Using scripts to test your code on a runner

How to use essential GitHub Actions features for continuous integration (CI).

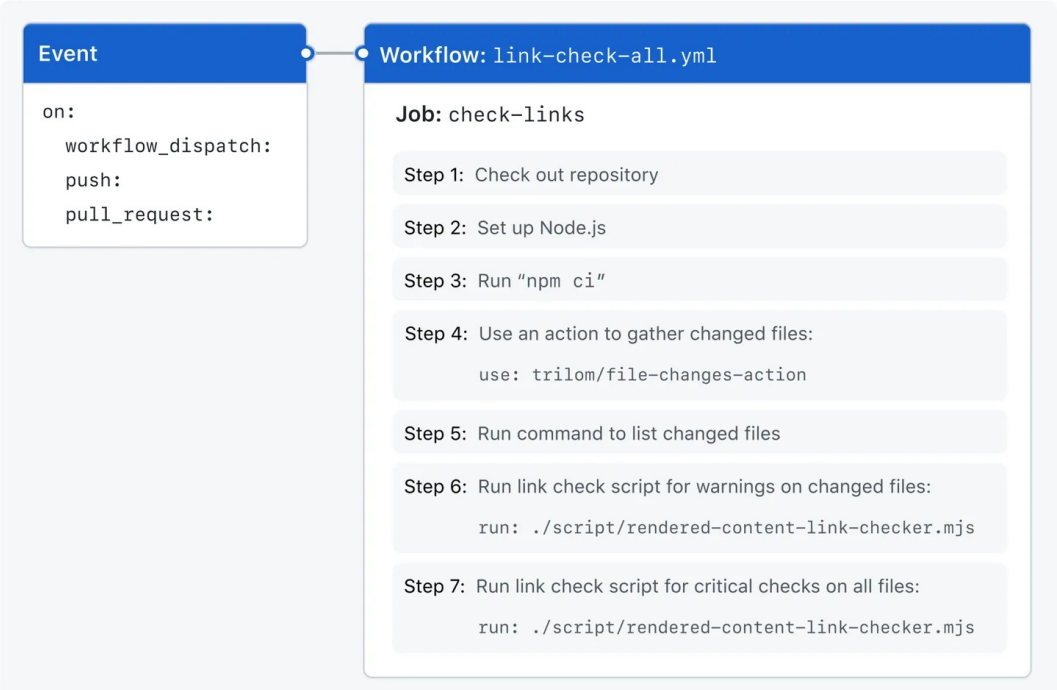
In this article

Note: GitHub-hosted runners are not currently supported on GitHub Enterprise Server. You can see more information about planned future support on the GitHub public roadmap.

Example overview

This article uses an example workflow to demonstrate some of the main CI features of GitHub Actions. When this workflow is triggered, it automatically runs a script that checks whether the GitHub Docs site has any broken links.

The following diagram shows a high level view of the workflow's steps and how they run within the job:



Features used in this example

The example workflow demonstrates the following capabilities of GitHub Actions.

Feature	Implementation
Triggering a workflow to run automatically	`push`
Triggering a workflow to run automatically	`pull_request`
Manually running a workflow from the UI	`workflow_dispatch`
Setting permissions for the token	`permissions`
Controlling how many workflow runs or jobs can run at the same time	`concurrency`
Running the job on different runners, depending on the repository	`runs-on`
Cloning your repository to the runner	`actions/checkout`
Installing `node` on the runner	`actions/setup-node`
Using a third-party action	`trilom/file-changes-action`
Running a script on the runner	Using `./script/rendered-content-link-checker.mjs`

Example workflow

The following workflow was created by the GitHub Docs Engineering team. To review the latest version of this file in the `github/docs` repository, see `check-broken-links-github-github.yml`.

The following workflow renders the content of every page in the documentation and checks all internal links to ensure they connect correctly.

YAML Beside Inline

```yaml
name: 'Link Checker: All English'
```

This defines the name of the workflow as it will appear in the "Actions" tab of the GitHub repository.

```yaml
on:
```

The `on` key lets you define the events that trigger when the workflow is run. You can define multiple events here. For more information, see "Triggering a workflow."

```yaml
  workflow_dispatch:
```

Add the `workflow_dispatch` event if you want to be able to manually run this workflow from the UI. For more information, see `workflow_dispatch`.

```yaml
  push:
    branches:
      - main
```

Add the `push` event, so that the workflow runs automatically every time a commit is

Add the `push` event, so that the workflow runs automatically every time a commit is pushed to a branch called `main`. For more information, see `push`.

```
  pull_request:
```

Add the `pull_request` event, so that the workflow runs automatically every time a pull request is created or updated. For more information, see `pull_request`.

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

This modifies the default permissions granted to `GITHUB_TOKEN`. This will vary depending on the needs of your workflow. For more information, see "Assigning permissions to jobs."

In this example, the `pull-requests: read` permission is needed for the `trilom/file-changes-action` action that is used later in this workflow.

```
concurrency:
  group: '${{ github.workflow }} @ ${{ github.event.pull_request.head.label
|| github.head_ref || github.ref }}'
  cancel-in-progress: true
```

The `concurrency` key ensures that only a single workflow in the same concurrency group will run at the same time. For more information, see "Using concurrency." `concurrency.group` generates a concurrency group name from the workflow name and pull request information. The `||` operator is used to define fallback values. `concurrency.cancel-in-progress` cancels any currently running job or workflow in the same concurrency group.

```
jobs:
```

The `jobs` key groups together all the jobs that run in the workflow file.

```
  check-links:
```

This line defines a job with the ID `check-links` that is stored within the `jobs` key.

```
    runs-on: ${{ fromJSON('["ubuntu-latest", "self-hosted"]')
[github.repository == 'github/docs-internal'] }}
```

The `runs-on` key in this example configures the job to run on a GitHub-hosted runner or a self-hosted runner, depending on the repository running the workflow.

In this example, the job will run on a self-hosted runner if the repository is named `docs-internal` and is within the `github` organization. If the repository doesn't match this path, then it will run on an `ubuntu-latest` runner hosted by GitHub. For more information on these options, see "Choosing the runner for a job."

```
    steps:
```

The `steps` key groups together all the steps that will run as part of the `check-links` job. Each job in a workflow has its own `steps` section.

job. Each job in a workflow has its own `steps` section.

```
      - name: Checkout
        uses: actions/checkout@v4
```

The `uses` key tells the job to retrieve the action named `actions/checkout`. This is an action that checks out your repository and downloads it to the runner, allowing you to run actions against your code (such as testing tools). You must use the checkout action any time your workflow will use the repository's code or you are using an action defined in the repository.

```
      - name: Setup node
        uses: actions/setup-node@v3
        with:
          node-version: 16.13.x
          cache: npm
```

This step uses the `actions/setup-node` action to install the specified version of the Node.js software package on the runner, which gives you access to the `npm` command.

```
      - name: Install
        run: npm ci
```

The `run` key tells the job to execute a command on the runner. In this example, `npm ci` is used to install the npm software packages for the project.

```
      - name: Gather files changed
        uses: trilom/file-changes-
action@a6ca26c14274c33b15e6499323aac178af06ad4b
        with:
          fileOutput: 'json'
```

This step uses the `trilom/file-changes-action` action to gather all the changed files. This example is pinned to a specific version of the action, using the `a6ca26c14274c33b15e6499323aac178af06ad4b` SHA.

In this example, this step creates the file "${{ env.HOME }}/files.json", among others.

```
      - name: Show files changed
        run: cat $HOME/files.json
```

To help with verification, this step lists the contents of `files.json`. This will be visible in the workflow run's log, and can be useful for debugging.

```
      - name: Link check (warnings, changed files)
        run: |
          ./script/rendered-content-link-checker.mjs \
            --language en \
            --max 100 \
            --check-anchors \
            --check-images \
            --verbose \
            --list $HOME/files.json
```

This step uses the `run` command to execute a script that is stored in the repository

This step uses the `run` command to execute a script that is stored in the repository at `script/rendered-content-link-checker.mjs` and passes all the parameters it needs to run.

```
      - name: Link check (critical, all files)
        run: |
          ./script/rendered-content-link-checker.mjs \
            --language en \
            --exit \
            --verbose \
            --check-images \
            --level critical
```

This step also uses `run` command to execute a script that is stored in the repository at `script/rendered-content-link-checker.mjs` and passes a different set of parameters.

Next steps

- To learn about GitHub Actions concepts, see "Understanding GitHub Actions."
- For more step-by-step guide for creating a basic workflow, see "Quickstart for GitHub Actions."
- If you're comfortable with the basics of GitHub Actions, you can learn about workflows and their features at "About workflows."