gh reference

gh reference

`gh alias <command>`
Create command shortcuts

`gh alias delete {<alias> &#124; --all} [flags]`
Delete set aliases

```
--all   Delete all aliases
```
`gh alias import [<filename> &#124; -] [flags]`
Import aliases from a YAML file

```
--clobber   Overwrite existing aliases of the same name
```
`gh alias list`
List your aliases

Aliases

gh alias ls

`gh alias set <alias> <expansion> [flags]`
Create a shortcut for a gh command

```
    --clobber   Overwrite existing aliases of the same name
-s, --shell     Declare an alias to be passed through a shell interpreter
```
`gh api <endpoint> [flags]`
Make an authenticated GitHub API request

```
    --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         The HTTP method for the request (default "GET")
    --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
```

`gh attestation [subcommand]`

Work with artifact attestations

Aliases

gh at

`gh attestation download [<file-path> &#124; oci://<image-uri>] [--owner &#124; --repo] [flags]`

Download an artifact's attestations for offline use

```
-d, --digest-alg string       The algorithm used to compute a digest of the
artifact: {sha256&#124;sha512} (default "sha256")
    --hostname string         Configure host to use
-L, --limit int               Maximum number of attestations to fetch
(default 30)
-o, --owner string            a GitHub organization to scope attestation
lookup by
    --predicate-type string   Filter attestations by provided predicate
type
-R, --repo string             Repository name in the format <owner>/<repo>
```

`gh attestation trusted-root [--tuf-url <url> --tuf-root <file-path>] [--verify-only] [flags]`

Output trusted_root.jsonl contents, likely for offline verification

```
--hostname string   Configure host to use
--tuf-root string   Path to the TUF root.json file on disk
--tuf-url string    URL to the TUF repository mirror
--verify-only       Don't output trusted_root.jsonl contents
```

`gh attestation verify [<file-path> &#124; oci://<image-uri>] [--owner &#124; --repo] [flags]`

Verify an artifact's integrity using attestations

```
-b, --bundle string                Path to bundle on disk, either a single
bundle in a JSON file or a JSON lines file with multiple bundles
    --bundle-from-oci              When verifying an OCI image, fetch the
attestation bundle from the OCI registry instead of from GitHub
    --cert-identity string         Enforce that the certificate's subject
alternative name matches the provided value exactly
-i, --cert-identity-regex string   Enforce that the certificate's subject
alternative name matches the provided regex
    --cert-oidc-issuer string      Issuer of the OIDC token (default
"https://token.actions.githubusercontent.com")
    --custom-trusted-root string   Path to a trusted_root.jsonl file;
likely for offline verification
    --deny-self-hosted-runners     Fail verification for attestations
generated on self-hosted runners
-d, --digest-alg string            The algorithm used to compute a digest
of the artifact: {sha256&#124;sha512} (default "sha256")
    --format string                Output format: {json}
    --hostname string              Configure host to use
-q, --jq expression                Filter JSON output using a jq expression
-L, --limit int                    Maximum number of attestations to fetch
(default 30)
    --no-public-good               Do not verify attestations signed with
Sigstore public good instance
```

```
-o, --owner string                 GitHub organization to scope attestation
lookup by
    --predicate-type string        Filter attestations by provided
predicate type (default "https://slsa.dev/provenance/v1")
-R, --repo string                  Repository name in the format
<owner>/<repo>
    --signer-repo string           Repository of reusable workflow that
signed attestation in the format <owner>/<repo>
    --signer-workflow string       Workflow that signed attestation in the
format [host/]<owner>/<repo>/<path>/<to>/<workflow>
-t, --template string              Format JSON output using a Go template;
see "gh help formatting"
```

`gh auth <command>`

Authenticate gh and git with GitHub

`gh auth login [flags]`

Log in to a GitHub account

```
-p, --git-protocol string   The protocol to use for git operations on this
host: {ssh&#124;https}
-h, --hostname string       The hostname of the GitHub instance to
authenticate with
    --insecure-storage      Save authentication credentials in plain text
instead of credential store
-s, --scopes strings        Additional authentication scopes to request
    --skip-ssh-key          Skip generate/upload SSH key prompt
-w, --web                   Open a browser to authenticate
    --with-token            Read token from standard input
```

`gh auth logout [flags]`

Log out of a GitHub account

```
-h, --hostname string   The hostname of the GitHub instance to log out of
-u, --user string       The account to log out of
```

`gh auth refresh [flags]`

Refresh stored authentication credentials

```
-h, --hostname string         The GitHub host to use for authentication
    --insecure-storage        Save authentication credentials in plain text
instead of credential store
-r, --remove-scopes strings   Authentication scopes to remove from gh
    --reset-scopes            Reset authentication scopes to the default
minimum set of scopes
-s, --scopes strings          Additional authentication scopes for gh to
have
```

`gh auth setup-git [flags]`

Setup git with GitHub CLI

```
-f, --force --hostname    Force setup even if the host is not known. Must be
used in conjunction with --hostname
-h, --hostname string    The hostname to configure git for
```

`gh auth status [flags]`

Display active account and authentication state on each known GitHub host

```
-a, --active            Display the active account only
-h, --hostname string   Check only a specific hostname's auth status
-t, --show-token        Display the auth token
```

`gh auth switch [flags]`

Switch active GitHub account

```
-h, --hostname string   The hostname of the GitHub instance to switch
account for
-u, --user string       The account to switch to
```
`gh auth token [flags]`

Print the authentication token gh uses for a hostname and account

```
-h, --hostname string   The hostname of the GitHub instance authenticated
with
-u, --user string       The account to output the token for
```
`gh browse [<number> &#124; <path> &#124; <commit-SHA>] [flags]`

Open repositories, issues, pull requests, and more in the browser

```
-b, --branch string            Select another branch by passing in the
branch name
-c, --commit string[="last"]   Select another commit by passing in the
commit SHA, default is the last commit
-n, --no-browser               Print destination URL instead of opening the
browser
-p, --projects                 Open repository projects
-r, --releases                 Open repository releases
-s, --settings                 Open repository settings
-w, --wiki                     Open repository wiki
```
`gh cache <command>`

Manage GitHub Actions caches

`gh cache delete [<cache-id>&#124; <cache-key> &#124; --all] [flags]`

Delete GitHub Actions caches

```
-a, --all   Delete all caches
```
`gh cache list [flags]`

List GitHub Actions caches

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-k, --key string        Filter by cache key prefix
-L, --limit int         Maximum number of caches to fetch (default 30)
-O, --order string      Order of caches returned: {asc&#124;desc} (default
"desc")
-r, --ref string        Filter by ref, formatted as refs/heads/<branch
name> or refs/pull/<number>/merge
-S, --sort string       Sort fetched caches:
{created_at&#124;last_accessed_at&#124;size_in_bytes} (default
"last_accessed_at")
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```
Aliases

gh cache ls

`gh codespace`

Connect to and manage codespaces

Aliases

gh cs

`gh codespace code [flags]`

Open a codespace in Visual Studio Code

```
-c, --codespace string    Name of the codespace
    --insiders            Use the insiders version of Visual Studio Code
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
-w, --web                 Use the web version of Visual Studio Code
```

`gh codespace cp [-e] [-r] [-- [<scp flags>...]] <sources>... <dest>`

Copy files between local and remote file systems

```
-c, --codespace string    Name of the codespace
-e, --expand              Expand remote file names on remote shell
-p, --profile string      Name of the SSH profile to use
-r, --recursive           Recursively copy directories
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
```

`gh codespace create [flags]`

Create a codespace

```
-b, --branch string                 repository branch
    --default-permissions           do not prompt to accept additional
permissions requested by the codespace
    --devcontainer-path string    path to the devcontainer.json file to use
when creating codespace
-d, --display-name string         display name for the codespace (48
characters or less)
    --idle-timeout duration       allowed inactivity before codespace is
stopped, e.g. "10m", "1h"
-l, --location string             location:
{EastUs&#124;SouthEastAsia&#124;WestEurope&#124;WestUs2} (determined
automatically if not provided)
-m, --machine string              hardware specifications for the VM
-R, --repo string                 repository name with owner: user/repo
    --retention-period duration   allowed time after shutting down before
the codespace is automatically deleted (maximum 30 days), e.g. "1h", "72h"
-s, --status                      show status of post-create command and
dotfiles
-w, --web                         create codespace from browser, cannot be
used with --display-name, --idle-timeout, or --retention-period
```

`gh codespace delete [flags]`

Delete codespaces

```
    --all                 Delete all codespaces
-c, --codespace string    Name of the codespace
    --days N              Delete codespaces older than N days
-f, --force               Skip confirmation for codespaces that contain
unsaved changes
-o, --org login           The login handle of the organization (admin-only)
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
-u, --user username       The username to delete codespaces for (used with
--org)
```

`gh codespace edit [flags]`

Edit a codespace

```
-c, --codespace string      Name of the codespace
-d, --display-name string   Set the display name
-m, --machine string        Set hardware specifications for the VM
-R, --repo string           Filter codespace selection by repository name
(user/repo)
    --repo-owner string     Filter codespace selection by repository owner
(username or org)
```

`gh codespace jupyter [flags]`

Open a codespace in JupyterLab

```
-c, --codespace string    Name of the codespace
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
```

`gh codespace list [flags]`

List codespaces

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-L, --limit int         Maximum number of codespaces to list (default 30)
-o, --org login         The login handle of the organization to list
codespaces for (admin-only)
-R, --repo string       Repository name with owner: user/repo
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-u, --user username     The username to list codespaces for (used with --
org)
-w, --web               List codespaces in the web browser, cannot be used
with --user or --org
```

Aliases

gh codespace ls, gh cs ls

`gh codespace logs [flags]`

Access codespace logs

```
-c, --codespace string    Name of the codespace
-f, --follow              Tail and follow the logs
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
```

`gh codespace ports [flags]`

List ports in a codespace

```
-c, --codespace string    Name of the codespace
-q, --jq expression       Filter JSON output using a jq expression
    --json fields         Output JSON with the specified fields
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
-t, --template string     Format JSON output using a Go template; see "gh
help formatting"
```

`gh codespace ports forward <remote-port>:<local-port>...`

Forward ports

`gh codespace ports visibility <port>:{public&#124;private&#124;org}...`

Change the visibility of the forwarded port

`gh codespace rebuild [flags]`

Rebuild a codespace

```
-c, --codespace string    Name of the codespace
    --full                perform a full rebuild
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
```

`gh codespace ssh [<flags>...] [-- <ssh-flags>...] [<command>]`

SSH into a codespace

```
-c, --codespace string    Name of the codespace
    --config              Write OpenSSH configuration to stdout
-d, --debug               Log debug data to a file
    --debug-file string   Path of the file log to
    --profile string      Name of the SSH profile to use
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
    --server-port int     SSH server port number (0 => pick unused)
```

`gh codespace stop [flags]`

Stop a running codespace

```
-c, --codespace string    Name of the codespace
-o, --org login           The login handle of the organization (admin-only)
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
-u, --user username       The username to stop codespace for (used with --
org)
```

`gh codespace view [flags]`

View details about a codespace

```
-c, --codespace string    Name of the codespace
-q, --jq expression       Filter JSON output using a jq expression
    --json fields         Output JSON with the specified fields
-R, --repo string         Filter codespace selection by repository name
(user/repo)
    --repo-owner string   Filter codespace selection by repository owner
(username or org)
-t, --template string     Format JSON output using a Go template; see "gh
help formatting"
```

`gh completion -s <shell>`

Generate shell completion scripts

```
-s, --shell string   Shell type: {bash&#124;zsh&#124;fish&#124;powershell}
```

`gh config <command>`

Manage configuration for gh

`gh config clear-cache`

Clear the cli cache

`gh config get <key> [flags]`

Print the value of a given configuration key

```
-h, --host string   Get per-host setting
```

`gh config list [flags]`

Print a list of configuration keys and values

```
-h, --host string   Get per-host configuration
```

Aliases

gh config ls

`gh config set <key> <value> [flags]`

Update configuration with a value for the given key

```
-h, --host string   Set per-host setting
```

`gh extension`

Manage gh extensions

Aliases

gh extensions, gh ext

`gh extension browse [flags]`

Enter a UI for browsing, adding, and removing extensions

```
    --debug           log to /tmp/extBrowse-*
-s, --single-column   Render TUI with only one column of text
```

`gh extension create [<name>] [flags]`

Create a new extension

```
--precompiled string   Create a precompiled extension. Possible values: go, other
```

`gh extension exec <name> [args]`

Execute an installed extension

`gh extension install <repository> [flags]`

Install a gh extension from a repository

```
--force        force upgrade extension, or ignore if latest already installed
--pin string   pin extension to a release tag or commit ref
```

`gh extension list`

List installed extension commands

Aliases

gh ext ls, gh extension ls, gh extensions ls

`gh extension remove <name>`

Remove an installed extension

gh extension search [<query>] [flags]

Search extensions to the GitHub CLI

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
    --license strings   Filter based on license type
-L, --limit int         Maximum number of extensions to fetch (default 30)
    --order string      Order of repositories returned, ignored unless '--
sort' flag is specified: {asc&#124;desc} (default "desc")
    --owner strings     Filter on owner
    --sort string       Sort fetched repositories: {forks&#124;help-wanted-
issues&#124;stars&#124;updated} (default "best-match")
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               Open the search query in the web browser
```

gh extension upgrade {<name> &#124; --all} [flags]

Upgrade installed extensions

```
--all        Upgrade all extensions
--dry-run    Only display upgrades
--force      Force upgrade extension
```

gh gist <command>

Manage gists

gh gist clone <gist> [<directory>] [-- <gitflags>...]

Clone a gist locally

gh gist create [<filename>... &#124; -] [flags]

Create a new gist

```
-d, --desc string       A description for this gist
-f, --filename string   Provide a filename to be used when reading from
standard input
-p, --public            List the gist publicly (default "secret")
-w, --web               Open the web browser with created gist
```

Aliases

gh gist new

gh gist delete {<id> &#124; <url>} [flags]

Delete a gist

```
--yes   confirm deletion without prompting
```

gh gist edit {<id> &#124; <url>} [<filename>] [flags]

Edit one of your gists

```
-a, --add string        Add a new file to the gist
-d, --desc string       New description for the gist
-f, --filename string   Select a file to edit
-r, --remove string     Remove a file from the gist
```

gh gist list [flags]

List your gists

```
    --filter expression   Filter gists using a regular expression
    --include-content     Include gists' file content when filtering
-L, --limit int           Maximum number of gists to fetch (default 10)
    --public              Show only public gists
    --secret              Show only secret gists
```

Aliases

gh gist ls

`gh gist rename {<id> &#124; <url>} <oldFilename> <newFilename>`

Rename a file in a gist

`gh gist view [<id> &#124; <url>] [flags]`

View a gist

```
-f, --filename string   Display a single file from the gist
    --files             List file names from the gist
-r, --raw               Print raw instead of rendered gist contents
-w, --web               Open gist in the browser
```

`gh gpg-key <command>`

Manage GPG keys

`gh gpg-key add [<key-file>] [flags]`

Add a GPG key to your GitHub account

```
-t, --title string   Title for the new key
```

`gh gpg-key delete <key-id> [flags]`

Delete a GPG key from your GitHub account

```
-y, --yes   Skip the confirmation prompt
```

`gh gpg-key list`

Lists GPG keys in your GitHub account

Aliases

gh gpg-key ls

`gh issue <command>`

Manage issues

`gh issue close {<number> &#124; <url>} [flags]`

Close issue

```
-c, --comment string   Leave a closing comment
-r, --reason string    Reason for closing: {completed&#124;not planned}
```

`gh issue comment {<number> &#124; <url>} [flags]`

Add a comment to an issue

```
-b, --body text        The comment body text
-F, --body-file file   Read body text from file (use "-" to read from
standard input)
    --edit-last        Edit the last comment of the same author
-e, --editor           Skip prompts and open the text editor to write the
body in
-w, --web              Open the web browser to write the comment
```

`gh issue create [flags]`

Create a new issue

```
-a, --assignee login   Assign people by their login. Use "@me" to self-
assign.
-b, --body string      Supply a body. Will prompt for one otherwise.
-F, --body-file file   Read body text from file (use "-" to read from
standard input)
-e, --editor           Skip prompts and open the text editor to write the
title and body in. The first line is the title and the remaining text is
the body.
-l, --label name       Add labels by name
-m, --milestone name   Add the issue to a milestone by name
-p, --project title    Add the issue to projects by title
    --recover string   Recover input from a failed run of create
-T, --template name    Template name to use as starting body text
-t, --title string     Supply a title. Will prompt for one otherwise.
-w, --web              Open the browser to create an issue
```

Aliases

gh issue new

`gh issue delete {<number> &#124; <url>} [flags]`

Delete issue

```
--yes   confirm deletion without prompting
```

`gh issue develop {<number> &#124; <url>} [flags]`

Manage linked branches for an issue

```
-b, --base string          Name of the remote branch you want to make your
new branch from
    --branch-repo string   Name or URL of the repository where you want to
create your new branch
-c, --checkout             Checkout the branch after creating it
-l, --list                 List linked branches for the issue
-n, --name string          Name of the branch to create
```

`gh issue edit {<numbers> &#124; <urls>} [flags]`

Edit issues

```
    --add-assignee login      Add assigned users by their login. Use "@me"
to assign yourself.
    --add-label name          Add labels by name
    --add-project title       Add the issue to projects by title
-b, --body string             Set the new body.
-F, --body-file file          Read body text from file (use "-" to read
from standard input)
-m, --milestone name          Edit the milestone the issue belongs to by
name
    --remove-assignee login   Remove assigned users by their login. Use
"@me" to unassign yourself.
    --remove-label name       Remove labels by name
    --remove-milestone        Remove the milestone association from the
issue
    --remove-project title    Remove the issue from projects by title
-t, --title string            Set the new title.
```

`gh issue list [flags]`

List issues in a repository

```
    --app string         Filter by GitHub App author
```

```
-a, --assignee string    Filter by assignee
-A, --author string      Filter by author
-q, --jq expression      Filter JSON output using a jq expression
    --json fields        Output JSON with the specified fields
-l, --label strings      Filter by label
-L, --limit int          Maximum number of issues to fetch (default 30)
    --mention string     Filter by mention
-m, --milestone string   Filter by milestone number or title
-S, --search query       Search issues with query
-s, --state string       Filter by state: {open&#124;closed&#124;all}
(default "open")
-t, --template string    Format JSON output using a Go template; see "gh
help formatting"
-w, --web                List issues in the web browser
```

Aliases

gh issue ls

`gh issue lock {<number> &#124; <url>} [flags]`

Lock issue conversation

```
-r, --reason string   Optional reason for locking conversation (off_topic,
resolved, spam, too_heated).
```

`gh issue pin {<number> &#124; <url>}`

Pin a issue

`gh issue reopen {<number> &#124; <url>} [flags]`

Reopen issue

```
-c, --comment string   Add a reopening comment
```

`gh issue status [flags]`

Show status of relevant issues

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

`gh issue transfer {<number> &#124; <url>} <destination-repo>`

Transfer issue to another repository

`gh issue unlock {<number> &#124; <url>}`

Unlock issue conversation

`gh issue unpin {<number> &#124; <url>}`

Unpin a issue

`gh issue view {<number> &#124; <url>} [flags]`

View an issue

```
-c, --comments          View issue comments
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               Open an issue in the browser
```

`gh label <command>`

Manage labels

`gh label clone <source-repository> [flags]`

Clones labels from one repository to another

```
-f, --force   Overwrite labels in the destination repository
```

`gh label create <name> [flags]`

Create a new label

```
-c, --color string         Color of the label
-d, --description string   Description of the label
-f, --force                Update the label color and description if label
already exists
```

`gh label delete <name> [flags]`

Delete a label from a repository

```
--yes   Confirm deletion without prompting
```

`gh label edit <name> [flags]`

Edit a label

```
-c, --color string         Color of the label
-d, --description string   Description of the label
-n, --name string          New name of the label
```

`gh label list [flags]`

List labels in a repository

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-L, --limit int         Maximum number of labels to fetch (default 30)
    --order string      Order of labels returned: {asc&#124;desc} (default
"asc")
-S, --search string     Search label names and descriptions
    --sort string       Sort fetched labels: {created&#124;name} (default
"created")
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               List labels in the web browser
```

Aliases

gh label ls

`gh org <command>`

Manage organizations

`gh org list [flags]`

List organizations for the authenticated user.

```
-L, --limit int   Maximum number of organizations to list (default 30)
```

Aliases

gh org ls

`gh pr <command>`

Manage pull requests

`gh pr checkout [<number> &#124; <url> &#124; <branch>] [flags]`

Check out a pull request in git

```
-b, --branch string        Local branch name to use (default [the name of
the head branch])
    --detach               Checkout PR with a detached HEAD
-f, --force                Reset the existing local branch to the latest
state of the pull request
    --recurse-submodules   Update all submodules after checkout
```

`gh pr checks [<number> &#124; <url> &#124; <branch>] [flags]`

Show CI status for a single pull request

```
    --fail-fast          Exit watch mode on first check failure
-i, --interval --watch   Refresh interval in seconds when using --watch
flag (default 10)
-q, --jq expression      Filter JSON output using a jq expression
    --json fields        Output JSON with the specified fields
    --required           Only show checks that are required
-t, --template string    Format JSON output using a Go template; see "gh
help formatting"
    --watch              Watch checks until they finish
-w, --web                Open the web browser to show details about checks
```

`gh pr close {<number> &#124; <url> &#124; <branch>} [flags]`

Close a pull request

```
-c, --comment string   Leave a closing comment
-d, --delete-branch    Delete the local and remote branch after close
```

`gh pr comment [<number> &#124; <url> &#124; <branch>] [flags]`

Add a comment to a pull request

```
-b, --body text        The comment body text
-F, --body-file file   Read body text from file (use "-" to read from
standard input)
    --edit-last        Edit the last comment of the same author
-e, --editor           Skip prompts and open the text editor to write the
body in
-w, --web              Open the web browser to write the comment
```

`gh pr create [flags]`

Create a pull request

```
-a, --assignee login       Assign people by their login. Use "@me" to self-
assign.
-B, --base branch          The branch into which you want your code merged
-b, --body string          Body for the pull request
-F, --body-file file       Read body text from file (use "-" to read from
standard input)
-d, --draft                Mark pull request as a draft
    --dry-run              Print details instead of creating the PR. May
still push git changes.
-e, --editor               Skip prompts and open the text editor to write
the title and body in. The first line is the title and the remaining text
is the body.
-f, --fill                 Use commit info for title and body
    --fill-first           Use first commit info for title and body
    --fill-verbose         Use commits msg+body for description
-H, --head branch          The branch that contains commits for your pull
request (default [current branch])
-l, --label name           Add labels by name
-m, --milestone name       Add the pull request to a milestone by name
```

```
    --no-maintainer-edit   Disable maintainer's ability to modify pull
request
-p, --project title        Add the pull request to projects by title
    --recover string       Recover input from a failed run of create
-r, --reviewer handle      Request reviews from people or teams by their
handle
-T, --template file        Template file to use as starting body text
-t, --title string         Title for the pull request
-w, --web                  Open the web browser to create a pull request
```

Aliases

gh pr new

`gh pr diff [<number> &#124; <url> &#124; <branch>] [flags]`

View changes in a pull request

```
    --color string   Use color in diff output:
{always&#124;never&#124;auto} (default "auto")
    --name-only      Display only names of changed files
    --patch          Display diff in patch format
-w, --web            Open the pull request diff in the browser
```

`gh pr edit [<number> &#124; <url> &#124; <branch>] [flags]`

Edit a pull request

```
    --add-assignee login      Add assigned users by their login. Use "@me"
to assign yourself.
    --add-label name          Add labels by name
    --add-project title       Add the pull request to projects by title
    --add-reviewer login      Add reviewers by their login.
-B, --base branch             Change the base branch for this pull request
-b, --body string             Set the new body.
-F, --body-file file          Read body text from file (use "-" to read
from standard input)
-m, --milestone name          Edit the milestone the pull request belongs
to by name
    --remove-assignee login   Remove assigned users by their login. Use
"@me" to unassign yourself.
    --remove-label name       Remove labels by name
    --remove-milestone        Remove the milestone association from the
pull request
    --remove-project title    Remove the pull request from projects by
title
    --remove-reviewer login   Remove reviewers by their login.
-t, --title string            Set the new title.
```

`gh pr list [flags]`

List pull requests in a repository

```
    --app string         Filter by GitHub App author
-a, --assignee string    Filter by assignee
-A, --author string      Filter by author
-B, --base string        Filter by base branch
-d, --draft              Filter by draft state
-H, --head string        Filter by head branch
-q, --jq expression      Filter JSON output using a jq expression
    --json fields        Output JSON with the specified fields
-l, --label strings      Filter by label
-L, --limit int          Maximum number of items to fetch (default 30)
-S, --search query       Search pull requests with query
```

```
-s, --state string       Filter by state:
{open&#124;closed&#124;merged&#124;all} (default "open")
-t, --template string    Format JSON output using a Go template; see "gh
help formatting"
-w, --web                List pull requests in the web browser
```

Aliases

gh pr ls

`gh pr lock {<number> &#124; <url>} [flags]`

Lock pull request conversation

```
-r, --reason string   Optional reason for locking conversation (off_topic,
resolved, spam, too_heated).
```

`gh pr merge [<number> &#124; <url> &#124; <branch>] [flags]`

Merge a pull request

```
    --admin                     Use administrator privileges to merge a pull
request that does not meet requirements
-A, --author-email text         Email text for merge commit author
    --auto                      Automatically merge only after necessary
requirements are met
-b, --body text                 Body text for the merge commit
-F, --body-file file            Read body text from file (use "-" to read
from standard input)
-d, --delete-branch             Delete the local and remote branch after
merge
    --disable-auto              Disable auto-merge for this pull request
    --match-head-commit SHA     Commit SHA that the pull request head must
match to allow merge
-m, --merge                     Merge the commits with the base branch
-r, --rebase                    Rebase the commits onto the base branch
-s, --squash                    Squash the commits into one commit and merge
it into the base branch
-t, --subject text              Subject text for the merge commit
```

`gh pr ready [<number> &#124; <url> &#124; <branch>] [flags]`

Mark a pull request as ready for review

```
--undo   Convert a pull request to "draft"
```

`gh pr reopen {<number> &#124; <url> &#124; <branch>} [flags]`

Reopen a pull request

```
-c, --comment string   Add a reopening comment
```

`gh pr review [<number> &#124; <url> &#124; <branch>] [flags]`

Add a review to a pull request

```
-a, --approve            Approve pull request
-b, --body string        Specify the body of a review
-F, --body-file file     Read body text from file (use "-" to read from
standard input)
-c, --comment            Comment on a pull request
-r, --request-changes    Request changes on a pull request
```

`gh pr status [flags]`

Show status of relevant pull requests

```
-c, --conflict-status    Display the merge conflict status of each pull
request
```

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```
`gh pr unlock {<number> &#124; <url>}`

Unlock pull request conversation

`gh pr update-branch [<number> &#124; <url> &#124; <branch>] [flags]`

Update a pull request branch

```
--rebase   Update PR branch by rebasing on top of latest base branch
```
`gh pr view [<number> &#124; <url> &#124; <branch>] [flags]`

View a pull request

```
-c, --comments          View pull request comments
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               Open a pull request in the browser
```
`gh project <command>`

Work with GitHub Projects.

`gh project close [<number>] [flags]`

Close a project

```
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
    --undo              Reopen a closed project
```
`gh project copy [<number>] [flags]`

Copy a project

```
    --drafts                Include draft issues when copying
    --format string         Output format: {json}
-q, --jq expression         Filter JSON output using a jq expression
    --source-owner string   Login of the source owner. Use "@me" for the
current user.
    --target-owner string   Login of the target owner. Use "@me" for the
current user.
-t, --template string       Format JSON output using a Go template; see "gh
help formatting"
    --title string          Title for the new project
```
`gh project create [flags]`

Create a project

```
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
    --title string      Title for the project
```
`gh project delete [<number>] [flags]`

Delete a project

```
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

gh project edit [<number>] [flags]

Edit a project

```
-d, --description string   New description of the project
    --format string        Output format: {json}
-q, --jq expression        Filter JSON output using a jq expression
    --owner string         Login of the owner. Use "@me" for the current
user.
    --readme string        New readme for the project
-t, --template string      Format JSON output using a Go template; see "gh
help formatting"
    --title string         New title for the project
    --visibility string    Change project visibility: {PUBLIC&#124;PRIVATE}
```

gh project field-create [<number>] [flags]

Create a field in a project

```
    --data-type string                 DataType of the new field.:
{TEXT&#124;SINGLE_SELECT&#124;DATE&#124;NUMBER}
    --format string                    Output format: {json}
-q, --jq expression                    Filter JSON output using a jq
expression
    --name string                      Name of the new field
    --owner string                     Login of the owner. Use "@me" for the
current user.
    --single-select-options strings   Options for SINGLE_SELECT data type
-t, --template string                  Format JSON output using a Go
template; see "gh help formatting"
```

gh project field-delete [flags]

Delete a field in a project

```
    --format string     Output format: {json}
    --id string         ID of the field to delete
-q, --jq expression     Filter JSON output using a jq expression
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

gh project field-list number [flags]

List the fields in a project

```
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
-L, --limit int         Maximum number of fields to fetch (default 30)
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

gh project item-add [<number>] [flags]

Add a pull request or an issue to a project

```
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
    --url string        URL of the issue or pull request to add to the
project
```

`gh project item-archive [<number>] [flags]`

Archive an item in a project

```
    --format string     Output format: {json}
    --id string         ID of the item to archive
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
    --undo              Unarchive an item
```

`gh project item-create [<number>] [flags]`

Create a draft issue item in a project

```
    --body string       Body for the draft issue
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
    --title string      Title for the draft issue
```

`gh project item-delete [<number>] [flags]`

Delete an item from a project by ID

```
    --format string     Output format: {json}
    --id string         ID of the item to delete
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

`gh project item-edit [flags]`

Edit an item in a project

```
    --body string                       Body of the draft issue item
    --clear                             Remove field value
    --date string                       Date value for the field (YYYY-MM-
DD)
    --field-id string                   ID of the field to update
    --format string                     Output format: {json}
    --id string                         ID of the item to edit
    --iteration-id string               ID of the iteration value to set on
the field
-q, --jq expression                     Filter JSON output using a jq
expression
    --number float                      Number value for the field
    --project-id string                 ID of the project to which the field
belongs to
    --single-select-option-id string   ID of the single select option value
to set on the field
-t, --template string                   Format JSON output using a Go
template; see "gh help formatting"
    --text string                       Text value for the field
    --title string                      Title of the draft issue item
```

`gh project item-list [<number>] [flags]`

List the items in a project

```
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
-L, --limit int         Maximum number of items to fetch (default 30)
    --owner string      Login of the owner. Use "@me" for the current user.
```

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

`gh project link [<number>] [flag] [flags]`

Link a project to a repository or a team

```
    --owner string   Login of the owner. Use "@me" for the current user.
-R, --repo string    The repository to be linked to this project
-T, --team string    The team to be linked to this project
```

`gh project list [flags]`

List the projects for an owner

```
    --closed            Include closed projects
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
-L, --limit int         Maximum number of projects to fetch (default 30)
    --owner string      Login of the owner
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               Open projects list in the browser
```

Aliases

gh project ls

`gh project mark-template [<number>] [flags]`

Mark a project as a template

```
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the org owner.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
    --undo              Unmark the project as a template.
```

`gh project unlink [<number>] [flag] [flags]`

Unlink a project from a repository or a team

```
    --owner string   Login of the owner. Use "@me" for the current user.
-R, --repo string    The repository to be unlinked from this project
-T, --team string    The team to be unlinked from this project
```

`gh project view [<number>] [flags]`

View a project

```
    --format string     Output format: {json}
-q, --jq expression     Filter JSON output using a jq expression
    --owner string      Login of the owner. Use "@me" for the current user.
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               Open a project in the browser
```

`gh release <command>`

Manage releases

`gh release create [<tag>] [<files>...]`

Create a new release

```
    --discussion-category string   Start a discussion in the specified
category
-d, --draft                        Save the release as a draft instead of
publishing it
```

```
    --generate-notes               Automatically generate title and notes
for the release
    --latest                       Mark this release as "Latest" (default
[automatic based on date and version]). --latest=false to explicitly NOT
set as latest
-n, --notes string                 Release notes
-F, --notes-file file              Read release notes from file (use "-" to
read from standard input)
    --notes-from-tag               Automatically generate notes from
annotated tag
    --notes-start-tag string       Tag to use as the starting point for
generating release notes
-p, --prerelease                   Mark the release as a prerelease
    --target branch                Target branch or full commit SHA
(default [main branch])
-t, --title string                 Release title
    --verify-tag                   Abort in case the git tag doesn't
already exist in the remote repository
```

Aliases

gh release new

`gh release delete <tag> [flags]`

Delete a release

```
    --cleanup-tag   Delete the specified tag in addition to its release
-y, --yes           Skip the confirmation prompt
```

`gh release delete-asset <tag> <asset-name> [flags]`

Delete an asset from a release

```
-y, --yes   Skip the confirmation prompt
```

`gh release download [<tag>] [flags]`

Download release assets

```
-A, --archive format        Download the source code archive in the
specified format (zip or tar.gz)
    --clobber               Overwrite existing files of the same name
-D, --dir directory         The directory to download files into (default
".")
-O, --output file           The file to write a single asset to (use "-" to
write to standard output)
-p, --pattern stringArray   Download only assets that match a glob pattern
    --skip-existing         Skip downloading when files of the same name
exist
```

`gh release edit <tag>`

Edit a release

```
    --discussion-category string   Start a discussion in the specified
category when publishing a draft
    --draft                        Save the release as a draft instead of
publishing it
    --latest                       Explicitly mark the release as "Latest"
-n, --notes string                 Release notes
-F, --notes-file file              Read release notes from file (use "-" to
read from standard input)
    --prerelease                   Mark the release as a prerelease
    --tag string                   The name of the tag
    --target branch                Target branch or full commit SHA
(default [main branch])
-t, --title string                 Release title
    --verify-tag                   Abort in case the git tag doesn't
already exist in the remote repository
```

`gh release list [flags]`

List releases in a repository

```
    --exclude-drafts         Exclude draft releases
    --exclude-pre-releases   Exclude pre-releases
-q, --jq expression          Filter JSON output using a jq expression
    --json fields            Output JSON with the specified fields
-L, --limit int              Maximum number of items to fetch (default 30)
-O, --order string           Order of releases returned: {asc&#124;desc}
(default "desc")
-t, --template string        Format JSON output using a Go template; see
"gh help formatting"
```

Aliases

gh release ls

`gh release upload <tag> <files>... [flags]`

Upload assets to a release

```
--clobber   Overwrite existing assets of the same name
```

`gh release view [<tag>] [flags]`

View information about a release

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               Open the release in the browser
```

`gh repo <command>`

Manage repositories

`gh repo archive [<repository>] [flags]`

Archive a repository

```
-y, --yes   Skip the confirmation prompt
```

`gh repo autolink <command>`

Manage autolink references

`gh repo autolink create <keyPrefix> <urlTemplate> [flags]`

Create a new autolink reference

```
-n, --numeric   Mark autolink as numeric
```

Aliases

gh repo autolink new

`gh repo autolink delete <id> [flags]`

Delete an autolink reference

```
--yes   Confirm deletion without prompting
```

`gh repo autolink list [flags]`

List autolink references for a GitHub repository

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               List autolink references in the web browser
```

Aliases

gh repo autolink ls

`gh repo autolink view <id> [flags]`

View an autolink reference

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

`gh repo clone <repository> [<directory>] [-- <gitflags>...]`

Clone a repository locally

```
-u, --upstream-remote-name string   Upstream remote name when cloning a
fork (default "upstream")
```

`gh repo create [<name>] [flags]`

Create a new repository

```
    --add-readme             Add a README file to the new repository
-c, --clone                  Clone the new repository to the current
directory
-d, --description string     Description of the repository
    --disable-issues         Disable issues in the new repository
    --disable-wiki           Disable wiki in the new repository
-g, --gitignore string       Specify a gitignore template for the
repository
-h, --homepage URL           Repository home page URL
    --include-all-branches   Include all branches from template repository
    --internal               Make the new repository internal
-l, --license string         Specify an Open Source License for the
repository
    --private                Make the new repository private
    --public                 Make the new repository public
    --push                   Push local commits to the new repository
-r, --remote string          Specify remote name for the new repository
-s, --source string          Specify path to local repository to use as
source
-t, --team name              The name of the organization team to be
granted access
-p, --template repository    Make the new repository based on a template
repository
```

Aliases

gh repo new

`gh repo delete [<repository>] [flags]`

Delete a repository

```
--yes   confirm deletion without prompting
```

`gh repo deploy-key <command>`

Manage deploy keys in a repository

`gh repo deploy-key add <key-file> [flags]`

Add a deploy key to a GitHub repository

```
-w, --allow-write    Allow write access for the key
-t, --title string   Title of the new key
```

`gh repo deploy-key delete <key-id>`

Delete a deploy key from a GitHub repository

`gh repo deploy-key list [flags]`

List deploy keys in a GitHub repository

```
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

Aliases

gh repo deploy-key ls

`gh repo edit [<repository>] [flags]`

Edit repository settings

```
    --accept-visibility-change-consequences    Accept the consequences of
changing the repository visibility
    --add-topic strings                        Add repository topic
    --allow-forking                            Allow forking of an
organization repository
    --allow-update-branch                      Allow a pull request head
branch that is behind its base branch to be updated
    --default-branch name                      Set the default branch name
for the repository
    --delete-branch-on-merge                   Delete head branch when pull
requests are merged
-d, --description string                       Description of the
repository
    --enable-advanced-security                 Enable advanced security in
the repository
    --enable-auto-merge                        Enable auto-merge
functionality
    --enable-discussions                       Enable discussions in the
repository
    --enable-issues                            Enable issues in the
repository
    --enable-merge-commit                      Enable merging pull requests
via merge commit
    --enable-projects                          Enable projects in the
repository
    --enable-rebase-merge                      Enable merging pull requests
via rebase
    --enable-secret-scanning                   Enable secret scanning in
the repository
    --enable-secret-scanning-push-protection   Enable secret scanning push
protection in the repository. Secret scanning must be enabled first
```

```
    --enable-squash-merge                      Enable merging pull requests
via squashed commit
    --enable-wiki                              Enable wiki in the
repository
-h, --homepage URL                             Repository home page URL
    --remove-topic strings                     Remove repository topic
    --template                                 Make the repository
available as a template repository
    --visibility string                        Change the visibility of the
repository to {public,private,internal}
```
gh repo fork [<repository>] [-- <gitflags>...] [flags]

Create a fork of a repository

```
--clone                   Clone the fork
--default-branch-only     Only include the default branch in the fork
--fork-name string        Rename the forked repository
--org string              Create the fork in an organization
--remote                  Add a git remote for the fork
--remote-name string      Specify the name for the new remote (default
"origin")
```
gh repo gitignore <command>

List and view available repository gitignore templates

gh repo gitignore list

List available repository gitignore templates

Aliases

gh repo gitignore ls

gh repo gitignore view <template>

View an available repository gitignore template

gh repo license <command>

Explore repository licenses

gh repo license list

List common repository licenses

Aliases

gh repo license ls

gh repo license view {<license-key> &#124; <SPDX-ID>} [flags]

View a specific repository license

```
-w, --web   Open https://choosealicense.com/ in the browser
```
gh repo list [<owner>] [flags]

List repositories owned by user or organization

```
    --archived             Show only archived repositories
    --fork                 Show only forks
-q, --jq expression        Filter JSON output using a jq expression
    --json fields          Output JSON with the specified fields
-l, --language string      Filter by primary coding language
```

```
-L, --limit int           Maximum number of repositories to list (default
30)
    --no-archived         Omit archived repositories
    --source              Show only non-forks
-t, --template string     Format JSON output using a Go template; see "gh
help formatting"
    --topic strings       Filter by topic
    --visibility string   Filter by repository visibility:
{public&#124;private&#124;internal}
```
Aliases

gh repo ls

`gh repo rename [<new-name>] [flags]`

Rename a repository

```
-y, --yes   Skip the confirmation prompt
```
`gh repo set-default [<repository>] [flags]`

Configure default repository for this directory

```
-u, --unset   unset the current default repository
-v, --view    view the current default repository
```
`gh repo sync [<destination-repository>] [flags]`

Sync a repository

```
-b, --branch string   Branch to sync (default [default branch])
    --force           Hard reset the branch of the destination repository
to match the source repository
-s, --source string   Source repository
```
`gh repo unarchive [<repository>] [flags]`

Unarchive a repository

```
-y, --yes   Skip the confirmation prompt
```
`gh repo view [<repository>] [flags]`

View a repository

```
-b, --branch string     View a specific branch of the repository
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-w, --web               Open a repository in the browser
```
`gh ruleset <command>`

View info about repo rulesets

Aliases

gh rs

`gh ruleset check [<branch>] [flags]`

View rules that would apply to a given branch

```
    --default   Check rules on default branch
-w, --web       Open the branch rules page in a web browser
```
`gh ruleset list [flags]`

List rulesets for a repository or organization

```
-L, --limit int    Maximum number of rulesets to list (default 30)
-o, --org string   List organization-wide rulesets for the provided
organization
-p, --parents      Whether to include rulesets configured at higher levels
that also apply (default true)
-w, --web          Open the list of rulesets in the web browser
```

Aliases

gh rs ls, gh ruleset ls

`gh ruleset view [<ruleset-id>] [flags]`

View information about a ruleset

```
-o, --org string   Organization name if the provided ID is an organization-
level ruleset
-p, --parents      Whether to include rulesets configured at higher levels
that also apply (default true)
-w, --web          Open the ruleset in the browser
```

`gh run <command>`

View details about workflow runs

`gh run cancel [<run-id>]`

Cancel a workflow run

`gh run delete [<run-id>]`

Delete a workflow run

`gh run download [<run-id>] [flags]`

Download artifacts generated by a workflow run

```
-D, --dir string            The directory to download artifacts into
(default ".")
-n, --name stringArray      Download artifacts that match any of the given
names
-p, --pattern stringArray   Download artifacts that match a glob pattern
```

`gh run list [flags]`

List recent workflow runs

```
-a, --all                Include disabled workflows
-b, --branch string      Filter runs by branch
-c, --commit SHA         Filter runs by the SHA of the commit
    --created date       Filter runs by the date it was created
-e, --event event        Filter runs by which event triggered the run
-q, --jq expression      Filter JSON output using a jq expression
    --json fields        Output JSON with the specified fields
-L, --limit int          Maximum number of runs to fetch (default 20)
-s, --status string      Filter runs by status:
{queued&#124;completed&#124;in_progress&#124;requested&#124;waiting&#124;pe
nding&#124;action_required&#124;cancelled&#124;failure&#124;neutral&#124;sk
ipped&#124;stale&#124;startup_failure&#124;success&#124;timed_out}
-t, --template string    Format JSON output using a Go template; see "gh
help formatting"
-u, --user string        Filter runs by user who triggered the run
-w, --workflow string    Filter runs by workflow
```

Aliases

gh run ls

gh run rerun [<run-id>] [flags]

Rerun a run

```
-d, --debug        Rerun with debug logging
    --failed       Rerun only failed jobs, including dependencies
-j, --job string   Rerun a specific job ID from a run, including
dependencies
```

gh run view [<run-id>] [flags]

View a summary of a workflow run

```
-a, --attempt uint      The attempt number of the workflow run
    --exit-status       Exit with non-zero status if run failed
-j, --job string        View a specific job ID from a run
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
    --log               View full log for either a run or specific job
    --log-failed        View the log for any failed steps in a run or
specific job
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-v, --verbose           Show job steps
-w, --web               Open run in the browser
```

gh run watch <run-id> [flags]

Watch a run until it completes, showing its progress

```
    --exit-status    Exit with non-zero status if run fails
-i, --interval int   Refresh interval in seconds (default 3)
```

gh search <command>

Search for repositories, issues, and pull requests

gh search code <query> [flags]

Search within code

```
    --extension string   Filter on file extension
    --filename string    Filter on filename
-q, --jq expression      Filter JSON output using a jq expression
    --json fields        Output JSON with the specified fields
    --language string    Filter results by language
-L, --limit int          Maximum number of code results to fetch (default
30)
    --match strings      Restrict search to file contents or file path:
{file&#124;path}
    --owner strings      Filter on owner
-R, --repo strings       Filter on repository
    --size string        Filter on size range, in kilobytes
-t, --template string    Format JSON output using a Go template; see "gh
help formatting"
-w, --web                Open the search query in the web browser
```

gh search commits [<query>] [flags]

Search for commits

```
    --author string          Filter by author
    --author-date date       Filter based on authored date
    --author-email string    Filter on author email
    --author-name string     Filter on author name
    --committer string       Filter by committer
    --committer-date date      Filter based on committed date
    --committer-email string   Filter on committer email
    --committer-name string    Filter on committer name
    --hash string              Filter by commit hash
-q, --jq expression            Filter JSON output using a jq expression
    --json fields              Output JSON with the specified fields
-L, --limit int                Maximum number of commits to fetch (default
30)
    --merge                    Filter on merge commits
    --order string             Order of commits returned, ignored unless '-
-sort' flag is specified: {asc&#124;desc} (default "desc")
    --owner strings            Filter on repository owner
    --parent string            Filter by parent hash
-R, --repo strings             Filter on repository
    --sort string              Sort fetched commits: {author-
date&#124;committer-date} (default "best-match")
-t, --template string          Format JSON output using a Go template; see
"gh help formatting"
    --tree string              Filter by tree hash
    --visibility strings       Filter based on repository visibility:
{public&#124;private&#124;internal}
-w, --web                      Open the search query in the web browser
```

`gh search issues [<query>] [flags]`

Search for issues

```
    --app string             Filter by GitHub App author
    --archived               Filter based on the repository archived state
{true&#124;false}
    --assignee string        Filter by assignee
    --author string          Filter by author
    --closed date            Filter on closed at date
    --commenter user         Filter based on comments by user
    --comments number        Filter on number of comments
    --created date           Filter based on created at date
    --include-prs            Include pull requests in results
    --interactions number    Filter on number of reactions and comments
    --involves user          Filter based on involvement of user
-q, --jq expression          Filter JSON output using a jq expression
    --json fields            Output JSON with the specified fields
    --label strings          Filter on label
    --language string        Filter based on the coding language
-L, --limit int              Maximum number of results to fetch (default
30)
    --locked                 Filter on locked conversation status
    --match strings          Restrict search to specific field of issue:
{title&#124;body&#124;comments}
    --mentions user          Filter based on user mentions
    --milestone title        Filter by milestone title
    --no-assignee            Filter on missing assignee
    --no-label               Filter on missing label
    --no-milestone           Filter on missing milestone
    --no-project             Filter on missing project
    --order string           Order of results returned, ignored unless '--
sort' flag is specified: {asc&#124;desc} (default "desc")
    --owner strings          Filter on repository owner
    --project owner/number   Filter on project board owner/number
    --reactions number       Filter on number of reactions
-R, --repo strings           Filter on repository
    --sort string            Sort fetched results:
{comments&#124;created&#124;interactions&#124;reactions&#124;reactions-
+1&#124;reactions--1&#124;reactions-heart&#124;reactions-
smile&#124;reactions-tada&#124;reactions-thinking_face&#124;updated}
(default "best-match")
    --state string           Filter based on state: {open&#124;closed}
    --team-mentions string   Filter based on team mentions
-t, --template string        Format JSON output using a Go template; see
"gh help formatting"
    --updated date           Filter on last updated at date
    --visibility strings     Filter based on repository visibility:
{public&#124;private&#124;internal}
-w, --web                    Open the search query in the web browser
```

`gh search prs [<query>] [flags]`

Search for pull requests

```
    --app string              Filter by GitHub App author
    --archived                Filter based on the repository archived state
{true&#124;false}
    --assignee string         Filter by assignee
    --author string           Filter by author
-B, --base string             Filter on base branch name
    --checks string           Filter based on status of the checks:
{pending&#124;success&#124;failure}
    --closed date             Filter on closed at date
    --commenter user          Filter based on comments by user
    --comments number         Filter on number of comments
    --created date            Filter based on created at date
    --draft                   Filter based on draft state
-H, --head string             Filter on head branch name
    --interactions number     Filter on number of reactions and comments
    --involves user           Filter based on involvement of user
-q, --jq expression           Filter JSON output using a jq expression
    --json fields             Output JSON with the specified fields
    --label strings           Filter on label
    --language string         Filter based on the coding language
-L, --limit int               Maximum number of results to fetch (default
30)
    --locked                  Filter on locked conversation status
    --match strings           Restrict search to specific field of issue:
{title&#124;body&#124;comments}
    --mentions user           Filter based on user mentions
    --merged                  Filter based on merged state
    --merged-at date          Filter on merged at date
    --milestone title         Filter by milestone title
    --no-assignee             Filter on missing assignee
    --no-label                Filter on missing label
    --no-milestone            Filter on missing milestone
    --no-project              Filter on missing project
    --order string            Order of results returned, ignored unless '--
sort' flag is specified: {asc&#124;desc} (default "desc")
    --owner strings           Filter on repository owner
    --project owner/number    Filter on project board owner/number
    --reactions number        Filter on number of reactions
-R, --repo strings            Filter on repository
    --review string           Filter based on review status:
{none&#124;required&#124;approved&#124;changes_requested}
    --review-requested user   Filter on user or team requested to review
    --reviewed-by user        Filter on user who reviewed
    --sort string             Sort fetched results:
{comments&#124;reactions&#124;reactions-+1&#124;reactions--
1&#124;reactions-smile&#124;reactions-thinking_face&#124;reactions-
heart&#124;reactions-tada&#124;interactions&#124;created&#124;updated}
(default "best-match")
```

```
    --state string            Filter based on state: {open&#124;closed}
    --team-mentions string    Filter based on team mentions
-t, --template string         Format JSON output using a Go template; see
"gh help formatting"
    --updated date            Filter on last updated at date
    --visibility strings      Filter based on repository visibility:
{public&#124;private&#124;internal}
-w, --web                     Open the search query in the web browser
```

gh search repos [<query>] [flags]

Search for repositories

```
    --archived                    Filter based on the repository archived
state {true&#124;false}
    --created date                Filter based on created at date
    --followers number            Filter based on number of followers
    --forks number                Filter on number of forks
    --good-first-issues number    Filter on number of issues with the 'good
first issue' label
    --help-wanted-issues number   Filter on number of issues with the 'help
wanted' label
    --include-forks string        Include forks in fetched repositories:
{false&#124;true&#124;only}
-q, --jq expression               Filter JSON output using a jq expression
    --json fields                 Output JSON with the specified fields
    --language string             Filter based on the coding language
    --license strings             Filter based on license type
-L, --limit int                   Maximum number of repositories to fetch
(default 30)
    --match strings               Restrict search to specific field of
repository: {name&#124;description&#124;readme}
    --number-topics number        Filter on number of topics
    --order string                Order of repositories returned, ignored
unless '--sort' flag is specified: {asc&#124;desc} (default "desc")
    --owner strings               Filter on owner
    --size string                 Filter on a size range, in kilobytes
    --sort string                 Sort fetched repositories:
{forks&#124;help-wanted-issues&#124;stars&#124;updated} (default "best-
match")
    --stars number                Filter on number of stars
-t, --template string             Format JSON output using a Go template;
see "gh help formatting"
    --topic strings               Filter on topic
    --updated date                Filter on last updated at date
    --visibility strings          Filter based on visibility:
{public&#124;private&#124;internal}
-w, --web                         Open the search query in the web browser
```

gh secret <command>

Manage GitHub secrets

gh secret delete <secret-name> [flags]

Delete secrets

```
-a, --app string   Delete a secret for a specific application:
{actions&#124;codespaces&#124;dependabot}
-e, --env string   Delete a secret for an environment
-o, --org string   Delete a secret for an organization
-u, --user         Delete a secret for your user
```

Aliases

gh secret remove

`gh secret list [flags]`

List secrets

```
-a, --app string        List secrets for a specific application:
{actions&#124;codespaces&#124;dependabot}
-e, --env string        List secrets for an environment
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-o, --org string        List secrets for an organization
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
-u, --user              List a secret for your user
```

Aliases

gh secret ls

`gh secret set <secret-name> [flags]`

Create or update secrets

```
-a, --app string           Set the application for a secret:
{actions&#124;codespaces&#124;dependabot}
-b, --body string          The value for the secret (reads from standard
input if not specified)
-e, --env environment      Set deployment environment secret
-f, --env-file file        Load secret names and values from a dotenv-
formatted file
    --no-store             Print the encrypted, base64-encoded value
instead of storing it on GitHub
-o, --org organization     Set organization secret
-r, --repos repositories   List of repositories that can access an
organization or user secret
-u, --user                 Set a secret for your user
-v, --visibility string    Set visibility for an organization secret:
{all&#124;private&#124;selected} (default "private")
```

`gh ssh-key <command>`

Manage SSH keys

`gh ssh-key add [<key-file>] [flags]`

Add an SSH key to your GitHub account

```
-t, --title string   Title for the new key
    --type string    Type of the ssh key: {authentication&#124;signing}
(default "authentication")
```

`gh ssh-key delete <id> [flags]`

Delete an SSH key from your GitHub account

```
-y, --yes   Skip the confirmation prompt
```

`gh ssh-key list`

Lists SSH keys in your GitHub account

Aliases

gh ssh-key ls

gh status [flags]

Print information about relevant issues, pull requests, and notifications across repositories

```
-e, --exclude strings   Comma separated list of repos to exclude in
owner/name format
-o, --org string        Report status within an organization
```

gh variable <command>

Manage GitHub Actions variables

gh variable delete <variable-name> [flags]

Delete variables

```
-e, --env string   Delete a variable for an environment
-o, --org string   Delete a variable for an organization
```

Aliases

gh variable remove

gh variable get <variable-name> [flags]

Get variables

```
-e, --env string        Get a variable for an environment
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-o, --org string        Get a variable for an organization
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

gh variable list [flags]

List variables

```
-e, --env string        List variables for an environment
-q, --jq expression     Filter JSON output using a jq expression
    --json fields       Output JSON with the specified fields
-o, --org string        List variables for an organization
-t, --template string   Format JSON output using a Go template; see "gh
help formatting"
```

Aliases

gh variable ls

gh variable set <variable-name> [flags]

Create or update variables

```
-b, --body string           The value for the variable (reads from standard
input if not specified)
-e, --env environment       Set deployment environment variable
-f, --env-file file         Load variable names and values from a dotenv-
formatted file
-o, --org organization      Set organization variable
-r, --repos repositories    List of repositories that can access an
organization variable
-v, --visibility string     Set visibility for an organization variable:
{all&#124;private&#124;selected} (default "private")
```

gh workflow <command>

View details about GitHub Actions workflows

```
gh workflow disable [<workflow-id> &#124; <workflow-name>]
```
Disable a workflow

```
gh workflow enable [<workflow-id> &#124; <workflow-name>]
```
Enable a workflow

```
gh workflow list [flags]
```
List workflows

```
-a, --all                 Include disabled workflows
-q, --jq expression       Filter JSON output using a jq expression
    --json fields         Output JSON with the specified fields
-L, --limit int           Maximum number of workflows to fetch (default 50)
-t, --template string     Format JSON output using a Go template; see "gh
help formatting"
```
Aliases

gh workflow ls

```
gh workflow run [<workflow-id> &#124; <workflow-name>] [flags]
```
Run a workflow by creating a workflow_dispatch event

```
-F, --field key=value       Add a string parameter in key=value format,
respecting @ syntax (see "gh help api").
    --json                  Read workflow inputs as JSON via STDIN
-f, --raw-field key=value   Add a string parameter in key=value format
-r, --ref string            The branch or tag name which contains the
version of the workflow file you'd like to run
gh workflow view [<workflow-id> &#124; <workflow-name> &#124; <filename>]
[flags]
```
View the summary of a workflow

```
-r, --ref string   The branch or tag name which contains the version of the
workflow file you'd like to view
-w, --web          Open workflow in the browser
-y, --yaml         View the workflow yaml file
```