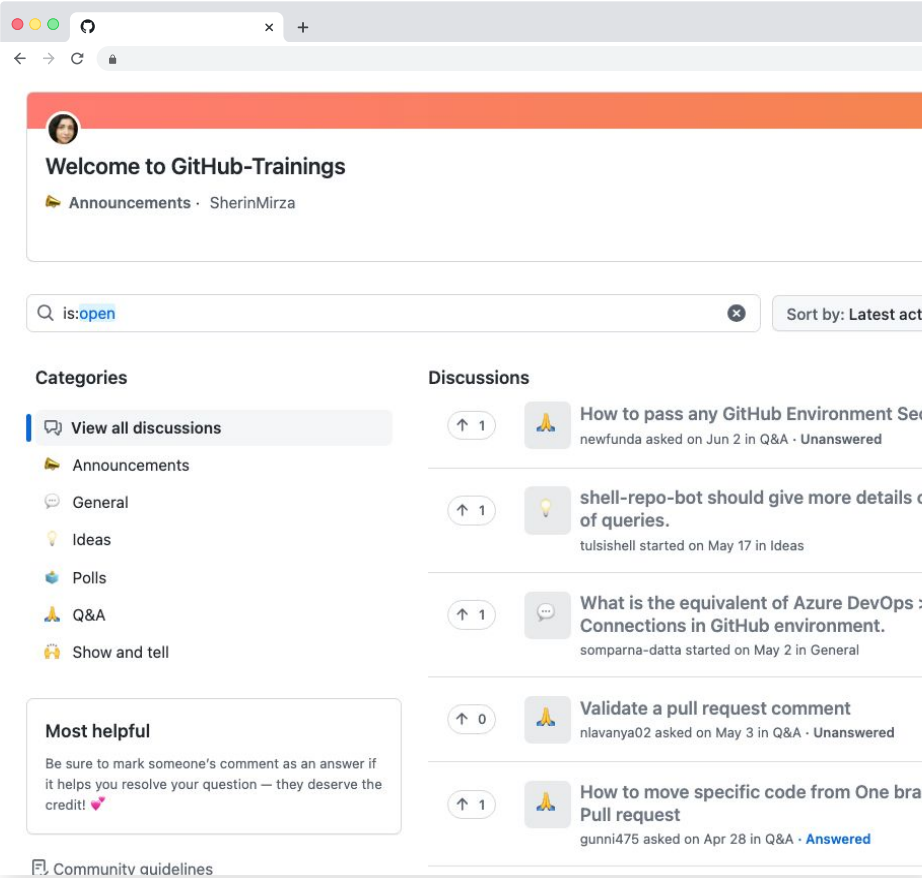
GitHub Actions Training

Stefan Stölzle
Staff Services Delivery Engineer
stefan@github.com

Shell GitHub Trainings

Discussions | sede-x/GitHub-Trainings



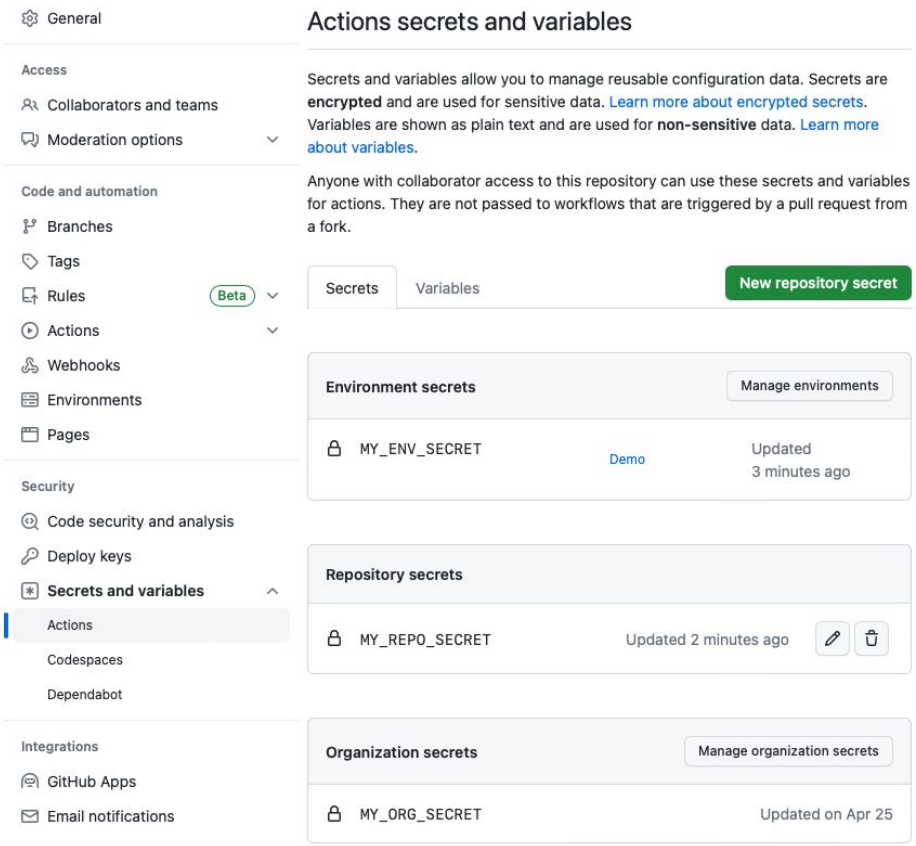
AGENDA

- Overview
- Secrets & variables
- Reusable Workflows
- Composite Actions
- Q&A

Secrets & variables

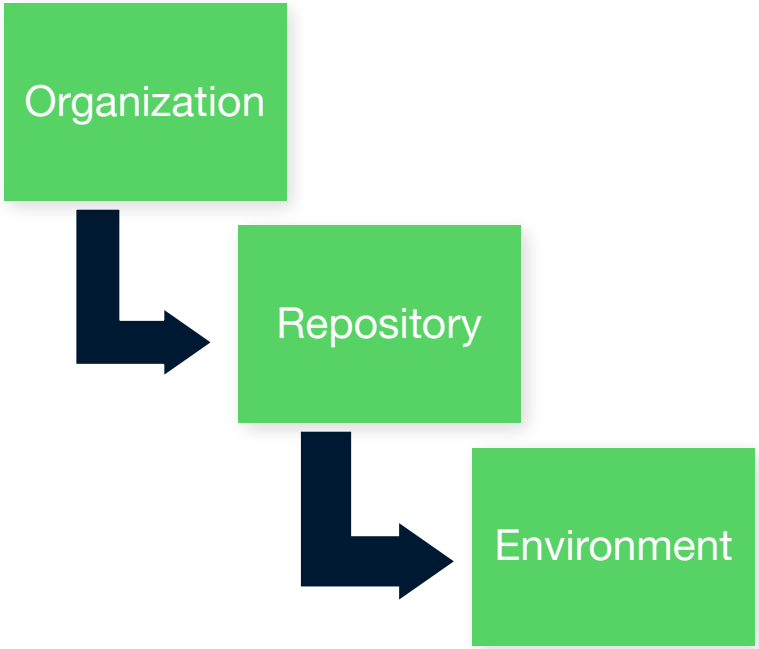
GitHub Secret store

- 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
- permissions - GitHub Enterprise Cloud Docs

```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
```

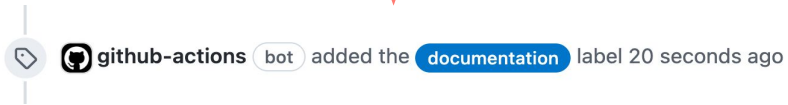
The GITHUB_TOKEN

Perform actions as “github-actions”:

```yaml
permissions:
  contents: read
  issues: write
```

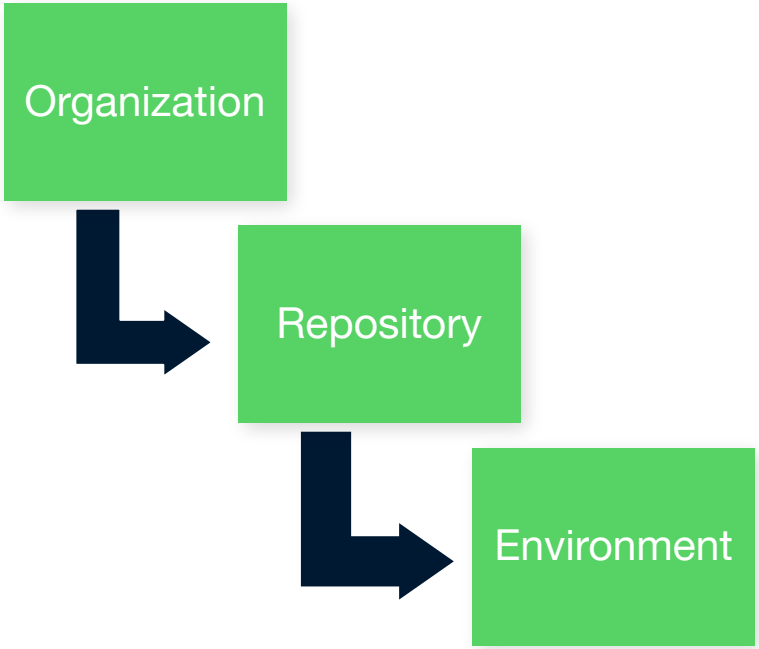
```yaml
label_issues:
  runs-on: ubuntu-latest
  if: github.event_name == 'issues'

  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



Reusable Workflows

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

```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

octokit.github.io/rest.js/v18#repos-create-release

Current (v18) search

Repos
- Create a repository dispatch event
- Create a repository for the authenticated user
- Create a fork
- Create an organization repository
- Create or update an environment
- Create or update file contents
- Create a GitHub Pages site
- Create a release
- Create a repository using a template
- Create a repository webhook
- Decline a repository invitation
- Delete a repository
- Delete access restrictions
- Delete admin branch protection
- Delete an environment
- Delete branch protection
- Delete a commit comment
- Delete commit signature protection
- Delete a deploy key
- Delete a deployment

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.

Parameters

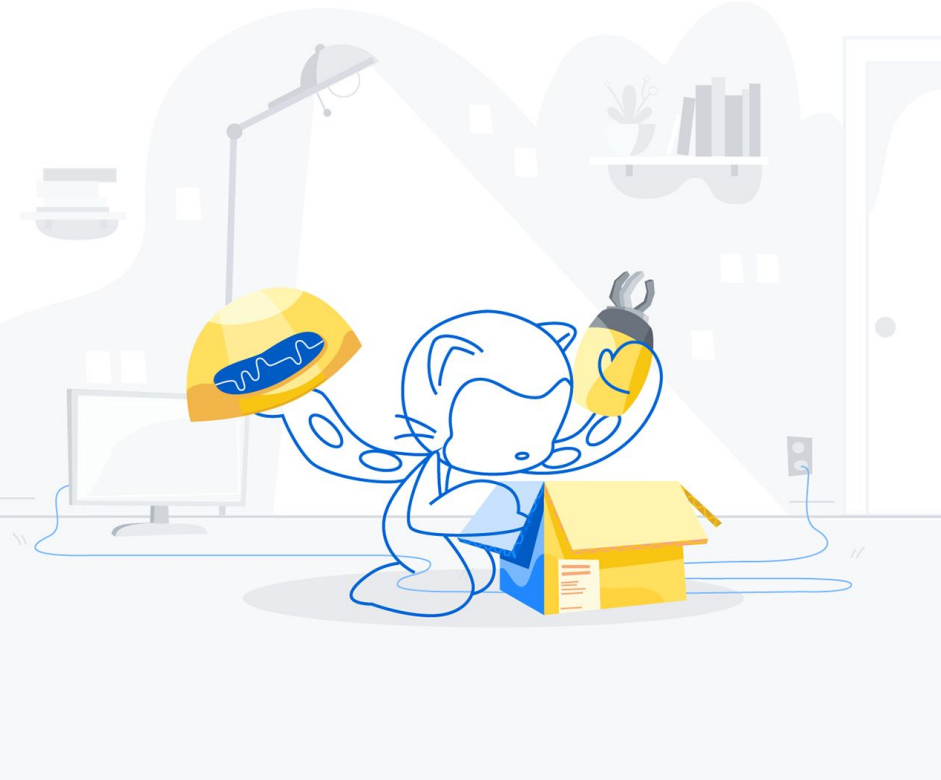
name	required	description
owner	yes	
repo	yes	
tag_name	yes	The name of the tag.
target_commitish	no	Specifies the commitish value that determines where the Git tag is created from. Can be any branch or commit SHA. Unused if the Git tag already exists. Default: the repository's default branch (usually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

```
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
- Publish your Action to the Marketplace



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 }}
```

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

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**

Composite Actions

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 }}
```

Q&A

Resources

- Using secrets in GitHub Actions - GitHub Enterprise Cloud Docs
- Variables - GitHub Enterprise Cloud Docs
- Reusing workflows - GitHub Enterprise Cloud Docs
- Creating a composite action - GitHub Enterprise Cloud Docs

- stoe/policies: Shared policies and workflows (github.com)
- stoe-actions-playground/read-json-action-demo: Composite Action that parses a JSON file and prints it to stdout (github.com)
- stoe-actions-playground/use-read-json-action-demo: Demo using stoe-demo/read-json-action-demo (github.com)

Thank you