# GitHub event types

**In this article**



---

For the GitHub Events API, learn about each event type, the triggering action on GitHub, and each event's unique properties.

The Events API can return different types of events triggered by activity on GitHub. Each event response contains shared properties, but has a unique `payload` object determined by its event type. The Event object common properties describes the properties shared by all events, and each event type describes the `payload` properties that are unique to the specific event.

## Event object common properties

The event objects returned from the Events API endpoints have the same structure.

| Event API attribute name | Type | Description |
|---|---|---|
| `id` | `string` | Unique identifier for the event. |
| `type` | `string` | The type of event. Events uses PascalCase for the name. |
| `actor` | `object` | The user that triggered the event. |
| `actor.id` | `string` | The unique identifier for the actor. |

| | | |
|---|---|---|
| `actor.login` | `string` | The username of the actor. |
| `actor.display_login` | `string` | The specific display format of the username. |
| `actor.gravatar_id` | `string` | The unique identifier of the Gravatar profile for the actor. |
| `actor.url` | `string` | The REST API URL used to retrieve the user object, which includes additional user information. |
| `actor.avatar_url` | `string` | The URL of the actor's profile image. |
| `repository` | `object` | The repository object where the event occurred. |
| `repository.id` | `string` | The unique identifier of the repository. |
| `repository.name` | `string` | The name of the repository, which includes the owner and repository name. For example, `octocat/hello-world` is the name of the `hello-world` repository owned by the `octocat` personal account. |
| `repository.url` | `string` | The REST API URL used to retrieve the repository object, which includes additional repository information. |
| `payload` | `object` | The event payload object is unique to the event type. See the event type below for the event API `payload` object. |
| `public` | `boolean` | Whether the event is visible to all users. |
| `created_at` | `string` | The date and time when the event was triggered. It is<br><br>formatted according to ISO 8601. |
| `org` | `object` | The organization that was chosen by the actor to perform action that triggers the event. *The property appears in the event object only if it is applicable.* |
| `org.id` | `string` | The unique identifier for the organization. |
| `org.login` | `string` | The name of the organization. |
| `org.gravatar_id` | `string` | The unique identifier of the Gravatar profile for the organization. |

| | | |
|---|---|---|
| `org.url` | `string` | The REST API URL used to retrieve the organization object, which includes additional organization information. |
| `org.avatar_url` | `string` | The URL of the organization's profile image. |

### Example WatchEvent event object

This example shows the format of the WatchEvent response when using the Events API.

```
HTTP/2 200
Link: <https://api.github.com/resource?page=2>; rel="next",
      <https://api.github.com/resource?page=5>; rel="last"
```


```
[
  {
    "type": "WatchEvent",
    "public": false,
    "payload": {
    },
    "repository": {
      "id": 3,
      "name": "octocat/Hello-World",
      "url": "https://api.github.com/repos/octocat/Hello-World"
    },
    "actor": {
      "id": 1,
      "login": "octocat",
      "gravatar_id": "",
      "avatar_url": "https://github.com/images/error/octocat_happy.gif",
      "url": "https://api.github.com/users/octocat"
    },
    "org": {
      "id": 1,
      "login": "github",
      "gravatar_id": "",
      "url": "https://api.github.com/orgs/github",
      "avatar_url": "https://github.com/images/error/octocat_happy.gif"
    },
    "created_at": "2011-09-06T17:26:27Z",
    "id": "12345"
  }
]
```


## CommitCommentEvent

A commit comment is created. The type of activity is specified in the `action` property of the payload object. For more information, see the "Commit comments" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for CommitCommentEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action performed. Can be `created`. |

| | | |
|---|---|---|
| `comment` | `object` | The commit comment resource. |

## CreateEvent

A Git branch or tag is created. For more information, see the "Git database" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for CreateEvent

| Key | Type | Description |
|---|---|---|
| `ref` | `string` | The `git ref` resource, or `null` if `ref_type` is `repository`. |
| `ref_type` | `string` | The type of Git ref object created in the repository. Can be either `branch`, `tag`, or `repository`. |
| `master_branch` | `string` | The name of the repository's default branch (usually `main`). |
| `description` | `string` | The repository's current description. |
| `pusher_type` | `string` | Can be either `user` or a deploy key. |

## DeleteEvent

A Git branch or tag is deleted. For more information, see the "Git database" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for DeleteEvent

| Key | Type | Description |
|---|---|---|
| `ref` | `string` | The `git ref` resource. |
| `ref_type` | `string` | The type of Git ref object deleted in the repository. Can be either `branch` or `tag`. |

## ForkEvent

A user forks a repository. For more information, see the "Repositories" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for ForkEvent

| Key | Type | Description |
|---|---|---|
| `forkee` | `object` | The created `repository` resource. |

## GollumEvent

A wiki page is created or updated. For more information, see "About wikis."

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for GollumEvent

| Key | Type | Description |
|---|---|---|
| `pages` | `array` | The pages that were updated. |
| `pages[][page_name]` | `string` | The name of the page. |
| `pages[][title]` | `string` | The current page title. |
| `pages[][action]` | `string` | The action that was performed on the page. Can be `created` or `edited`. |
| `pages[][sha]` | `string` | The latest commit SHA of the page. |
| `pages[][html_url]` | `string` | Points to the HTML wiki page. |

## IssueCommentEvent

Activity related to an issue or pull request comment. The type of activity is specified in the `action` property of the payload object. For more information, see the "Issues" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for IssueCommentEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action that was performed on the comment. Can be one of `created`, `edited`, or `deleted`. |
| `changes` | `object` | The changes to the comment if |

| | | |
|---|---|---|
| `changes` | `object` | The changes to the comment if the action was `edited`. |
| `changes[body][from]` | `string` | The previous version of the body if the action was `edited`. |
| `issue` | `object` | The issue the comment belongs to. |
| `comment` | `object` | The comment itself. |

## IssuesEvent

Activity related to an issue. The type of activity is specified in the `action` property of the payload object. For more information, see the "Issues" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for IssuesEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action that was performed. Can be one of `opened`, `edited`, `closed`, `reopened`, `assigned`, `unassigned`, `labeled`, or `unlabeled`. |
| `issue` | `object` | The issue itself. |
| `changes` | `object` | The changes to the issue if the action was `edited`. |
| `changes[title][from]` | `string` | The previous version of the title if the action was `edited`. |
| `changes[body][from]` | `string` | The previous version of the body if the action was `edited`. |
| `assignee` | `object` | The optional user who was assigned or unassigned from the issue. |
| `label` | `object` | The optional label that was added or removed from the issue. |

## MemberEvent

Activity related to repository collaborators. The type of activity is specified in the `action` property of the payload object. For more information, see the "Collaborators" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for MemberEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action that was performed. Can be `added` to indicate a user accepted an invitation to a repository. |
| `member` | `object` | The user that was added. |
| `changes` | `object` | The changes to the collaborator permissions if the action was `edited`. |
| `changes[old_permission][from]` | `string` | The previous permissions of the collaborator if the action was `edited`. |

## PublicEvent

When a private repository is made public. Without a doubt: the best GitHub Enterprise Cloud event.

### Event `payload` object for PublicEvent

This event returns an empty `payload` object.

## PullRequestEvent

Activity related to pull requests. The type of activity is specified in the `action` property of the payload object. For more information, see the "Pulls" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for PullRequestEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action that was performed. Can be one of `opened`, `edited`, `closed`, `reopened`, `assigned`, `unassigned`, `review_requested`, `review_request_removed`, `labeled`, `unlabeled`, and `synchronize`. |
| `number` | `integer` | The pull request number. |
| `changes` | `object` | The changes to the comment if the action was `edited`. |
| `changes[title][from]` | `string` | The previous version of the title if the action was `edited`. |
| `changes[body][from]` | `string` | The previous version of the |

| | | |
|---|---|---|
| | | body if the action was `edited`. |
| `pull_request` | `object` | The pull request itself. |
| `reason` | `string` | The reason the pull request was removed from a merge queue if the action was `dequeued`. |

## PullRequestReviewEvent

Activity related to pull request reviews. The type of activity is specified in the `action` property of the payload object. For more information, see the "Pulls" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for PullRequestReviewEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action that was performed. Can be `created`. |
| `pull_request` | `object` | The pull request the review pertains to. |
| `review` | `object` | The review that was affected. |

## PullRequestReviewCommentEvent

Activity related to pull request review comments in the pull request's unified diff. The type of activity is specified in the `action` property of the payload object. For more information, see the "Pulls" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for PullRequestReviewCommentEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action that was performed on the comment. Can be `created`. |
| `changes` | `object` | The changes to the comment if the action was `edited`. |
| `changes[body][from]` | `string` | The previous version of the body if the action was `edited`. |
| `pull_request` | `object` | The pull request the comment belongs to. |
| `comment` | `object` | The comment itself. |

| comment | object | The comment itself. |
|---|---|---|

## PullRequestReviewThreadEvent

Activity related to a comment thread on a pull request being marked as resolved or unresolved. The type of activity is specified in the `action` property of the payload object.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for PullRequestReviewThreadEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action that was performed. Can be one of:<br>• `resolved` - A comment thread on a pull request was marked as resolved.<br>• `unresolved` - A previously resolved comment thread on a pull request was marked as unresolved. |
| `pull_request` | `object` | The pull request the thread pertains to. |
| `thread` | `object` | The thread that was affected. |

## PushEvent

One or more commits are pushed to a repository branch or tag.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for PushEvent

| Key | Type | Description |
|---|---|---|
| `push_id` | `integer` | Unique identifier for the push. |
| `size` | `integer` | The number of commits in the push. |
| `distinct_size` | `integer` | The number of distinct commits in the push. |
| `ref` | `string` | The full `git ref` that was pushed. Example: `refs/heads/main`. |
| `head` | `string` | The SHA of the most recent commit on `ref` after the push. |
| `before` | `string` | The SHA of the most recent commit on `ref` before the |

| | | |
|---|---|---|
| | | push. |
| `commits` | `array` | An array of commit objects describing the pushed commits. (The array includes a maximum of 20 commits. If necessary, you can use the Commits API to fetch additional commits. This limit is applied to timeline events only and isn't applied to webhook deliveries.) |
| `commits[][sha]` | `string` | The SHA of the commit. |
| `commits[][message]` | `string` | The commit message. |
| `commits[][author]` | `object` | The git author of the commit. |
| `commits[][author][name]` | `string` | The git author's name. |
| `commits[][author][email]` | `string` | The git author's email address. |
| `commits[][url]` | `url` | URL that points to the commit API resource. |
| `commits[][distinct]` | `boolean` | Whether this commit is distinct from any that have been pushed before. |

## ReleaseEvent

Activity related to a release. The type of activity is specified in the `action` property of the payload object. For more information, see the "Releases" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for ReleaseEvent

| Key | Type | Description |
|---|---|---|
| `action` | `string` | The action that was performed. Can be `published`. |
| `changes[body][from]` | `string` | The previous version of the body if the action was `edited`. |
| `changes[name][from]` | `string` | The previous version of the name if the action was `edited`. |
| `release` | `object` | The release object. |

## SponsorshipEvent

Activity related to a sponsorship listing. The type of activity is specified in the `action` property of the payload object. For more information, see "About GitHub Sponsors".

### Event `payload` object for SponsorshipEvent

| Key | Type | Description |
| --- | --- | --- |
| `action` | `string` | The action that was performed. This can be `created`. |
| `effective_date` | `string` | The `pending_cancellation` and `pending_tier_change` event types will include the date the cancellation or tier change will take effect. |
| `changes[tier][from]` | `object` | The `tier_changed` and `pending_tier_change` will include the original tier before the change or pending change. For more information, see the pending tier change payload. |
| `changes[privacy_level][from]` | `string` | The `edited` event types include the details about the change when someone edits a sponsorship to change the privacy. |

## WatchEvent

When someone stars a repository. The type of activity is specified in the `action` property of the payload object. For more information, see the "Activity" REST API.

The event object includes properties that are common for all events. Each event object includes a `payload` property and the value is unique to each event type. The `payload` object for this event is described below.

### Event `payload` object for WatchEvent

| Key | Type | Description |
| --- | --- | --- |
| `action` | `string` | The action that was performed. Currently, can only be `started`. |