The REST API is now versioned. For more information, see "About API versioning."

Commit statuses

Use the REST API to interact with commit statuses.

About commit statuses

You can use the REST API to allow external services to mark commits with an `error`, `failure`, `pending`, or `success` state, which is then reflected in pull requests involving those commits. Statuses can also include an optional `description` and `target_url`, and we highly recommend providing them as they make statuses much more useful in the GitHub UI.

As an example, one common use is for continuous integration services to mark commits as passing or failing builds using status. The `target_url` would be the full URL to the build output, and the `description` would be the high level summary of what happened with the build.

Statuses can include a `context` to indicate what service is providing that status. For example, you may have your continuous integration service push statuses with a context of `ci`, and a security audit tool push statuses with a context of `security`. You can then use the REST API to Get the combined status for a specific reference to retrieve the whole status for a commit.

Note that the `repo:status` OAuth scope grants targeted access to statuses **without** also granting access to repository code, while the `repo` scope grants permission to code as well as statuses.

If you are developing a GitHub App and want to provide more detailed information about an external service, you may want to use the REST API to manage checks. For more information, see "Checks."

Get the combined status for a specific reference

Works with GitHub Apps

Users with pull access in a repository can access a combined view of commit statuses for a given ref. The ref can be a SHA, a branch name, or a tag name.

Additionally, a combined `state` is returned. The `state` is one of:

- **failure** if any of the contexts report as `error` or `failure`
- **pending** if there are no statuses or a context is `pending`
- **success** if the latest status for all contexts is `success`

Parameters for "Get the combined status for a specific reference"

Headers

`accept` string

Setting to `application/vnd.github+json` is recommended.

Path parameters

`owner` string Required

The account owner of the repository. The name is not case sensitive.

`repo` string Required

The name of the repository without the `.git` extension. The name is not case sensitive.

`ref` string Required

The commit reference. Can be a commit SHA, branch name (`heads/BRANCH_NAME`), or tag name (`tags/TAG_NAME`). For more information, see "Git References" in the Git documentation.

Query parameters

`per_page` integer

The number of results per page (max 100).

Default: `30`

`page` integer

Page number of the results to fetch.

Default: `1`

HTTP response status codes for "Get the combined status for a specific reference"

Status code	Description
`200`	OK
`404`	Resource not found

Code samples for "Get the combined status for a specific reference"

`GET` `/repos/{owner}/{repo}/commits/{ref}/status`

cURL JavaScript GitHub CLI

```
curl -L \ -H "Accept: application/vnd.github+json" \ -H "Authorization: Bearer <YOUR-TOKEN>" \ -H "X-GitHub-Api-Version: 2022-11-28" \ http(s)://HOSTNAME/api/v3/repos/OWNER/REPO/commits/REF/status
```

Response

Example response Response schema

```
Status: 200
```


```
{ "state": "success", "statuses": [ { "url": "https://HOSTNAME/repos/octocat/Hello-World/statuses/6dcb09b5b57875f334f61aebed695e2e4193db5e", "avatar_url": "https://github.com/images/error/hubot_happy.gif", "id": 1, "node_id": "MDY6U3RhdHVzMQ==", "state": "success", "description": "Build has completed successfully", "target_url": "https://ci.example.com/1000/output", "context": "continuous-integration/jenkins", "created_at": "2012-07-20T01:19:13Z", "updated_at": "2012-07-20T01:19:13Z" }, { "url": "https://HOSTNAME/repos/octocat/Hello-World/statuses/6dcb09b5b57875f334f61aebed695e2e4193db5e", "avatar_url": "https://github.com/images/error/other_user_happy.gif",
```

List commit statuses for a reference

Works with GitHub Apps

Users with pull access in a repository can view commit statuses for a given ref. The ref can be a SHA, a branch name, or a tag name. Statuses are returned in reverse chronological order. The first status in the list will be the latest one.

This resource is also available via a legacy route: `GET /repos/:owner/:repo/statuses/:ref`.

Parameters for "List commit statuses for a reference"

Headers

`accept` string

Setting to `application/vnd.github+json` is recommended.

Path parameters

`owner` string Required

The account owner of the repository. The name is not case sensitive.

`repo` string Required

The name of the repository without the `.git` extension. The name is not case sensitive.

`ref` string Required

The commit reference. Can be a commit SHA, branch name (`heads/BRANCH_NAME`), or tag name (`tags/TAG_NAME`). For more information, see "Git References" in the Git documentation.

Query parameters

`per_page` integer

The number of results per page (max 100).

Default: `30`

`page` integer

Page number of the results to fetch.

Default: `1`

HTTP response status codes for "List commit statuses for a reference"

Status code	Description
`200`	OK
`301`	Moved permanently

Code samples for "List commit statuses for a reference"

`GET` `/repos/{owner}/{repo}/commits/{ref}/statuses`

cURL | JavaScript | GitHub CLI

```
curl -L \ -H "Accept: application/vnd.github+json" \ -H "Authorization: Bearer <YOUR-TOKEN>" \ -H "X-GitHub-Api-Version: 2022-11-28" \ http(s)://HOSTNAME/api/v3/repos/OWNER/REPO/commits/REF/statuses
```

Response

Example response | Response schema

```
Status: 200
```


```
[ { "url": "https://HOSTNAME/repos/octocat/Hello-World/statuses/6dcb09b5b57875f334f61aebed695e2e4193db5e", "avatar_url": "https://github.com/images/error/hubot_happy.gif", "id": 1, "node_id": "MDY6U3RhdHVzMQ==", "state": "success", "description": "Build has completed successfully", "target_url": "https://ci.example.com/1000/output", "context": "continuous-integration/jenkins", "created_at": "2012-07-20T01:19:13Z", "updated_at": "2012-07-20T01:19:13Z", "creator": { "login": "octocat", "id": 1, "node_id": "MDQ6VXNlcjE=", "avatar_url": "https://github.com/images/error/octocat_happy.gif", "gravatar_id": "", "url": "https://HOSTNAME/users/octocat", "html_url": "https://github.com/octocat", "followers_url":
```


Create a commit status

Works with GitHub Apps

Users with push access in a repository can create commit statuses for a given SHA.

Note: there is a limit of 1000 statuses per `sha` and `context` within a repository. Attempts to create more than 1000 statuses will result in a validation error.

Parameters for "Create a commit status"

Headers

`accept` string

Setting to `application/vnd.github+json` is recommended.

Path parameters

`owner` string Required

The account owner of the repository. The name is not case sensitive.

`repo` string Required

The name of the repository without the `.git` extension. The name is not case sensitive.

`sha` string Required

Body parameters

`state` string Required

The state of the status.

Can be one of: `error`, `failure`, `pending`, `success`

`target_url` string or null

The target URL to associate with this status. This URL will be linked from the GitHub UI to allow users to easily see the source of the status.
For example, if your continuous integration system is posting build status, you would want to provide the deep link for the build output for this specific SHA:
`http://ci.example.com/user/repo/build/sha`

`description` string or null

A short description of the status.

`context` string

A string label to differentiate this status from the status of other systems. This field is case-insensitive.

Default: `default`

HTTP response status codes for "Create a commit status"

Status code	Description
`201`	Created

Code samples for "Create a commit status"

`POST` `/repos/{owner}/{repo}/statuses/{sha}`

cURL | JavaScript | GitHub CLI

```
curl -L \ -X POST \ -H "Accept: application/vnd.github+json" \ -H "Authorization: Bearer <YOUR-TOKEN>" \ -H "X-GitHub-Api-Version: 2022-11-28" \ http(s)://HOSTNAME/api/v3/repos/OWNER/REPO/statuses/SHA \ -d '{"state":"success","target_url":"https://example.com/build/status","description":"The build succeeded!","context":"continuous-integration/jenkins"}'
```

Response

Example response | Response schema

```
Status: 201
```


```
{ "url": "https://HOSTNAME/repos/octocat/Hello-World/statuses/6dcb09b5b57875f334f61aebed695e2e4193db5e", "avatar_url": "https://github.com/images/error/hubot_happy.gif", "id": 1, "node_id": "MDY6U3RhdHVzMQ==", "state": "success", "description": "Build has completed successfully", "target_url": "https://ci.example.com/1000/output", "context": "continuous-integration/jenkins", "created_at": "2012-07-20T01:19:13Z", "updated_at": "2012-07-20T01:19:13Z", "creator": { "login": "octocat", "id": 1, "node_id": "MDQ6VXNlcjE=", "avatar_url": "https://github.com/images/error/octocat_happy.gif", "gravatar_id": "", "url": "https://HOSTNAME/users/octocat", "html_url": "https://github.com/octocat", "followers_url":
```