

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

Commits

Use the REST API to interact with commits.

List commits [↗](#)

✔ Works with [GitHub Apps](#)

Signature verification object

The response will include a `verification` object that describes the result of verifying the commit's signature. The following fields are included in the `verification` object:

Name	Type	Description
<code>verified</code>	<code>boolean</code>	Indicates whether GitHub considers the signature in this commit to be verified.
<code>reason</code>	<code>string</code>	The reason for verified value. Possible values and their meanings are enumerated in table below.
<code>signature</code>	<code>string</code>	The signature that was extracted from the commit.
<code>payload</code>	<code>string</code>	The value that was signed.

These are the possible values for `reason` in the `verification` object:

Value	Description
<code>expired_key</code>	The key that made the signature is expired.
<code>not_signing_key</code>	The "signing" flag is not among the usage flags in the GPG key that made the signature.
<code>gpgverify_error</code>	There was an error communicating with the signature verification service.
<code>gpgverify_unavailable</code>	The signature verification service is currently unavailable.
<code>unsigned</code>	The object does not include a signature.
<code>unknown_signature_type</code>	A non-PGP signature was found in the commit.
<code>no_user</code>	No user was associated with the <code>committer</code> email address in the commit.
<code>unverified_email</code>	The <code>committer</code> email address in the commit was associated with a user, but the email address is not verified on their account.

<code>bad_email</code>	The <code>committer</code> email address in the commit is not included in the identities of the PGP key that made the signature.
<code>unknown_key</code>	The key that made the signature has not been registered with any user's account.
<code>malformed_signature</code>	There was an error parsing the signature.
<code>invalid</code>	The signature could not be cryptographically verified using the key whose key-id was found in the signature.
<code>valid</code>	None of the above errors applied, so the signature is considered to be verified.

Parameters for "List commits"

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.

Query parameters

`sha` string

SHA or branch to start listing commits from. Default: the repository's default branch (usually `main`).

`path` string

Only commits containing this file path will be returned.

`author` string

GitHub username or email address to use to filter by commit author.

`committer` string

GitHub username or email address to use to filter by commit committer.

`since` string

Only show results that were last updated after the given time. This is a timestamp in [ISO 8601](#) format: `YYYY-MM-DDTHH:MM:SSZ`.

`until` string

Only commits before this date will be returned. This is a timestamp in [ISO 8601](#) format: `YYYY-MM-DDTHH:MM:SSZ`.

`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 commits"

Status code	Description
200	OK
400	Bad Request
404	Resource not found
409	Conflict
500	Internal Error

Code samples for "List commits"

GET

/repos/{owner}/{repo}/commits

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
```

Response

Example response

Response schema

Status: 200

```
[ { "url": "https://HOSTNAME/repos/octocat/Hello-World/commits/6dcb09b5b57875f334f61aebd695e2e4193db5e", "sha": "6dcb09b5b57875f334f61aebd695e2e4193db5e", "node_id": "MDY6Q29tbWl0NmRjYjA5YjViNTc4NzVmMzM0ZjYxYWViZWQ2OTVlMmU0MTkzZGI1ZQ==", "html_url": "https://github.com/octocat/Hello-World/commit/6dcb09b5b57875f334f61aebd695e2e4193db5e", "comments_url": "https://HOSTNAME/repos/octocat/Hello-World/commits/6dcb09b5b57875f334f61aebd695e2e4193db5e/comments", "commit": { "url": "https://HOSTNAME/repos/octocat/Hello-World/git/commits/6dcb09b5b57875f334f61aebd695e2e4193db5e", "author": { "name": "Monalisa Octocat", "email": "support@github.com", "date": "2011-04-14T16:00:49Z" }, "committer": { "name": "Monalisa Octocat", "email":
```

List branches for HEAD commit

Works with [GitHub Apps](#)

Protected branches are available in public repositories with GitHub Free and GitHub Free for organizations, and in public and private repositories with GitHub Pro, GitHub Team, GitHub Enterprise Cloud, and GitHub Enterprise Server. For more information, see [GitHub's products](#) in the GitHub Help documentation.

Returns all branches where the given commit SHA is the HEAD, or latest commit for the branch.

Parameters for "List branches for HEAD commit"

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.

commit_sha string **Required**
The SHA of the commit.

HTTP response status codes for "List branches for HEAD commit"

Status code	Description
200	OK
422	Validation failed, or the endpoint has been spammed.

Code samples for "List branches for HEAD commit"

GET

/repos/{owner}/{repo}/commits/{commit_sha}/branches-where-head

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/COMMIT_SHA/branches-where-head
```

Response

Example response

Response schema

Status: 200

```
[ { "name": "branch_5", "commit": { "sha": "c5b97d5ae6c19d5c5df71a34c7fbeeda2479ccbc", "url": "https://HOSTNAME/repos/octocat/Hello-World/commits/c5b97d5ae6c19d5c5df71a34c7fbeeda2479ccbc" }, "protected": false } ]
```

List pull requests associated with a commit

✔ Works with [GitHub Apps](#)

Lists the merged pull request that introduced the commit to the repository. If the commit is not present in the default branch, will only return open pull requests associated with the commit.

To list the open or merged pull requests associated with a branch, you can set the `commit_sha` parameter to the branch name.

Parameters for "List pull requests associated with a commit"

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.

commit_sha string **Required**
The SHA of the commit.

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 pull requests associated with a commit"

Status code	Description
<code>200</code>	OK

Code samples for "List pull requests associated with a commit"


GET

/repos/{owner}/{repo}/commits/{commit_sha}/pulls

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/COMMIT_SHA/pulls
```

Response

Example response Response schema

Status: 200

[{ "url": "https://HOSTNAME/repos/octocat/Hello-World/pulls/1347", "id": 1, "node_id": "MDExO1B1bGxSZXF1ZXN0MQ==", "html_url": "https://github.com/octocat/Hello-World/pull/1347", "diff_url": "https://github.com/octocat/Hello-World/pull/1347.diff", "patch_url": "https://github.com/octocat/Hello-World/pull/1347.patch", "issue_url": "https://HOSTNAME/repos/octocat/Hello-World/issues/1347", "commits_url": "https://HOSTNAME/repos/octocat/Hello-World/pulls/1347/commits", "review_comments_url": "https://HOSTNAME/repos/octocat/Hello-World/pulls/1347/comments", "review_comment_url": "https://HOSTNAME/repos/octocat/Hello-World/pulls/comments{/number}", "comments_url": "https://HOSTNAME/repos/octocat/Hello-World/issues/1347/comments", "statuses_url":

Get a commit

✔ Works with [GitHub Apps](#)

Returns the contents of a single commit reference. You must have `read` access for the repository to use this endpoint.

Note: If there are more than 300 files in the commit diff, the response will include pagination link headers for the remaining files, up to a limit of 3000 files. Each page contains the static commit information, and the only changes are to the file listing.

You can pass the appropriate [media type](#) to fetch `diff` and `patch` formats. Diffs with binary data will have no `patch` property.

To return only the SHA-1 hash of the commit reference, you can provide the `sha` custom [media type](#) in the `Accept` header. You can use this endpoint to check if a remote reference's SHA-1 hash is the same as your local reference's SHA-1 hash by providing the local SHA-1 reference as the ETag.

Signature verification object

The response will include a `verification` object that describes the result of verifying the commit's signature. The following fields are included in the `verification` object:

Name	Type	Description
<code>verified</code>	<code>boolean</code>	Indicates whether GitHub considers the signature in this commit to be verified.
<code>reason</code>	<code>string</code>	The reason for verified value. Possible values and their meanings are enumerated in table below.
<code>signature</code>	<code>string</code>	The signature that was extracted from the commit.
<code>payload</code>	<code>string</code>	The value that was signed.

These are the possible values for `reason` in the `verification` object:

Value	Description
-------	-------------

<code>expired_key</code>	The key that made the signature is expired.
<code>not_signing_key</code>	The "signing" flag is not among the usage flags in the GPG key that made the signature.
<code>gpgverify_error</code>	There was an error communicating with the signature verification service.
<code>gpgverify_unavailable</code>	The signature verification service is currently unavailable.
<code>unsigned</code>	The object does not include a signature.
<code>unknown_signature_type</code>	A non-PGP signature was found in the commit.
<code>no_user</code>	No user was associated with the <code>committer</code> email address in the commit.
<code>unverified_email</code>	The <code>committer</code> email address in the commit was associated with a user, but the email address is not verified on their account.
<code>bad_email</code>	The <code>committer</code> email address in the commit is not included in the identities of the PGP key that made the signature.
<code>unknown_key</code>	The key that made the signature has not been registered with any user's account.
<code>malformed_signature</code>	There was an error parsing the signature.
<code>invalid</code>	The signature could not be cryptographically verified using the key whose key-id was found in the signature.
<code>valid</code>	None of the above errors applied, so the signature is considered to be verified.

Parameters for "Get a commit"

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

`page` integer

Page number of the results to fetch.

Default: 1

per_page integer

The number of results per page (max 100).

Default: 30

HTTP response status codes for "Get a commit"

Status code	Description
200	OK
404	Resource not found
422	Validation failed, or the endpoint has been spammed.
500	Internal Error
503	Service unavailable

Code samples for "Get a commit"

GET

/repos/{owner}/{repo}/commits/{ref}

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
```

Response

Example response

Response schema

Status: 200

```
{ "url": "https://HOSTNAME/repos/octocat/Hello-World/commits/6dcb09b5b57875f334f61aebd695e2e4193db5e", "sha": "6dcb09b5b57875f334f61aebd695e2e4193db5e", "node_id": "MDY6Q29tbWl0NmRjYjA5YjViNTc4NzVmMzM0ZjYxYWViZWQ20TVlMmU0MTkzZGI1ZQ==", "html_url": "https://github.com/octocat/Hello-World/commit/6dcb09b5b57875f334f61aebd695e2e4193db5e", "comments_url": "https://HOSTNAME/repos/octocat/Hello-World/commits/6dcb09b5b57875f334f61aebd695e2e4193db5e/comments", "commit": { "url": "https://HOSTNAME/repos/octocat/Hello-World/git/commits/6dcb09b5b57875f334f61aebd695e2e4193db5e", "author": { "name": "Monalisa Octocat", "email": "mona@github.com", "date": "2011-04-14T16:00:49Z" }, "committer": { "name": "Monalisa Octocat", "email":
```

Compare two commits

Works with [GitHub Apps](#)

Compares two commits against one another. You can compare branches in the same repository, or you can compare branches that exist in different repositories within the same repository network, including fork branches. For more information about how to view a repository's network, see "[Understanding connections between repositories](#)."

This endpoint is equivalent to running the `git log BASE..HEAD` command, but it returns commits in a different order. The `git log BASE..HEAD` command returns commits in reverse chronological order, whereas the API returns commits in chronological order. You can pass the appropriate [media type](#) to fetch diff and patch formats.

The API response includes details about the files that were changed between the two commits. This includes the status of the change (if a file was added, removed, modified, or renamed), and details of the change itself. For example, files with a `renamed` status have a `previous_filename` field showing the previous filename of the file, and files with a `modified` status have a `patch` field showing the changes made to the file.

When calling this endpoint without any paging parameter (`per_page` or `page`), the returned list is limited to 250 commits, and the last commit in the list is the most recent of the entire comparison.

Working with large comparisons

To process a response with a large number of commits, use a query parameter (`per_page` or `page`) to paginate the results. When using pagination:

- The list of changed files is only shown on the first page of results, but it includes all changed files for the entire comparison.
- The results are returned in chronological order, but the last commit in the returned list may not be the most recent one in the entire set if there are more pages of results.

For more information on working with pagination, see "[Using pagination in the REST API](#)."

Signature verification object

The response will include a `verification` object that describes the result of verifying the commit's signature. The `verification` object includes the following fields:

Name	Type	Description
<code>verified</code>	<code>boolean</code>	Indicates whether GitHub considers the signature in this commit to be verified.
<code>reason</code>	<code>string</code>	The reason for verified value. Possible values and their meanings are enumerated in table below.
<code>signature</code>	<code>string</code>	The signature that was extracted from the commit.
<code>payload</code>	<code>string</code>	The value that was signed.

These are the possible values for `reason` in the `verification` object:

Value	Description
<code>expired_key</code>	The key that made the signature is expired.
<code>not_signing_key</code>	The "signing" flag is not among the usage flags in the GPG key that made the signature.
<code>gpgverify_error</code>	There was an error communicating with the signature verification service.
<code>gpgverify_unavailable</code>	The signature verification service is currently unavailable.
<code>unsigned</code>	The object does not include a signature.
<code>unknown_signature_type</code>	A non-PGP signature was found in the commit.
<code>no_user</code>	No user was associated with the <code>committer</code> email address in the commit.

unverified_email	The <code>committer</code> email address in the commit was associated with a user, but the email address is not verified on their account.
bad_email	The <code>committer</code> email address in the commit is not included in the identities of the PGP key that made the signature.
unknown_key	The key that made the signature has not been registered with any user's account.
malformed_signature	There was an error parsing the signature.
invalid	The signature could not be cryptographically verified using the key whose key-id was found in the signature.
valid	None of the above errors applied, so the signature is considered to be verified.

Parameters for "Compare two commits"

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.

`basehead` string Required
The base branch and head branch to compare. This parameter expects the format `BASE...HEAD`. Both must be branch names in `repo`. To compare with a branch that exists in a different repository in the same network as `repo`, the `basehead` parameter expects the format `USERNAME:BASE...USERNAME:HEAD`.

Query parameters

`page` integer
Page number of the results to fetch.
Default: `1`

`per_page` integer
The number of results per page (max 100).
Default: `30`

HTTP response status codes for "Compare two commits"

Status code	Description
200	OK

404	Resource not found
500	Internal Error
503	Service unavailable

Code samples for "Compare two commits"

GET

/repos/{owner}/{repo}/compare/{basehead}

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/compare/BASEHEAD

Response

Example response

Response schema

Status: 200

{ "url": "https://HOSTNAME/repos/octocat/Hello-World/compare/master...topic", "html_url": "https://github.com/octocat/Hello-World/compare/master...topic", "permalink_url": "https://github.com/octocat/Hello-World/compare/octocat:bbcd538c8e72b8c175046e27cc8f907076331401...octocat:0328041d1152db8ae77652d1618a02e57f745f17", "diff_url": "https://github.com/octocat/Hello-World/compare/master...topic.diff", "patch_url": "https://github.com/octocat/Hello-World/compare/master...topic.patch", "base_commit": { "url": "https://HOSTNAME/repos/octocat/Hello-World/commits/6dcb09b5b57875f334f61aebd695e2e4193db5e", "sha": "6dcb09b5b57875f334f61aebd695e2e4193db5e", "node_id":

Legal