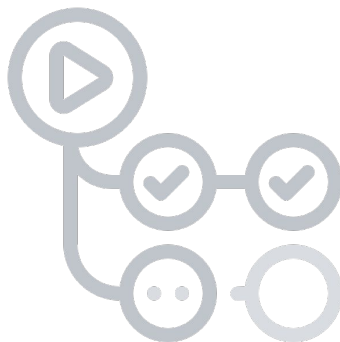




**Stefan Stölzle**

Staff Services Delivery Engineer

[stefan@github.com](mailto:stefan@github.com)

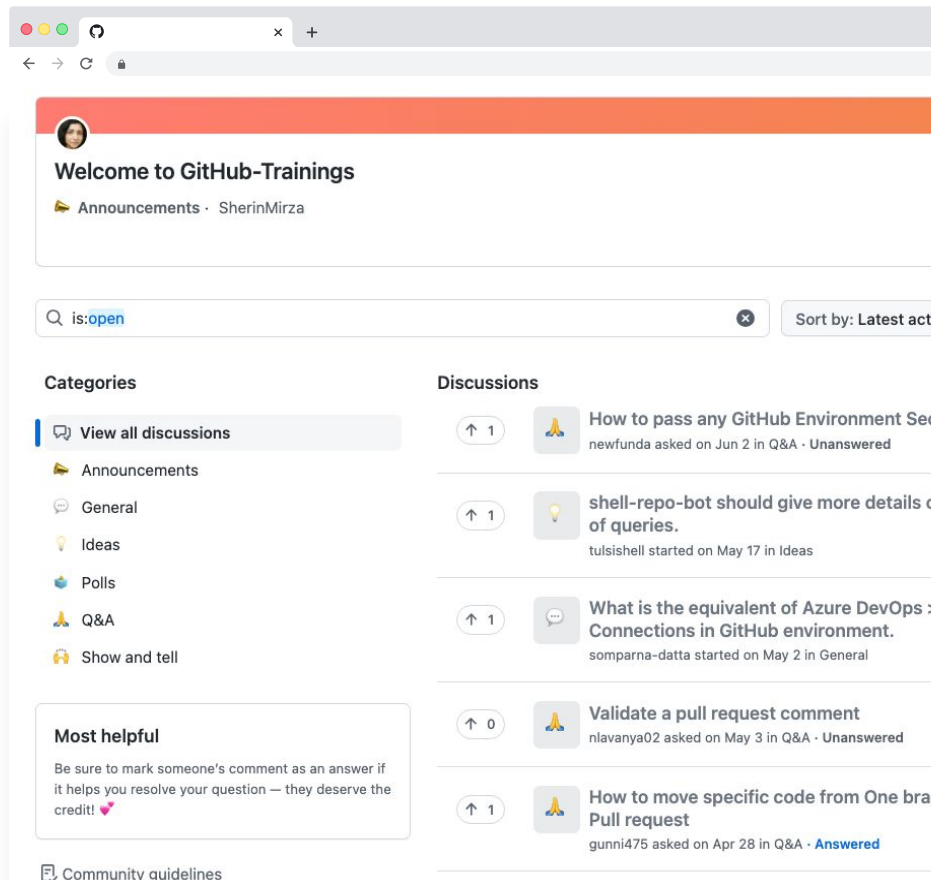


# GitHub Actions Training



# Shell GitHub Trainings

[Discussions](#) | [sede-x/GitHub-Trainings](#)



The screenshot shows the GitHub Discussions interface for the repository 'sede-x/GitHub-Trainings'. At the top, there's a header with the repository name and a welcome message from SherinMirza. Below this is a search bar with the text 'is:open' and a 'Sort by: Latest activity' button. The main content area is divided into two columns. The left column, titled 'Categories', lists various discussion topics: 'View all discussions', 'Announcements', 'General', 'Ideas', 'Polls', 'Q&A', and 'Show and tell'. The right column, titled 'Discussions', displays a list of discussion topics. Each topic includes a title, a user icon, the number of replies (indicated by an up arrow and a number), and the status (e.g., 'Unanswered' or 'Answered'). The topics listed are: 'How to pass any GitHub Environment Se...', 'shell-repo-bot should give more details of queries.', 'What is the equivalent of Azure DevOps Connections in GitHub environment.', 'Validate a pull request comment', and 'How to move specific code from One bra Pull request'. At the bottom of the page, there is a link to 'Community guidelines'.

Welcome to GitHub-Trainings

Announcements · SherinMirza

Search: is:open Sort by: Latest activity

### Categories

- View all discussions
- Announcements
- General
- Ideas
- Polls
- Q&A
- Show and tell

### Discussions

- How to pass any GitHub Environment Se  
newfunda asked on Jun 2 in Q&A · Unanswered
- shell-repo-bot should give more details of queries.  
tulsishell started on May 17 in Ideas
- What is the equivalent of Azure DevOps Connections in GitHub environment.  
somparna-datta started on May 2 in General
- Validate a pull request comment  
nlavanya02 asked on May 3 in Q&A · Unanswered
- How to move specific code from One bra Pull request  
gunni475 asked on Apr 28 in Q&A · Answered

Most helpful

Be sure to mark someone's comment as an answer if it helps you resolve your question — they deserve the credit! 💖

Community guidelines



## 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!

The screenshot shows the GitHub Actions 'Secrets and variables' page. On the left is a sidebar with navigation links: General, Access (Collaborators and teams, Moderation options), Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables, Actions, Codespaces, Dependabot), and Integrations (GitHub Apps, Email notifications). The 'Secrets and variables' link is highlighted. The main content area is titled 'Actions secrets and variables'. It contains a paragraph explaining that secrets and variables are used for sensitive data and are encrypted. Below this is a section with tabs for 'Secrets' and 'Variables', and a 'New repository secret' button. The 'Environment secrets' section has a 'Manage environments' button and lists a secret named 'MY\_ENV\_SECRET' with a 'Demo' label and an update time of '3 minutes ago'. The 'Repository secrets' section lists a secret named 'MY\_REPO\_SECRET' with an update time of '2 minutes ago' and edit/delete icons. The 'Organization secrets' section has a 'Manage organization secrets' button and lists a secret named 'MY\_ORG\_SECRET' with an update time of 'Updated on Apr 25'.

General

Access

- Collaborators and teams
- Moderation options

Code and automation

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

Security

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

Integrations

- GitHub Apps
- Email notifications

## Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets.](#) Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables.](#)

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets Variables New repository secret

### Environment secrets

Manage environments

MY_ENV_SECRET	Demo	Updated 3 minutes ago
---------------	------	-----------------------

### Repository secrets

MY_REPO_SECRET	Updated 2 minutes ago	<span>Edit</span> <span>Delete</span>
----------------	-----------------------	---------------------------------------

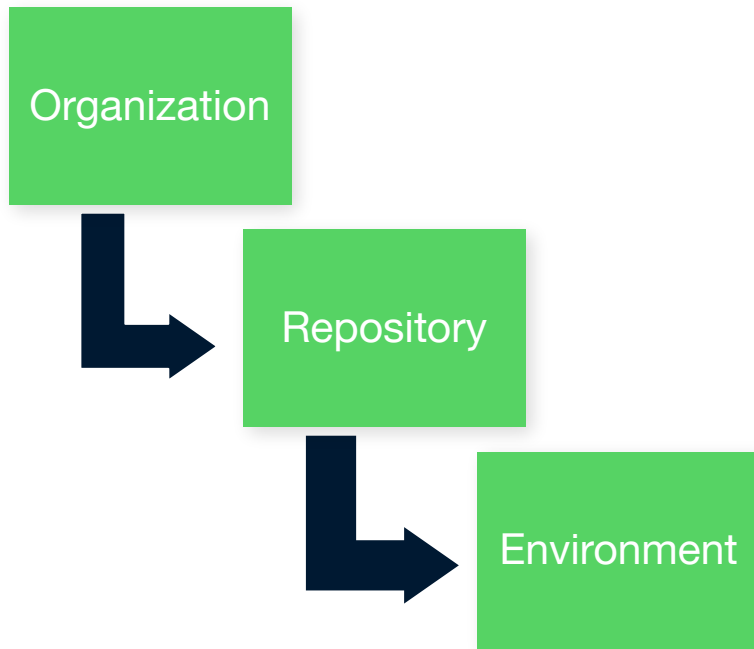
### Organization secrets

Manage organization secrets

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

# Secrets

- Defined on org, repo, or environment level
- Secret context
  - `{{ secrets.MY_SECRET }}`
  - Set as input (`with:`) or environment (`env:`) for actions
- Set in UI or CLI
  - `$ gh secret set MY_SECRET -body "$value"`
  - `$ gh secret set MY_SECRET --env Prod`
  - `$ gh secret set MY_SECRET --org my-org`
- Masked in log



# The GITHUB\_TOKEN

- `${{ secrets.GITHUB_TOKEN }}` or `${{ github.token }}`
- Authenticate to GitHub to perform automation inside the workflow's repo
- Default permission read/write for all scopes (old default) or set to read
- [permissions - GitHub Enterprise Cloud Docs](#)

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

```
permissions: read-all
```

```
permissions:  
  actions: read|write|none  
  checks: read|write|none  
  contents: read|write|none  
  deployments: read|write|none  
  issues: read|write|none  
  packages: read|write|none  
  pull-requests: read|write|none  
  repository-projects: read|write|none  
  security-events: read|write|none  
  statuses: read|write|none
```

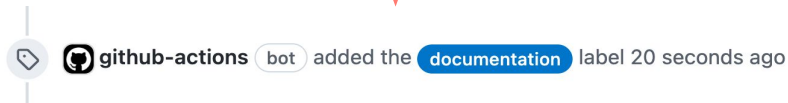
# The GITHUB\_TOKEN

Perform actions as “github-actions”:

```
permissions:  
  contents: read  
  issues: write  
  
label_issues:  
  runs-on: ubuntu-latest  
  if: github.event_name == 'issues'  
  
steps:  
  - uses: andymckay/labeler@e6c4322d0397f3240f0e7e30a33b5c5df2d39e90  
    with:  
      add-labels: documentation  
      repo-token: ${{ secrets.GITHUB_TOKEN }}
```



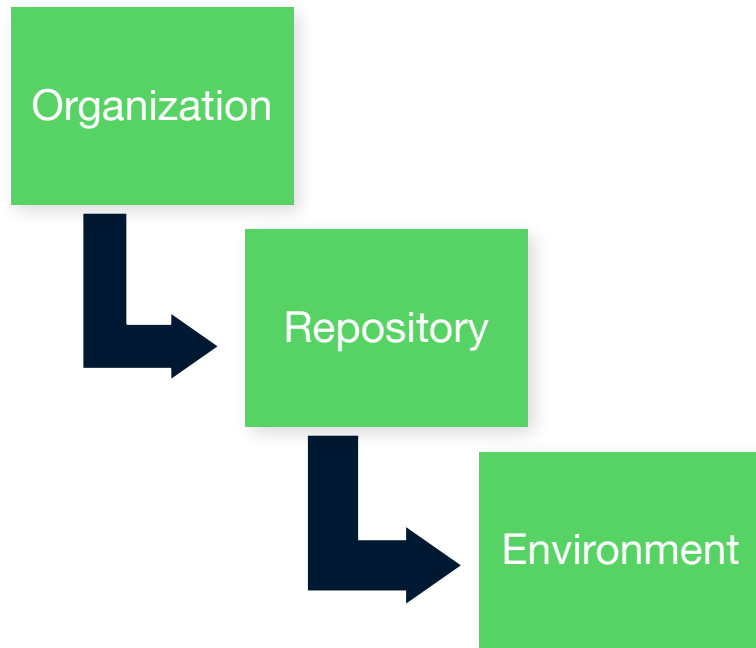
The diagram consists of two red arrows. The first arrow starts at the 'permissions' section of the YAML code and points down to the 'label\_issues' section. The second arrow starts at the 'repo-token' field in the 'steps' section, which is highlighted with a red border, and points down to the 'github-actions' bot action in the GitHub interface.





# 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
```

```
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 REST 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. 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.' Below this is a 'Parameters' table with columns 'name', 'required', and 'description'. The table lists parameters: 'owner' (required), 'repo' (required), 'tag\_name' (required, with description 'The name of the tag.'), and 'target\_commitish' (not required, with description '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).'). On the right, a code block shows the JavaScript usage: 

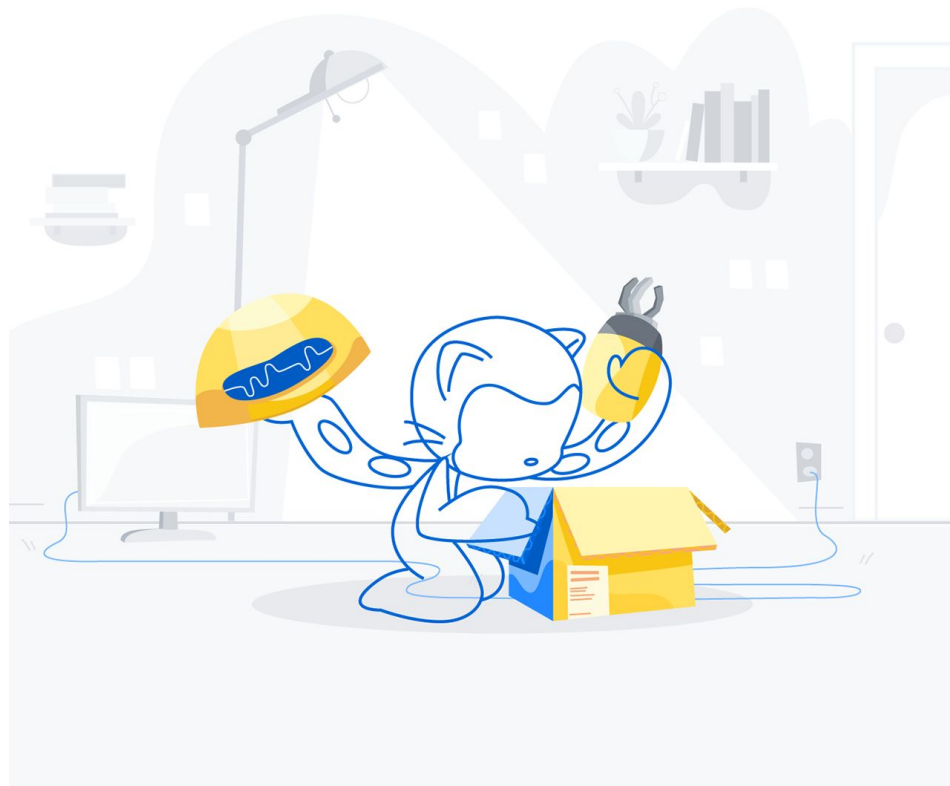
```
octokit.rest.repos.createRelease({  
  owner,  
  repo,  
  tag_name,  
});
```

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> ).

# 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



# 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

```
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

```
1  name: 'Hello World'
2  description: 'Greet someone'
3  inputs:
4    who-to-greet:
5      description: 'Who to greet'
6      required: true
7      default: 'World'
8  outputs:
9    random-number:
10     description: "Random number"
11     value: "${{ steps.random-number-generator.outputs.random-id }}"
12  runs:
13    using: "composite"
14    steps:
15      - run: echo Hello ${ inputs.who-to-greet }.
16        shell: bash
17
18      - id: random-number-generator
19        run: echo " :set-output name=random-id:$(echo $RANDOM)"
20        shell: bash
21
22      - run: echo "${{ github.action_path }}" >> $GITHUB_PATH
23        shell: bash
24
25      - run: echo "Goodbye $YOU"
26        shell: bash
27        env:
28          YOU: "${{ inputs.who-to-greet }}"
```



# 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

