GitHub CLI quickstart

In this article

Start using GitHub CLI to work with GitHub in the command line.

About GitHub CLI

GitHub CLI is an open source tool for using GitHub from your computer's command line. When you're working from the command line, you can use the GitHub CLI to save time and avoid switching context.

Prerequisites

1. Install GitHub CLI on macOS, Windows, or Linux. For more information, see Installation in the GitHub CLI repository.
2. Authenticate with GitHub by running this command from your terminal.

   ```
   gh auth login
   ```

3. Follow the on-screen prompts.

   GitHub CLI automatically stores your Git credentials for you when you choose HTTPS as your preferred protocol for Git operations and answer "yes" to the prompt asking if you would like to authenticate to Git with your GitHub credentials. This can be useful as it allows you to use `git push`, `git pull`, and so on, without needing to set up a separate credential manager or use SSH.

Some useful commands

Note: When you use some commands for the first time - for example, `gh codespace SUBCOMMAND` - you'll be prompted to add extra scopes to your authentication token. Follow the onscreen instructions.

Viewing your status

Enter `gh status` to see details of your current work on GitHub across all the repositories

you're subscribed to.

Viewing a repository

Enter `gh repo view OWNER/REPO` to see the repository description and `README.md` for the repository. Enter `gh repo view OWNER/REPO --web` to view the repository in your default browser.

If you run the `repo` subcommand from within the directory of a local Git repository that has a remote on GitHub you can omit `OWNER/REPO`.

Cloning a repository

Enter `gh repo clone OWNER/REPO`. For example, `gh repo clone octo-org/octo-repo` clones the `octo-org/octo-repo` repository to the directory from which you ran this command on your local computer.

Creating a repository

Enter `gh repo create` and follow the on-screen instructions. You can create a new, empty repository on GitHub and then, optionally, clone it locally. Alternatively, you can push an existing local repository to GitHub, and optionally set it as the remote for your local repository. For information on setting a local directory as a Git repository, see "Adding locally hosted code to GitHub."

Working with issues

Enter `gh issue list --repo OWNER/REPO` to list the most recently created issues that are currently open for the specified repository. If you run the `issue` subcommand from within the directory of a local Git repository that has a remote on GitHub you can omit `--repo OWNER/REPO`. For example, enter `gh issue list --assignee "@me"` to list issues assigned to you in this repository, or `gh issue list --author monalisa` to list issues created by the user "monalisa."

You can also create a new issue, see "Creating an issue," or search for an issue, see "Filtering and searching issues and pull requests."

Working with pull requests

Enter `gh pr list --repo OWNER/REPO` to list the most recently created pull requests that are currently open for the specified repository. If you run the `pr` subcommand from within the directory of a local Git repository that has a remote on GitHub you can omit `--repo OWNER/REPO`. For example, enter `gh pr list --author "@me"` to list open pull requests that you created in this repository.

Enter `gh pr list --label LABEL-NAME` to list open pull requests with a specific label. Enter `gh search prs --review-requested=@me --state=open` to list pull requests that you've been asked to review.

To create a pull request, enter `gh pr create` and follow the on-screen instructions. For more information, see "Creating a pull request."

Working with codespaces

To create a new codespace, enter `gh codespace create` and follow the on-screen instructions.

To display your existing codespaces, enter `gh codespace list`. To open a codespace in

the web version of VS Code enter `gh codespace code -w` and choose a codespace .

In all of these commands you can substitute `cs` for `codespace`.

Getting help

Enter `gh` for a reminder of the top-level GitHub CLI commands that you can use. For example, `issue`, `pr`, `repo`, and so on.

For each command, and each subsidiary subcommand, you can append the `--help` flag to find out how it's used. For example, `gh issue --help` or `gh issue create --help`.

Customizing GitHub CLI

You can change configuration settings and add aliases or extensions, to make GitHub CLI work the way that suits you best.

- Enter `gh config set SUBCOMMANDS` to configure GitHub CLI's settings, replacing `SUBCOMMANDS` with the setting you want to adjust.

  For example, you can specify the text editor that's used when a GitHub CLI command requires you to edit text - such as when you add the body text for a new issue you're creating. To set your preferred text editor to Visual Studio Code enter `gh config set editor "code -w"`. The `-w` (or `--wait`) flag in this example causes the command to wait for the file to be closed in Visual Studio Code before proceeding with the next step in your terminal.

  For more information, see `gh config set`.

- Define aliases for commands that you commonly run. For example, if you run `gh alias set prd "pr create --draft"`, you will then be able to run `gh prd` to quickly open a draft pull request. For more information, see `gh alias`.

- Create or add custom commands with GitHub CLI extensions. For more information, see "Using GitHub CLI extensions" and "Creating GitHub CLI extensions."

Further reading

- GitHub CLI reference
- GitHub CLI online manual