

GitHub event types

In this article

- Event object common properties
- CommitCommentEvent
- CreateEvent
- DeleteEvent
- ForkEvent
- GollumEvent
- IssueCommentEvent
- IssuesEvent
- MemberEvent
- PublicEvent
- PullRequestEvent
- PullRequestReviewEvent
- PullRequestReviewCommentEvent
- PullRequestReviewThreadEvent
- PushEvent
- ReleaseEvent
- WatchEvent

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
<code>id</code>	<code>string</code>	Unique identifier for the event.
<code>type</code>	<code>string</code>	The type of event. Events uses PascalCase for the name.
<code>actor</code>	<code>object</code>	The user that triggered the event.
<code>actor.id</code>	<code>string</code>	The unique identifier for the actor.
<code>actor.login</code>	<code>string</code>	The username of the actor.

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

object, which includes additional organization information.

The URL of the organization's profile image.

org.avatar_urlstring

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
<code>ref</code>	<code>string</code>	The git_ref resource, or <code>null</code> if <code>ref_type</code> is <code>repository</code> .
<code>ref_type</code>	<code>string</code>	The type of Git ref object created in the repository. Can be either <code>branch</code> , <code>tag</code> , or <code>repository</code> .
<code>master_branch</code>	<code>string</code>	The name of the repository's default branch (usually <code>main</code>).
<code>description</code>	<code>string</code>	The repository's current description.
<code>pusher_type</code>	<code>string</code>	Can be either <code>user</code> 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
<code>ref</code>	<code>string</code>	The git_ref resource.
<code>ref_type</code>	<code>string</code>	The type of Git ref object deleted in the repository. Can be either <code>branch</code> or <code>tag</code> .

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 <code>created</code> or <code>edited</code> .
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 <code>created</code> , <code>edited</code> , or <code>deleted</code> .
changes	object	The changes to the comment if the action was <code>edited</code> .
changes[body][from]	string	The previous version of the body if the action was <code>edited</code> .

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 <code>opened</code> , <code>edited</code> , <code>closed</code> , <code>reopened</code> , <code>assigned</code> , <code>unassigned</code> , <code>labeled</code> , or <code>unlabeled</code> .
issue	object	The issue itself.
changes	object	The changes to the issue if the action was <code>edited</code> .
changes[title][from]	string	The previous version of the title if the action was <code>edited</code> .
changes[body][from]	string	The previous version of the body if the action was <code>edited</code> .
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 <code>added</code> to indicate a user accepted an invitation to a repository.
<code>member</code>	<code>object</code>	The user that was added.
<code>changes</code>	<code>object</code>	The changes to the collaborator permissions if the action was <code>edited</code> .
<code>changes[old_permission][from]</code>	<code>string</code>	The previous permissions of the collaborator if the action was <code>edited</code> .

PublicEvent [↗](#)

When a private repository is made public. Without a doubt: the best GitHub Enterprise Server 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
<code>action</code>	<code>string</code>	The action that was performed. Can be one of <code>opened</code> , <code>edited</code> , <code>closed</code> , <code>reopened</code> , <code>assigned</code> , <code>unassigned</code> , <code>review_requested</code> , <code>review_request_removed</code> , <code>labeled</code> , <code>unlabeled</code> , and <code>synchronize</code> .
<code>number</code>	<code>integer</code>	The pull request number.
<code>changes</code>	<code>object</code>	The changes to the comment if the action was <code>edited</code> .
<code>changes[title][from]</code>	<code>string</code>	The previous version of the title if the action was <code>edited</code> .
<code>changes[body][from]</code>	<code>string</code>	The previous version of the body if the action was <code>edited</code> .
<code>pull_request</code>	<code>object</code>	The pull request itself.

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
<code>action</code>	<code>string</code>	The action that was performed. Can be <code>created</code> .
<code>pull_request</code>	<code>object</code>	The pull request the review pertains to.
<code>review</code>	<code>object</code>	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
<code>action</code>	<code>string</code>	The action that was performed on the comment. Can be <code>created</code> .
<code>changes</code>	<code>object</code>	The changes to the comment if the action was <code>edited</code> .
<code>changes[body][from]</code>	<code>string</code>	The previous version of the body if the action was <code>edited</code> .
<code>pull_request</code>	<code>object</code>	The pull request the comment belongs to.
<code>comment</code>	<code>object</code>	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
<code>action</code>	<code>string</code>	The action that was performed. Can be one of: <ul style="list-style-type: none"><code>resolved</code> - A comment thread on a pull request was marked as resolved.<code>unresolved</code> - A previously resolved comment thread on a pull request was marked as unresolved.
<code>pull_request</code>	<code>object</code>	The pull request the thread pertains to.
<code>thread</code>	<code>object</code>	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
<code>push_id</code>	<code>integer</code>	Unique identifier for the push.
<code>size</code>	<code>integer</code>	The number of commits in the push.
<code>distinct_size</code>	<code>integer</code>	The number of distinct commits in the push.
<code>ref</code>	<code>string</code>	The full git ref that was pushed. Example: <code>refs/heads/main</code> .
<code>head</code>	<code>string</code>	The SHA of the most recent commit on <code>ref</code> after the push.
<code>before</code>	<code>string</code>	The SHA of the most recent commit on <code>ref</code> before the push.
<code>commits</code>	<code>array</code>	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 <code>published</code> .
changes[body][from]	string	The previous version of the body if the action was <code>edited</code> .
changes[name][from]	string	The previous version of the name if the action was <code>edited</code> .
release	object	The release object.

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 .

Legal