Using concurrency, expressions, and a test matrix

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

In this article

Example overview

Features used in this example

Example workflow

Next steps

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 tests your code using a matrix of test combinations with `npm test`.

The following diagram shows a high level view of the workflow's steps and how they run within the job:
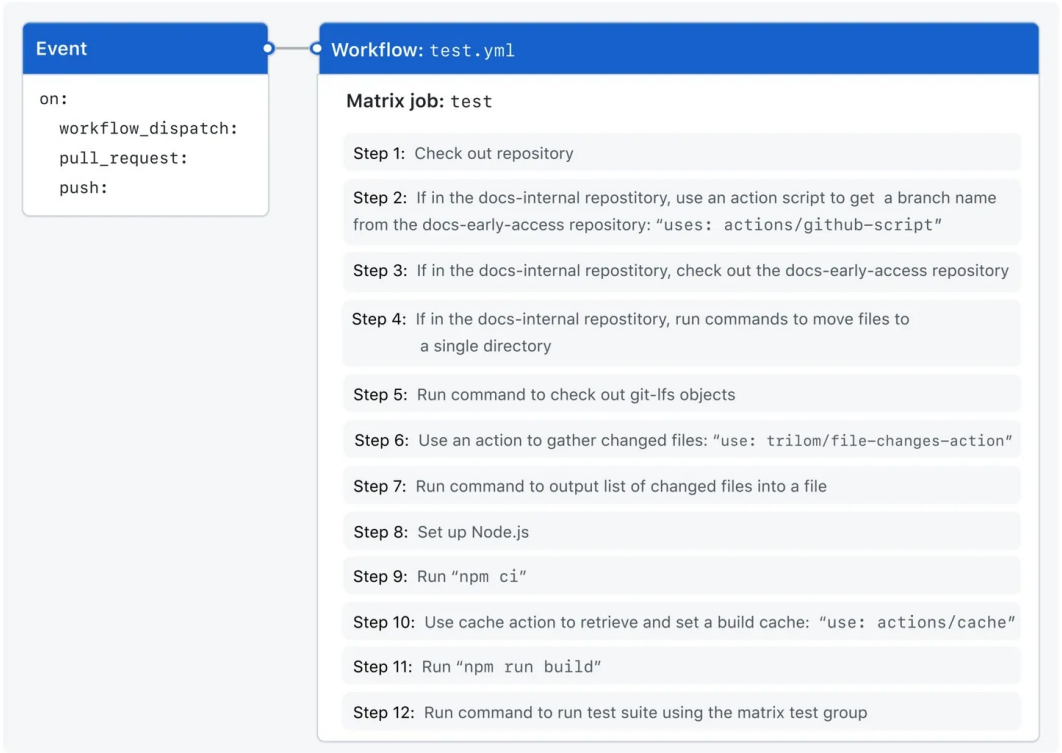
Features used in this example 🔗

The example workflow demonstrates the following capabilities of GitHub Actions.

Feature	Implementation

Manually running a workflow from the UI	`workflow_dispatch`
Triggering a workflow to run automatically	`pull_request`
Running a workflow at regular intervals	`schedule`
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`
Preventing a job from running unless specific conditions are met	`if`
Using a matrix to create different test configurations	`matrix`
Cloning your repository to the runner	`actions/checkout`
Installing `node` on the runner	`actions/setup-node`
Caching dependencies	`actions/cache`
Running tests on the runner	`npm test`

Example workflow 🔗

The following workflow was created by the GitHub Docs Engineering team. The workflow runs tests against the code in a pull request. To review the latest version of this file in the `github/docs` repository, see `test.yml` .

YAML   Beside   Inline   ⧉

```yaml
name: Node.js Tests
```

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

```yaml
on:
```

The `on` keyword 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. For more information, see `workflow_dispatch` .

```yaml
  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` .

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

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

```
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."

```
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:
```

This groups together all the jobs that run in the workflow file.

```
test:
```

This defines a job with the ID `test` that is stored within the `jobs` key.

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

This 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."

```
    timeout-minutes: 60
```

This sets the maximum number of minutes to let the job run before it is automatically canceled. For more information, see `timeout-minutes` .

```
    strategy:
```

This section defines the build matrix for your jobs.

```
      fail-fast: false
```

Setting `fail-fast` to `false` prevents GitHub from cancelling all in-progress jobs if any matrix job fails.

```
      matrix:
        test-group:
          [
            content,
            graphql,
            meta,
            rendering,
            routing,
            unit,
            linting,
            translations,
          ]
```

This creates a matrix named `test-group`, with an array of test groups. These values match the names of test groups that will be run by `npm test`.

```
    steps:
```

This groups together all the steps that will run as part of the `test` job. Each job in a workflow has its own `steps` section.

```
      - name: Check out repo
        uses: actions/checkout@v4
        with:
          lfs: ${{ matrix.test-group == 'content' }}
          persist-credentials: 'false'
```

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 use your repository's code. Some extra options are provided to the action using the `with` key.

```
      - name: Figure out which docs-early-access branch to checkout, if
  internal repo
        if: ${{ github.repository == 'github/docs-internal' }}
        id: check-early-access
        uses: actions/github-script@v6
        env:
          BRANCH_NAME: ${{ github.head_ref || github.ref_name }}
        with:
          github-token: ${{ secrets.DOCUBOT_REPO_PAT }}
          result-encoding: string
          script: |
            const { BRANCH_NAME } = process.env
            try {
              const response = await github.repos.getBranch({
```

```
                owner: 'github',
                repo: 'docs-early-access',
                BRANCH_NAME,
            })
            console.log(`Using docs-early-access branch called
'${BRANCH_NAME}'.`)
            return BRANCH_NAME
        } catch (err) {
            if (err.status === 404) {
            console.log(`There is no docs-early-access branch called
'${BRANCH_NAME}' so checking out 'main' instead.`)
                return 'main'
            }
            throw err
        }
```

If the current repository is the `github/docs-internal` repository, this step uses the `actions/github-script` action to run a script to check if there is a branch called `docs-early-access`.

```
    - name: Check out docs-early-access too, if internal repo
      if: ${{ github.repository == 'github/docs-internal' }}
      uses: actions/checkout@v4
      with:
        repository: github/docs-early-access
        token: ${{ secrets.DOCUBOT_REPO_PAT }}
        path: docs-early-access
        ref: ${{ steps.check-early-access.outputs.result }}
```

If the current repository is the `github/docs-internal` repository, this step checks out the branch from the `github/docs-early-access` that was identified in the previous step.

```
    - name: Merge docs-early-access repo's folders
      if: ${{ github.repository == 'github/docs-internal' }}
      run: |
        mv docs-early-access/assets assets/images/early-access
        mv docs-early-access/content content/early-access
        mv docs-early-access/data data/early-access
        rm -r docs-early-access
```

If the current repository is the `github/docs-internal` repository, this step uses the `run` keyword to execute shell commands to move the `docs-early-access` repository's folders into the main repository's folders.

```
    - name: Checkout LFS objects
      run: git lfs checkout
```

This step runs a command to check out large file storage (LFS) objects from the repository.

```
    - name: Gather files changed
      uses: trilom/file-changes-
action@a6ca26c14274c33b15e6499323aac178af06ad4b
      id: get_diff_files
      with:
        output: ' '
```

This step uses the `trilom/file-changes-action` action to gather the files changed in the pull request, so they can be analyzed in the next step. This example is pinned to

a specific version of the action, using the `a6ca26c14274c33b15e6499323aac178af06ad4b` SHA.

```
      - name: Insight into changed files
        run: |
          echo "${{ steps.get_diff_files.outputs.files }}" >
get_diff_files.txt
```

This step runs a shell command that uses an output from the previous step to create a file containing the list of files changed in the pull request.

```
      - name: Setup node
        uses: actions/setup-node@v3
        with:
          node-version: 16.14.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.

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

This step runs the `npm ci` shell command to install the npm software packages for the project.

```
      - name: Cache nextjs build
        uses: actions/cache@v3
        with:
          path: .next/cache
          key: ${{ runner.os }}-nextjs-${{ hashFiles('package*.json') }}
```

This step uses the `actions/cache` action to cache the Next.js build, so that the workflow will attempt to retrieve a cache of the build, and not rebuild it from scratch every time. For more information, see "Caching dependencies to speed up workflows."

```
      - name: Run build script
        run: npm run build
```

This step runs the build script.

```
      - name: Run tests
        env:
          DIFF_FILE: get_diff_files.txt
          CHANGELOG_CACHE_FILE_PATH: tests/fixtures/changelog-feed.json
        run: npm test -- tests/${{ matrix.test-group }}/
```

This step runs the tests using `npm test`, and the test matrix provides a different value for `${{ matrix.test-group }}` for each job in the matrix. It uses the `DIFF_FILE` environment variable to know which files have changed, and uses the `CHANGELOG_CACHE_FILE_PATH` environment variable for the changelog cache file.

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](#)."