Using the GitHub CLI on a runner

In this article

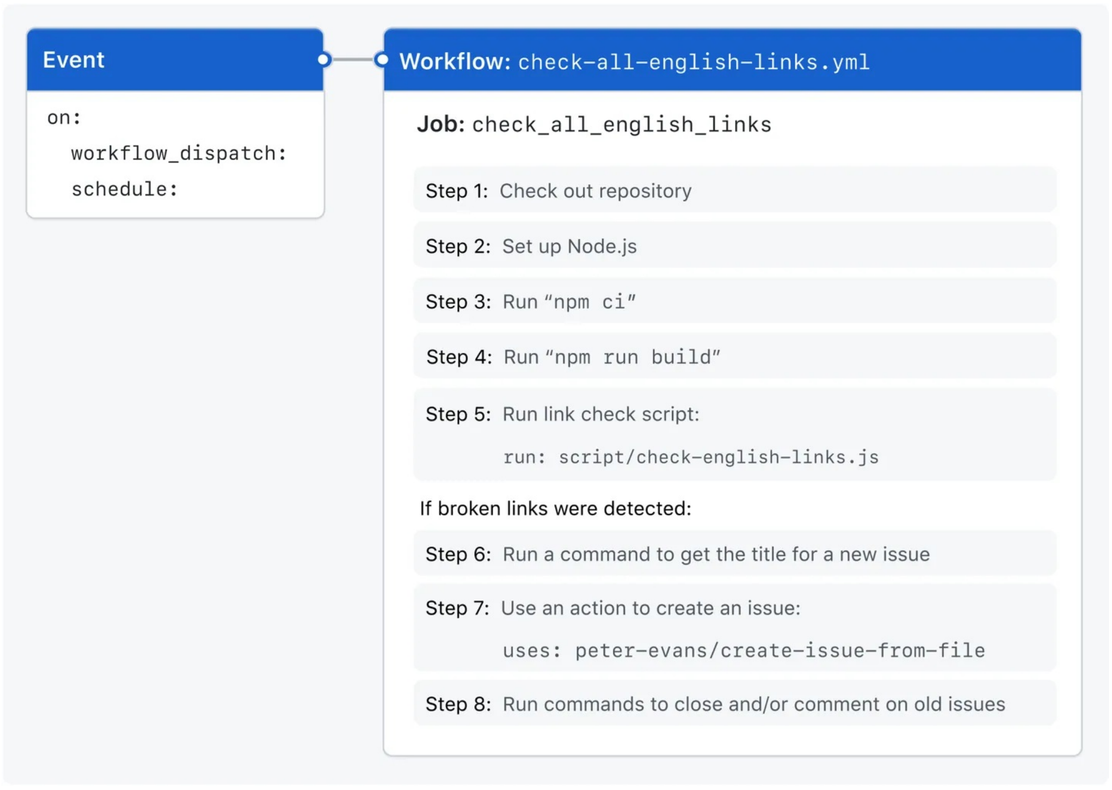
How to use advanced GitHub Actions features for continuous integration (CI).

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. If any broken links are found, the workflow uses the GitHub CLI to create a GitHub issue with the details.

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


Event
on:
workflow_dispatch:
schedule:
Workflow: check-all-english-links.yml
Job: check_all_english_links
Step 1: Check out repository
Step 2: Set up Node.js
Step 3: Run "npm ci"
Step 4: Run "npm run build"
Step 5: Run link check script:
run: script/check-english-links.js
If broken links were detected:
Step 6: Run a command to get the title for a new issue
Step 7: Use an action to create an issue:
uses: peter-evans/create-issue-from-file
Step 8: Run commands to close and/or comment on old issues

Features used in this example

The example workflow demonstrates the following capabilities of GitHub Actions.

Feature	Implementation
Running a workflow at regular intervals	`schedule`
Setting permissions for the token	`permissions`
Preventing a job from running unless specific conditions are met	`if`
Referencing secrets in a workflow	Secrets
Cloning your repository to the runner	`actions/checkout`
Installing `node` on the runner	`actions/setup-node`
Using a third-party action	`peter-evans/create-issue-from-file`
Running shell commands on the runner	`run`
Running a script on the runner	Using `script/check-english-links.js`
Generating an output file	Piping the output using the `>` operator
Checking for existing issues using GitHub CLI	`gh issue list`
Commenting on an issue using GitHub CLI	`gh issue comment`

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-all-english-`

`links.yml`.

Note: Each line of this workflow is explained in the next section at "Understanding the example."

YAML

```yaml
name: Check all English links

# **What it does**: This script once a day checks all English links and reports in i
# **Why we have it**: We want to know if any links break.
# **Who does it impact**: Docs content.

on:
  workflow_dispatch:
  schedule:
    - cron: '40 19 * * *' # once a day at 19:40 UTC / 11:40 PST

permissions:
  contents: read
  issues: write

jobs:
  check_all_english_links:
    name: Check all links
    if: github.repository == 'github/docs-internal'
    runs-on: ubuntu-latest
    env:
      GITHUB_TOKEN: ${{ secrets.DOCUBOT_READORG_REPO_WORKFLOW_SCOPES }}
      FIRST_RESPONDER_PROJECT: Docs content first responder
      REPORT_AUTHOR: docubot
      REPORT_LABEL: broken link report
      REPORT_REPOSITORY: github/docs-content
    steps:
      - name: Check out repo's default branch
        uses: actions/checkout@v2
      - name: Setup Node
        uses: actions/setup-node@v2
        with:
          node-version: 16.13.x
          cache: npm
      - name: npm ci
        run: npm ci
      - name: npm run build
        run: npm run build
      - name: Run script
        run: |
          script/check-english-links.js > broken_links.md

      # check-english-links.js returns 0 if no links are broken, and 1 if any links
      # are broken. When an Actions step's exit code is 1, the action run's job stat
      # is failure and the run ends. The following steps create an issue for the
      # broken link report only if any links are broken, so `if: ${{ failure() }}`
      # ensures the steps run despite the previous step's failure of the job.

      - if: ${{ failure() }}
        name: Get title for issue
        id: check
        run: echo "::set-output name=title::$(head -1 broken_links.md)"
      - if: ${{ failure() }}
        name: Create issue from file
        id: broken-link-report
        uses: peter-evans/create-issue-from-file@ceef9be92406ace67ab5421f66570acf213
        with:
          token: ${{ env.GITHUB_TOKEN }}

          title: ${{ steps.check.outputs.title }}
          content-filepath: ./broken_links.md
          repository: ${{ env.REPORT_REPOSITORY }}
          labels: ${{ env.REPORT_LABEL }}
      - if: ${{ failure() }}
        name: Close and/or comment on old issues
        env:
          NEW_REPORT_URL: 'https://github.com/${{ env.REPORT_REPOSITORY }}/issues/$
        run: |
          gh alias set list-reports "issue list \
                                       --repo ${{ env.REPORT_REPOSITORY }} \
                                       --author ${{ env.REPORT_AUTHOR }} \
                                       --label '${{ env.REPORT_LABEL }}'"

          # Link to the previous report from the new report that triggered this
          # workflow run.

          previous_report_url=$(gh list-reports \
                                  --state all \
                                  --limit 2 \
                                  --json url \
                                  --jq '.[].url' \
                                  | grep -v ${{ env.NEW_REPORT_URL }} | head -1)

          gh issue comment ${{ env.NEW_REPORT_URL }} --body "⬅ [Previous report]($p

          # If an old report is open and assigned to someone, link to the newer
          # report without closing the old report.

          for issue_url in $(gh list-reports \
                                  --json assignees,url \
                                  --jq '.[] | select (.assignees != []) | .url'); d
            if [ "$issue_url" != "${{ env.NEW_REPORT_URL }}" ]; then
              gh issue comment $issue_url --body "➡ [Newer report](${{ env.NEW_REPO
            fi
          done

          # Link to the newer report from any older report that is still open,
          # then close the older report and remove it from the first responder's
          # project board.

          for issue_url in $(gh list-reports \
                                  --search 'no:assignee' \
                                  --json url \
                                  --jq '.[].url'); do
            if [ "$issue_url" != "${{ env.NEW_REPORT_URL }}" ]; then
              gh issue comment $issue_url --body "➡ [Newer report](${{ env.NEW_REPO
              gh issue close $issue_url
              gh issue edit $issue_url --remove-project "${{ env.FIRST_RESPONDER_PR
            fi
          done
```

Understanding the example

The following table explains how each of these features are used when creating a GitHub Actions workflow.

Code	**Explanation**
YAML `name: Check all English links`	The name of the workflow as it will appear in the "Actions" tab of the GitHub repository.
YAML	Defines the `workflow_dispatch` and `scheduled` as triggers for the workflow: - The `workflow_dispatch` lets you

```yaml
on:
  workflow_dispatch:
  schedule:
    - cron: '40 20 * * *' # once a day at 20:40 UTC / 12:40 PST
```

- The `workflow_dispatch` lets you manually run this workflow from the UI. For more information, see [`workflow_dispatch`](#).
- The `schedule` event lets you use `cron` syntax to define a regular interval for automatically triggering the workflow. For more information, see [`schedule`](#).

YAML

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

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](#)."

YAML

```yaml
jobs:
```

Groups together all the jobs that run in the workflow file.

YAML

```yaml
  check_all_english_links:
    name: Check all links
```

Defines a job with the ID `check_all_english_links`, and the name `Check all links`, that is stored within the `jobs` key.

YAML

```yaml
if: github.repository == 'github/docs-internal'
```

Only run the `check_all_english_links` job if the repository is named `docs-internal` and is within the `github` organization. Otherwise, the job is marked as *skipped*.

YAML

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

Configures the job to run on an Ubuntu Linux runner. This means that the job will execute on a fresh virtual machine hosted by GitHub. For syntax examples using other runners, see "[Workflow syntax for GitHub Actions](#)."

YAML

```yaml
    env:
      GITHUB_TOKEN: ${{ secrets.DOCUBOT_READORG_REPO_WORKFLOW_SCOPE
      REPORT_AUTHOR: docubot

      REPORT_LABEL: broken link report
      REPORT_REPOSITORY: github/docs-content
```

Creates custom environment variables, and redefines the built-in `GITHUB_TOKEN` variable to use a custom [secret](#). These variables will be referenced later in the workflow.

YAML

```yaml
    steps:
```

Groups together all the steps that will run as part of the `check_all_english_links` job. Each job in the workflow has its own `steps` section.

YAML

```yaml
- name: Check out repo's default branch
  uses: actions/checkout@v2
```

The `uses` keyword 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 run against the repository's code or you are using an action defined in the repository.

YAML

```yaml
- name: Setup Node
  uses: actions/setup-node@v2
  with:
    node-version: 16.8.x
    cache: npm
```

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

YAML

```yaml
- name: Run the "npm ci" command
  run: npm ci
- name: Run the "npm run build" command
  run: npm run build
```

The `run` keyword tells the job to execute a command on the runner. In this case, the `npm ci` and `npm run build` commands are run as separate steps to install and build the Node.js application in the repository.

YAML

```yaml
- name: Run script
  run: |
    script/check-english-links.js > broken_links.md
```

This `run` command executes a script that is stored in the repository at `script/check-english-links.js`, and pipes the output to a file called `broken_links.md`.

YAML

```yaml
- if: ${{ failure() }}
  name: Get title for issue
  id: check
  run: echo "::set-output name=title::$(head -1 broken_links.n
```

If the `check-english-links.js` script detects broken links and returns a non-zero (failure) exit status, then use a workflow command to set an output that has the value of the first line of the `broken_links.md` file (this is used the next step).

YAML

```yaml
- if: ${{ failure() }}
  name: Create issue from file
  id: broken-link-report
  uses: peter-evans/create-issue-from-file@ceef9be92406ace67ab
  with:
    token: ${{ env.GITHUB_TOKEN }}

    title: ${{ steps.check.outputs.title }}
    content-filepath: ./broken_links.md
    repository: ${{ env.REPORT_REPOSITORY }}
```

Uses the `peter-evans/create-issue-from-file` action to create a new GitHub issue. This example is pinned to a specific version of the action, using the `ceef9be92406ace67ab5421f66570acf213ec395` SHA.

```
        labels: ${{ env.REPORT_LABEL }}
```

YAML

```
    - if: ${{ failure() }}
      name: Close and/or comment on old issues
      env:
        NEW_REPORT_URL: 'https://github.com/${{ env.REPORT_REPOSIT
      run: |
        gh alias set list-reports "issue list \
                                  --repo ${{ env.REPORT_REPOSIT
                                  --author ${{ env.REPORT_AUTHO
                                  --label '${{ env.REPORT_LABEL
        previous_report_url=$(gh list-reports \
                              --state all \
                              --limit 2 \
                              --json url \
                              --jq '.[].url' \
                              | grep -v ${{ env.NEW_REPORT_URL }

        gh issue comment ${{ env.NEW_REPORT_URL }} --body "⬅ [Prev
```

Uses `gh issue list` to locate the previously created issue from earlier runs. This is [aliased](#) to `gh list-reports` for simpler processing in later steps. To get the issue URL, the `jq` expression processes the resulting JSON output.

`gh issue comment` is then used to add a comment to the new issue that links to the previous one.

YAML

```
        for issue_url in $(gh list-reports \
                              --json assignees,url \
                              --jq '.[] | select (.assignees !=
          if [ "$issue_url" != "$" ]; then
            gh issue comment $issue_url --body "➡ [Newer report](
          fi
        done
```

If an issue from a previous run is open and assigned to someone, then use `gh issue comment` to add a comment with a link to the new issue.

YAML

```
        for issue_url in $(gh list-reports \
                              --search 'no:assignee' \
                              --json url \
                              --jq '.[].url'); do
          if [ "$issue_url" != "${{ env.NEW_REPORT_URL }}" ]; then
            gh issue comment $issue_url --body "➡ [Newer report](
            gh issue close $issue_url
            gh issue edit $issue_url --remove-project "${{ env.FIR
          fi
        done
```

If an issue from a previous run is open and is not assigned to anyone, then:

- Use `gh issue comment` to add a comment with a link to the new issue.
- Use `gh issue close` to close the old issue.
- Use `gh issue edit` to edit the old issue to remove it from a specific GitHub project board.

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."