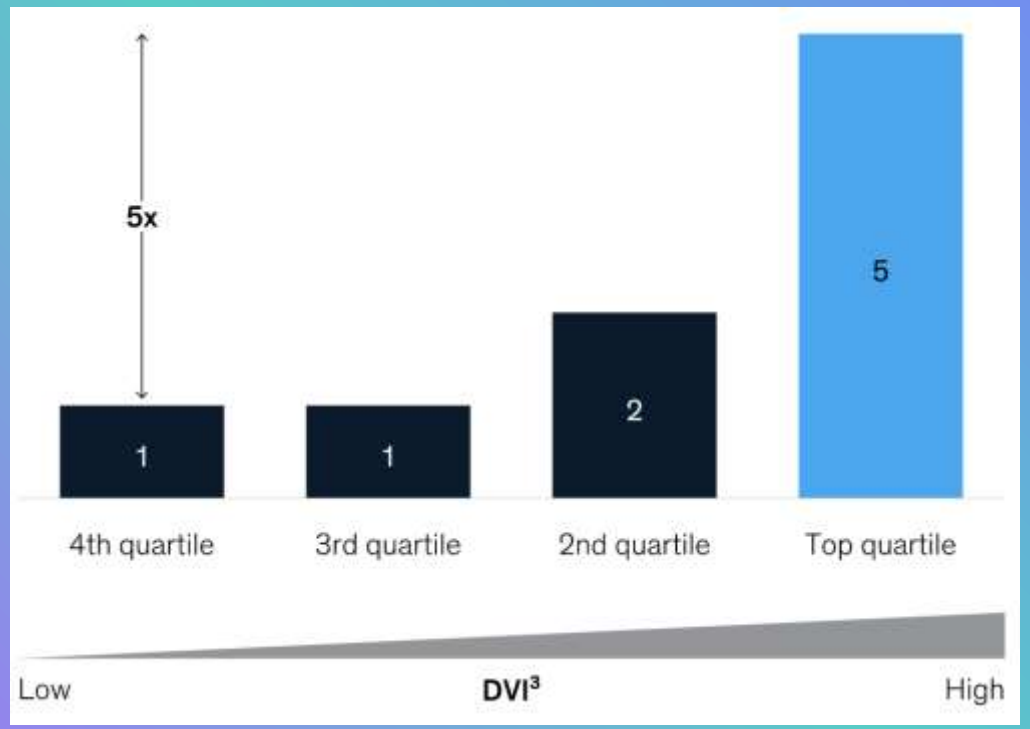
1,000s
Open-Source Communities

2.6B+
Contributions / Year

84%
Fortune 500 companies

CI/CD adoption feeds developer velocity

Faster developer velocity=faster time to market

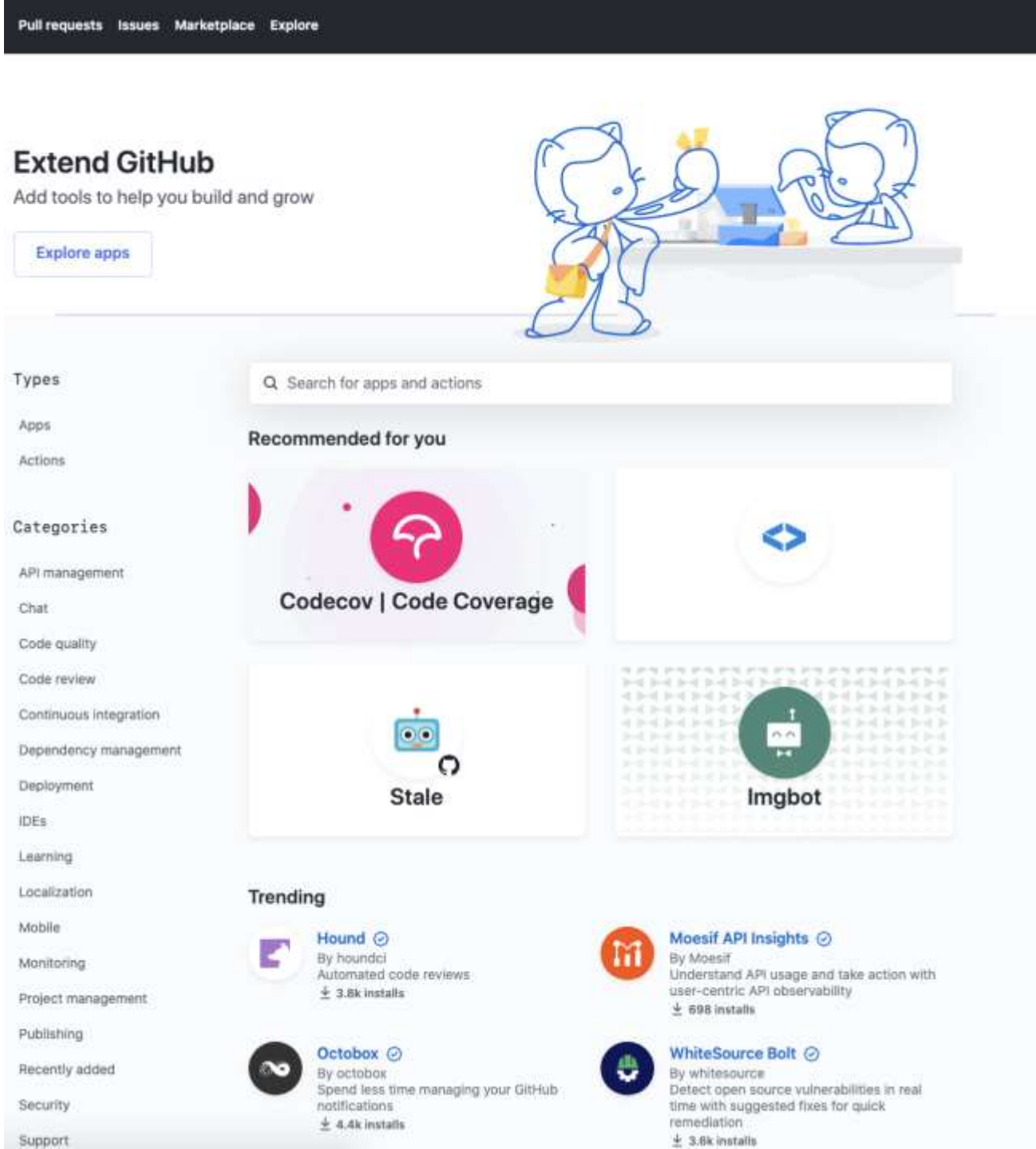


Source: Developer Velocity: How software excellence fuels business performance, McKinsey & Company, 2020

- Faster time to market
- Increased product innovation
- Increased customer satisfaction

GitHub Marketplace

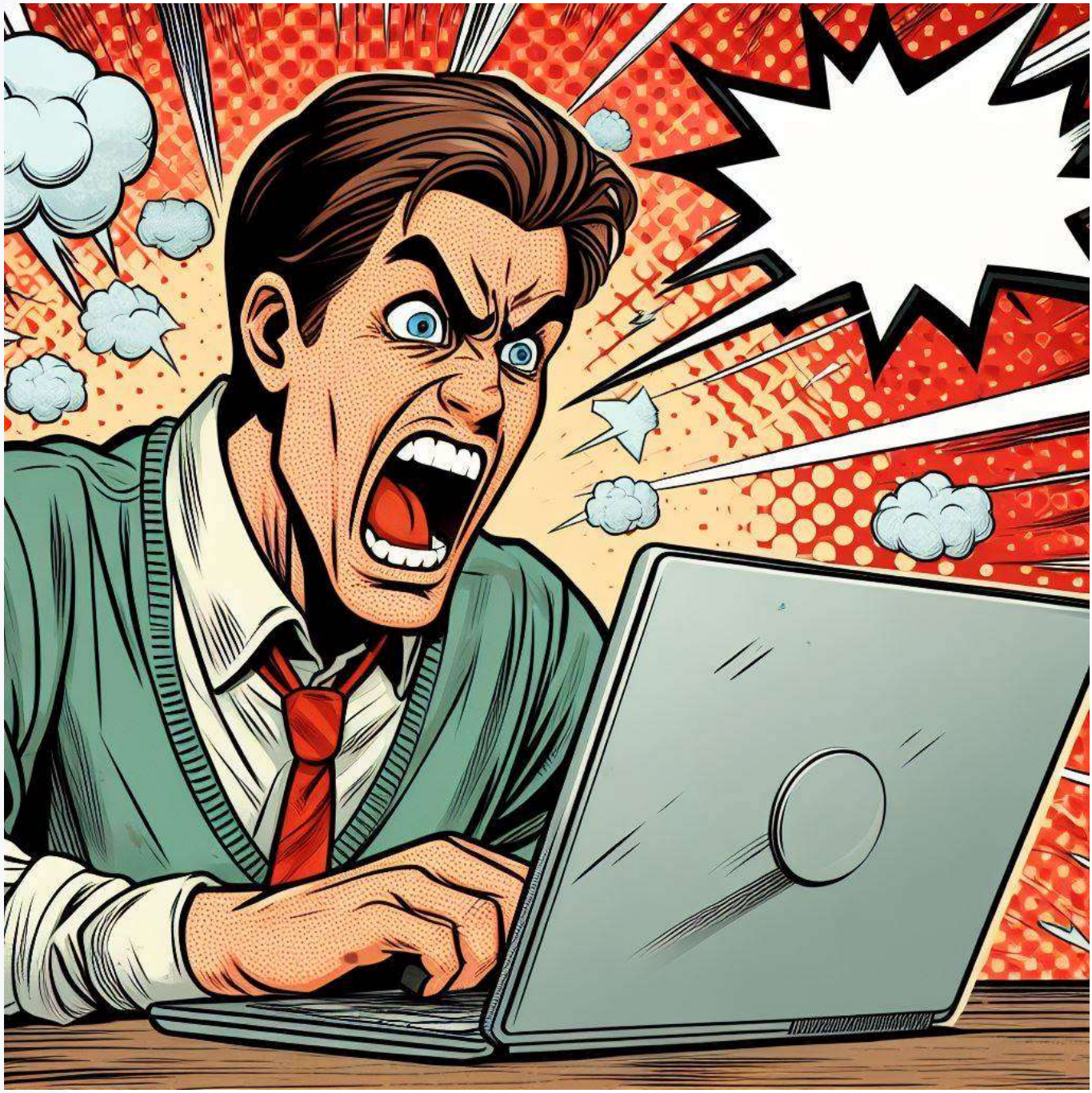
- Discover open-source Actions across multiple domains
- ~15,000 Actions (and counting...)
- Verified creators
  (*Publisher domain and email verified*)
- Reference these Actions directly in your workflow
- Integrated into the GitHub editor



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

AccelerateDevOps / .github / workflows / pr-validation.yml in main

Edit file | Preview changes | Spaces | 2 | No wrap

```yaml
name: PR-Validation

on: [pull_request]

jobs:
  Build:
    runs-on: ubuntu-latest

    steps:
    - name: 'Checkout Github Action'
      uses: actions/checkout@master

    - name: Set up .NET Core
      uses: actions/setup-dotnet@v1
      with:
        dotnet-version: '5.0.x'

    - name: Setup Node
      uses: actions/setup-node@v2.5.1
      with:
        node-version: 10.16.3

    - name: Install dependencies in client app
      working-directory: src/Tailwind.Traders.Web/ClientApp
      run: npm install

    - name: Build and publish with dotnet
      working-directory: src/Tailwind.Traders.Web
      run: |
        dotnet build --configuration Release

```

Use Control + Space to trigger autocomplete in most situations.

Basic syntax

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

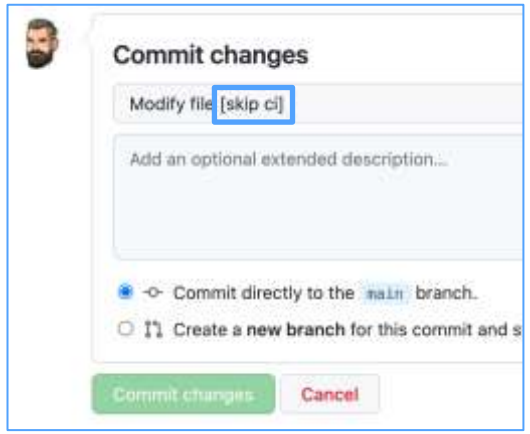
events →
jobs →
runner →
steps →
actions →
secrets →
shell →

```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      - run: echo "Hello world"
```

Workflow triggers

Events that triggers workflows

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



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

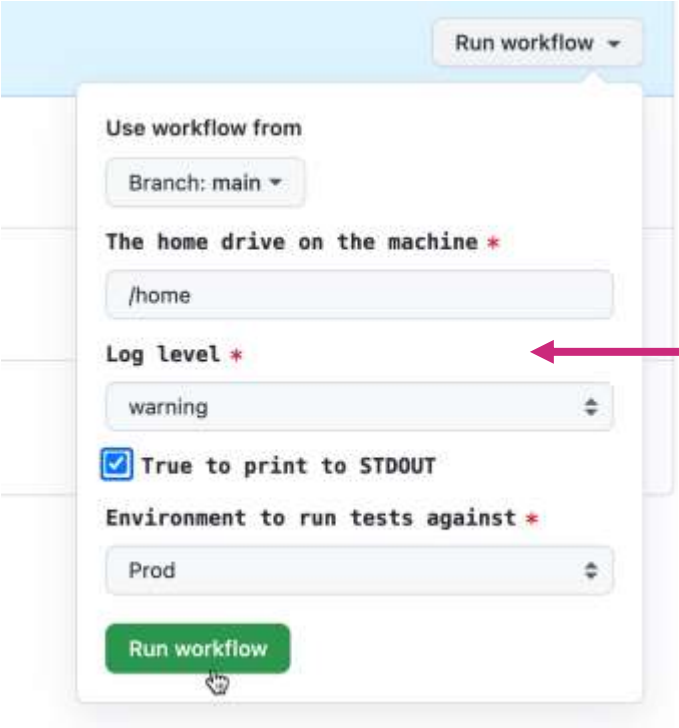
Edit file | Preview changes

```yaml
name: Starter Workflow

on:
  # Webhook events
  push:
    branches:
      - main
  issues:
    types: [opened, edited, milestoned]

  # Scheduled events

  schedule:
    - cron: '*/15 * * * *'
    - cron: '0 9-17 * * *'
    - cron: '11 11 * * 5'

  # Manual events
  workflow_dispatch:
    inputs:
      homedrive:
        description: 'The home drive on the machine'
        required: true
        default: '/home'
```

Runs every 15 minutes.
Actions schedules run at most every 5 minutes. Learn more

Events that triggers workflows

Manual events




```yaml
# 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)

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

```yaml
# 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

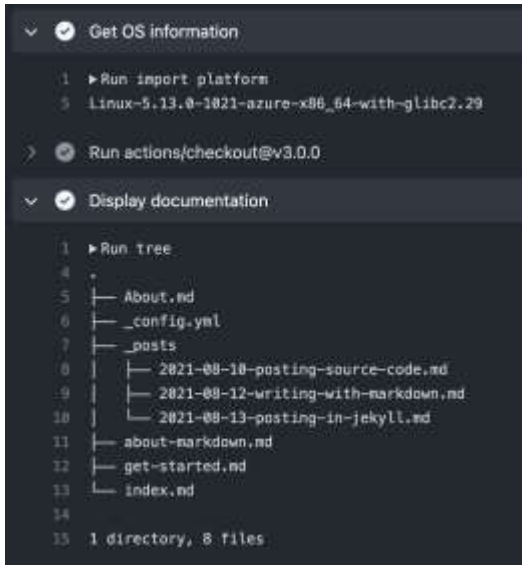
- Map – run in parallel by default
- Can be chained using **needs** keyword
- Runs on a **runner** in one process
- Contains a sequence of steps
- Steps can be a shell command (run) or an action (uses)

```yaml
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.




```yaml
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)
~/source/checkout  main                                                    15:2
❯
```

```
- uses: actions/checkout@a12a3943b4bdde767164f792f33f40b04645d846
- uses: actions/checkout@v3.0.0
- uses: actions/checkout@v3
- uses: actions/checkout@main
```

Actions

User docker images as actions

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

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

main container-demo / Dockerfile

wulfland Update Dockerfile

1 contributor

2 lines (2 sloc) 48 Bytes

```
1 FROM alpine:3.14.2
2 CMD ["echo", "Hello World!"]
```

Run ghcr.io/wulfland/container-demo:latest

```
1 ▶Run docker://ghcr.io/wulfland/container-demo:latest
2 /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
3 Hello World!
```

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`

```yaml
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

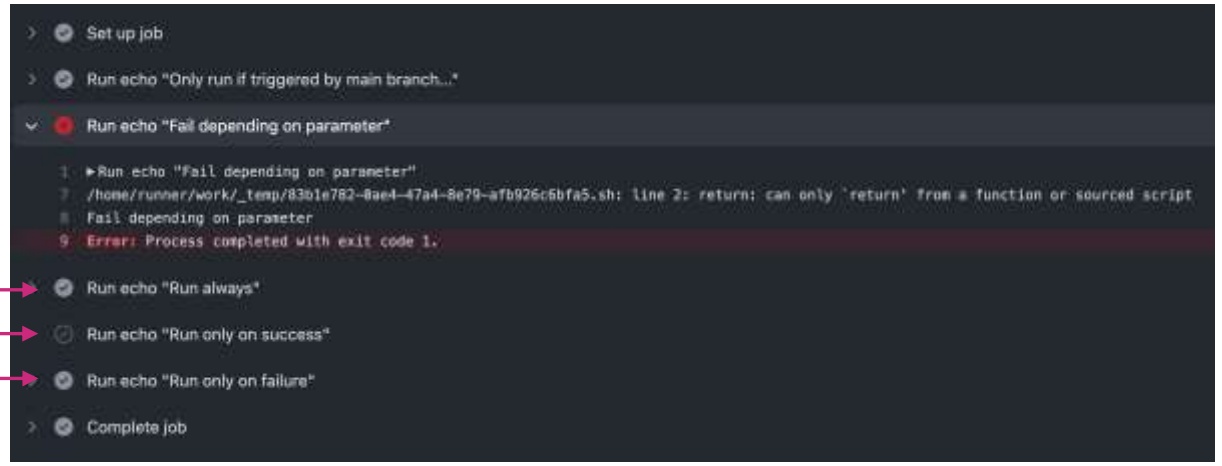
Set up job

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

```yaml
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()
```

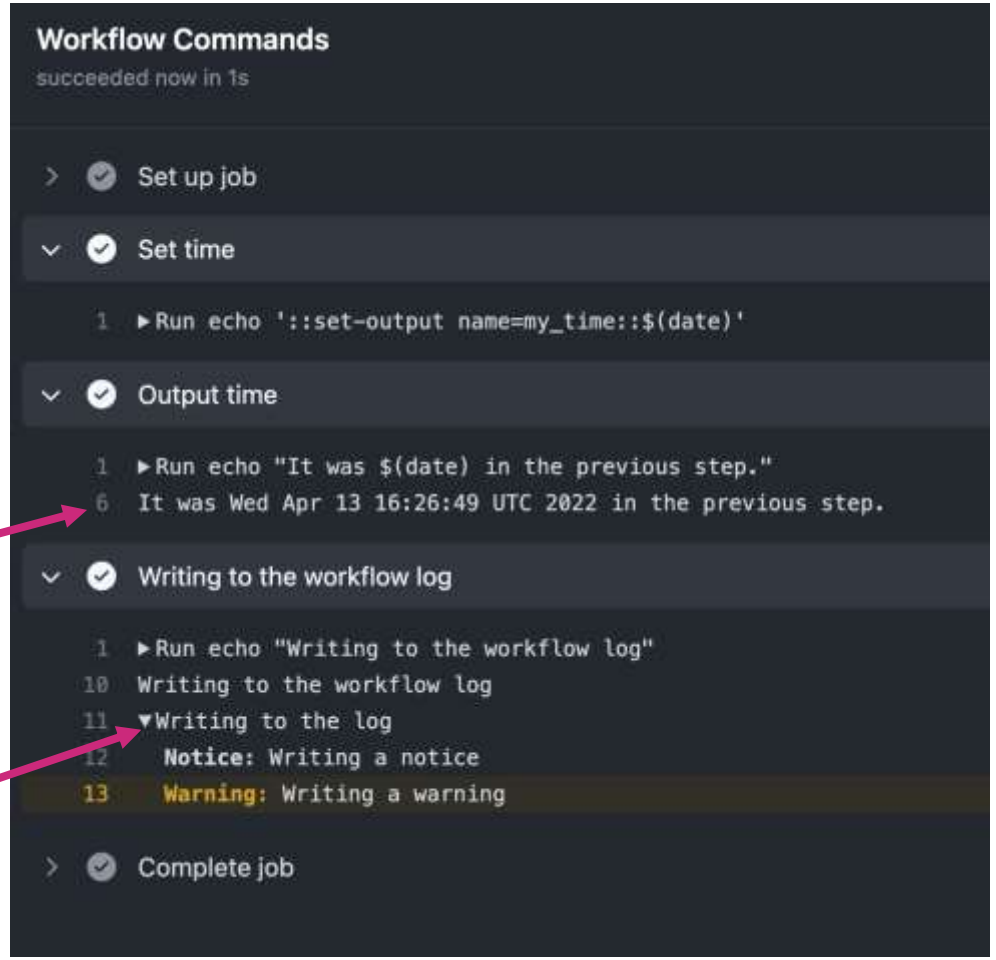


Workflow commands

Workflow commands

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

```yaml
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::"
```



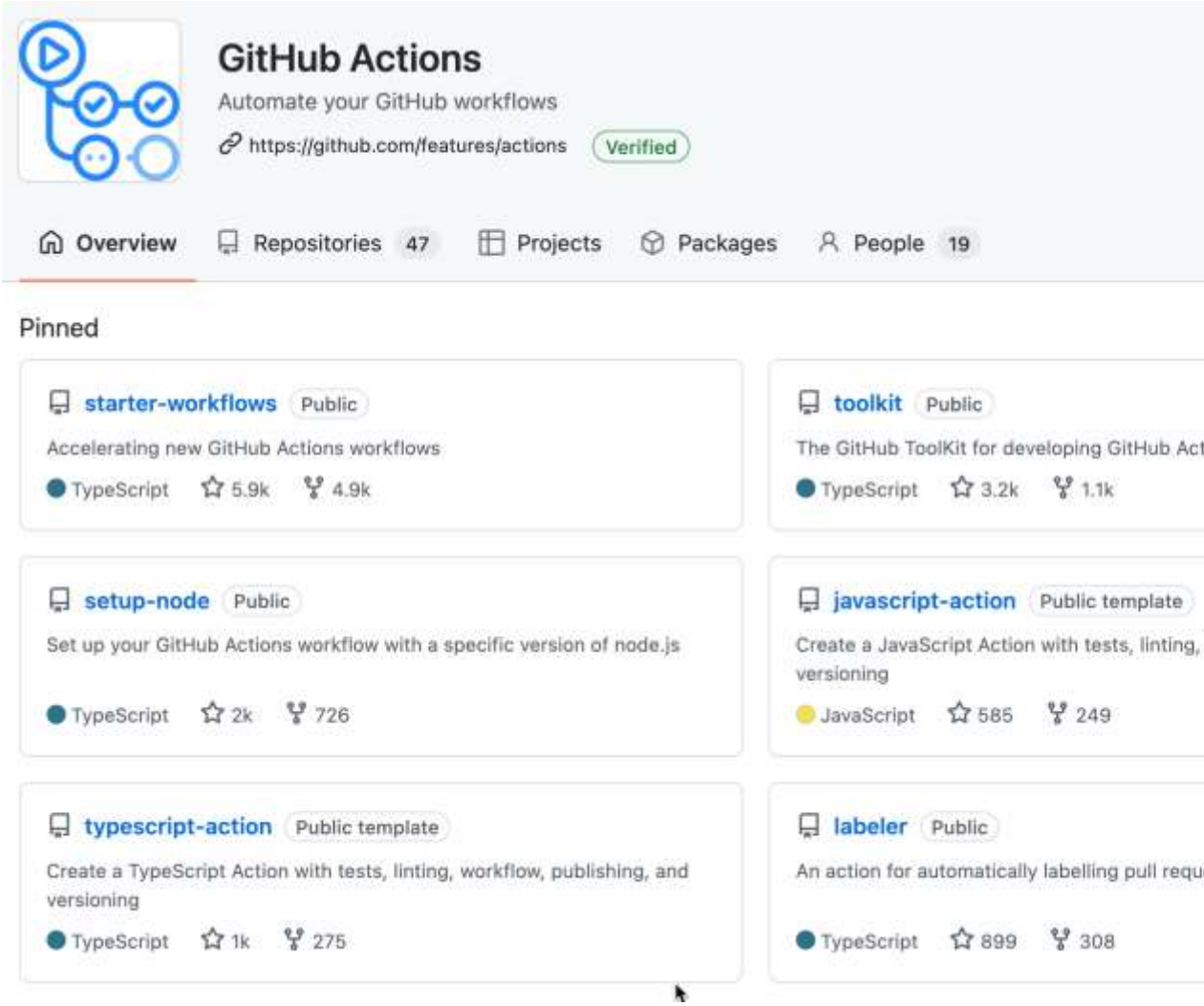
More advanced syntax elements

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

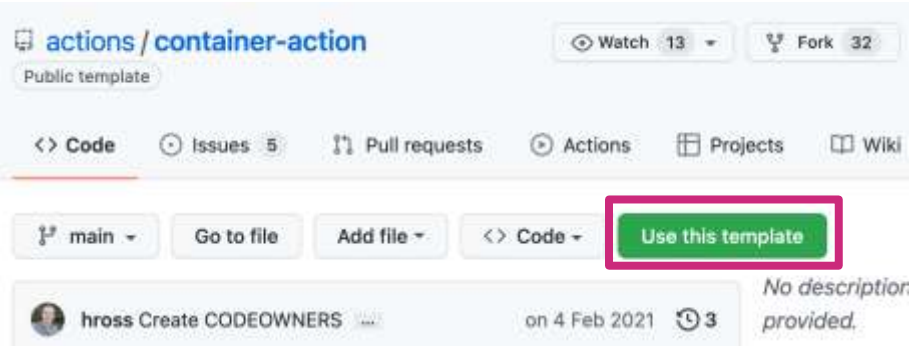
GitHub Actions

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



Container Actions

- Dockerfile or existing image
- Inputs



EXPLORER

HELLO-WORLD-DOCKE...

- action.yml
- CODEOWNERS
- Dockerfile
- entrypoint.sh
- LICENSE
- README.md

action.yml

```yaml
name: 'Wulfland Action'
description: 'Get started with Conta
author: '@wulfland'
branding:
  icon: 'award'
  color: 'green'
inputs:
  myInput:
    description: 'Input to use'
    default: 'world'
runs:
  using: 'docker'
  image: 'Dockerfile'
  args:
    - ${{ inputs.myInput }}
```

Dockerfile

```dockerfile
FROM alpine:3.10

COPY LICENSE README.md /

COPY entrypoint.sh /entrypoint.sh

ENTRYPOINT ["/entrypoint.sh"]
```

entrypoint.sh

```shell
#!/bin/sh -l

echo "hello $1"
```

Container Actions

- Dockerfile or existing image
- inputs

15 lines (13 sloc) | 398 Bytes

```yaml
name: Test Action

on: [workflow_dispatch]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Run my own container action
        id: hello-action
        uses: wulfland/hello-world-docker-action@v1.2
        with:
          myInput: '@wulfland'
      - name: Output time set in the container
        run: echo "The time in the container was ${{ steps.hello-action.outputs.time }}"
```

test

succeeded 17 seconds ago in 5s

Set up job

Build wulfland/hello-world-docker-action@v1.2

```
▶Build container for action use: '/home/runner/work/_actions/wulfland/hello-world-docker-action/v1.2/Dockerfile'.
```

Run my own container action

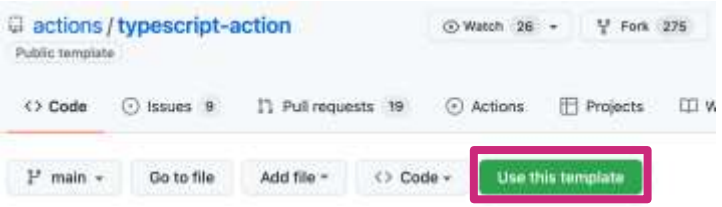
```
▶Run wulfland/hello-world-docker-action@v1.2
/usr/bin/docker run --name bcf090f977186e9874b92a188d8a409df5216_6a7e50 --label 2bcf09 --workdir /github/workspace --rm -e INPUT_MYI
GITHUB_REPOSITORY_OWNER -e GITHUB_RUN_ID -e GITHUB_RUN_NUMBER -e GITHUB_RETENTION_DAYS -e GITHUB_RUN_ATTEMPT -e GITHUB_ACTOR -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 GITHUB_WO
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
action":"/github/workspace" 2bcf09:0f977186e9874b92a188d8a409df5216  "@wulfland"
hello @wulfland
```

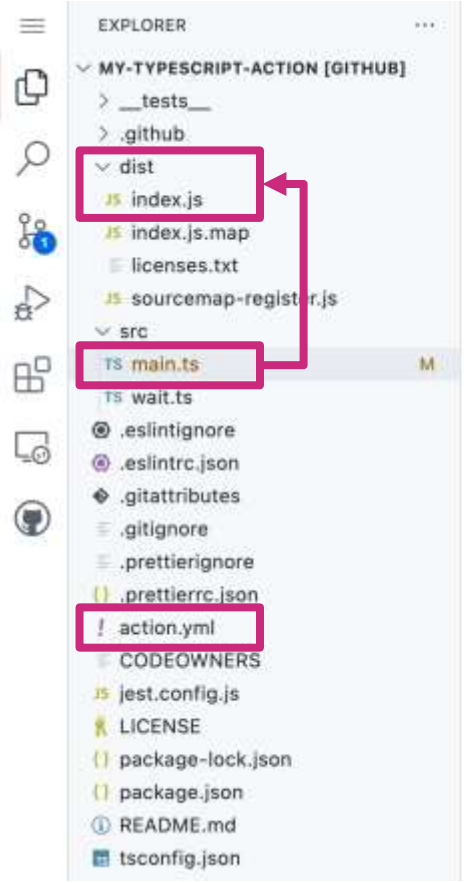
Output time set in the container

```
▶Run echo "The time in the container was Wed Apr 13 18:16:52 UTC 2022"
The time in the container was Wed Apr 13 18:16:52 UTC 2022
```

Complete job

JavaScript Actions





action.yml

```yaml
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'
```

main.ts M

src > main.ts > ...

```typescript
import * as core from '@actions/core'
import {wait} 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 wait(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()
```

Composite Actions

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

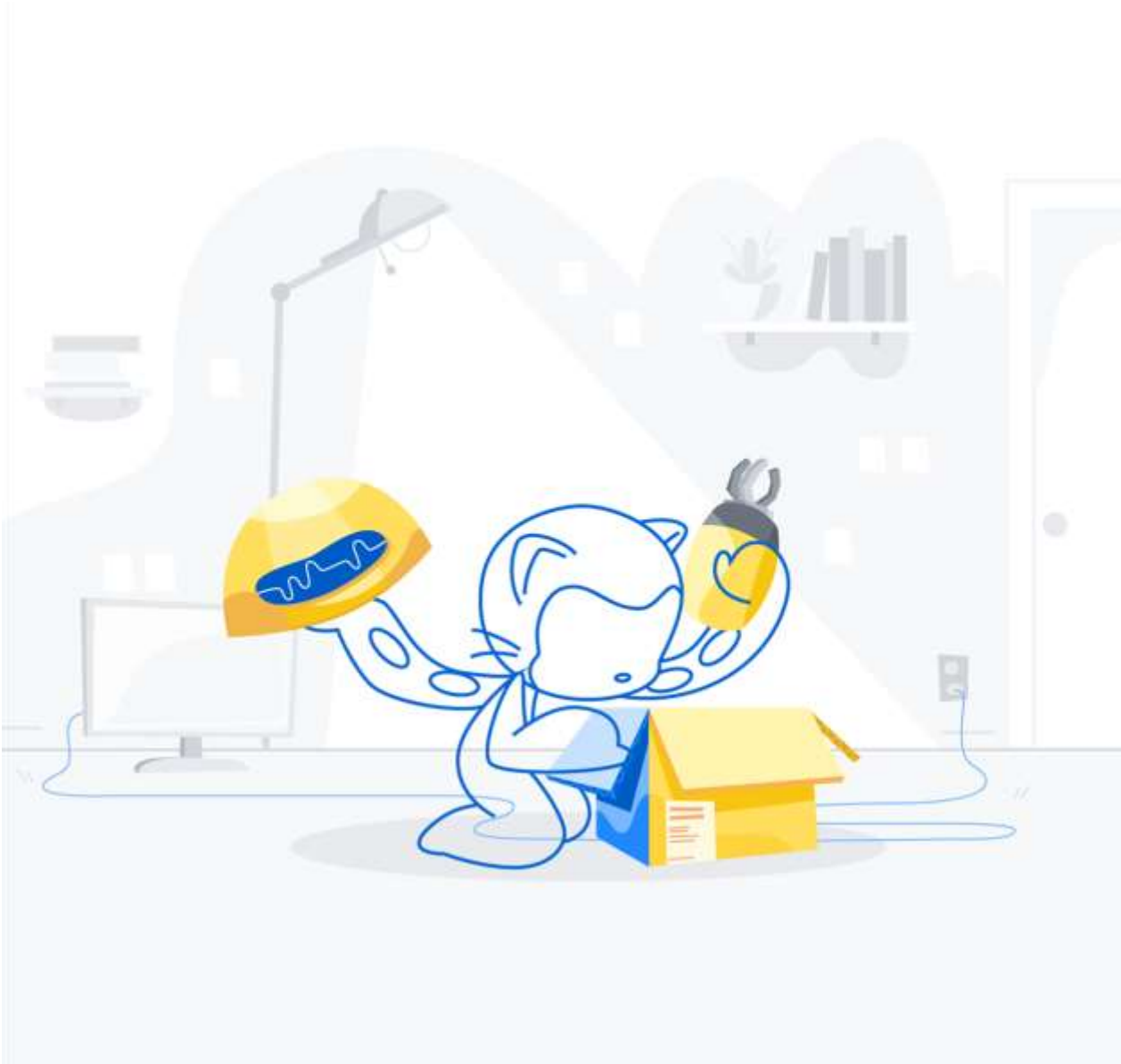
28 lines (25 sloc) | 689 Bytes

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

    - id: random-number-generator
      run: echo "::set-output name=random-id::$(echo $RANDOM)"
      shell: bash

    - run: echo "${{ github.action_path }}" >> $GITHUB_PATH
      shell: bash

    - run: echo "Goodbuye $YOU"
      shell: bash
      env:
        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

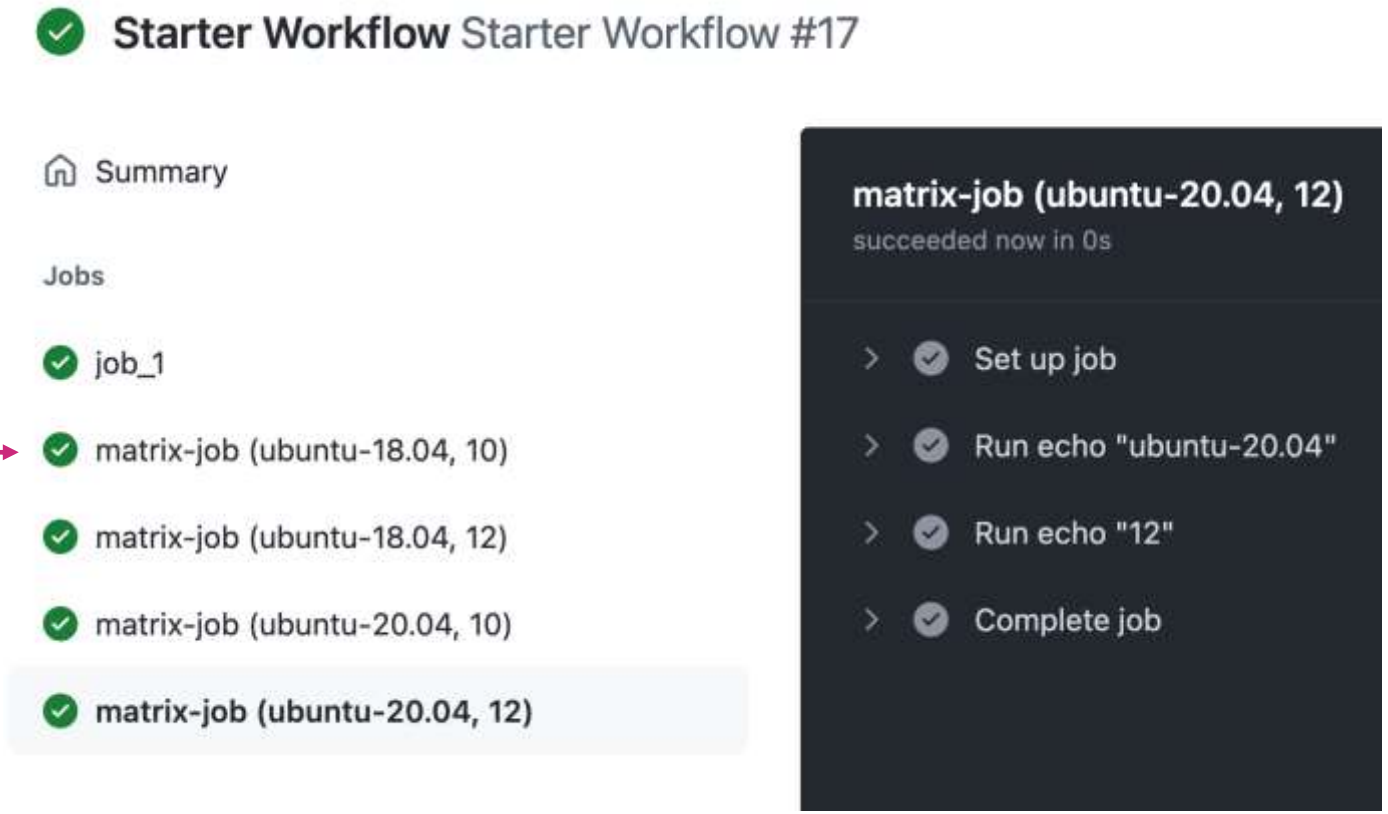


Actions for CI / CD

Strategy

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

```yaml
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 }}"
```



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

```yaml
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

```yaml
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

```yaml
- 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

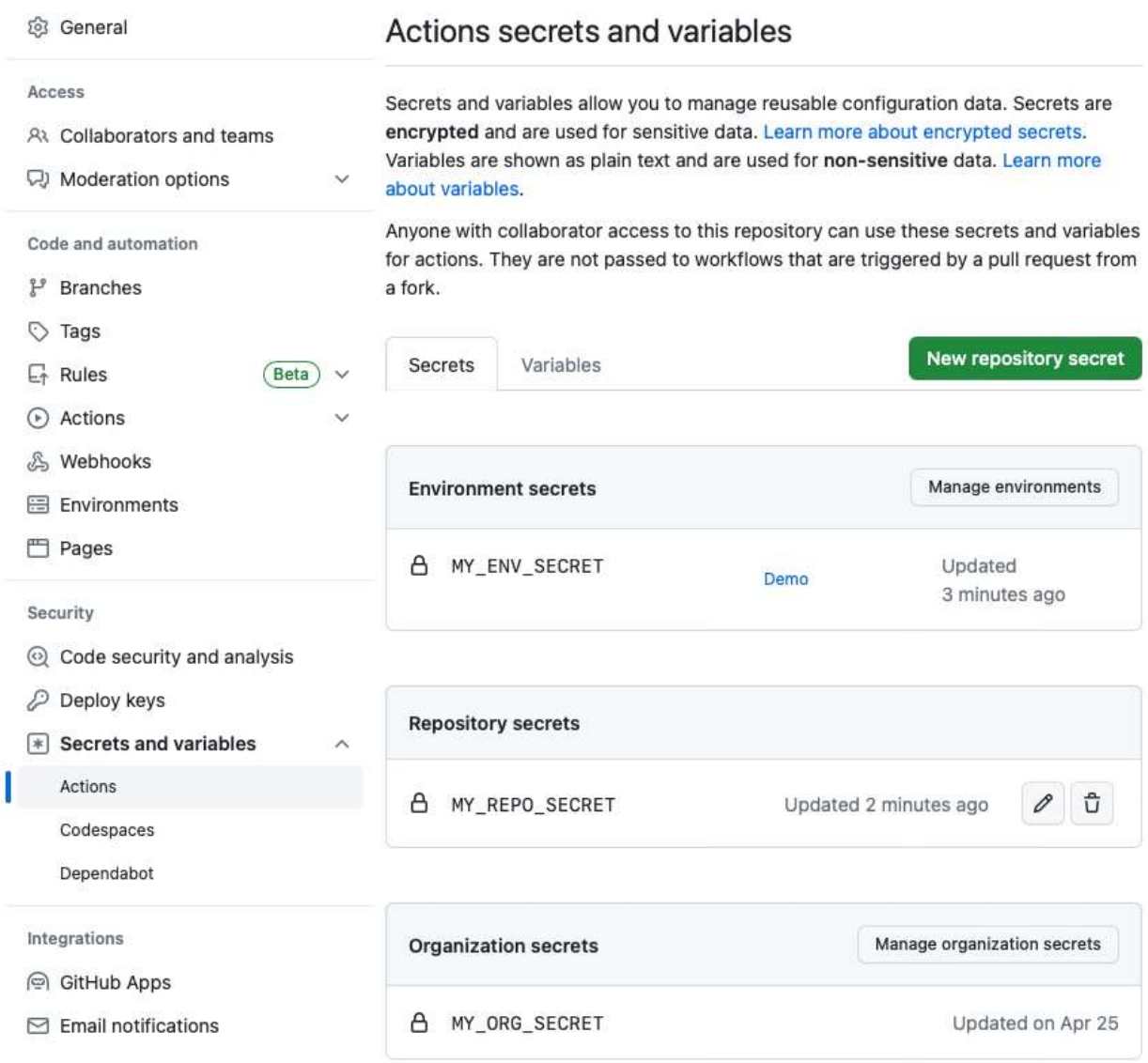
Best Practices

- 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

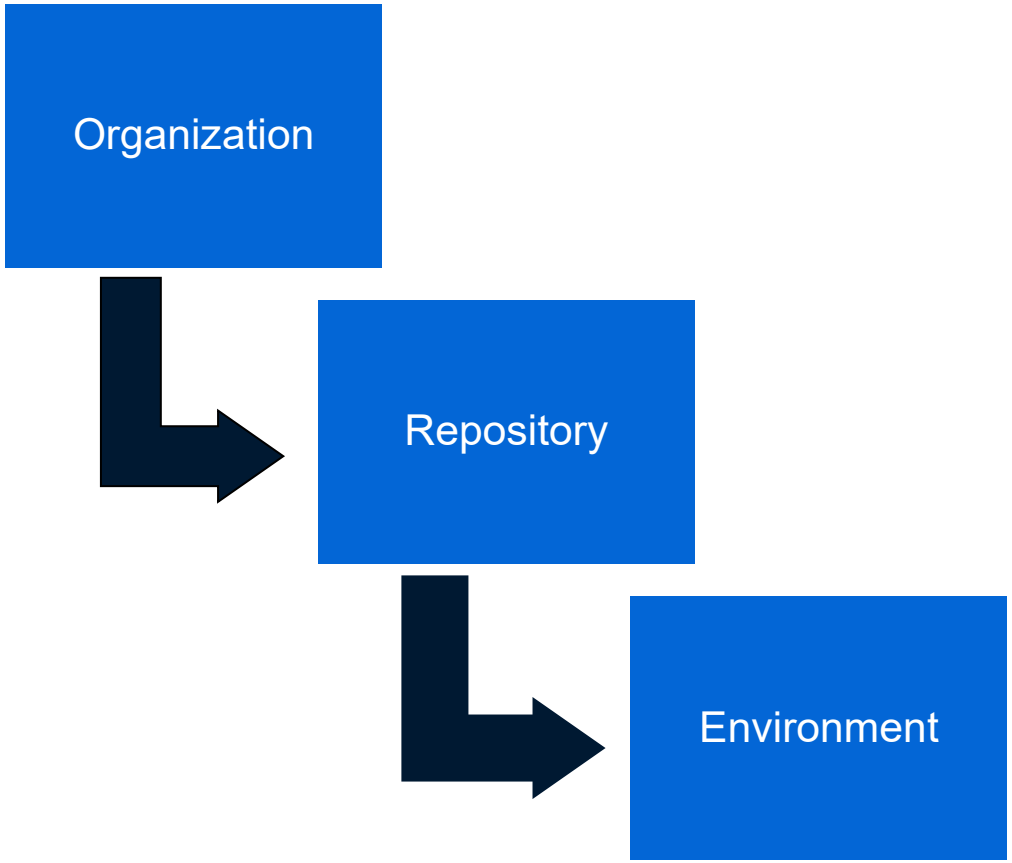
GitHub Secret store

- 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!



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

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

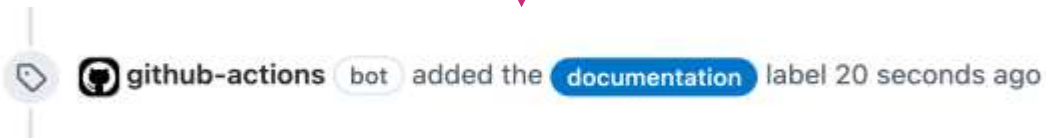
```yaml
permissions: read-all
```

```yaml
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

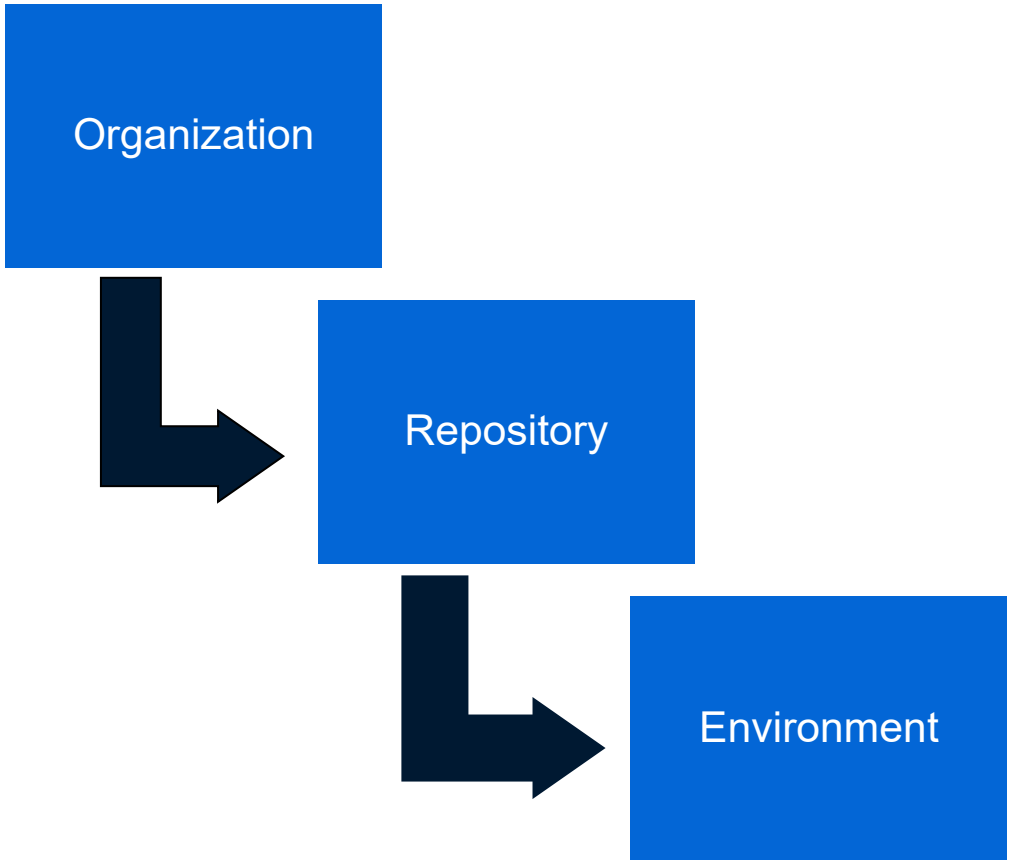
Perform actions as “github-actions”:

```yaml
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

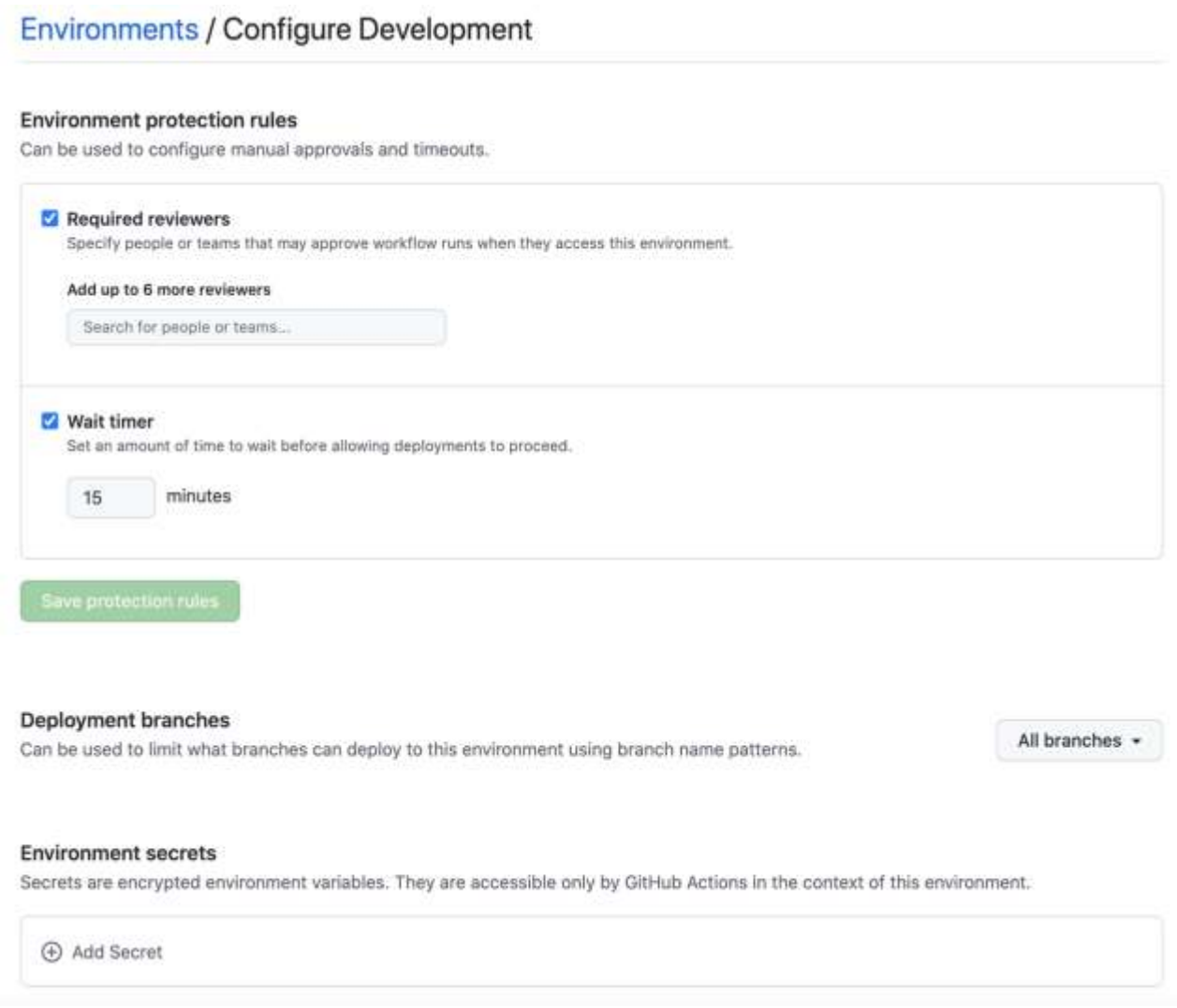
- Same setup as secrets, but no redacting
- Defined on org, repo, or environment level
- vars context
  - `${{ vars.MY_VAR }}`
  - Set as input (`with:`) or environment (`env:`) for actions
- **Not** masked in log



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

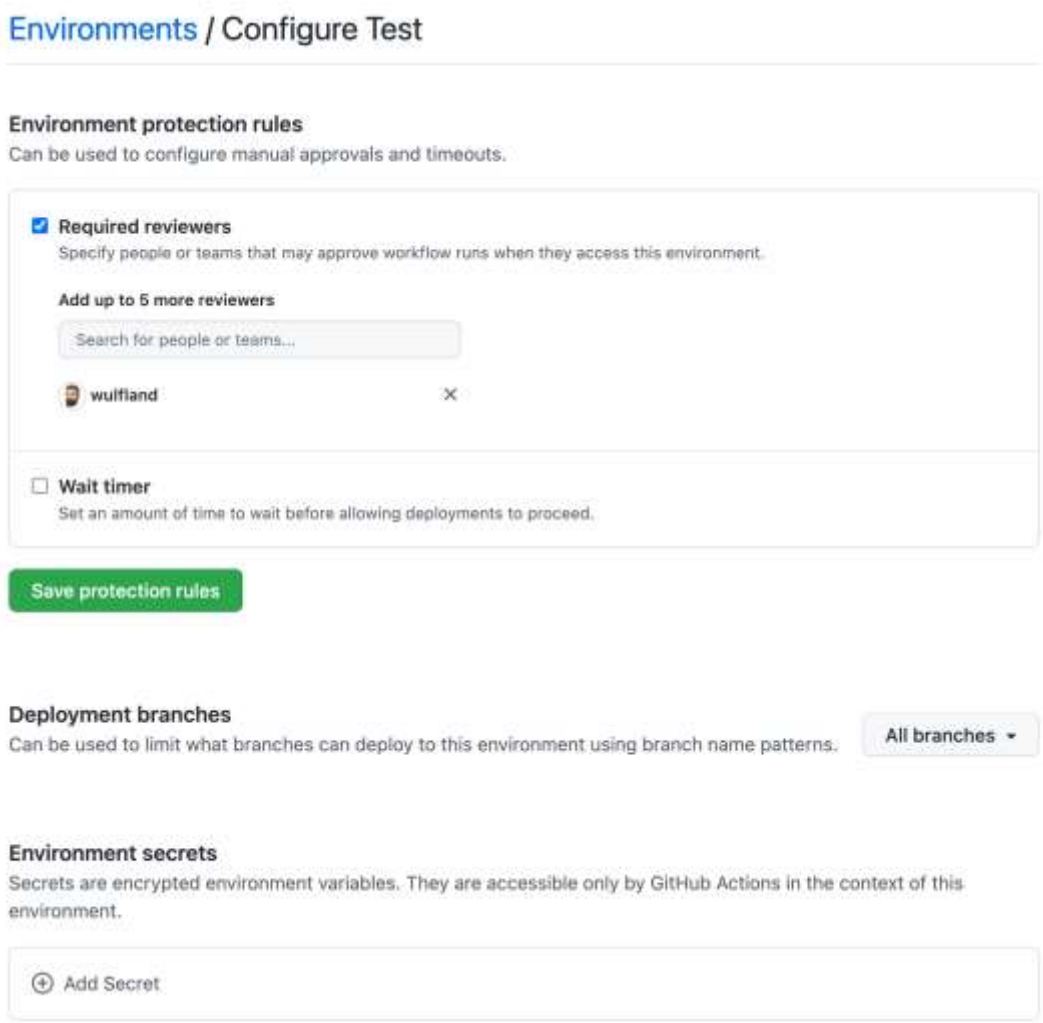


Environments

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

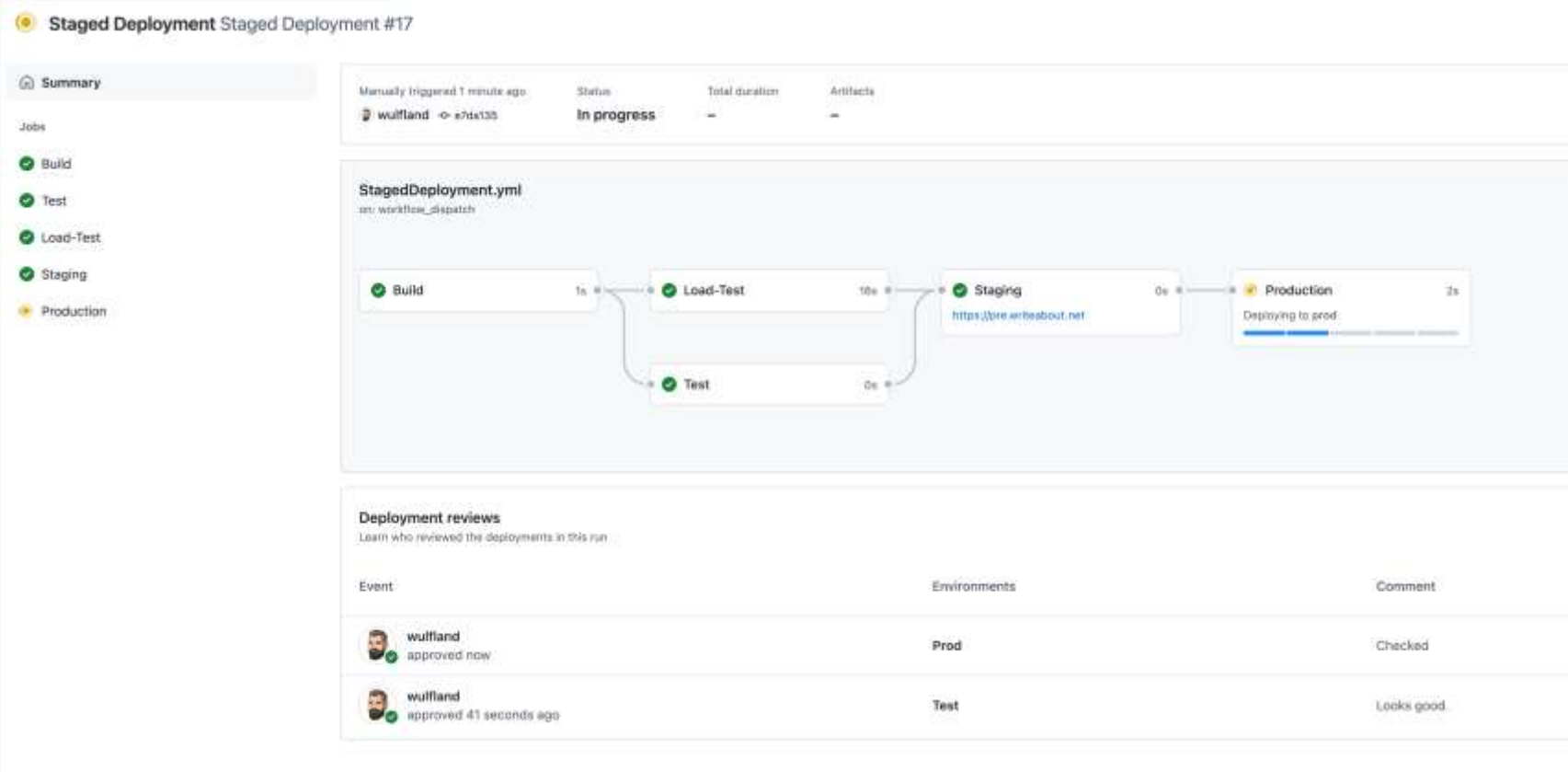
```yaml
Test:
  runs-on: ubuntu-latest
  environment: Test
  needs: Build
  steps:
  - name: Test app
    run: echo "Testing..."
```

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



Environments

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



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: • **Standard_DS2_v2** virtual machines in Microsoft **Azure** • 2-core CPU • 7 GB of RAM • 14 GB of SSD disk space	**Hardware**: • 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	ubuntu-22.04
Ubuntu 20.04	ubuntu-20.04	ubuntu-20.04
macOS 11	macos-latest or macos-11	macOS-11
macOS 10.15	macos-10.15	macOS-10.15
Windows Server 2022	windows-latest or windows-2022	windows-2022
Windows Server 2019	windows-2019	windows-2019
Windows Server 2016	windows-2016	windows-2016

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
Current runner version: '2.289.2'
▼Operating System
  Ubuntu
  20.04.4
  LTS
▼Virtual Environment
  Environment: ubuntu-20.04
  Version: 20220405.4
  Included Software: https://github.com/actions/virtual-environments/blob/ubuntu20/20220405.4/images/linux
  Image Release: https://github.com/actions/virtual-environments/releases/tag/ubuntu20%2F20220405.4
▶Virtual Environment Provisioner
▼GITHUB_TOKEN Permissions
  Actions: write
  Checks: write
  Contents: write
  Deployments: write
  Discussions: write
  Issues: write
  Metadata: read
  Packages: write
  Pages: write
  PullRequests: write
  RepositoryProjects: write
  SecurityEvents: write
  Statuses: write
Secret source: Actions
Prepare workflow directory
Prepare all required actions
Getting action download info
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."
```

https://docs.github.com/en/billing/managing-billing-for-github-actions/about-billing-for-github-actions

Larger runners

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

sitories 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

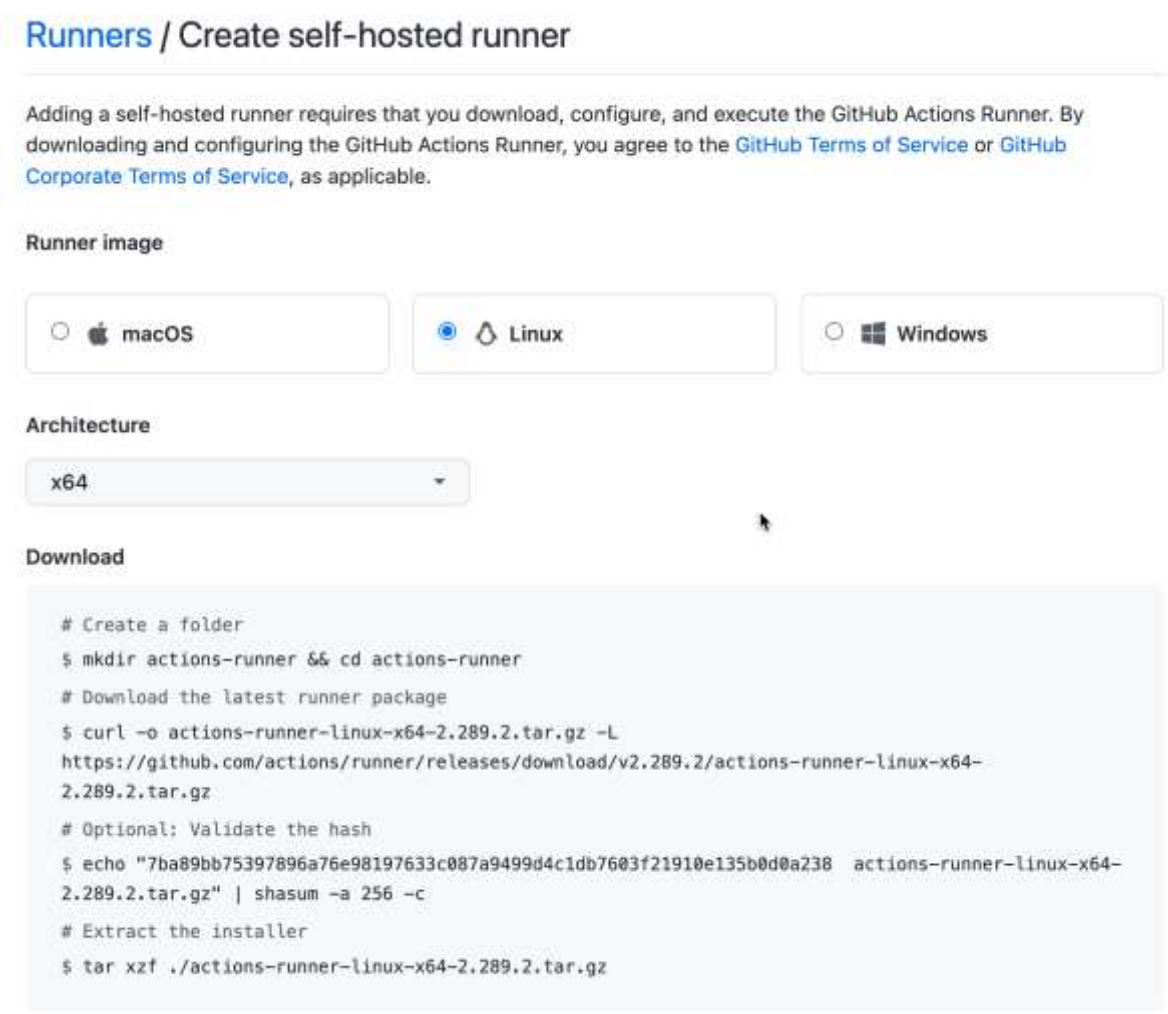
Image: Ubuntu Latest (20.04)

Public IP range: 20.237.78.80/28

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

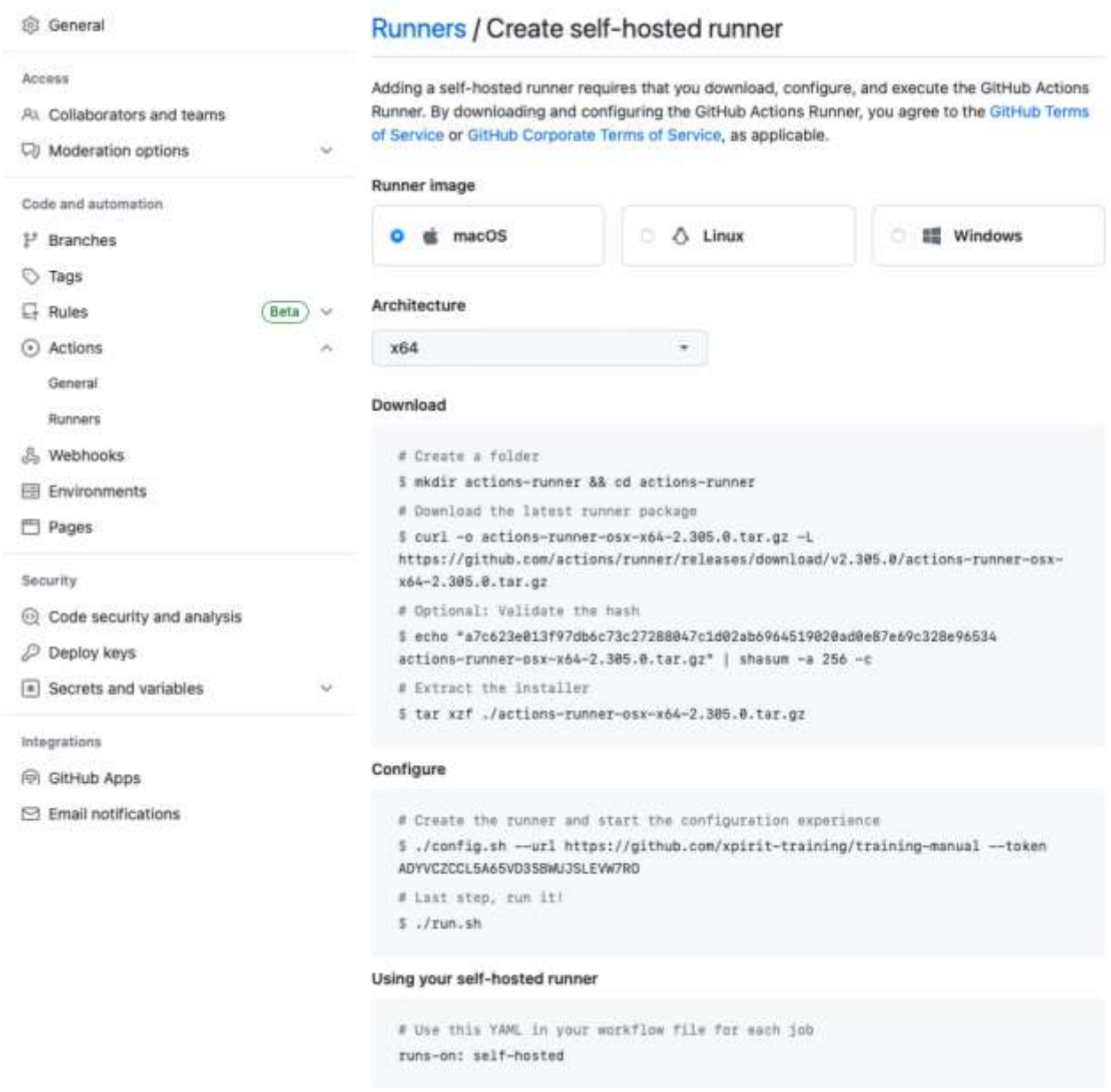
Self-hosted runners

- **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**



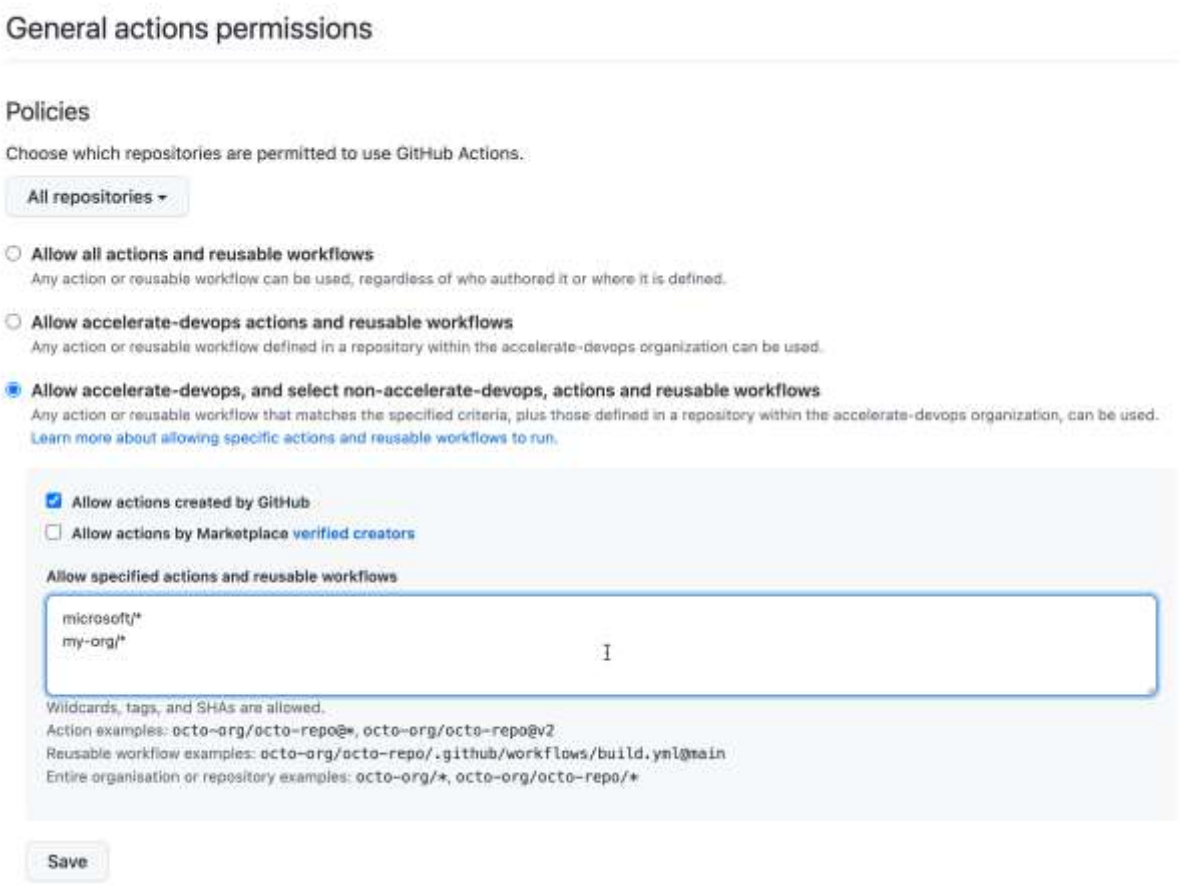
Adding self-hosted runners

- 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)



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)



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

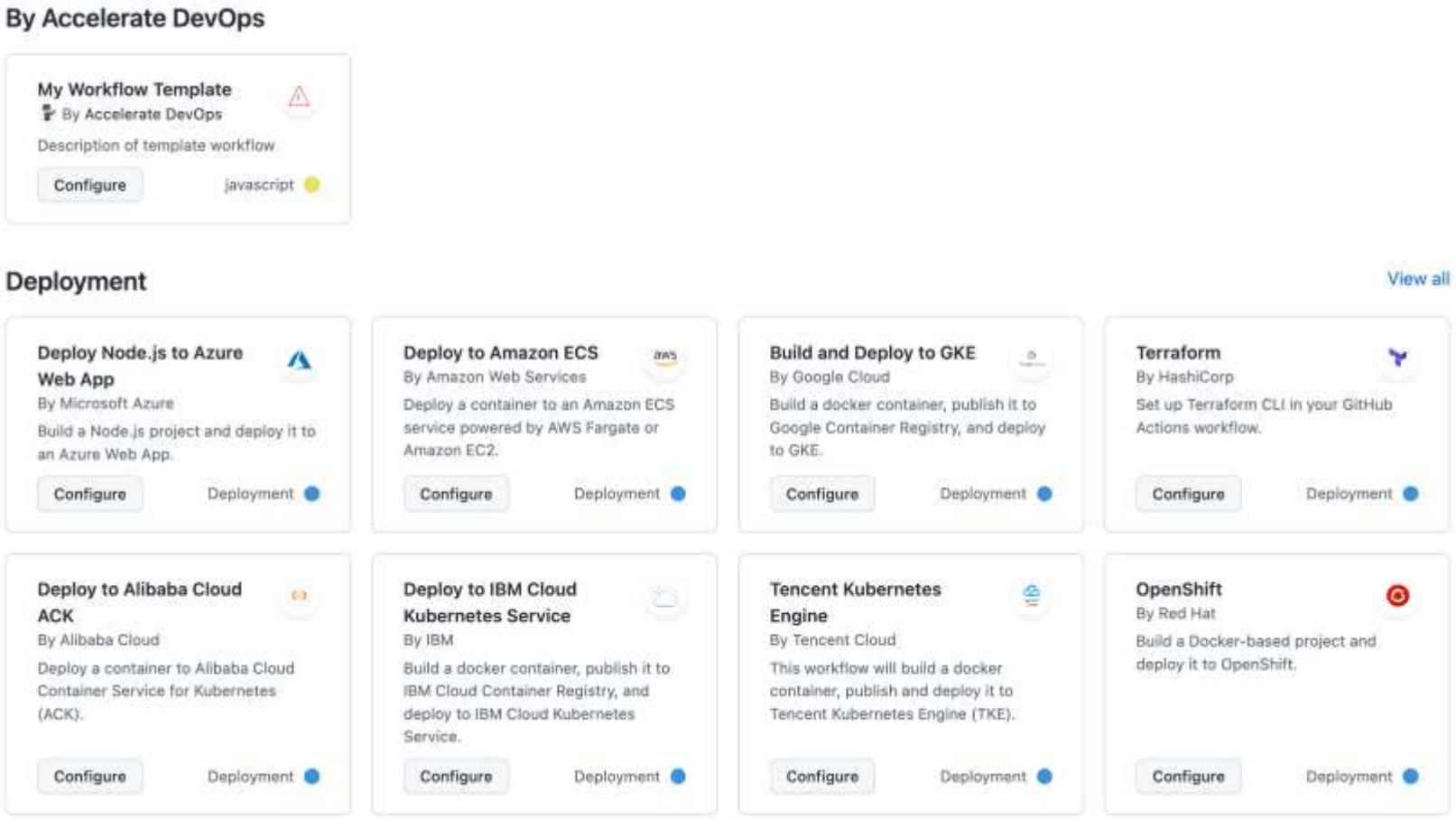
Forked repositories will contain the same Actions configuration as the parent repository, including the self-hosted runners. Creates the potential for a fork to run malicious code on a runner inside your network. For this reason, it is highly recommended to use self-hosted runners only with **private**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

Sharing Workflows

Workflow templates

Workflow templates

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



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

```yaml
name: Reusable workflow

on:
  workflow_call:
    inputs:
      who-to-greet:
        description: 'The person to greet'
        type: string
        required: true
        default: World
    outputs:
      current-time:
        description: 'The time when greeting.'
        value: ${{ jobs.reusable-job.outputs.current-time }}

jobs:
  reusable-job:
    runs-on: ubuntu-latest
    outputs:
      current-time: ${{ steps.time.outputs.current-time }}
    steps:
      - name: Greet someone
        run: echo "Hello ${{ inputs.who-to-greet }}"
      - name: Set time
        id: time
        run: echo "::set-output name=current-time::$(date)"
```

```yaml
name: Reuse other workflow

on: [workflow_dispatch]

jobs:
  call-workflow:
    uses: ./.github/workflows/reusable.yml
    with:
      who-to-greet: '@wulfland'

  use-output:
    runs-on: ubuntu-latest
    needs: [call-workflow]
    steps:
      - run: echo "Time was ${{ needs.call-workflow.outputs.current-time }}"
```

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

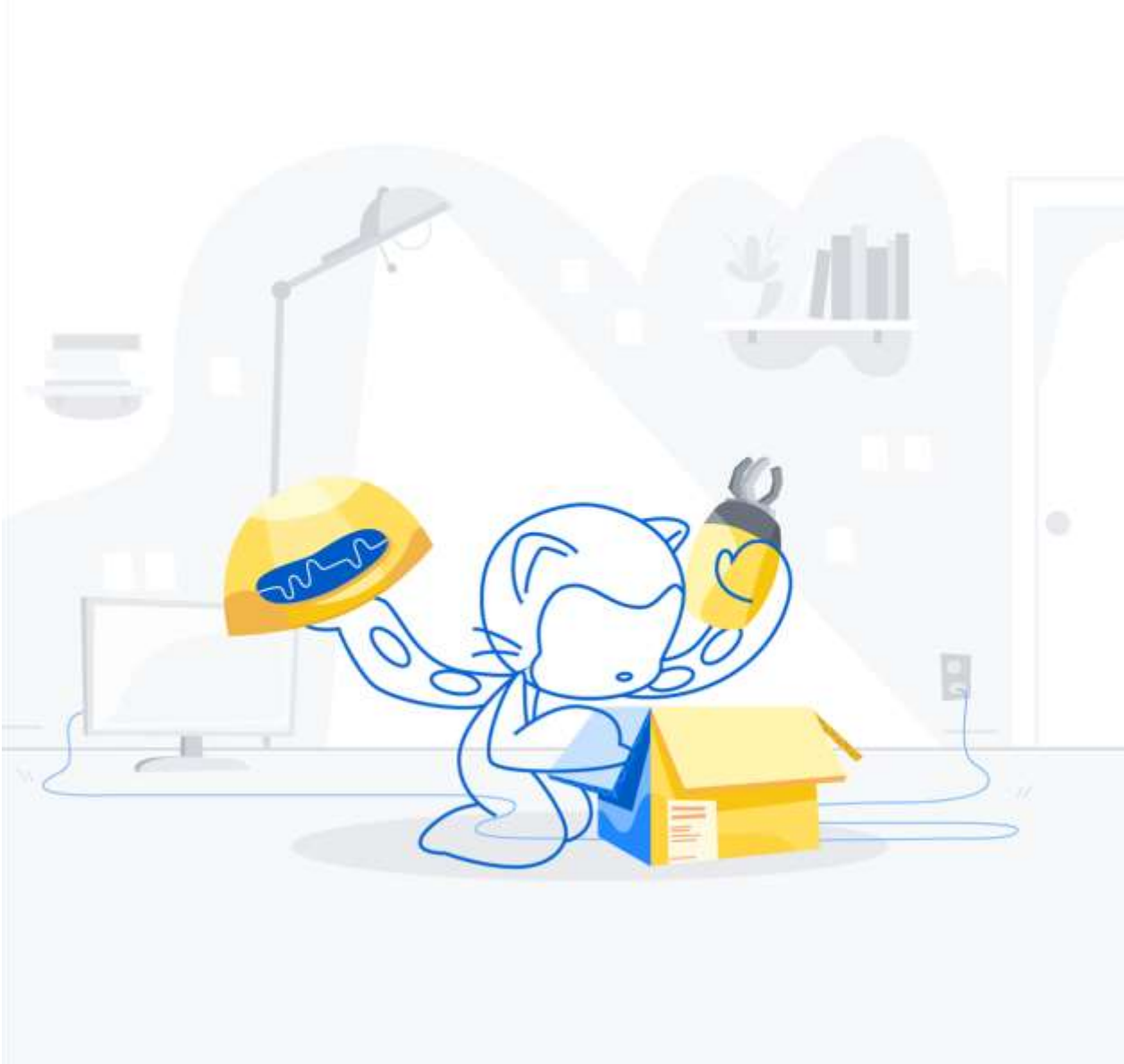
Composite action:

- 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