gh api

```
gh api <endpoint> [flags]
```

Makes an authenticated HTTP request to the GitHub API and prints the response.

The endpoint argument should either be a path of a GitHub API v3 endpoint, or `graphql` to access the GitHub API v4.

Placeholder values `{owner}`, `{repo}`, and `{branch}` in the endpoint argument will get replaced with values from the repository of the current directory or the repository specified in the `GH_REPO` environment variable. Note that in some shells, for example PowerShell, you may need to enclose any value that contains `{...}` in quotes to prevent the shell from applying special meaning to curly braces.

The default HTTP request method is `GET` normally and `POST` if any parameters were added. Override the method with `--method`.

Pass one or more `-f/--raw-field` values in `key=value` format to add static string parameters to the request payload. To add non-string or placeholder-determined values, see `-F/--field` below. Note that adding request parameters will automatically switch the request method to `POST`. To send the parameters as a `GET` query string instead, use `--method GET`.

The `-F/--field` flag has magic type conversion based on the format of the value:

- literal values `true`, `false`, `null`, and integer numbers get converted to appropriate JSON types;
- placeholder values `{owner}`, `{repo}`, and `{branch}` get populated with values from the repository of the current directory;
- if the value starts with `@`, the rest of the value is interpreted as a filename to read the value from. Pass `-` to read from standard input.

For GraphQL requests, all fields other than `query` and `operationName` are interpreted as GraphQL variables.

To pass nested parameters in the request payload, use `key[subkey]=value` syntax when declaring fields. To pass nested values as arrays, declare multiple fields with the syntax `key[]=value1`, `key[]=value2`. To pass an empty array, use `key[]` without a value.

To pass pre-constructed JSON or payloads in other formats, a request body may be read from file specified by `--input`. Use `-` to read from standard input. When passing the request body this way, any parameters specified via field flags are added to the query string of the endpoint URL.

In `--paginate` mode, all pages of results will sequentially be requested until there are no more pages of results. For GraphQL requests, this requires that the original query accepts an `$endCursor: String` variable and that it fetches the `pageInfo{ hasNextPage, endCursor }` set of fields from a collection. Each page is a separate JSON array or object. Pass `--slurp` to wrap all pages of JSON arrays or objects into an outer JSON array.

Options

`--cache <duration>`
Cache the response, e.g. "3600s", "60m", "1h"

`-F`, `--field <key=value>`
Add a typed parameter in key=value format

`-H`, `--header <key:value>`
Add a HTTP request header in key:value format

`--hostname <string>`
The GitHub hostname for the request (default "github.com")

`-i`, `--include`
Include HTTP response status line and headers in the output

`--input <file>`
The file to use as body for the HTTP request (use "-" to read from standard input)

`-q`, `--jq <string>`
Query to select values from the response using jq syntax

`-X`, `--method <string> (default "GET")`
The HTTP method for the request

`--paginate`
Make additional HTTP requests to fetch all pages of results

`-p`, `--preview <names>`
GitHub API preview names to request (without the "-preview" suffix)

`-f`, `--raw-field <key=value>`

Add a string parameter in key=value format

`--silent`
Do not print the response body

`--slurp`
Use with "--paginate" to return an array of all pages of either JSON arrays or objects

`-t`, `--template <string>`
Format JSON output using a Go template; see "gh help formatting"

`--verbose`
Include full HTTP request and response in the output

Examples

```
# list releases in the current repository
$ gh api repos/{owner}/{repo}/releases

# post an issue comment
$ gh api repos/{owner}/{repo}/issues/123/comments -f body='Hi from CLI'

# post nested parameter read from a file
$ gh api gists -F 'files[myfile.txt][content]=@myfile.txt'

# add parameters to a GET request
$ gh api -X GET search/issues -f q='repo:cli/cli is:open remote'

# set a custom HTTP header
$ gh api -H 'Accept: application/vnd.github.v3.raw+json' ...

# opt into GitHub API previews
$ gh api --preview baptiste,nebula ...

# print only specific fields from the response
$ gh api repos/{owner}/{repo}/issues --jq '.[].title'

# use a template for the output
$ gh api repos/{owner}/{repo}/issues --template \
  '{{range .}}{{.title}} ({{.labels | pluck "name" | join ", " | color "yellow"}})

# update allowed values of the "environment" custom property in a deeply nested ar
gh api -X PATCH /orgs/{org}/properties/schema \
   -F 'properties[][property_name]=environment' \
   -F 'properties[][default_value]=production' \
   -F 'properties[][allowed_values][]=staging' \
   -F 'properties[][allowed_values][]=production'

# list releases with GraphQL
$ gh api graphql -F owner='{owner}' -F name='{repo}' -f query='
```

```
  query($name: String!, $owner: String!) {
    repository(owner: $owner, name: $name) {
      releases(last: 3) {
        nodes { tagName }
      }
    }
  }
'

# list all repositories for a user
$ gh api graphql --paginate -f query='
  query($endCursor: String) {
    viewer {
      repositories(first: 100, after: $endCursor) {
        nodes { nameWithOwner }
        pageInfo {
          hasNextPage
          endCursor
        }
      }
    }
  }
'

# get the percentage of forks for the current user
$ gh api graphql --paginate --slurp -f query='
  query($endCursor: String) {
    viewer {
      repositories(first: 100, after: $endCursor) {
        nodes { isFork }
        pageInfo {
          hasNextPage
          endCursor
        }
      }
    }
  }
' | jq 'def count(e): reduce e as $_ (0;.+1);
[.[].data.viewer.repositories.nodes[]] as $r | count(select($r[].isFork))/count($r
```

See also

- gh

GitHub

Product

Features

Security

Enterprise

Customer stories

Pricing

Resources

Platform

Developer API

Partners

GitHub Desktop

GitHub Mobile

Support

Help

Community Forum

Expert Services

Status

Contact GitHub

Company

About

Blog

Careers

Press

Shop