



# GitHub Actions

## Fundamentals

Presented by GitHub Professional Services

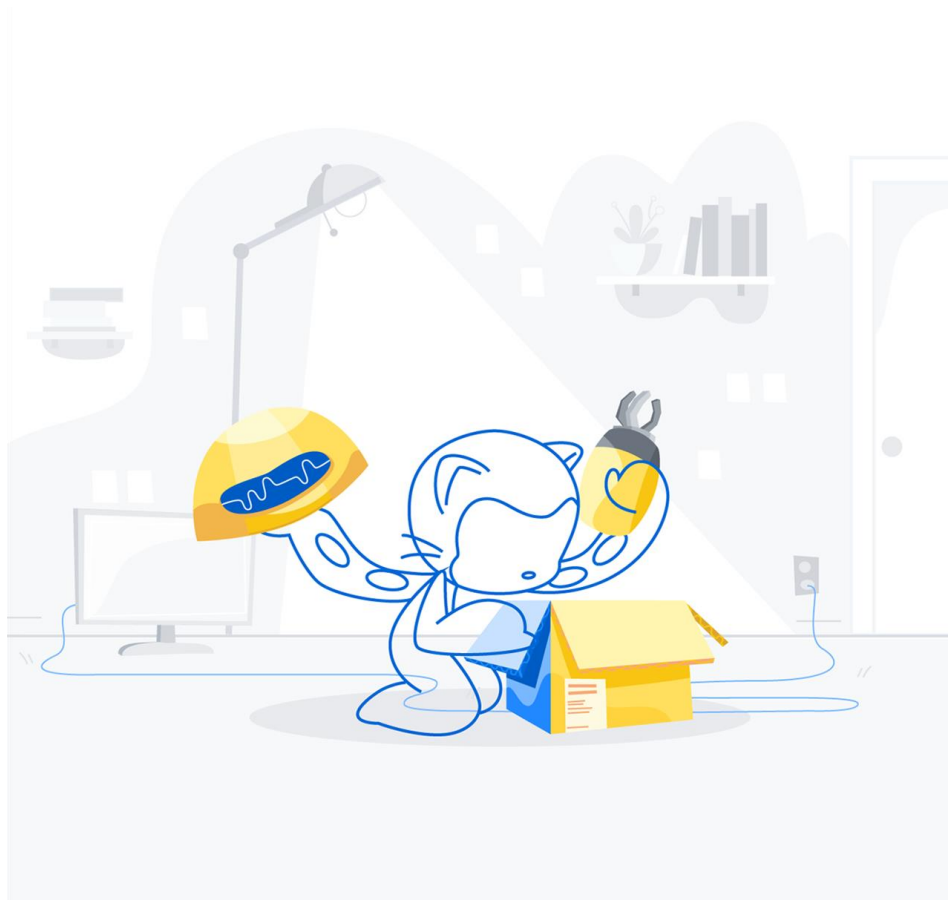
# Objectives



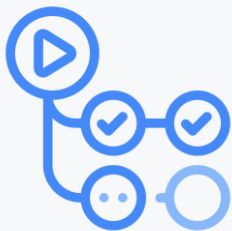
- Understand the **basic components** of GitHub Actions and its **use cases**
- Understand the GitHub Actions **syntax**, both for actions and workflows
- Know how to navigate GitHub Actions **GUI and documentation**
- Understand how to leverage actions written by the **community**
- Create **custom 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

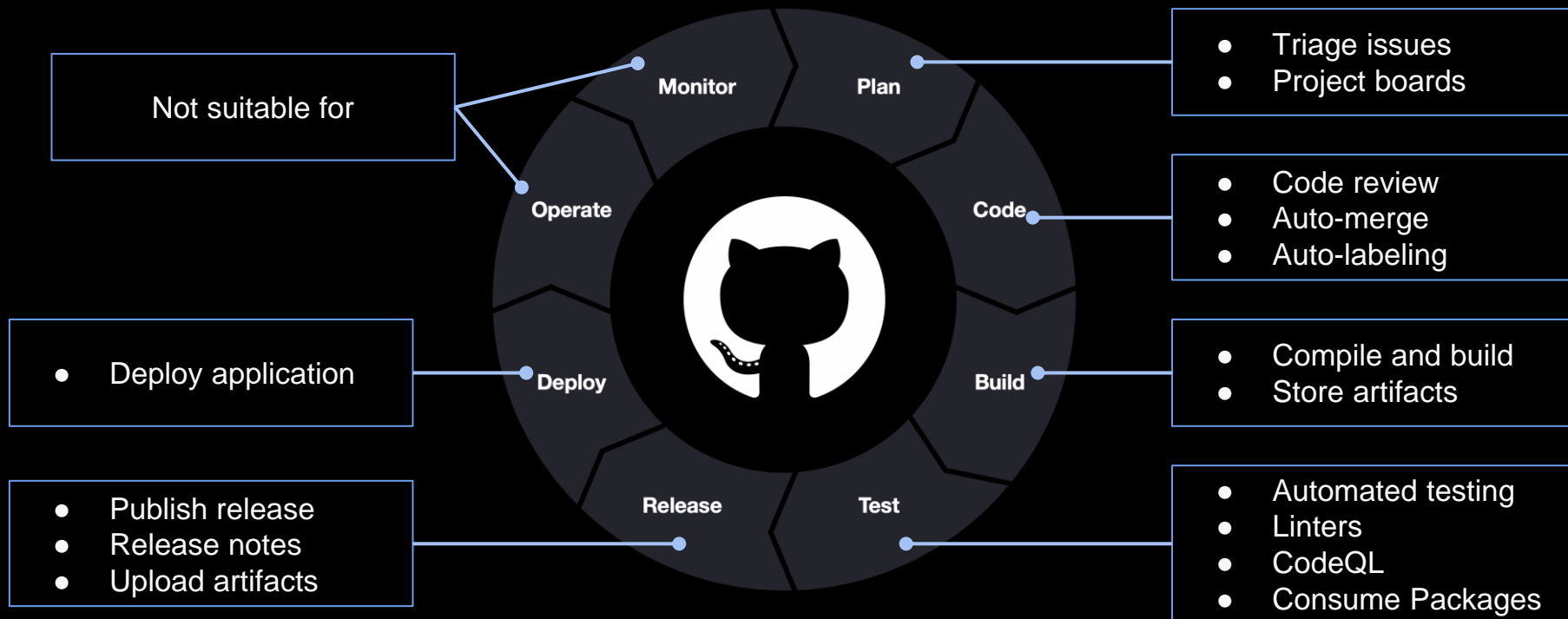


# What is GitHub Actions

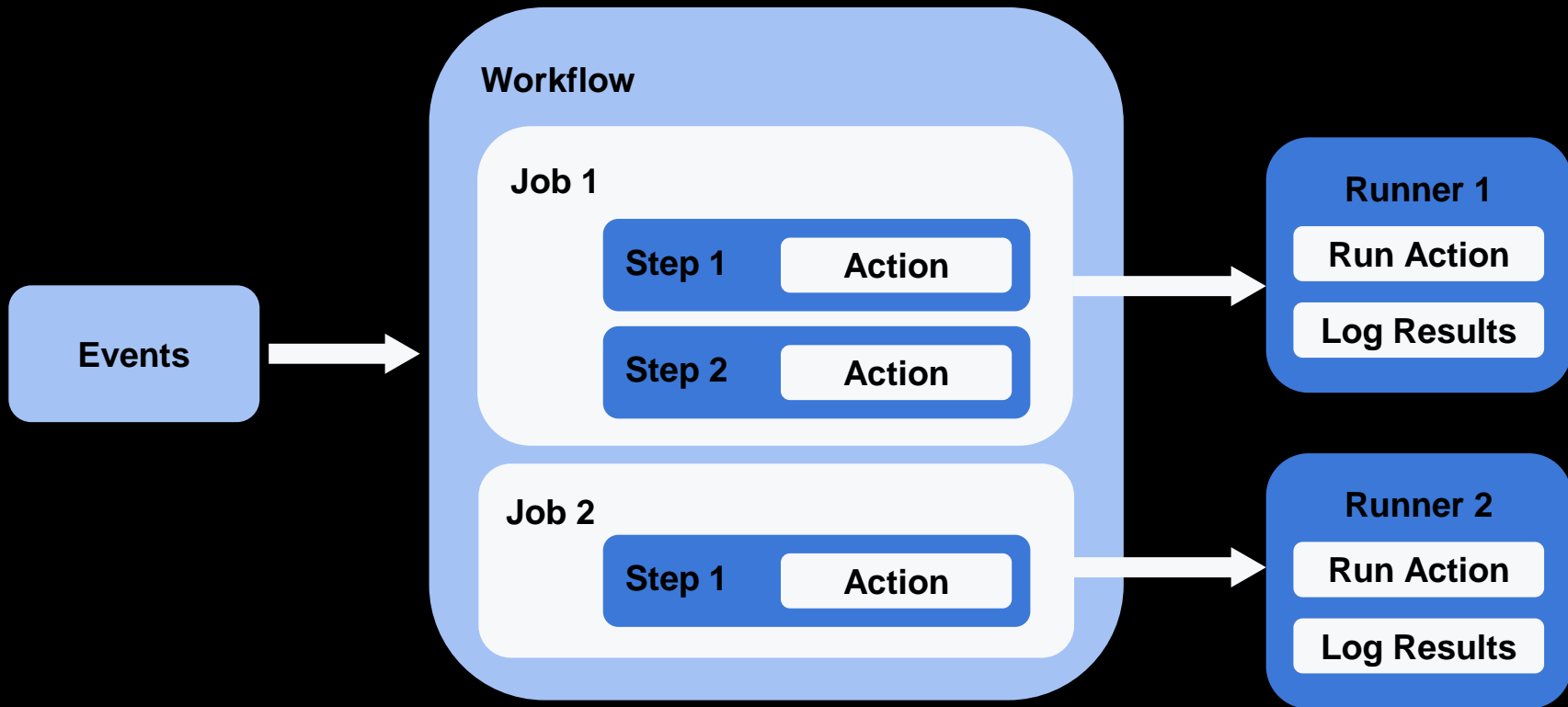
GitHub Actions is a GitHub product that allows you to **automate your workflows**.

- 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



# Key components



# Workflow syntax



# Basic syntax

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

	<code>name: Super Linter workflow</code>
events →	<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>
actions →	<code>- uses: actions/checkout@v2</code>  <code>- uses: github/super-linter@v3</code>
secrets →	<code>  env:</code> <code>    GITHUB_TOKEN: \${ secrets.GITHUB_TOKEN }</code>

# Events

## Webhook events

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

events

## Scheduled events

## Manual events

```
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

events

```
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

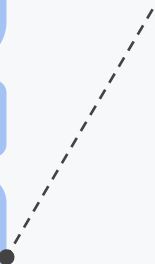
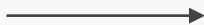
- Pull request
- Issues
- Push
- Release
- ...

## Scheduled events

## Manual events

- workflow\_dispatch
- repository\_dispatch

events



```
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 }
```

Event ▾ Status ▾ Branch ▾ Actor ▾

This workflow has a workflow\_dispatch event trigger.

Run workflow ▾

# Runners



GitHub-hosted runner



Self-hosted runner

runner

```
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



```
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



```
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 →

```
name: Super workflow
```

```
on:
```

```
  push:
```

```
jobs:
```

```
  lint:
```

```
    name: Lint Code Base
```

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

```
    steps:
```

```
      - run: echo "Hello World"
```

public actions →

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

```
      - uses: github/super-linter@v3
```

```
        env:
```

```
          GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
```

local action →

```
      - uses: ./path/to/action
```

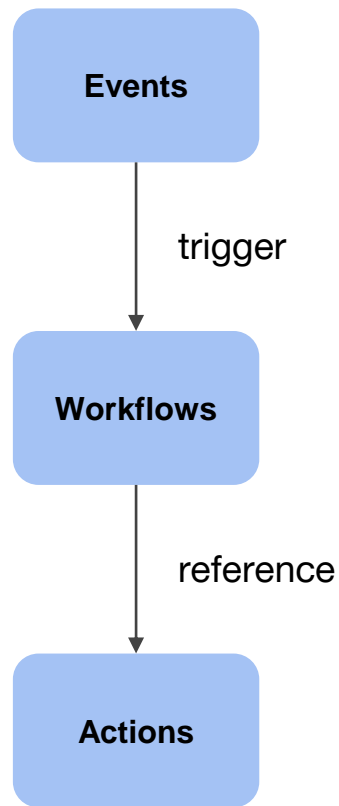
docker image →

```
      - 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

## 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

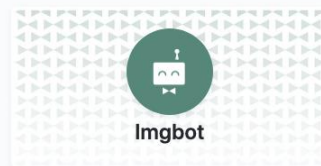
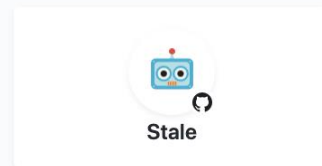
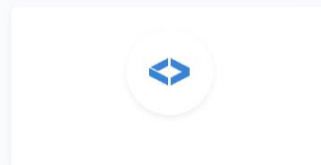
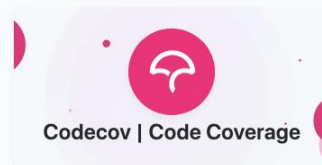
Recently added

Security

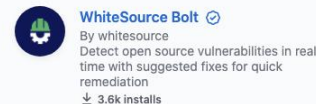
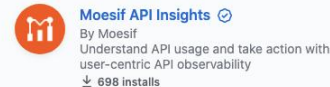
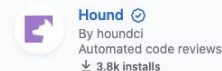
Support

Search for apps and actions

### Recommended for you



### Trending



# 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.

**Deploy Node.js to Azure Web App**  
By Microsoft Azure  
Build a Node.js project and deploy it to an Azure Web App.  
[Set up this workflow](#)

actions/starter-workflows

Deployment

**Deploy to Amazon ECS**  
By Amazon Web Services  
Deploy a container to an Amazon ECS service powered by AWS Fargate or Amazon EC2.  
[Set up this workflow](#)

actions/starter-workflows

Deployment

**Ruby**  
By GitHub Actions  
Build and test a Ruby project with Rake.  
[Set up this workflow](#)

actions/starter-workflows

Ruby

**Python application**  
By GitHub Actions  
Create and test a Python application.  
[Set up this workflow](#)

actions/starter-workflows

Python

**Clojure**  
By GitHub Actions  
Build and test a Clojure project with Leiningen.  
[Set up this workflow](#)

actions/starter-workflows

Clojure

**Node.js**  
By GitHub Actions  
Build and test a Node.js project with npm.  
[Set up this workflow](#)

actions/starter-workflows

JavaScript

**Publish Docker Container**  
By GitHub Actions  
Build, test and push Docker image to GitHub Packages.  
[Set up this workflow](#)

actions/starter-workflows

Dockerfile

**(...)**

# Workflow logs

🔒 [stebje-actions-packages / demo-publish](#) Private

👁 Watch 0 ⭐ Star 0 🍴 Fork 0

<> Code ⓘ Issues 🔗 Pull requests 1 🏠 Actions 📁 Projects 📖 Wiki ⓘ Security 📈 Insights ⚙ Settings

✅ Update README.md Chatops-cmd #8

🔄 Re-run jobs ⋮

🏠 Summary

Jobs

✅ publish-command

## publish-command

succeeded 25 minutes ago in 5s

🔍 Search logs



### Set up job 3s

```
1 Current runner version: '2.278.0'
2 ▶ Operating System
6 ▶ Virtual Environment
11 ▶ GITHUB_TOKEN Permissions
24 Prepare workflow directory
25 Prepare all required actions
26 Getting action download info
27 Download action repository 'peter-evans/slash-command-dispatch@v2'
```

### Slash Command Dispatch 2s

```
1 ▼ Run peter-evans/slash-command-dispatch@v2
2   with:
3     token: ***
4     commands: publish
5
6     reactions: true
7     issue-type: pull-request
8     permission: maintain
9     reaction-token: ***
10    allow-edits: false
11    repository: stebje-actions-packages/demo-publish
12    event-type-suffix: -command
13    dispatch-type: repository
14 Using configuration from yaml inputs.
15 Command 'publish' to be dispatched.
16 Command 'publish' dispatched to 'stebje-actions-packages/demo-publish' with event type 'publish-command'.
```

# Advanced syntax

Syntax element	Description
<code>permissions</code>	Set workflow permissions for <code>GITHUB_TOKEN</code>
<code>env</code>	Set environment variables 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 jobs dependent of each other. Share outputs
<code>if</code>	Check whether a job should run based on variables. <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>services</code>	Create sidecar docker images for integration dependencies
<code>container</code>	Use a container for the steps execution

# Function expressions

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

# Usage limits

Limit	Description	Notes
<b>Concurrent Jobs</b> (Based on Enterprise Plan)	180 <b>GHEC</b>  32 core: 1,500 concurrent jobs 64 core: 1,900 concurrent jobs <b>GHEC</b>	50 maximum concurrent macOS jobs
<b>Job Execution</b>	6 hours	On error, jobs terminated and seen as failed This limit <b>doesn't</b> apply to self-hosted runners
<b>Workflow Run</b>	72 hours	On error, workflow cancelled when reached This limit also applies to self-hosted runners
<b>Job Queue</b>	24 hours	Jobs terminated when the limit is reached This limit <b>only</b> applies to self-hosted runners
<b>API Requests</b>	1000/hour per repo	On error, API calls fail. Can cause jobs to fail This limit also applies to self-hosted runners
<b>Job Matrix</b>	256 per run	This limit also applies to self-hosted runners

*Subject to change*



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

## 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

# 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

The screenshot shows the GitHub interface for the repository 'stebje-actions-packages / demo-ci'. The 'Secrets' tab is selected in the left sidebar. The main content area is titled 'Actions secrets' and includes a 'New repository secret' button. It explains that secrets are encrypted environment variables and are not passed to workflows triggered by pull requests from forks. The page is divided into three sections: 'Environment secrets' (showing 'MY\_ENV\_SECRET' with a 'Manage environment' button), 'Repository secrets' (showing 'MY\_REPO\_SECRET' with 'Update' and 'Remove' buttons), and 'Organization secrets' (showing 'MY\_ORG\_SECRET' with a 'Manage organization secrets' button). A note at the bottom states that secrets can also be created at the organization level.

stebje-actions-packages / demo-ci Private

Watch 0 Star 0 Fork 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights ...

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Autolink references

Actions

Environments

Secrets

Actions

Codespaces

Dependabot

Pages

## Actions secrets

New repository secret

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more](#).

### Environment secrets

MY\_ENV\_SECRET TEST\_ENV Updated 1 minute ago Manage environment

### Repository secrets

MY\_REPO\_SECRET Updated 2 minutes ago Update Remove

Secrets can also be created at the organization level and authorized for use in this repository.

### Organization secrets

MY\_ORG\_SECRET Updated 1 minute ago Manage organization 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

The screenshot shows the GitHub Actions secrets page for the repository 'demo-ci'. The page is divided into three main sections: Environment secrets, Repository secrets, and Organization secrets, each highlighted with a pink border. The left sidebar contains a list of navigation options: Options, Manage access, Security & analysis, Branches, Webhooks, Notifications, Integrations, Deploy keys, Autolink references, Actions, Environments, Secrets (highlighted), Actions, Codespaces, Dependabot, and Pages. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. The top right corner shows repository statistics: Watch (0), Star (0), and Fork (0). The 'New repository secret' button is located in the top right corner of the 'Actions secrets' section. The 'Environment secrets' section shows a secret named 'MY\_ENV\_SECRET' with the value 'TEST\_ENV', updated 1 minute ago, and a 'Manage environment' button. The 'Repository secrets' section shows a secret named 'MY\_REPO\_SECRET', updated 2 minutes ago, with 'Update' and 'Remove' buttons. The 'Organization secrets' section shows a secret named 'MY\_ORG\_SECRET', updated 1 minute ago, and a 'Manage organization secrets' button. The 'Secrets' section also includes a description: 'Secrets are environment variables that are encrypted. Anyone with collaborator access to this repository can use these secrets for Actions.' and 'Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more.](#)'

stebje-actions-packages / demo-ci Private

Watch 0 Star 0 Fork 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights ...

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Autolink references

Actions

Environments

Secrets

Actions

Codespaces

Dependabot

Pages

### Actions secrets

New repository secret

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more.](#)

#### Environment secrets

MY\_ENV\_SECRET  
TEST\_ENV Updated 1 minute ago Manage environment

#### Repository secrets

MY\_REPO\_SECRET Updated 2 minutes ago Update Remove

Secrets can also be created at the organization level and authorized for use in this repository.

#### Organization secrets

MY\_ORG\_SECRET Updated 1 minute ago Manage organization secrets

# 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
  - Enterprise/organization/repository policies for default permissions
  - `permissions` syntax for granular permissions on workflow- or job-level
  - Can't trigger other workflows
- Marketplace Actions exist for integration with other secret stores

```
name: Pull request labeler
```

```
on:
```

```
  pull_request:
```

```
jobs:
```

```
  triage:
```

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

```
  permissions:
```

```
    contents: read
```

```
    actions: read
```

```
    issues: write
```

```
steps:
```

```
- uses: actions/labeler@v2
```

```
  with:
```

```
    repo-token: ${{ secrets.GITHUB_TOKEN }}
```

```
- uses: myAction@v1
```

```
  with:
```

```
    mySecret: ${{ secrets.MY_SECRET }}
```



## Vault Secrets

By hashicorp ✓

A Github Action that allows you to consume HashiCorp Vault™ secrets as secure environment variables



## Azure key vault - Get Secrets

By Azure ✓

Get Secrets from Azure Key Vault instance and set as output variables.  
[github.com/azure/actions](https://github.com/azure/actions)

# Permissions for GITHUB\_TOKEN

## Workflow permissions

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

☐ Read and write permissions

Workflows have read and write permissions in the repository for all scopes.

☒ Read repository contents permission

Workflows have read permissions in the repository for the contents scope only.

Save

- Default permissions can be set at the org level
- `permissions` key supported at the workflow and job level allows you to specify which permissions you want for the token
- Any permission that is absent from the list will be set to `none`.
- Pull requests from public forks are still considered a special case and will receive a read token regardless of these settings.

`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`



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`

Account settings

Profile

Billing & plans

Member privileges

Organization security

Security & analysis

Verified & approved domains

Audit log

Webhooks

Third-party access

Installed GitHub Apps

Scheduled reminders

Repository topics

Repository defaults

Deleted repositories

Projects

Teams

Actions

General

Runners

Packages

Secrets

Developer settings

Moderation settings

## Actions permissions

### Policies

Choose which repositories are permitted to use GitHub Actions.

All repositories ▾

☒ **Allow all actions**

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

☐ **Allow local actions only**

Only actions defined in a repository within the enterprise can be used.

☐ **Allow select actions**

Only actions that match specified criteria, plus actions defined in a repository within the enterprise, can be used. [Learn more about allowing specific actions to run.](#)

Save

### Artifact and log retention

This is the default duration that repositories will retain all artifacts and logs. Your enterprise administrator has set a maximum limit of 90 days.

90 days Save

### Fork pull request workflows

These settings apply to private repositories. Repository administrators will only be able to change the settings that are *enabled* here.

☐ **Run workflows from fork pull requests**

This tells Actions to run workflows from pull requests originating from repository forks. Note that doing so will give maintainers of those forks the ability to use tokens with read permissions on the source repository.

Save

### Workflow permissions

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

Repository administrators will only be able to change the default permissions to a more restrictive setting.

☒ **Read and write permissions**

Workflows have read and write permissions in the repository for all scopes.

☐ **Read repository contents permission**

Workflows have read permissions in the repository for the contents scope only.

Save

# 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>
- **(Beta)** Reusable workflows in **GitHub Enterprise Cloud**



# Sharing actions

Actions stored in **public** repositories can be accessed by anyone

Sharing actions stored in **private** repositories:

- Use GitHub packages and `ghcr.io` to share actions
- Use a **GitHub App** to authenticate during the workflow when checking out the private action

Sharing actions stored in **internal** repositories:

- (**Beta**) Allow sharing via Actions policies

```
# Use an action stored in a private repository
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 }
```

# 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
- Never cache sensitive data
- Only works on GitHub Hosted Runners



## [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
- Use actions for CI/CD but also **\*-ops**



Demo

# Building Actions

# Writing your own Actions

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

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

```
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

The screenshot shows the Octokit GitHub API documentation for the 'Create a release' endpoint. The browser address bar shows 'octokit.github.io/rest.js/v18#repos-create-release'. The left sidebar lists various API endpoints under the 'Repos' category, with 'Create a release' selected. The main content area has a title 'Create a release' and a description: 'Users with push access to the repository can create a release.' It also includes a warning about rate limiting. Below this is a 'Parameters' table with columns 'name', 'required', and 'description'. The table lists parameters: 'owner' (required), 'repo' (required), 'tag\_name' (required), and 'target\_commitish' (not required). To the right of the table is a code block showing the JavaScript usage of the 'createRelease' method.

Current (v18) search

**Create a release**

Users with push access to the repository can create a release.

This endpoint triggers [notifications](#). Creating content too quickly using this endpoint may result in abuse rate limiting. See "[Abuse rate limits](#)" and "[Dealing with abuse rate limits](#)" for details.

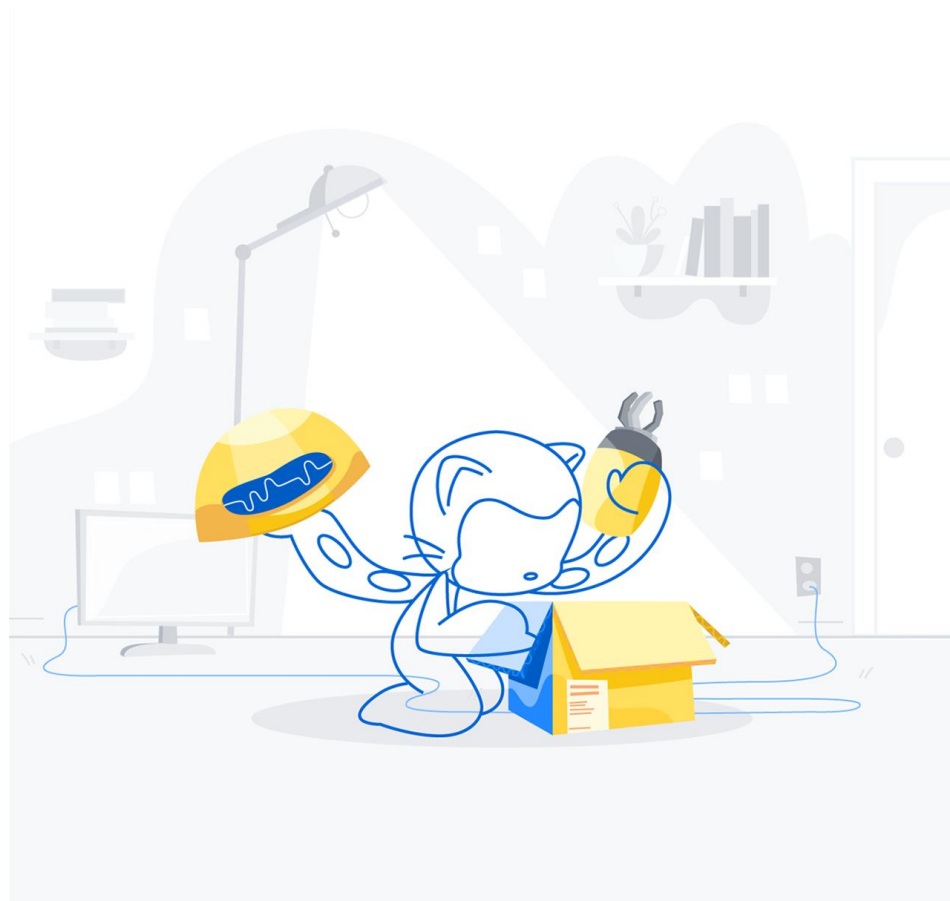
name	required	description
owner	yes	
repo	yes	
tag_name	yes	The name of the tag.
target_commitish	no	Specifies the commitish value that determines where the Git tag is created from. Can be any branch or commit SHA. Unused if the Git tag already exists. Default: the repository's default branch (usually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

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

# Writing your own Actions

## Best Practices

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





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

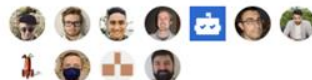
MIT License

## Releases 32

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

+ 31 releases

## Contributors 28



+ 17 contributors

## Languages

TypeScript 100.0%



Demo

# 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](https://github.com/actions)
  - npm run by shell
  - Artifact uploaded separately

```
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

```
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: mattedavis0351/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

# 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.

# 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)



## Actions & Packages

Organization account

[Switch to another account](#)

[Go to your organization profile](#)

Account settings

Profile

Billing & plans

Member privileges

Organization security

Security & analysis

Verified & approved domains

Audit log

Webhooks

Third-party access

Installed GitHub Apps

Scheduled reminders

Repository topics

Repository defaults

Deleted repositories

Projects

Teams

Actions

General

Runners

## Actions / Add 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.

Operating System: Linux

Architecture: X64

### Download

```
# Create a folder
$ mkdir actions-runner && cd actions-runner
# Download the latest runner package
$ curl -o actions-runner-linux-x64-2.278.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.278.0/actions-runner-linux-x64-2.278.0.tar.gz
# Extract the installer
$ tar xzf ./actions-runner-linux-x64-2.278.0.tar.gz
```

### Configure

```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/stebye-actions-packages --token
AMVHBKYWH2WBXCII474ZW6DAX2RRI
# 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
```

For additional details about configuring, running, or shutting down the runner, please check out our [product docs](#).

[Back to runner settings](#)

# 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

## Self-hosted runners

Host your own runners and customize the environment used to run jobs in your GitHub Actions workflows. Runners added to this organization can be used to process jobs in multiple repositories in your organization. [Learn more about self-hosted runners](#)

Add new

New runner

New group

### Runner groups

☐ Default ⓘ

All repositories

0 runners

...

### Create group

×

#### Group name

macosx


Repository access: Selected repositories ▾

#### ✓ Selected repositories

Runners can be used by specifically selected repositories

#### All repositories

Runners can be used by private and internal repositories

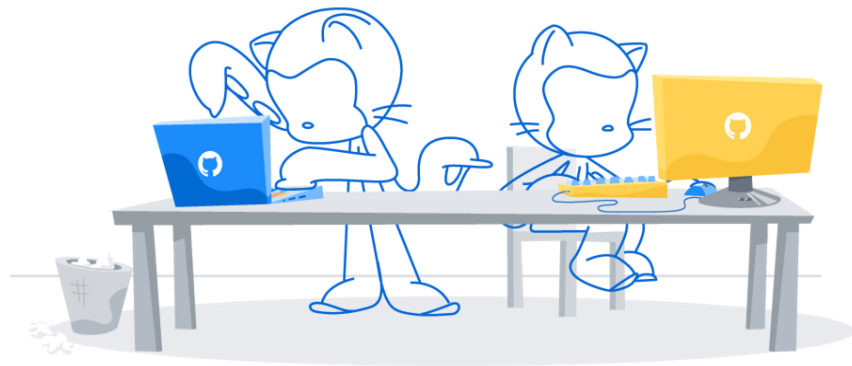
☒  admin-tasks

# 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

The screenshot shows the GitHub repository page for `jonico/awesome-runners`. At the top, the repository name is displayed with navigation links for Watch (11), Unstar (110), and Fork (11). Below this are tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, and Security. The main content area shows a merge pull request #2 from `callum-tait-pbx/pa...` merged 16 days ago. A table lists recent updates: `assets/css` (Update style.scss, 3 months ago), `LICENSE` (Apache 2 license, 4 months ago), `README.md` (Merge pull request #2 from callum-tait-pb..., 16 days ago), and `_config.yml` (Set theme jekyll-theme-hacker, 4 months ago). Below the table is a section for the `README.md` file, featuring a large graphic with the GitHub logo, the text "awesome-runners", and a background of binary code. Under the graphic, the repository name "awesome-runners" is shown with a runner icon. At the bottom of the repository page, there are badges for "awesome", "license Apache-2.0", "Made with Markdown", "Maintained? yes", and "Open Source? Yes!". On the right side, the "About" section describes the repository as "A curated list of awesome self-hosted GitHub Action runners in a large comparison matrix" and provides a link to `jonico.github.io/aweso...`. Below this is a list of tags including `github`, `docker`, `kubernetes`, `aws`, `security`, `arm`, `collection`, `azure`, `actions`, `gcp`, `self-hosted`, `comparison`, `operator`, `awesome-list`, `elastic`, `auto-scaling`, `scaling`, `actions-runner`, `comparisons-page`, and `self-hosted-actions`. Further down, it shows "Readme" and "Apache-2.0 License". At the bottom right, the "Contributors" section lists 4 contributors: `jonico` (Johannes Nicolai), `tetchel` (Tim Etchells), and `MonolithProjects` (Mic...).



Demo





**Q&A**



**Thank you**