GitHub

GitHub Actions

Fundamentals

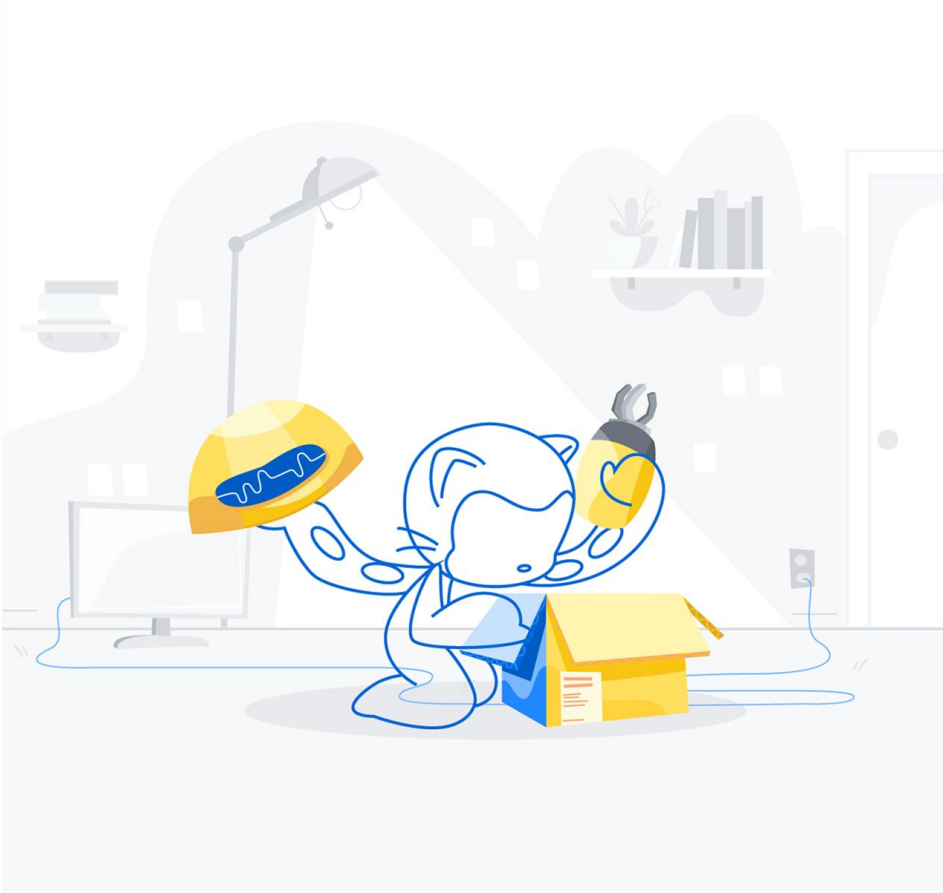
Presented by GitHub Professional Services

Objectives

- Understand the **basic components** of GitHub Actions and its **use cases**
- Understand the GitHub Actions **syntax**, both for actions and workflows
- Know how to navigate GitHub Actions **GUI and documentation**
- Understand how to leverage actions written by the **community**
- Create **custom actions**
- Automate both **CI/CD and non-CI/CD** use cases
- Know how to use **environments and secrets**
- Understand how to **migrate** to GitHub Actions from a different CI/CD system
- Understand the differences between **GitHub-hosted and self-hosted runners**
- Understand **best practices** related to GitHub Actions

Agenda

- Introduction to GitHub Actions
- Workflow syntax
- Environments and secrets
- Managing workflows & Actions
- Building Actions
- Migration
- Runners
- CI/CD workflows 🎉
- Demos!



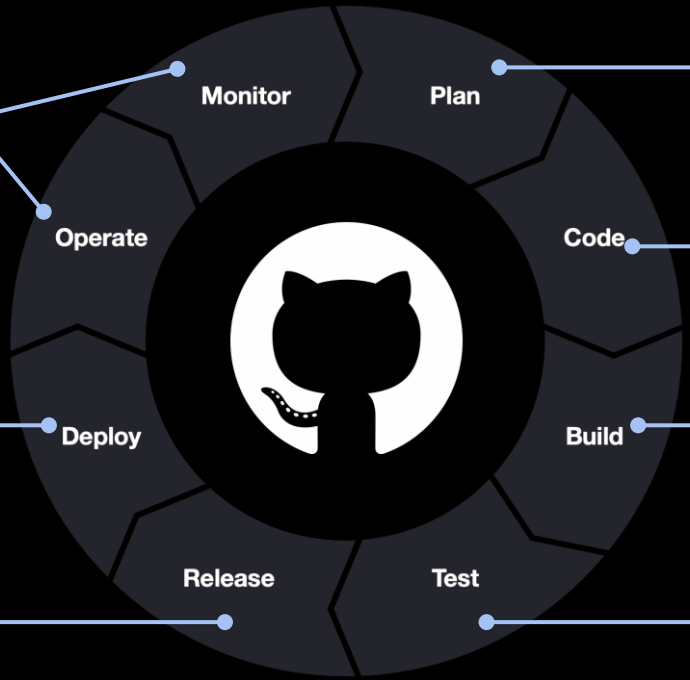
Introduction

What is GitHub Actions

GitHub Actions is a GitHub product that allows you to **automate your workflows.**

- Workflows stored as `yml` files
- Fully integrated with GitHub
- Respond to GitHub events
- Live logs and visualized workflow execution
- Community-powered workflows
- GitHub-hosted or self-hosted runners
- Built-in secret store

Use cases across your SDLC



Not suitable for

- Triage issues
- Project boards

- Code review
- Auto-merge
- Auto-labeling

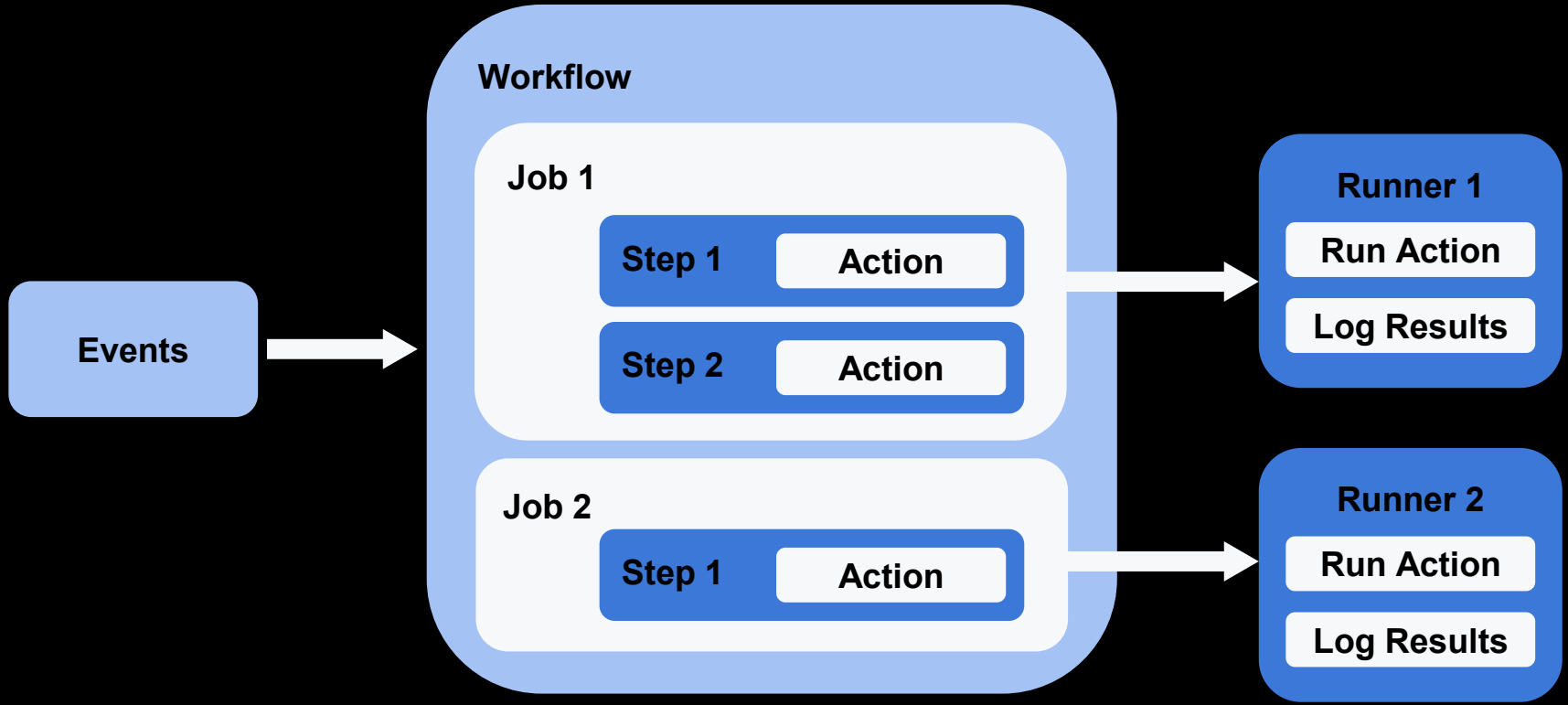
- Deploy application

- Compile and build
- Store artifacts

- Publish release
- Release notes
- Upload artifacts

- Automated testing
- Linters
- CodeQL
- Consume Packages

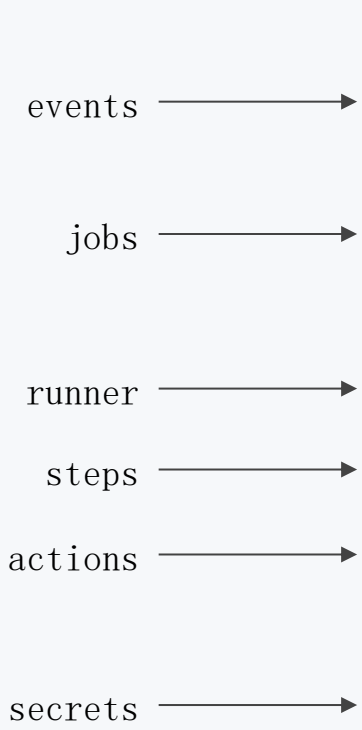
Key components



Workflow syntax

Basic syntax

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




```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 }}
```

Events

Webhook events

- Pull request
- Issues
- Push
- Release
- ...

Scheduled events

Manual events


```yaml
name: Super Linter workflow

on:
  issues:
    types: [closed, reopened]

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 }}
```

Events

Webhook events

- Pull request
- Issues
- Push
- Release
- ...

Scheduled events

Manual events


```yaml
name: Super Linter workflow

on:
  schedule:
    - cron: '30 6 * * 5' # every Friday 06:30 UTC

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 }}
```

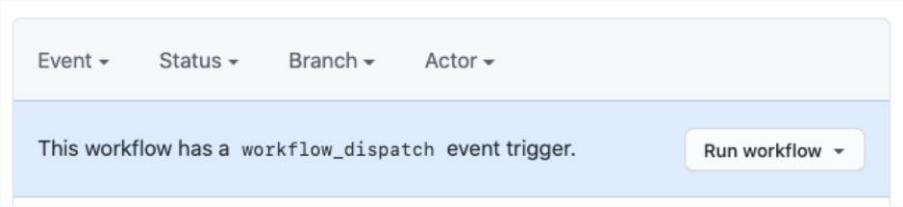
Events

Webhook events
- Pull request
- Issues
- Push
- Release
- ...

Scheduled events

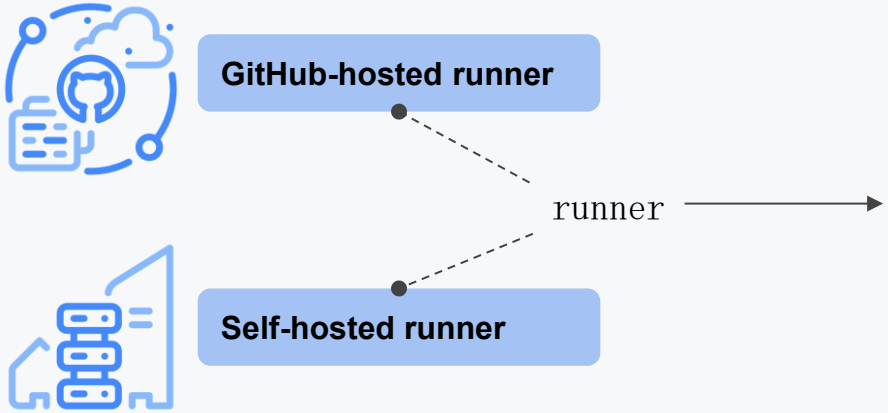
Manual events
- workflow_dispatch
- repository_dispatch

events →




```yaml
name: Super Linter workflow

on:
  workflow_dispatch:

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 }}
```

Runners




```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 }}
```

Runners

GitHub-hosted runner

- OS: ubuntu, windows, or macOS
- Ephemeral
- 2-core CPU (macOS: 3-core)
- 7 GB RAM (macOS: 14 GB)
- 14 GB SSD disk space
- Software installed: wget, GH CLI, AWS CLI, Java, …
- Not currently available on GHES

runner →

```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base
    runs-on: windows-latest
    steps:
      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Runners

Self-hosted runner

- Custom hardware config
- Run on OS not supported on GitHub-hosted runner
- Reference runner using custom labels
- Can be grouped together
- Control which organizations/repositories have access to which runners/runner groups
- Do not use with public repositories!

runner →

```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: [self-hosted, linux, ARM64]

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Actions

Reusable units of code that can be referenced in a workflow

GitHub runs them in Node.js runtime, or in Docker containers

Reference an Action, or run scripts directly

Can be referenced in three ways:

- Public repository
- The same repository as your workflow (local actions)
- A published Docker container image on DockerHub

script →

public actions →

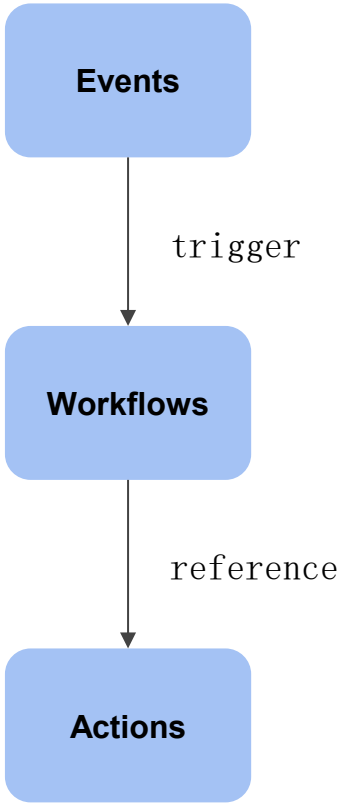
local action →

docker image →

```yaml
name: Super workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - run: echo "Hello World"

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}

      - uses: ./path/to/action

      - uses: docker://alpine:3.8
```

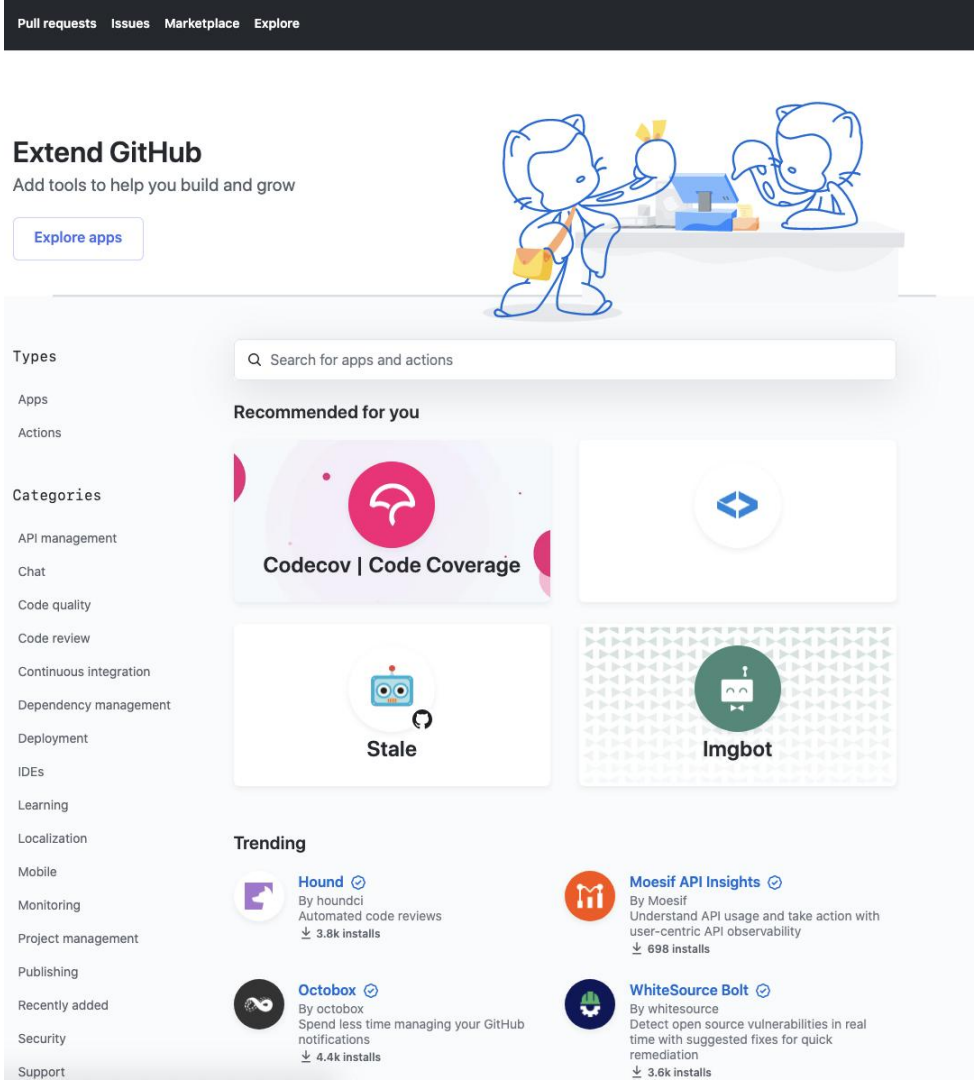
Quick summary

- Events trigger workflows, e.g. a push to a branch
- Workflows contain one or more jobs, which contains one or more steps
- These steps can reference actions or execute commands
- The term “*GitHub Actions*” include all components, not just the Actions themselves



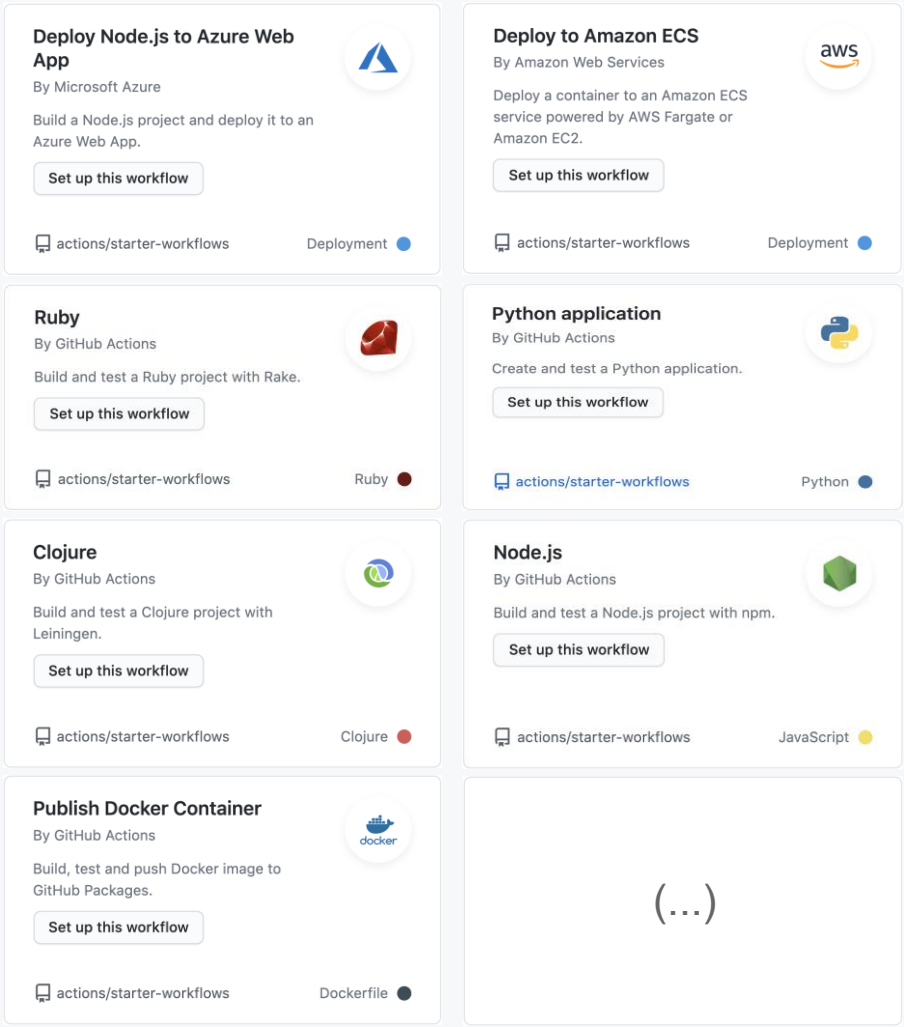
GitHub Marketplace

- Discover open-source Actions across multiple domains
- ~9,000 Actions (and counting...)
- Verified creators
- Reference these Actions directly in your workflow
- Integrated into the GitHub editor

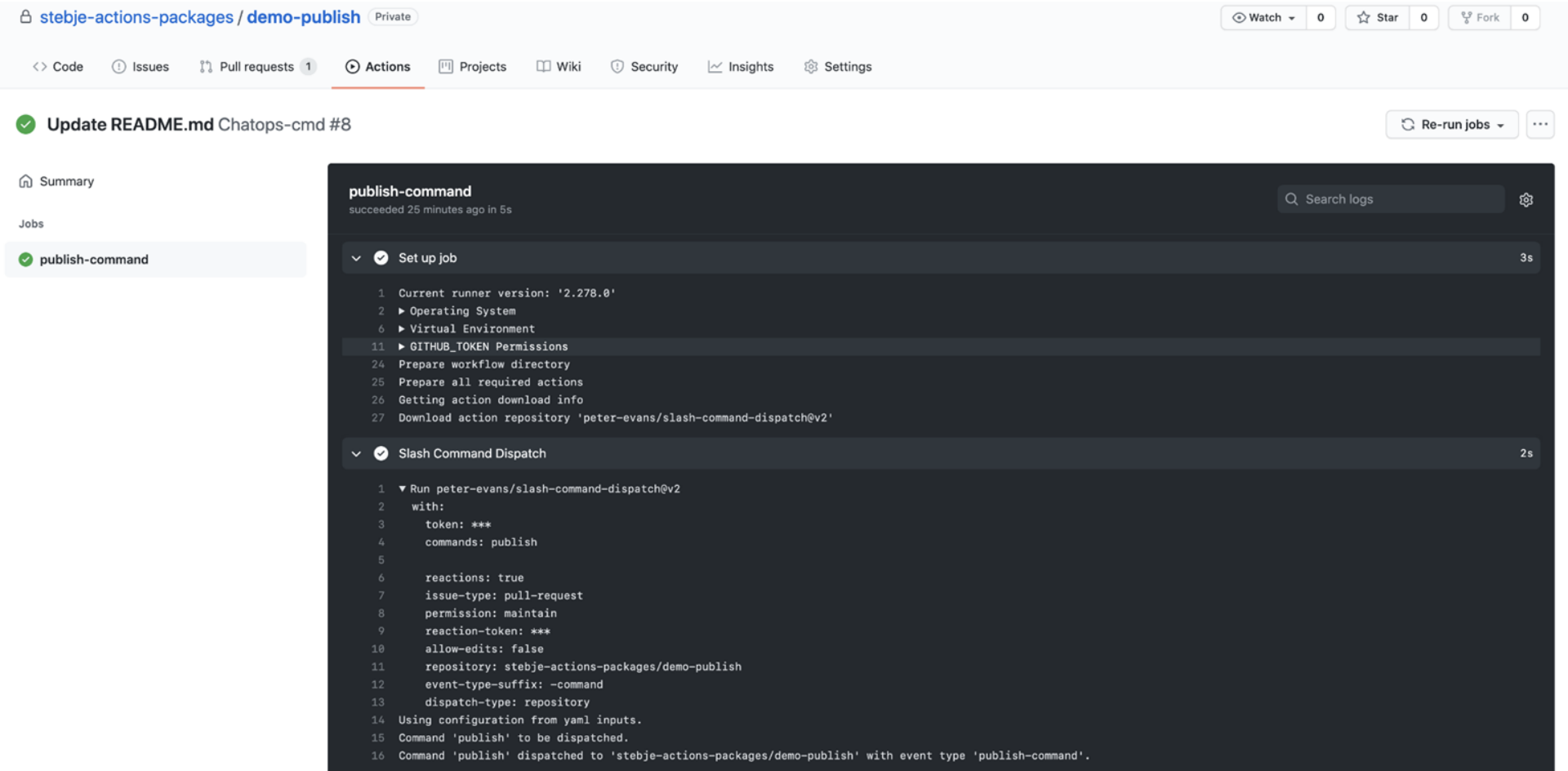


Starter workflows

- Preconfigured for specific languages and frameworks
- GitHub analyzes your code and suggests the workflows based on your language and framework
- For GHES 3.x: A number of starter workflows come pre-packaged with the release.



Workflow logs



Advanced syntax

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

Function expressions

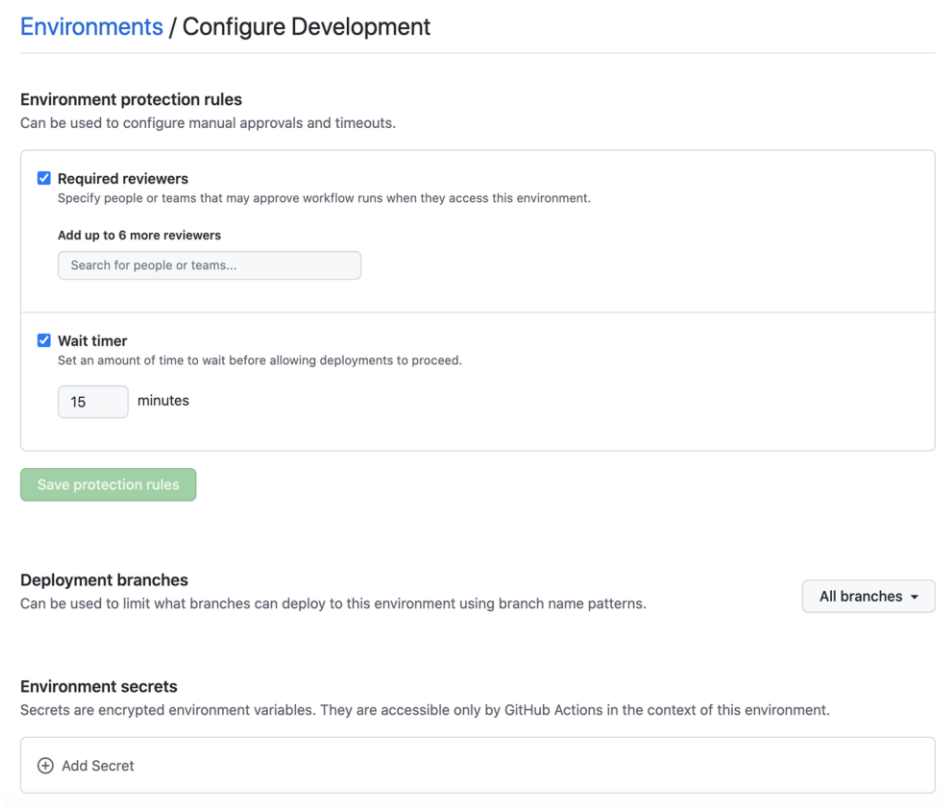
Syntax element	Description
`contains`	Check if a string is contained in another
`startsWith/endsWith`	Check start/end of a string
`format`	Format outputs
`join`	Join arrays into strings
`toJSON/fromJSON`	Make string JSON and JSON strings
`hashFiles`	Create a hash from an input file. Useful for caching
`always/success/failure/cancelled`	Workflow statuses. Useful for conditional runs

Demo

Environments and secrets

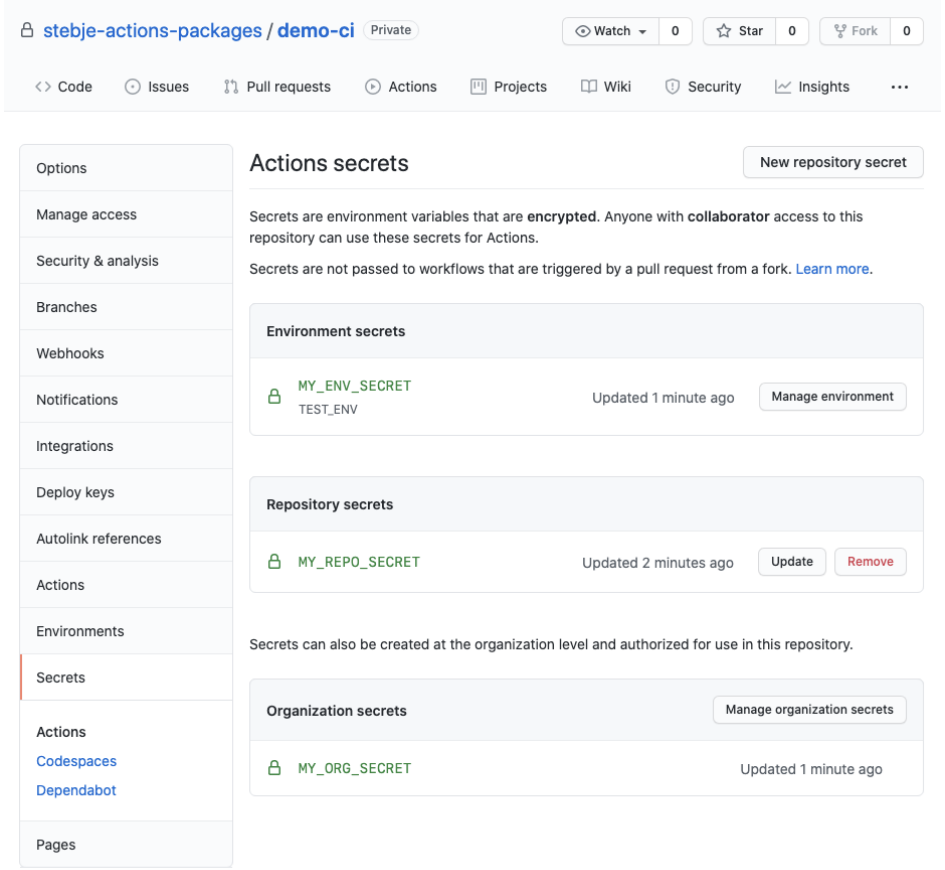
Environments

- 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



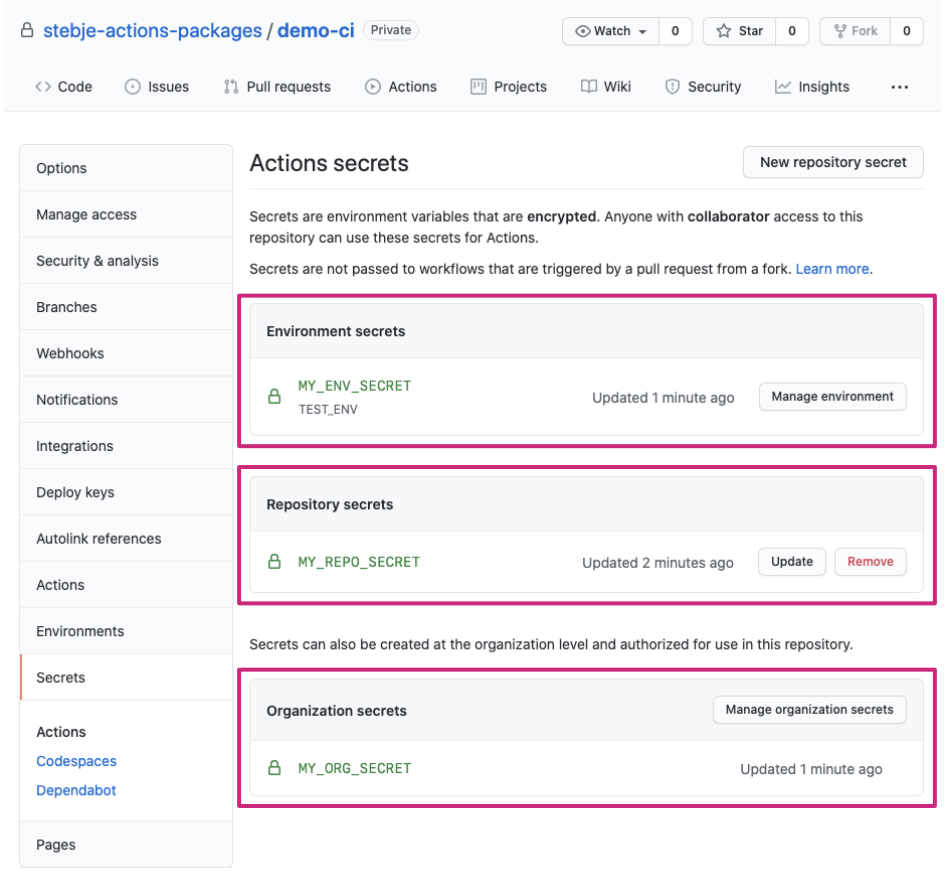
GitHub Secret store

- Built-in secret store
- Encrypted
  - LibSodium sealed box
- Use directly from your workflow
- Redacted in workflow logs
- API support
- Organization / repository / environment secrets



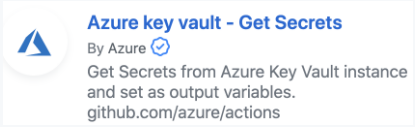
Types of secrets

- Environment secrets
  - Scoped to a single environment
  - The secret is not accessible by workflow unless the deployment to that environment is approved
- Repository secrets
  - Scoped to a single repository
  - Can override org-level secrets
- Organization secrets
  - Managed at org-level
  - Can be scoped to specific repositories



Using secrets in workflows

- All secrets can be accessed using the same syntax; `${{ secrets.<SECRET_NAME> }}`
- Every workflow run provisions a `GITHUB_TOKEN` secret by default
  - Scoped to a single repository
  - Granular permissions
  - Can't trigger other workflows
- Marketplace Actions exist for integration with other secret stores



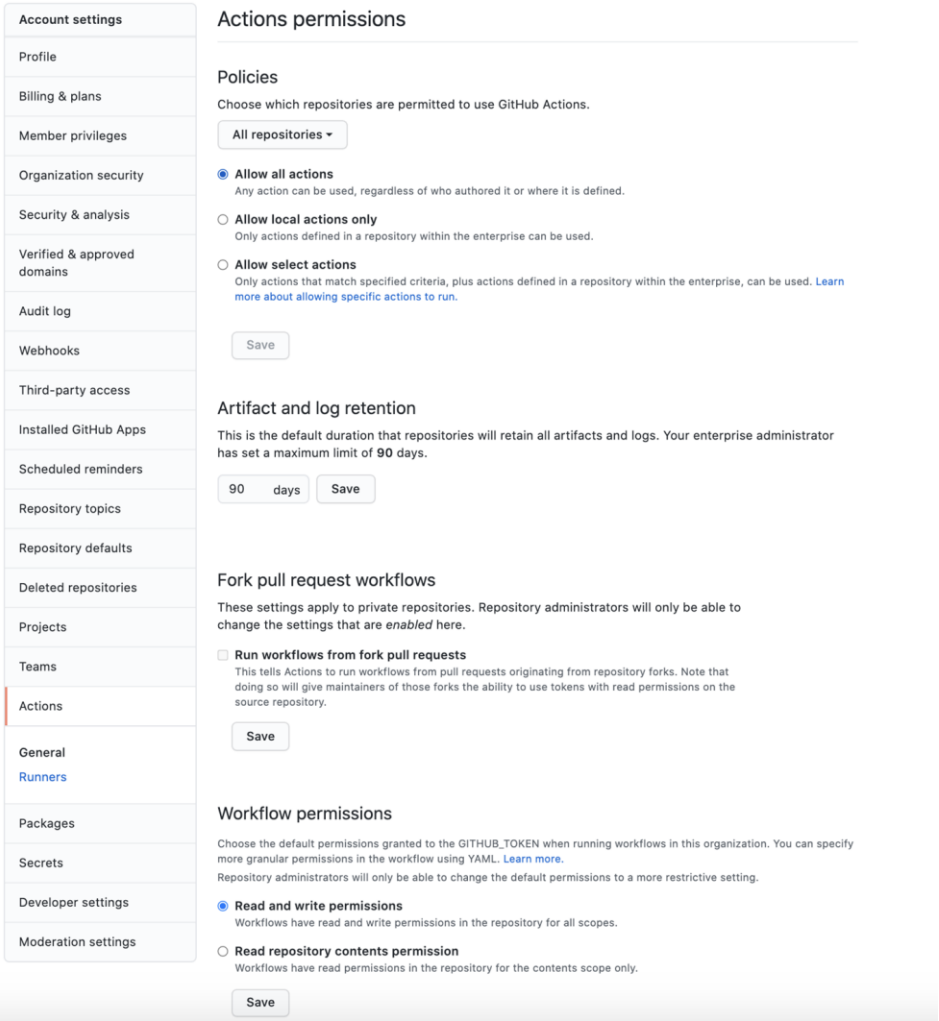

```yaml
name: Pull request labeler

on:
  pull_request:

jobs:
  triage:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/labeler@v2
        with:
          repo-token: ${{ secrets.GITHUB_TOKEN }}

      - uses: myAction@v1
        with:
          mySecret: ${{ secrets.MY_SECRET }}
```

Demo

Managing workflows & Actions

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



Sharing workflows in an organization

- Create GitHub **actions starter templates** in .github

  repository to share workflows

- (Upcoming) Organization workflow execution. Open source concept: https://github.com/SvanBoxel/organization-workflows

Reusable Workflows

- A workflow that can be called by other workflows
- Public repository or same repository
- workflow_call event
- Accepts inputs

```yaml
on:
  workflow_call:
    inputs:
      username:
        required: true
        type: string
    secrets:
      envPAT:
        required: true
```

Sharing private actions

Use GitHub packages and `ghcr.io` to share actions using docker execution and **package registry** permissions

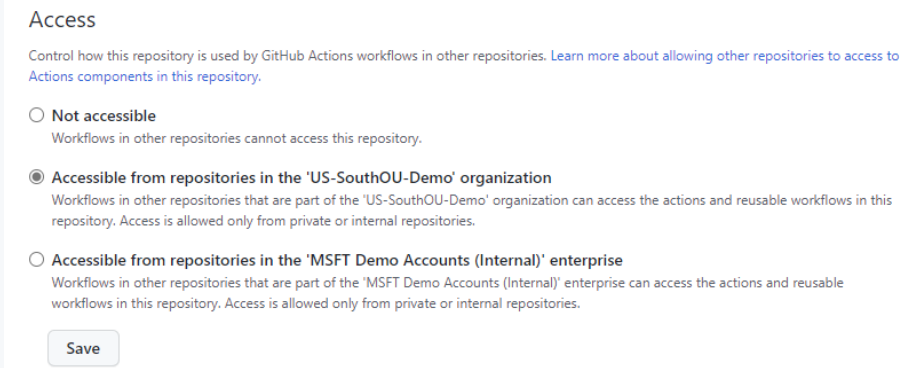
Use a **GitHub App** to clone actions from:

- Actions in different repositories
- Actions monorepo
- Actions separate organization

```yaml
jobs:
  do-something:
    runs-on: ubuntu-latest

    steps:
      - name: Generate app installation token
        id: app
        uses: peter-murray/workflow-application-token-action@v1
        with:
          application_id: ${{ secrets.APP_ID }}
          application_private_key: ${{ secrets.PRIV_KEY }}

      - name: Checkout private repository
        id: checkout_repo
        uses: actions/checkout@v2
        with:
          repository: my-org/repo
          path: path/to/privateAction
          token: ${{ steps.app.outputs.token }}
```

Sharing organization actions

- Enabled for Organization by default
- GHES
  - Internal
  - Private
- Configurable at Repository hosting Action



Caching

Optimizing your workflow performance with caching:

- Temporarily save files between workflow runs
- 5GB max cache size per repo
- 7 days retention
- Scoped to key and branch
- Avoid caching sensitive data
- 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

Best practices on Actions in an organization

- Use the **GITHUB_TOKEN** when possible, as a second option GitHub Apps
- **Limit token permissions**
- Run only **trusted actions**
- Protect your secrets with **environments**
- Create **starter workflows** for reusability

Demo

Building Actions

Writing your own Actions

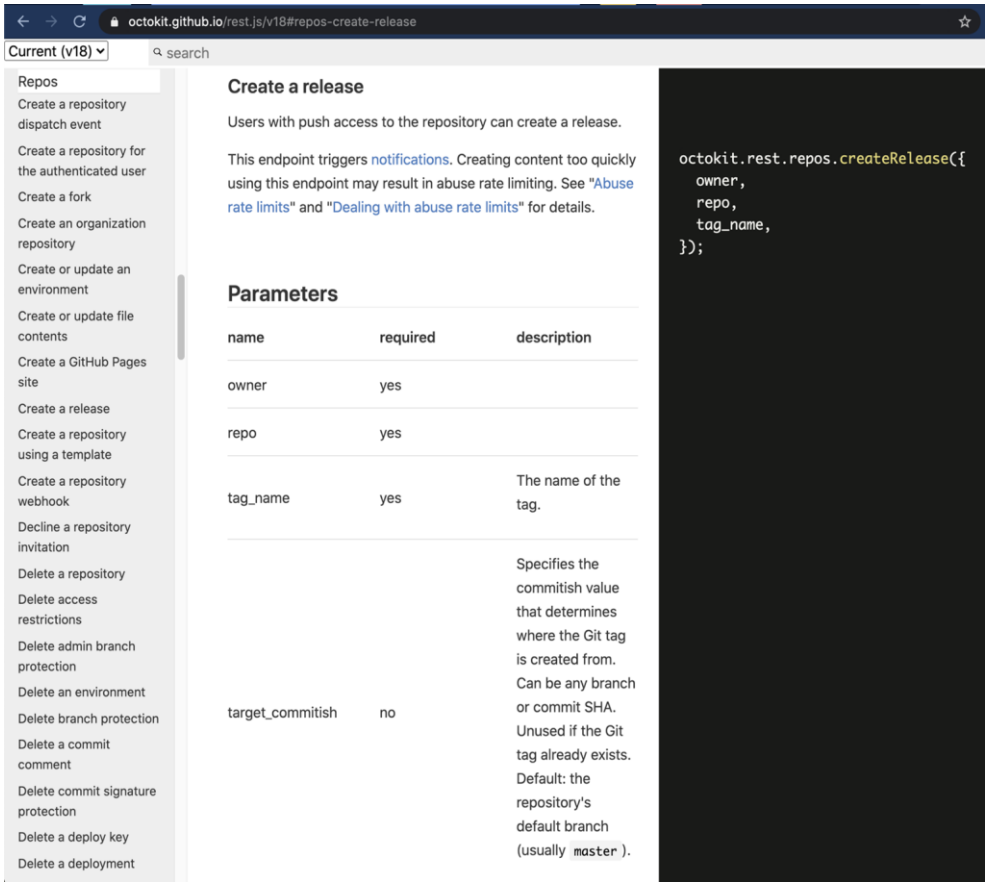
- 3 types of Actions
  - JavaScript
  - Docker
  - Composite run step
- Metadata defined in `action.yml` file
  - Inputs
  - Outputs
  - Branding
  - Pre-/post-scripts
  - ...

`./path/to/action/action.yml`

```yaml
name: "Hello Action"
description: "Greet someone"
author: "octocat@github.com"

inputs:
  MY_NAME:
    description: "Who to greet"
    required: true
    default: "World"

outputs:
  GREETING:
    description: "Full greeting"

runs:
  using: "docker"
  image: "Dockerfile"

branding:
  icon: "mic"
  color: "purple"
```

Using the GitHub API

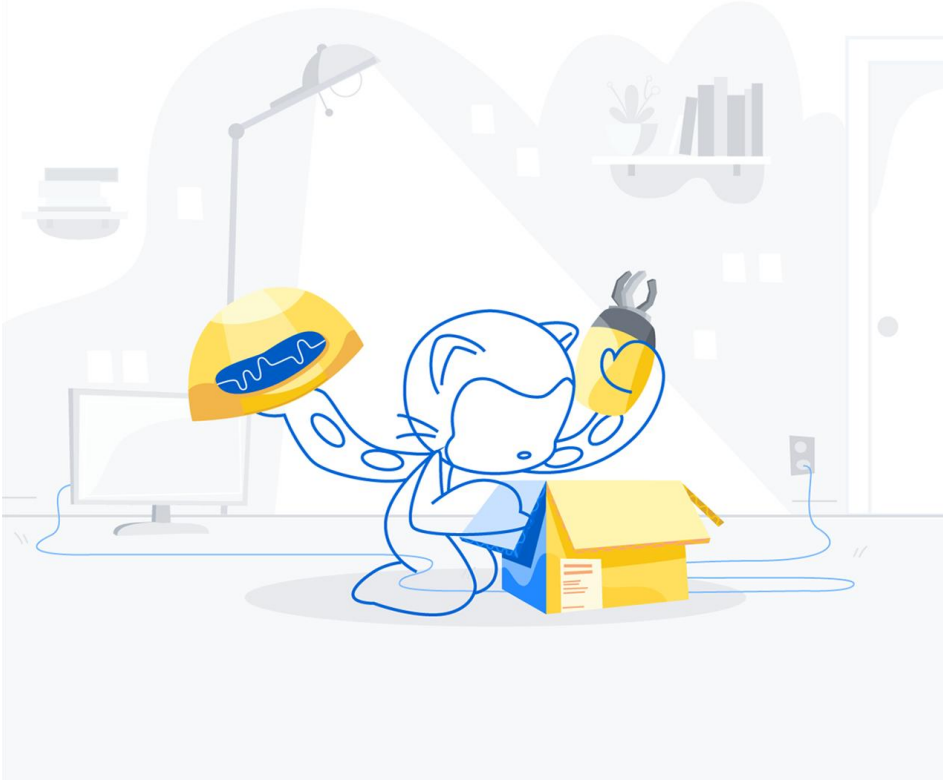
- REST API (v3)
  - Libraries available for most languages
  - Octokit
- GraphQL (v4)
  - The future of the GitHub API
  - A query language allowing granular control of request and response



Writing your own Actions

Best Practices

- Design for reusability
- Write tests
- Versioning
- Documentation
- Proper `action.yml` metadata
- github.com/actions/toolkit
- Publish your Action to the Marketplace 🎉



github.com/actions/github-script

actions / github-script

Watch 15 | Star 1.1k | Fork 123

Code | Issues 7 | Pull requests | Actions | Wiki | Security | Insights

Use this GitHub Action with your project
Add this Action to an existing workflow or create a new one.

View on Marketplace

main | Branches | Tags | Go to file | Add file | Code

joshmgross Merge pull request #137 from actions/joshgross/update-actions... a3e7071 28 days ago 247 commits

.github	Workflow syntax error	last month
.licenses/npm	Update license for @actions/core	28 days ago
.vscode	Check in .vscode	12 months ago
__test__	Add ESLint and Prettier	12 months ago
dist	Update @actions/core to 1.2.7	28 days ago
docs	Update development.md	6 months ago
src	Remove require search fallback	last month
types	Pass nativeRequire, as well	last month
.eslintrc.yml	Add ESLint and Prettier	12 months ago
.gitattributes	Mark licenses as generated	10 months ago
.gitignore	Check in .vscode	12 months ago
.licensed.yml	Add production licenses with licensed	10 months ago
.prettierrc.yml	Add ESLint and Prettier	12 months ago

About

Write workflows scripting the GitHub API in JavaScript

javascript | github-api | actions

Readme

MIT License

Releases 32

Update @actions/core pac... Latest
28 days ago

+ 31 releases

Contributors 28

+ 17 contributors

Languages

TypeScript 100.0%

Demo

Migration

Runners

Runners

GitHub-hosted

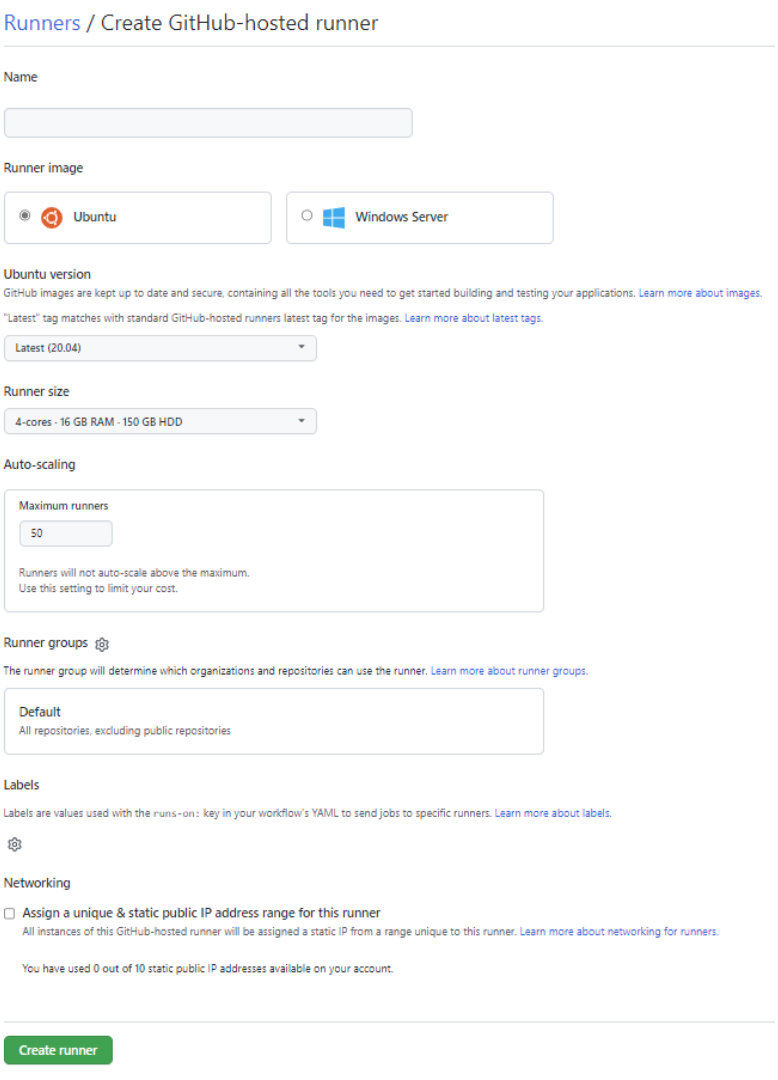
- Receive automatic updates for the operating system, pre-installed packages and tools, and the self-hosted runner application.
- Are managed and maintained by GitHub.
- Provide a clean instance for every job execution.
- Use free minutes on your GitHub plan, with per-minute rates applied after surpassing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

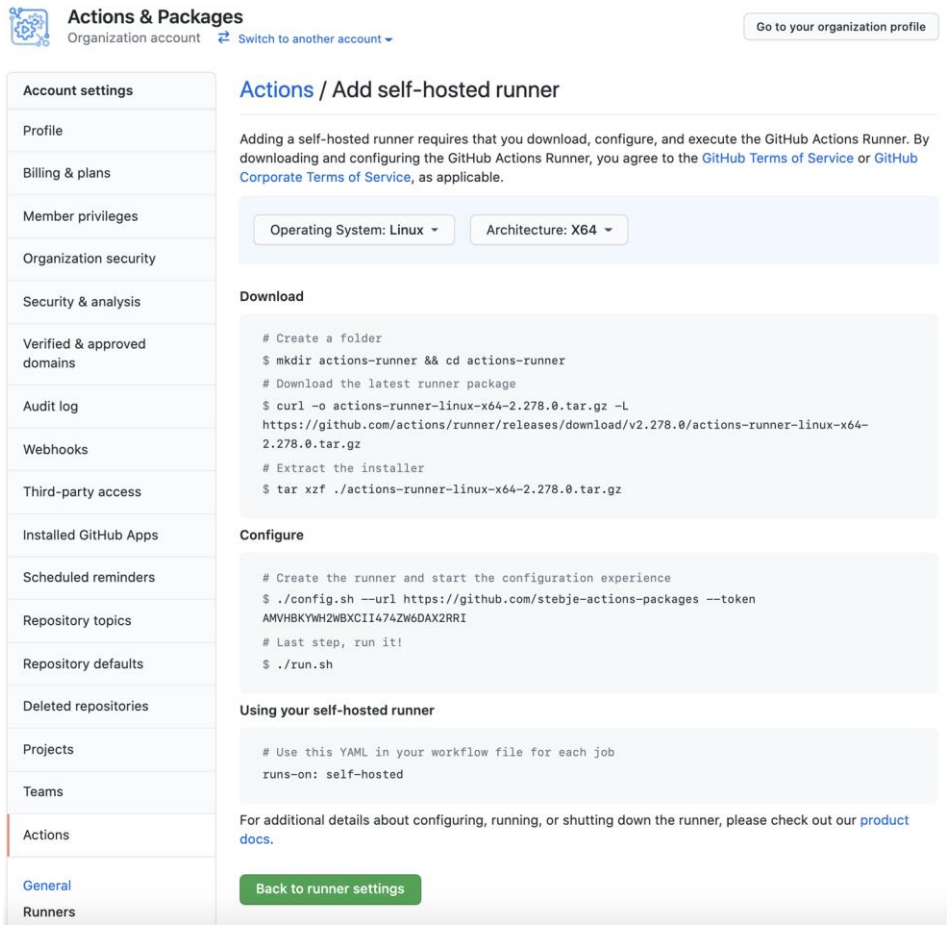
Larger Runners (GitHub Hosted)

- Ubuntu & Linux
- CPU Cores / RAM
- Auto-Scaling
- Runner Groups
- Static IP Addresses
- Public Preview



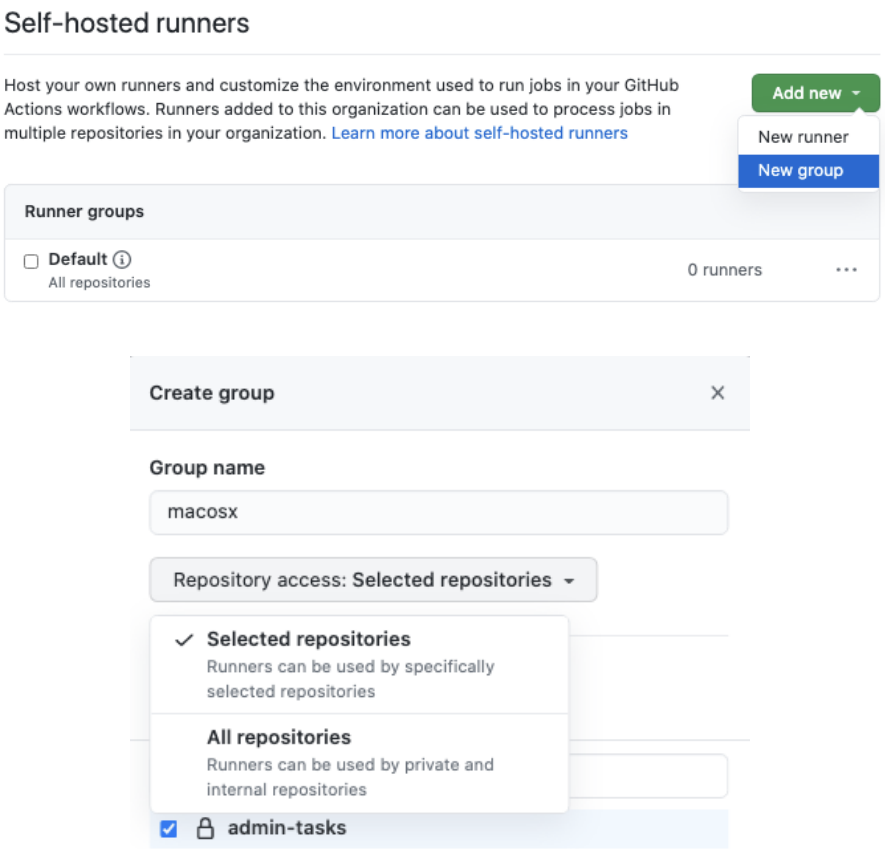
Adding self-hosted runners

- 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)



Runner groups

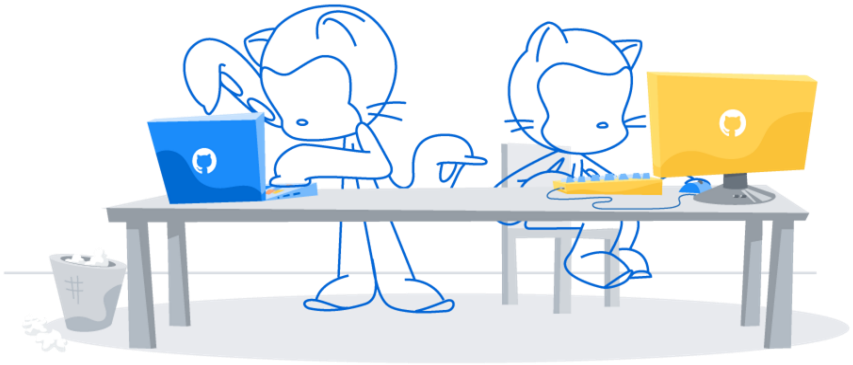
- Can be set up on enterprise and/or organization level
- Can be scoped to specific organizations and/or repositories
- Runners can be moved between groups
- A runner can only be in one group at a time



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

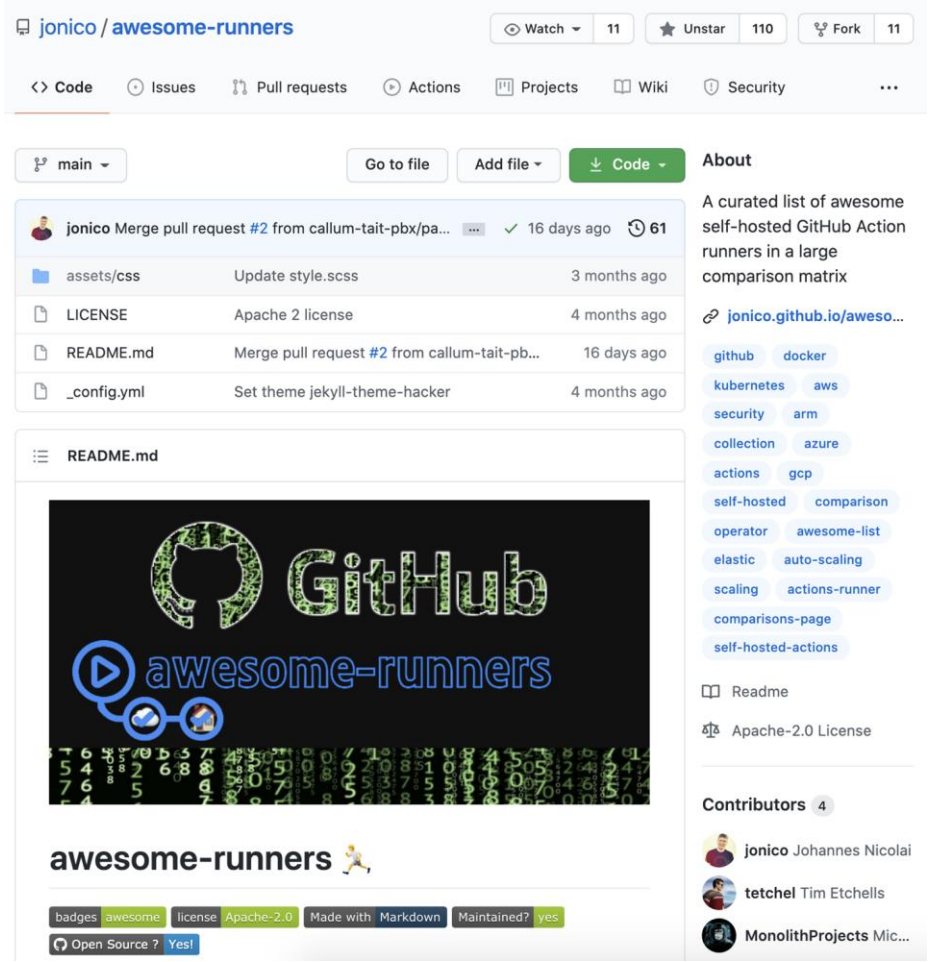


Self-hosted runners and Security

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.

Scaling runners

- Auto-scaling is not yet supported with GitHub-hosted runners
- Open-source solutions do exist for scaling self-hosted runners, e.g.
  - https://github.com/actions-runner-controller/actions-runner-controller
  - https://github.com/philips-labs/terraform-aws-github-runner
- See https://github.com/jonico/awesome-runners for an open source list of options



Demo

CI / CD workflows

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 uploaded separately

```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/
```

Linting

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

```yaml
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 }}
```

Basic CD workflow

- Starter workflows available for most cloud providers
- Store the image in GitHub
- Jobs run on different envs
  - Uses the Docker image
  - Deploys the container image to Azure

```
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
80
```

Demo

Q&A

Thank you