Events that trigger workflows

In this article

You can configure your workflows to run when specific activity on GitHub happens, at a scheduled time, or when an event outside of GitHub occurs.

About events that trigger workflows

Workflow triggers are events that cause a workflow to run. For more information about how to use workflow triggers, see "Triggering a workflow."

Some events have multiple activity types. For these events, you can specify which activity types will trigger a workflow run. For more information about what each activity type means, see "Webhook events and payloads."

> **Note:** Not all webhook events trigger workflows.

`branch_protection_rule`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`branch_protection_rule`	- `created` - `edited` - `deleted`	Last commit on default branch	Default branch

> **Note**: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

> **Note:** This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when branch protection rules in the workflow repository are changed. For more information about branch protection rules, see "About protected branches." For information about the branch protection rule APIs, see "Objects" in the GraphQL API documentation or "Branches" in the REST API documentation.

For example, you can run a workflow when a branch protection rule has been `created` or `deleted`:

```yaml
on:
  branch_protection_rule:
    types: [created, deleted]
```

`check_run`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`check_run`	- `created` - `rerequested` - `completed` - `requested_action`	Last commit on default branch	Default branch

> **Note**: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

> **Note:** This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when activity related to a check run occurs. A check run is an individual test that is part of a check suite. For information, see "Using the REST API to interact with checks." For information about the check run APIs, see "Objects" in the GraphQL API documentation or "Checks" in the REST API documentation.

For example, you can run a workflow when a check run has been `rerequested` or `completed`.

```yaml
on:
  check_run:
    types: [rerequested, completed]
```

`check_suite`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`check_suite`	- `completed`	Last commit on default branch	Default branch

> **Note**: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." Although only the `completed` activity type is supported, specifying the activity type will keep your workflow specific if more activity types are added in the future. By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

> **Note:** This event will only trigger a workflow run if the workflow file is on the default branch.

> **Note:** To prevent recursive workflows, this event does not trigger workflows if the check suite was created by GitHub Actions.

Runs your workflow when check suite activity occurs. A check suite is a collection of the check runs created for a specific commit. Check suites summarize the status and conclusion of the check runs that are in the suite. For information, see "Using the REST API to interact with checks." For information about the check suite APIs, see "Objects" in the GraphQL API documentation or "Checks" in the REST API documentation.

For example, you can run a workflow when a check suite has been `completed`.

```yaml
on:
  check_suite:
    types: [completed]
```

`create`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`create`	Not applicable	Last commit on the created branch or tag	Branch or tag created

> **Note**: An event will not be created when you create more than three tags at once.

Runs your workflow when someone creates a Git reference (Git branch or tag) in the workflow's repository. For information about the APIs to create a Git reference, see "Mutations" in the GraphQL API documentation or "Git database" in the REST API documentation.

For example, you can run a workflow when the `create` event occurs.

```yaml
on:
  create
```

`delete`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`delete`	Not applicable	Last commit on default branch	Default branch

> **Note:** This event will only trigger a workflow run if the workflow file is on the default branch.

> **Note**: An event will not be created when you delete more than three tags at once.

Runs your workflow when someone deletes a Git reference (Git branch or tag) in the workflow's repository. For information about the APIs to delete a Git reference, see "Mutations" in the GraphQL API documentation or "Git database" in the REST API documentation.

For example, you can run a workflow when the `delete` event occurs.

```yaml
on:
  delete
```

`deployment`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`deployment`	Not applicable	Commit to be deployed	Branch or tag to be deployed (empty if created with a commit SHA)

Runs your workflow when someone creates a deployment in the workflow's repository. Deployments created with a commit SHA may not have a Git ref. For information about the APIs to create a deployment, see "Mutations" in the GraphQL API documentation or "Repositories" in the REST API documentation.

For example, you can run a workflow when the `deployment` event occurs.

```yaml
on:
  deployment
```

`deployment_status`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`deployment_status`	Not applicable	Commit to be deployed	Branch or tag to be deployed (empty if commit)

Note: When a deployment status's state is set to `inactive`, a workflow run will not be triggered.

Runs your workflow when a third party provides a deployment status. Deployments created with a commit SHA may not have a Git ref. For information about the APIs to create a deployment status, see "Mutations" in the GraphQL API documentation or "Deployments" in the REST API documentation.

For example, you can run a workflow when the `deployment_status` event occurs.

```
on:
  deployment_status
```

`discussion`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`discussion`	- `created` - `edited` - `deleted` - `transferred` - `pinned` - `unpinned` - `labeled` - `unlabeled` - `locked` - `unlocked` - `category_changed` - `answered` - `unanswered`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Note: Webhook events for GitHub Discussions are currently in beta and subject to change.

Runs your workflow when a discussion in the workflow's repository is created or modified. For activity related to comments on a discussion, use the `discussion_comment` event. For more information about discussions, see "About discussions." For information about the GraphQL API, see "Objects."

For example, you can run a workflow when a discussion has been `created`, `edited`, or

`answered`.

```yaml
on:
  discussion:
    types: [created, edited, answered]
```

`discussion_comment`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`discussion_comment`	- `created` - `edited` - `deleted`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Note: Webhook events for GitHub Discussions are currently in beta and subject to change.

Runs your workflow when a comment on a discussion in the workflow's repository is created or modified. For activity related to a discussion as opposed to comments on the discussion, use the `discussion` event. For more information about discussions, see "About discussions." For information about the GraphQL API, see "Objects."

For example, you can run a workflow when a discussion comment has been `created` or `deleted`.

```yaml
on:
  discussion_comment:
    types: [created, deleted]
```

`fork`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`fork`	Not applicable	Last commit on default branch	Default branch

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when someone forks a repository. For information about the REST API, see "Repositories."

For example, you can run a workflow when the `fork` event occurs.

```yaml
on:
  fork
```

`gollum`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`gollum`	Not applicable	Last commit on default branch	Default branch

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when someone creates or updates a Wiki page. For more information, see "About wikis."

For example, you can run a workflow when the `gollum` event occurs.

```yaml
on:
  gollum
```

`issue_comment`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`issue_comment`	- `created` - `edited` - `deleted`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when an issue or pull request comment is created, edited, or deleted. For information about the issue comment APIs, see "Objects" in the GraphQL API documentation or "Webhook events and payloads" in the REST API documentation.

For example, you can run a workflow when an issue or pull request comment has been `created` or `deleted`.

```yaml
on:
  issue_comment:
    types: [created, deleted]
```

`issue_comment` on issues only or pull requests only

The `issue_comment` event occurs for comments on both issues and pull requests. You can use the `github.event.issue.pull_request` property in a conditional to take different action depending on whether the triggering object was an issue or pull request.

For example, this workflow will run the `pr_commented` job only if the `issue_comment` event

originated from a pull request. It will run the `issue_commented` job only if the `issue_comment` event originated from an issue.

```yaml
on: issue_comment

jobs:
  pr_commented:
    # This job only runs for pull request comments
    name: PR comment
    if: ${{ github.event.issue.pull_request }}
    runs-on: ubuntu-latest
    steps:
      - run: |
          echo A comment on PR $NUMBER
        env:
          NUMBER: ${{ github.event.issue.number }}

  issue_commented:
    # This job only runs for issue comments
    name: Issue comment
    if: ${{ !github.event.issue.pull_request }}
    runs-on: ubuntu-latest
    steps:
      - run: |
          echo A comment on issue $NUMBER
        env:
          NUMBER: ${{ github.event.issue.number }}
```

`issues`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`issues`	- `opened` - `edited` - `deleted` - `transferred` - `pinned` - `unpinned` - `closed` - `reopened` - `assigned` - `unassigned` - `labeled` - `unlabeled` - `locked` - `unlocked` - `milestoned` - `demilestoned`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when an issue in the workflow's repository is created or modified. For activity related to comments in an issue, use the `issue_comment` event. For more information about issues, see "About issues." For information about the issue APIs, see "Objects" in the GraphQL API documentation or "Issues" in the REST API documentation.

For example, you can run a workflow when an issue has been `opened`, `edited`, or `milestoned`.

```yaml
on:
  issues:
    types: [opened, edited, milestoned]
```

`label`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`label`	- `created` - `edited` - `deleted`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when a label in your workflow's repository is created or modified. For more information about labels, see "Managing labels." For information about the label APIs, see "Objects" in the GraphQL API documentation or "Issues" in the REST API documentation.

If you want to run your workflow when a label is added to or removed from an issue, pull request, or discussion, use the `labeled` or `unlabeled` activity types for the `issues`, `pull_request`, `pull_request_target`, or `discussion` events instead.

For example, you can run a workflow when a label has been `created` or `deleted`.

```yaml
on:
  label:
    types: [created, deleted]
```

`merge_group`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`merge_group`	`checks_requested`	SHA of the merge group	Ref of the merge group

Note: More than one activity type triggers this event. Although only the `checks_requested` activity type is supported, specifying the activity type will keep your workflow specific if more activity types are added in the future. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Runs your workflow when a pull request is added to a merge queue, which adds the pull

request to a merge group. For more information see "Merging a pull request with a merge queue".

For example, you can run a workflow when the `checks_requested` activity has occurred.

```yaml
on:
  merge_group:
    types: [checks_requested]
```

`milestone`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`milestone`	- `created` - `closed` - `opened` - `edited` - `deleted`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when a milestone in the workflow's repository is created or modified. For more information about milestones, see "About milestones." For information about the milestone APIs, see "Objects" in the GraphQL API documentation or "Issues" in the REST API documentation.

If you want to run your workflow when an issue is added to or removed from a milestone, use the `milestoned` or `demilestoned` activity types for the `issues` event instead.

For example, you can run a workflow when a milestone has been `opened` or `deleted`.

```yaml
on:
  milestone:
    types: [opened, deleted]
```

`page_build`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`page_build`	Not applicable	Last commit on default branch	Not applicable

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when someone pushes to a branch that is the publishing source for GitHub Pages, if GitHub Pages is enabled for the repository. For more information about GitHub Pages publishing sources, see "Configuring a publishing source for your GitHub

Pages site." For information about the REST API, see "Repositories."

For example, you can run a workflow when the `page_build` event occurs.

```yaml
on:
  page_build
```

`project`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`project`	- `created` - `closed` - `reopened` - `edited` - `deleted`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. The `edited` activity type refers to when a project board, not a column or card on the project board, is edited. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Note: This event only occurs for projects owned by the workflow's repository, not for organization-owned or user-owned projects or for projects owned by another repository.

Note: This event only occurs for projects (classic).

Runs your workflow when a project board is created or modified. For activity related to cards or columns in a project board, use the `project_card` or `project_column` events instead. For more information about project boards, see "About projects (classic)." For information about the project board APIs, see "Objects" in the GraphQL API documentation or "Projects (classic)" in the REST API documentation.

For example, you can run a workflow when a project has been `created` or `deleted`.

```yaml
on:
  project:
    types: [created, deleted]
```

`project_card`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`project_card`	- `created` - `moved` - `converted` to an issue - `edited` - `deleted`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Note: This event only occurs for projects owned by the workflow's repository, not for organization-owned or user-owned projects or for projects owned by another repository.

Note: This event only occurs for projects (classic).

Runs your workflow when a card on a project board is created or modified. For activity related to project boards or columns in a project board, use the `project` or `project_column` event instead. For more information about project boards, see "About projects (classic)." For information about the project card APIs, see "Objects" in the GraphQL API documentation or "Projects (classic)" in the REST API documentation.

For example, you can run a workflow when a project card has been `created` or `deleted`.

```yaml
on:
  project_card:
    types: [created, deleted]
```

`project_column`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`project_column`	- `created` - `updated` - `moved` - `deleted`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Note: This event only occurs for projects owned by the workflow's repository, not for organization-owned or user-owned projects or for projects owned by another repository.

Note: This event only occurs for projects (classic).

Runs your workflow when a column on a project board is created or modified. For activity related to project boards or cards in a project board, use the `project` or `project_card` event instead. For more information about project boards, see "About projects (classic)." For information about the project column APIs, see "Objects" in the GraphQL API

documentation or "Projects (classic)" in the REST API documentation.

For example, you can run a workflow when a project column has been `created` or `deleted`.

```yaml
on:
  project_column:
    types: [created, deleted]
```

`public`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`public`	Not applicable	Last commit on default branch	Default branch

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when your workflow's repository changes from private to public. For information about the REST API, see "Repositories."

For example, you can run a workflow when the `public` event occurs.

```yaml
on:
  public
```

`pull_request`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`pull_request`	- `assigned` - `unassigned` - `labeled` - `unlabeled` - `opened` - `edited` - `closed` - `reopened` - `synchronize` - `converted_to_draft` - `ready_for_review` - `locked` - `unlocked` - `review_requested` - `review_request_removed` - `auto_merge_enabled` - `auto_merge_disabled`	Last merge commit on the `GITHUB_REF` branch	PR merge branch `refs/pull/:prNumber/merge`

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, a workflow only runs when a `pull_request` event's activity type is `opened`, `synchronize`, or `reopened`. To trigger workflows by different activity types, use the `types` keyword. For more information, see "Workflow syntax for GitHub

Actions."

Note: Workflows will not run on `pull_request` activity if the pull request has a merge conflict. The merge conflict must be resolved first.

Conversely, workflows with the `pull_request_target` event will run even if the pull request has a merge conflict. Before using the `pull_request_target` trigger, you should be aware of the security risks. For more information, see `pull_request_target`.

Runs your workflow when activity on a pull request in the workflow's repository occurs. For example, if no activity types are specified, the workflow runs when a pull request is opened or reopened or when the head branch of the pull request is updated. For activity related to pull request reviews, pull request review comments, or pull request comments, use the `pull_request_review`, `pull_request_review_comment`, or `issue_comment` events instead. For information about the pull request APIs, see "Objects" in the GraphQL API documentation or "Pulls" in the REST API documentation.

Note that `GITHUB_SHA` for this event is the last merge commit of the pull request merge branch. If you want to get the commit ID for the last commit to the head branch of the pull request, use `github.event.pull_request.head.sha` instead.

For example, you can run a workflow when a pull request has been opened or reopened.

```yaml
on:
  pull_request:
    types: [opened, reopened]
```

You can use the event context to further control when jobs in your workflow will run. For example, this workflow will run when a review is requested on a pull request, but the `specific_review_requested` job will only run when a review by `octo-team` is requested.

```yaml
on:
  pull_request:
    types: [review_requested]
jobs:
  specific_review_requested:
    runs-on: ubuntu-latest
    if: ${{ github.event.requested_team.name == 'octo-team'}}
    steps:
      - run: echo 'A review from octo-team was requested'
```

Running your `pull_request` workflow based on the head or base branch of a pull request

You can use the `branches` or `branches-ignore` filter to configure your workflow to only run on pull requests that target specific branches. For more information, see "Workflow syntax for GitHub Actions."

For example, this workflow will run when someone opens a pull request that targets a branch whose name starts with `releases/`:

```yaml
on:
  pull_request:
    types:
      - opened
    branches:
      - 'releases/**'
```

Note: If you use both the `branches` filter and the `paths` filter, the workflow will only run when

both filters are satisfied. For example, the following workflow will only run when a pull request that includes a change to a JavaScript (`.js`) file is opened on a branch whose name starts with `releases/` :

```yaml
on:
  pull_request:
    types:
      - opened
    branches:
      - 'releases/**'
    paths:
      - '**.js'
```

To run a job based on the pull request's head branch name (as opposed to the pull request's base branch name), use the `github.head_ref` context in a conditional. For example, this workflow will run whenever a pull request is opened, but the `run_if` job will only execute if the head of the pull request is a branch whose name starts with `releases/` :

```yaml
on:
  pull_request:
    types:
      - opened
jobs:
  run_if:
    if:  startsWith(github.head_ref, 'releases/')
    runs-on: ubuntu-latest
    steps:
      - run: echo "The head of this PR starts with 'releases/'"
```

Running your `pull_request` workflow based on files changed in a pull request

You can also configure your workflow to run when a pull request changes specific files. For more information, see "Workflow syntax for GitHub Actions."

For example, this workflow will run when a pull request includes a change to a JavaScript file (`.js`):

```yaml
on:
  pull_request:
    paths:
      - '**.js'
```

Note: If you use both the `branches` filter and the `paths` filter, the workflow will only run when both filters are satisfied. For example, the following workflow will only run when a pull request that includes a change to a JavaScript (`.js`) file is opened on a branch whose name starts with `releases/` :

```yaml
on:
  pull_request:
    types:
      - opened
    branches:
      - 'releases/**'
    paths:
      - '**.js'
```

Running your `pull_request` workflow when a pull request merges

When a pull request merges, the pull request is automatically closed. To run a workflow

when a pull request merges, use the `pull_request` `closed` event type along with a conditional that checks the `merged` value of the event. For example, the following workflow will run whenever a pull request closes. The `if_merged` job will only run if the pull request was also merged.

```yaml
on:
  pull_request:
    types:
      - closed

jobs:
  if_merged:
    if: github.event.pull_request.merged == true
    runs-on: ubuntu-latest
    steps:
    - run: |
        echo The PR was merged
```

Workflows in forked repositories

Workflows don't run in forked repositories by default. You must enable GitHub Actions in the **Actions** tab of the forked repository.

With the exception of `GITHUB_TOKEN`, secrets are not passed to the runner when a workflow is triggered from a forked repository. The `GITHUB_TOKEN` has read-only permissions in pull requests from forked repositories. For more information, see "Automatic token authentication."

Pull request events for forked repositories

For pull requests from a forked repository to the base repository, GitHub sends the `pull_request`, `issue_comment`, `pull_request_review_comment`, `pull_request_review`, and `pull_request_target` events to the base repository. No pull request events occur on the forked repository.

When a first-time contributor submits a pull request to a public repository, a maintainer with write access may need to approve running workflows on the pull request. For more information, see "Approving workflow runs from public forks."

For pull requests from a forked repository to a private repository, workflows only run when they are enabled, see "Managing GitHub Actions settings for a repository."

> **Note:** Workflows triggered by Dependabot pull requests are treated as though they are from a forked repository, and are also subject to these restrictions.

`pull_request_comment` (use `issue_comment`)

To run your workflow when a comment on a pull request (not on a pull request's diff) is created, edited, or deleted, use the `issue_comment` event. For activity related to pull request reviews or pull request review comments, use the `pull_request_review` or `pull_request_review_comment` events.

`pull_request_review`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`pull_request_review`	- `submitted`	Last merge commit on	PR merge branch

- `edited` - `dismissed`	the `GITHUB_REF` branch	`refs/pull/:prNumber/merge`

> **Note**: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Runs your workflow when a pull request review is submitted, edited, or dismissed. A pull request review is a group of pull request review comments in addition to a body comment and a state. For activity related to pull request review comments or pull request comments, use the `pull_request_review_comment` or `issue_comment` events instead. For information about the pull request review APIs, see "Objects" in the GraphQL API documentation or "Pulls" in the REST API documentation.

For example, you can run a workflow when a pull request review has been `edited` or `dismissed`.

```yaml
on:
  pull_request_review:
    types: [edited, dismissed]
```

Running a workflow when a pull request is approved

To run your workflow when a pull request has been approved, you can trigger your workflow with the `submitted` type of `pull_request_review` event, then check the review state with the `github.event.review.state` property. For example, this workflow will run whenever a pull request review is submitted, but the `approved` job will only run if the submitted review is an approving review:

```yaml
on:
  pull_request_review:
    types: [submitted]

jobs:
  approved:
    if: github.event.review.state == 'APPROVED'
    runs-on: ubuntu-latest
    steps:
      - run: echo "This PR was approved"
```

Workflows in forked repositories

Workflows don't run in forked repositories by default. You must enable GitHub Actions in the **Actions** tab of the forked repository.

With the exception of `GITHUB_TOKEN`, secrets are not passed to the runner when a workflow is triggered from a forked repository. The `GITHUB_TOKEN` has read-only permissions in pull requests from forked repositories. For more information, see "Automatic token authentication."

Pull request events for forked repositories

For pull requests from a forked repository to the base repository, GitHub sends the `pull_request`, `issue_comment`, `pull_request_review_comment`, `pull_request_review`, and `pull_request_target` events to the base repository. No pull request events occur on the forked repository.

When a first-time contributor submits a pull request to a public repository, a maintainer

with write access may need to approve running workflows on the pull request. For more information, see "Approving workflow runs from public forks."

For pull requests from a forked repository to a private repository, workflows only run when they are enabled, see "Managing GitHub Actions settings for a repository."

> **Note:** Workflows triggered by Dependabot pull requests are treated as though they are from a forked repository, and are also subject to these restrictions.

`pull_request_review_comment`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`pull_request_review_comment`	- `created` - `edited` - `deleted`	Last merge commit on the `GITHUB_REF` branch	PR merge branch `refs/pull/:prNumber/merge`

> **Note**: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Runs your workflow when a pull request review comment is modified. A pull request review comment is a comment on a pull request's diff. For activity related to pull request reviews or pull request comments, use the `pull_request_review` or `issue_comment` events instead. For information about the pull request review comment APIs, see "Objects" in the GraphQL API documentation or "Pulls" in the REST API documentation.

For example, you can run a workflow when a pull request review comment has been `created` or `deleted`.

```yaml
on:
  pull_request_review_comment:
    types: [created, deleted]
```

Workflows in forked repositories

Workflows don't run in forked repositories by default. You must enable GitHub Actions in the **Actions** tab of the forked repository.

With the exception of `GITHUB_TOKEN`, secrets are not passed to the runner when a workflow is triggered from a forked repository. The `GITHUB_TOKEN` has read-only permissions in pull requests from forked repositories. For more information, see "Automatic token authentication."

Pull request events for forked repositories

For pull requests from a forked repository to the base repository, GitHub sends the `pull_request`, `issue_comment`, `pull_request_review_comment`, `pull_request_review`, and `pull_request_target` events to the base repository. No pull request events occur on the forked repository.

When a first-time contributor submits a pull request to a public repository, a maintainer with write access may need to approve running workflows on the pull request. For more information, see "Approving workflow runs from public forks."

For pull requests from a forked repository to a private repository, workflows only run when they are enabled, see "Managing GitHub Actions settings for a repository."

> **Note:** Workflows triggered by Dependabot pull requests are treated as though they are from a forked repository, and are also subject to these restrictions.

`pull_request_target`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`pull_request`	- `assigned` - `unassigned` - `labeled` - `unlabeled` - `opened` - `edited` - `closed` - `reopened` - `synchronize` - `converted_to_draft` - `ready_for_review` - `locked` - `unlocked` - `review_requested` - `review_request_removed` - `auto_merge_enabled` - `auto_merge_disabled`	Last commit on the PR base branch	PR base branch

> **Note**: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, a workflow only runs when a `pull_request_target` event's activity type is `opened`, `synchronize`, or `reopened`. To trigger workflows by different activity types, use the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Runs your workflow when activity on a pull request in the workflow's repository occurs. For example, if no activity types are specified, the workflow runs when a pull request is opened or reopened or when the head branch of the pull request is updated.

This event runs in the context of the base of the pull request, rather than in the context of the merge commit, as the `pull_request` event does. This prevents execution of unsafe code from the head of the pull request that could alter your repository or steal any secrets you use in your workflow. This event allows your workflow to do things like label or comment on pull requests from forks. Avoid using this event if you need to build or run code from the pull request.

To ensure repository security, branches with names that match certain patterns (such as those which look similar to SHAs) may not trigger workflows with the `pull_request_target` event.

> **Warning:** For workflows that are triggered by the `pull_request_target` event, the `GITHUB_TOKEN` is granted read/write repository permission unless the `permissions` key is specified and the workflow can access secrets, even when it is triggered from a fork. Although the workflow runs in the context of the base of the pull request, you should make sure that you do not check out, build, or run untrusted code from the pull request with this event. Additionally, any caches share the same scope as the base branch. To help prevent cache poisoning, you should not save the cache if there is a possibility that the cache contents were altered. For more information, see "Keeping your GitHub Actions and workflows secure: Preventing pwn requests" on the GitHub

Security Lab website.

For example, you can run a workflow when a pull request has been `assigned`, `opened`, `synchronize`, or `reopened`.

```yaml
on:
  pull_request_target:
    types: [assigned, opened, synchronize, reopened]
```

Running your `pull_request_target` workflow based on the head or base branch of a pull request

You can use the `branches` or `branches-ignore` filter to configure your workflow to only run on pull requests that target specific branches. For more information, see "Workflow syntax for GitHub Actions."

For example, this workflow will run when someone opens a pull request that targets a branch whose name starts with `releases/`:

```yaml
on:
  pull_request_target:
    types:
      - opened
    branches:
      - 'releases/**'
```

Note: If you use both the `branches` filter and the `paths` filter, the workflow will only run when both filters are satisfied. For example, the following workflow will only run when a pull request that includes a change to a JavaScript (`.js`) file is opened on a branch whose name starts with `releases/`:

```yaml
on:
  pull_request_target:
    types:
      - opened
    branches:
      - 'releases/**'
    paths:
      - '**.js'
```

To run a job based on the pull request's head branch name (as opposed to the pull request's base branch name), use the `github.head_ref` context in a conditional. For example, this workflow will run whenever a pull request is opened, but the `run_if` job will only execute if the head of the pull request is a branch whose name starts with `releases/`:

```yaml
on:
  pull_request_target:
    types:
      - opened
jobs:
  run_if:
    if:  startsWith(github.head_ref, 'releases/')
    runs-on: ubuntu-latest
    steps:
      - run: echo "The head of this PR starts with 'releases/'"
```

Running your `pull_request_target` workflow based on files changed in a pull request

You can use the `paths` or `paths-ignore` filter to configure your workflow to run when a pull request changes specific files. For more information, see "Workflow syntax for GitHub Actions."

For example, this workflow will run when a pull request includes a change to a JavaScript file (`.js`):

```yaml
on:
  pull_request_target:
    paths:
      - '**.js'
```

Note: If you use both the `branches` filter and the `paths` filter, the workflow will only run when both filters are satisfied. For example, the following workflow will only run when a pull request that includes a change to a JavaScript (`.js`) file is opened on a branch whose name starts with `releases/` :

```yaml
on:
  pull_request_target:
    types:
      - opened
    branches:
      - 'releases/**'
    paths:
      - '**.js'
```

Running your `pull_request_target` workflow when a pull request merges

When a pull request merges, the pull request is automatically closed. To run a workflow when a pull request merges, use the `pull_request_target` `closed` event type along with a conditional that checks the `merged` value of the event. For example, the following workflow will run whenever a pull request closes. The `if_merged` job will only run if the pull request was also merged.

```yaml
on:
  pull_request_target:
    types:
      - closed

jobs:
  if_merged:
    if: github.event.pull_request.merged == true
    runs-on: ubuntu-latest
    steps:
    - run: |
        echo The PR was merged
```

`push`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`push`	Not applicable	When you delete a branch, the SHA in the workflow run (and its associated refs) reverts to the default branch of the repository.	Updated ref

Note: The webhook payload available to GitHub Actions does not include the `added`, `removed`, and `modified` attributes in the `commit` object. You can retrieve the full commit object using the API. For information, see "Objects" in the GraphQL API documentation or "Commits" in the REST API documentation.

Note: An event will not be created when you push more than three tags at once.

Runs your workflow when you push a commit or tag, or when you create a repository from a template.

For example, you can run a workflow when the `push` event occurs.

```yaml
on:
  push
```

Note: When a `push` webhook event triggers a workflow run, the Actions UI's "pushed by" field shows the account of the pusher and not the author or committer. However, if the changes are pushed to a repository using SSH authentication with a deploy key, then the "pushed by" field will be the repository admin who verified the deploy key when it was added it to a repository.

Running your workflow only when a push to specific branches occurs

You can use the `branches` or `branches-ignore` filter to configure your workflow to only run when specific branches are pushed. For more information, see "Workflow syntax for GitHub Actions."

For example, this workflow will run when someone pushes to `main` or to a branch that starts with `releases/`.

```yaml
on:
  push:
    branches:
      - 'main'
      - 'releases/**'
```

Note: If you use both the `branches` filter and the `paths` filter, the workflow will only run when both filters are satisfied. For example, the following workflow will only run when a push that includes a change to a JavaScript (`.js`) file is made to a branch whose name starts with `releases/` :

```yaml
on:
  push:
    branches:
      - 'releases/**'
    paths:
      - '**.js'
```

Running your workflow only when a push of specific tags occurs

You can use the `tags` or `tags-ignore` filter to configure your workflow to only run when specific tags are pushed. For more information, see "Workflow syntax for GitHub Actions."

For example, this workflow will run when someone pushes a tag that starts with `v1.`.

```yaml
on:
  push:
    tags:
      - v1.**
```

Running your workflow only when a push affects specific files

You can use the `paths` or `paths-ignore` filter to configure your workflow to run when a push to specific files occurs. For more information, see "Workflow syntax for GitHub Actions."

For example, this workflow will run when someone pushes a change to a JavaScript file (`.js`):

```yaml
on:
  push:
    paths:
      - '**.js'
```

Note: If you use both the `branches` filter and the `paths` filter, the workflow will only run when both filters are satisfied. For example, the following workflow will only run when a push that includes a change to a JavaScript (`.js`) file is made to a branch whose name starts with `releases/`:

```yaml
on:
  push:
    branches:
      - 'releases/**'
    paths:
      - '**.js'
```

`registry_package`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`registry_package`	- `published` - `updated`	Commit of the published package	Branch or tag of the published package

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Note: When pushing multi-architecture container images, this event occurs once per manifest, so you might observe your workflow triggering multiple times. To mitigate this, and only run your workflow job for the event that contains the actual image tag information, use a conditional:

```yaml
jobs:
    job_name:
        if: ${{ github.event.registry_package.package_version.container_metadata.tag.name != '' }}
```

Runs your workflow when activity related to GitHub Packages occurs in your repository. For more information, see "GitHub Packages Documentation."

For example, you can run a workflow when a new package version has been `published`.

```yaml
on:
  registry_package:
    types: [published]
```

`release`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`release`	- `published` - `unpublished` - `created` - `edited` - `deleted` - `prereleased` - `released`	Last commit in the tagged release	Tag ref of release `refs/tags/<tag_name>`

Note: More than one activity type triggers this event. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: Workflows are not triggered for the `created`, `edited`, or `deleted` activity types for draft releases. When you create your release through the GitHub browser UI, your release may automatically be saved as a draft.

Note: The `prereleased` type will not trigger for pre-releases published from draft releases, but the `published` type will trigger. If you want a workflow to run when stable *and* pre-releases publish, subscribe to `published` instead of `released` and `prereleased`.

Runs your workflow when release activity in your repository occurs. For information about the release APIs, see "Objects" in the GraphQL API documentation or "Releases" in the REST API documentation.

For example, you can run a workflow when a release has been `published`.

```yaml
on:
  release:
    types: [published]
```

`repository_dispatch`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
repository_dispatch	Custom	Last commit on default branch	Default branch

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

You can use the GitHub API to trigger a webhook event called `repository_dispatch` when you want to trigger a workflow for activity that happens outside of GitHub. For more information, see "Repositories."

When you make a request to create a `repository_dispatch` event, you must specify an `event_type` to describe the activity type. By default, all `repository_dispatch` activity types trigger a workflow to run. You can use the `types` keyword to limit your workflow to run when a specific `event_type` value is sent in the `repository_dispatch` webhook payload.

```yaml
on:
  repository_dispatch:
    types: [test_result]
```

Note: The `event_type` value is limited to 100 characters.

Any data that you send through the `client_payload` parameter will be available in the `github.event` context in your workflow. For example, if you send this request body when you create a repository dispatch event:

```json
{
  "event_type": "test_result",
  "client_payload": {
    "passed": false,
    "message": "Error: timeout"
  }
}
```

then you can access the payload in a workflow like this:

```yaml
on:
  repository_dispatch:
    types: [test_result]

jobs:
  run_if_failure:
    if: ${{ !github.event.client_payload.passed }}
    runs-on: ubuntu-latest
    steps:
      - env:
          MESSAGE: ${{ github.event.client_payload.message }}
        run: echo $MESSAGE
```

Notes:

- The maximum number of top-level properties in `client_payload` is 10.
- The payload can contain a maximum of 65,535 characters.

`schedule`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
Not applicable	Not applicable	Last commit on default branch	Default branch

> Note: The `schedule` event can be delayed during periods of high loads of GitHub Actions workflow runs. High load times include the start of every hour. If the load is sufficiently high enough, some queued jobs may be dropped. To decrease the chance of delay, schedule your workflow to run at a different time of the hour.

The `schedule` event allows you to trigger a workflow at a scheduled time.

You can schedule a workflow to run at specific UTC times using POSIX cron syntax. Scheduled workflows run on the latest commit on the default or base branch. The shortest interval you can run scheduled workflows is once every 5 minutes.

This example triggers the workflow every day at 5:30 and 17:30 UTC:

```yaml
on:
  schedule:
    # * is a special character in YAML so you have to quote this string
    - cron:  '30 5,17 * * *'
```

A single workflow can be triggered by multiple `schedule` events. You can access the schedule event that triggered the workflow through the `github.event.schedule` context. This example triggers the workflow to run at 5:30 UTC every Monday-Thursday, but skips the `Not on Monday or Wednesday` step on Monday and Wednesday.

```yaml
on:
  schedule:
    - cron: '30 5 * * 1,3'
    - cron: '30 5 * * 2,4'

jobs:
  test_schedule:
    runs-on: ubuntu-latest
    steps:
      - name: Not on Monday or Wednesday
        if: github.event.schedule != '30 5 * * 1,3'
        run: echo "This step will be skipped on Monday and Wednesday"
      - name: Every time
        run: echo "This step will always run"
```

Cron syntax has five fields separated by a space, and each field represents a unit of time.

```
┌─────────────── minute (0 - 59)
│ ┌─────────────── hour (0 - 23)
│ │ ┌─────────────── day of the month (1 - 31)
│ │ │ ┌─────────────── month (1 - 12 or JAN-DEC)
│ │ │ │ ┌─────────────── day of the week (0 - 6 or SUN-SAT)
│ │ │ │ │
│ │ │ │ │
│ │ │ │ │
* * * * *
```

You can use these operators in any of the five fields:

Operator	Description	Example
*	Any value	`15 * * * *` runs at every minute 15 of every hour of every day.
,	Value list separator	`2,10 4,5 * * *` runs at minute 2 and 10 of the 4th and 5th hour of every day.

-	Range of values	`30 4-6 * * *` runs at minute 30 of the 4th, 5th, and 6th hour.
/	Step values	`20/15 * * * *` runs every 15 minutes starting from minute 20 through 59 (minutes 20, 35, and 50).

Note: GitHub Actions does not support the non-standard syntax `@yearly`, `@monthly`, `@weekly`, `@daily`, `@hourly`, and `@reboot`.

You can use crontab guru to help generate your cron syntax and confirm what time it will run. To help you get started, there is also a list of crontab guru examples.

Notifications for scheduled workflows are sent to the user who last modified the cron syntax in the workflow file. For more information, see "Notifications for workflow runs."

`status`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`status`	Not applicable	Last commit on default branch	Not applicable

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when the status of a Git commit changes. For example, commits can be marked as `error`, `failure`, `pending`, or `success`. If you want to provide more details about the status change, you may want to use the `check_run` event. For information about the commit status APIs, see "Objects" in the GraphQL API documentation or "Commits" in the REST API documentation.

For example, you can run a workflow when the `status` event occurs.

```yaml
on:
  status
```

If you want to run a job in your workflow based on the new commit state, you can use the `github.event.state` context. For example, the following workflow triggers when a commit status changes, but the `if_error_or_failure` job only runs if the new commit state is `error` or `failure`.

```yaml
on:
  status
jobs:
  if_error_or_failure:
    runs-on: ubuntu-latest
    if: >-
      github.event.state == 'error' ||
      github.event.state == 'failure'
    steps:
      - env:
          DESCRIPTION: ${{ github.event.description }}
        run: |
          echo The status is error or failed: $DESCRIPTION
```

`watch`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`watch`	- `started`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. Although only the `started` activity type is supported, specifying the activity type will keep your workflow specific if more activity types are added in the future. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Runs your workflow when the workflow's repository is starred. For information about the pull request APIs, see "Mutations" in the GraphQL API documentation or "Activity" in the REST API documentation.

For example, you can run a workflow when someone stars a repository, which is the `started` activity type for a watch event.

```yaml
on:
  watch:
    types: [started]
```

`workflow_call`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
Same as the caller workflow	Not applicable	Same as the caller workflow	Same as the caller workflow

`workflow_call` is used to indicate that a workflow can be called by another workflow. When a workflow is triggered with the `workflow_call` event, the event payload in the called workflow is the same event payload from the calling workflow. For more information see, "Reusing workflows."

The example below only runs the workflow when it's called from another workflow:

```yaml
on: workflow_call
```

`workflow_dispatch`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
workflow_dispatch	Not applicable	Last commit on the `GITHUB_REF` branch or tag	Branch or tag that received dispatch

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

To enable a workflow to be triggered manually, you need to configure the `workflow_dispatch` event. You can manually trigger a workflow run using the GitHub API, GitHub CLI, or GitHub browser interface. For more information, see "Manually running a workflow."

```yaml
on: workflow_dispatch
```

Providing inputs

You can configure custom-defined input properties, default input values, and required inputs for the event directly in your workflow. When you trigger the event, you can provide the `ref` and any `inputs`. When the workflow runs, you can access the input values in the `inputs` context. For more information, see "Contexts."

Notes:

- The workflow will also receive the inputs in the `github.event.inputs` context. The information in the `inputs` context and `github.event.inputs` context is identical except that the `inputs` context preserves Boolean values as Booleans instead of converting them to strings. The `choice` type resolves to a string and is a single selectable option.
- The maximum number of top-level properties for `inputs` is 10.
- The maximum payload for `inputs` is 65,535 characters.

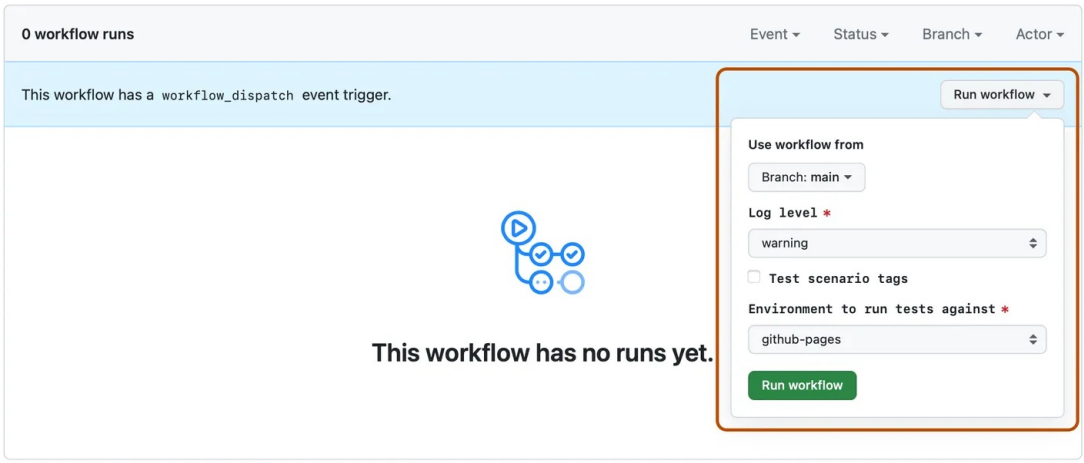
This example defines inputs called `logLevel`, `tags`, and `environment`. You pass values for these inputs to the workflow when you run it. This workflow then prints the values to the log, using the `inputs.logLevel`, `inputs.tags`, and `inputs.environment` context properties.

```yaml
on:
  workflow_dispatch:
    inputs:
      logLevel:
        description: 'Log level'
        required: true
        default: 'warning'
        type: choice
        options:
        - info
        - warning
        - debug
      tags:
        description: 'Test scenario tags'
        required: false
        type: boolean
      environment:
        description: 'Environment to run tests against'
        type: environment
        required: true

jobs:
  log-the-inputs:
    runs-on: ubuntu-latest
    steps:
      - run: |
          echo "Log level: $LEVEL"
          echo "Tags: $TAGS"
          echo "Environment: $ENVIRONMENT"
        env:
          LEVEL: ${{ inputs.logLevel }}
          TAGS: ${{ inputs.tags }}
```

```
          ENVIRONMENT: ${{ inputs.environment }}
```

If you run this workflow from a browser you must enter values for the required inputs manually before the workflow will run.



You can also pass inputs when you run a workflow from a script, or by using GitHub CLI. For example:

```
gh workflow run run-tests.yml -f logLevel=warning -f tags=false -f
environment=staging
```

For more information, see the GitHub CLI information in "Manually running a workflow."

`workflow_run`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`workflow_run`	- `completed` - `requested` - `in_progress`	Last commit on default branch	Default branch

Note: More than one activity type triggers this event. The `requested` activity type does not occur when a workflow is re-run. For information about each activity type, see "Webhook events and payloads." By default, all activity types trigger workflows that run on this event. You can limit your workflow runs to specific activity types using the `types` keyword. For more information, see "Workflow syntax for GitHub Actions."

Note: This event will only trigger a workflow run if the workflow file is on the default branch.

Note: You can't use `workflow_run` to chain together more than three levels of workflows. For example, if you attempt to trigger five workflows (named `B` to `F`) to run sequentially after an initial workflow `A` has run (that is: `A` → `B` → `C` → `D` → `E` → `F`), workflows `E` and `F` will not be run.

This event occurs when a workflow run is requested or completed. It allows you to execute a workflow based on execution or completion of another workflow. The workflow started by the `workflow_run` event is able to access secrets and write tokens, even if the previous workflow was not. This is useful in cases where the previous workflow is intentionally not privileged, but you need to take a privileged action in a later workflow.

In this example, a workflow is configured to run after the separate "Run Tests" workflow

completes.

```yaml
on:
  workflow_run:
    workflows: [Run Tests]
    types:
      - completed
```

If you specify multiple `workflows` for the `workflow_run` event, only one of the workflows needs to run. For example, a workflow with the following trigger will run whenever the "Staging" workflow or the "Lab" workflow completes.

```yaml
on:
  workflow_run:
    workflows: [Staging, Lab]
    types:
      - completed
```

Running a workflow based on the conclusion of another workflow

A workflow run is triggered regardless of the conclusion of the previous workflow. If you want to run a job or step based on the result of the triggering workflow, you can use a conditional with the `github.event.workflow_run.conclusion` property. For example, this workflow will run whenever a workflow named "Build" completes, but the `on-success` job will only run if the "Build" workflow succeeded, and the `on-failure` job will only run if the "Build" workflow failed:

```yaml
on:
  workflow_run:
    workflows: [Build]
    types: [completed]

jobs:
  on-success:
    runs-on: ubuntu-latest
    if: ${{ github.event.workflow_run.conclusion == 'success' }}
    steps:
      - run: echo 'The triggering workflow passed'
  on-failure:
    runs-on: ubuntu-latest
    if: ${{ github.event.workflow_run.conclusion == 'failure' }}
    steps:
      - run: echo 'The triggering workflow failed'
```

Limiting your workflow to run based on branches

You can use the `branches` or `branches-ignore` filter to specify what branches the triggering workflow must run on in order to trigger your workflow. For more information, see "Workflow syntax for GitHub Actions." For example, a workflow with the following trigger will only run when the workflow named `Build` runs on a branch named `canary`.

```yaml
on:
  workflow_run:
    workflows: [Build]
    types: [requested]
    branches: [canary]
```

Using data from the triggering workflow

You can access the `workflow_run` event payload that corresponds to the workflow that triggered your workflow. For example, if your triggering workflow generates artifacts, a workflow triggered with the `workflow_run` event can access these artifacts.

The following workflow uploads data as an artifact. (In this simplified example, the data is the pull request number.)

```
name: Upload data

on:
  pull_request:

jobs:
  upload:
    runs-on: ubuntu-latest

    steps:
      - name: Save PR number
        env:
          PR_NUMBER: ${{ github.event.number }}
        run: |
          mkdir -p ./pr
          echo $PR_NUMBER > ./pr/pr_number
      - uses: actions/upload-artifact@v3
        with:
          name: pr_number
          path: pr/
```

When a run of the above workflow completes, it triggers a run of the following workflow. The following workflow uses the `github.event.workflow_run` context and the GitHub REST API to download the artifact that was uploaded by the above workflow, unzips the downloaded artifact, and comments on the pull request whose number was uploaded as an artifact.

```
name: Use the data

on:
  workflow_run:
    workflows: [Upload data]
    types:
      - completed

jobs:
  download:
    runs-on: ubuntu-latest
    steps:
      - name: 'Download artifact'
        uses: actions/github-script@v6
        with:
          script: |
            let allArtifacts = await
github.rest.actions.listWorkflowRunArtifacts({
               owner: context.repo.owner,
               repo: context.repo.repo,
               run_id: context.payload.workflow_run.id,
            });
            let matchArtifact = allArtifacts.data.artifacts.filter((artifact) =>
{
              return artifact.name == "pr_number"
            })[0];
            let download = await github.rest.actions.downloadArtifact({
               owner: context.repo.owner,
               repo: context.repo.repo,
               artifact_id: matchArtifact.id,
               archive_format: 'zip',
            });
            let fs = require('fs');
```

```
            fs.writeFileSync(`${process.env.GITHUB_WORKSPACE}/pr_number.zip`,
Buffer.from(download.data));

      - name: 'Unzip artifact'
        run: unzip pr_number.zip

      - name: 'Comment on PR'
        uses: actions/github-script@v6
        with:
          github-token: ${{ secrets.GITHUB_TOKEN }}
          script: |
            let fs = require('fs');
            let issue_number = Number(fs.readFileSync('./pr_number'));
            await github.rest.issues.createComment({
              owner: context.repo.owner,
              repo: context.repo.repo,
              issue_number: issue_number,
              body: 'Thank you for the PR!'
            });
```

Legal

© 2023 GitHub, Inc. Terms Privacy Status Pricing Expert services Blog