GitHub CLI

Thodi charcha, thoda gyaan 😎

Resources used :

Official GitHub CLI docs. It’s kinda too wordy and unnecessarily **long and hard** (😏) so this is like a 1hr refresher directly into using most of the useful stuff w/o any probs

Checkout this video just to get refamiliarized with some of the git concepts :

https://www.youtube.com/watch?v=8JJ101D3knE

GitHub CLI

`gh` is GitHub on the command line. It brings pull requests, issues, and other GitHub concepts to the terminal next to where you are already working with git and your code.

Installation

Just head over to https://cli.github.com/ to download the msi file which you can install manually

Authentication

Run `gh auth login` to authenticate with your GitHub account. `gh` will respect tokens set using `GITHUB_TOKEN`.

Setting an editor

To set your preferred editor, you can use `gh config set editor <editor>`. Read more about `gh config`.
Additionally if the above is not set, for macOS and Linux,`gh` will respect the following environment variables, in this order, based on your OS and shell setup:

1. `GIT_EDITOR`
2. `VISUAL`
3. `EDITOR`

On Windows, the editor will currently always be Notepad.

1) HELP (--help)

Option available with –help flag universally can be combined with any cli command

```
     --help   Show help for command
```

2) SHORTCUTS (gh alias set)

Create a shortcut for a gh command

Synopsis

Declare a word as a command alias that will expand to the specified command(s).

The expansion may specify additional arguments and flags. If the expansion includes positional placeholders such as ‘$1’, ‘$2’, etc., any extra arguments that follow the invocation of an alias will be inserted appropriately (**kinda like tab stops**)

This rule only applies for shell scripts (**Windows Powershell**) :

If ‘–shell’ is specified, the alias will be run through a shell interpreter (sh). This allows you to compose commands with “|” or redirect with “>”. Note that extra arguments following the alias will not be automatically passed to the expanded expression. To have a shell alias receive arguments, you must explicitly accept them using “$1”, “$2”, etc., or “$@” to accept all of them.

Platform note: on Windows, shell aliases are executed via “sh” as installed by Git For Windows. If you have installed git on Windows in some other way, shell aliases may not work for you.

Quotes must always be used when defining a command as in the examples.

```
gh alias set <alias> <expansion> [flags]
```

Examples

```
$ gh alias set pv 'pr view'
$ gh pv -w 123
#=> gh pr view -w 123

$ gh alias set bugs 'issue list --label="bugs"'

$ gh alias set epicsBy 'issue list --author="$1" --label="epic"'
$ gh epicsBy vilmibm
#=> gh issue list --author="vilmibm" --label="epic"

$ gh alias set --shell igrep 'gh issue list --label="$1" | grep $2'
$ gh igrep epic foo
#=> gh issue list --label="epic" | grep "foo"
```

Options

```
  -s, --shell   Declare an alias to be passed through a shell interpreter
```

➢ DELETING SHORTCUTS (gh alias delete)

Delete an alias

Synopsis

Delete an alias

```
gh alias delete <alias> [flags]
```

➢ LISTING SHORTCUTS (gh alias list)

List your aliases

Synopsis

This command prints out all of the aliases gh is configured to use.

```
gh alias list [flags]
```

3) CLONE REPO (gh repo clone)

Clone a GitHub repository locally.

If the “OWNER/” portion of the “OWNER/REPO” repository argument is omitted, it defaults to the name of the authenticating user.

Pass additional ‘git clone’ flags by listing them after ‘–’.

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

In use

Using OWNER/REPO syntax

You can clone any repository using OWNER/REPO syntax.

```
# Cloning a repository
~/Projects$ gh repo clone cli/cli
```

Using other selectors

You can also use GitHub URLs to clone repositories.

```
# Cloning a repository
~/Projects/my-project$ gh repo clone https://github.com/cli/cli
```

4) CREATE REPO (gh repo create)

Create a new GitHub repository.

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

Examples

```
# create a repository under your account using the current directory name
$ gh repo create

# create a repository with a specific name
$ gh repo create my-project

# create a repository in an organization
$ gh repo create cli/my-project
```

Options

```
  -y, --confirm               Confirm the submission directly
  -d, --description string    Description of repository
      --enable-issues         Enable issues in the new repository (default true)
      --enable-wiki           Enable wiki in the new repository (default true)
  -h, --homepage string       Repository home page URL
      --internal              Make the new repository internal
      --private               Make the new repository private
      --public                Make the new repository public
  -t, --team string           The name of the organization team to be granted
access
  -p, --template string       Make the new repository based on a template
repository
```

In use

With no arguments

Inside a git repository, and with no arguments, `gh` will automatically create a repository on GitHub on your account for your current directory, using the directory name.

```
# Create a repository for the current directory.
~/Projects/my-project$ gh repo create
```

Setting a repository name

Enter a name to set a repository name other than the directory name.

```
# Create a repository in your organization
~/Projects/my-project$ gh repo create my-cool-project
```

Setting your organization as an owner

Use OWNER/REPO syntax to create a repository under an organization that you are a part of.

```
# Create a repository in your organization
~/Projects/my-project$ gh repo create org/repo
```

With flags

Use flags to choose your repository settings.

```
# Create a repository using flags
~/Projects/my-project$ gh repo create --enable-issues=false -public
```

5) FORK REPO (gh repo fork)

Create a fork of a repository.

With no argument, creates a fork of the current repository. Otherwise, forks the specified repository.

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

Options

```
      --clone    Clone the fork {true|false}
      --remote   Add remote for fork {true|false}
```

In use

With no arguments

Inside a git repository, and without any arguments, we will automatically create a fork on GitHub on your account for your current directory. It will then prompt if you want to set an upstream remote.

```
# Create a fork for the current repository.
```

```
~/Projects/cli$ gh repo fork
```

With arguments

If you pass a repository in OWNER/REPO format, `gh` will automatically create a fork on GitHub on your account and ask if you want to clone it. This works inside or outside of a git repository.

```
# Create a fork for another repository.
~/Projects$ gh repo fork cli/cli
```

Using flags

Use flags to skip prompts about adding a git remote for the fork, or about cloning the forked repository locally.

```
# Skipping remote prompts using flags
~/Projects/cli$ gh repo fork --remote=false
```

6) VIEW REPO (gh repo view)

Synopsis

Display the description and the README of a GitHub repository.

With no argument, the repository for the current directory is displayed.

With '–web', open the repository in a web browser instead.

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

Options

```
  -w, --web    Open a repository in the browser
```

In use

In terminal

By default, we will display items in the terminal.

```
# Viewing a repository in terminal
~/Projects/my-project$ gh repo view owner/repo
```

```
owner/repo
Repository description

  Repository README

View this repository on GitHub: https://github.com/owner/repo/
~/Projects/my-project$
```

In the browser

Quickly open an item in the browser using `--web` or `-w`

```
# Viewing a repository in the browser
~/Projects$ gh repo view owner/repo --web
Opening https://github.com/owner/repo/ in your browser.
~/Projects$
```

With no arguments

Display the repository you're currently in.

```
# Viewing the repository you're in
~/Projects/my-project$ gh repo view
```

```
owner/my-project
Repository description

  Repository README

View this repository on GitHub: https://github.com/owner/repo/
~/Projects/my-project$
```

7) PULL REQUESTS (PR)

➢ CHECKOUT PRs (gh pr checkout)

Check out a pull request in git

Synopsis

Check out a pull request in git

```
gh pr checkout {<number> | <url> | <branch>} [flags]
```

Options

```
      --recurse-submodules   Update all active submodules (recursively)
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

In use

Using pull request number

You can check out any pull request, including from forks, in a repository using its pull request number

```
// Checking out a pull request locally
~/Projects/my-project$ gh pr checkout 12
```

Using other selectors

You can also use URLs and branch names to checkout pull requests.

```
// Checking out a pull request locally
~/Projects/my-project$ gh pr checkout branch-name

~/Projects/my-project$
```

➢ CHECK PR STATUS (gh pr checks)

Show CI status for a single pull request – **mhanje approve kiya ya pending hai ya reject kiya pr ko**

Synopsis

Show CI status for a single pull request

```
gh pr checks [flags]
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/RE
```

➢ CLOSING PR (gh pr close)

Close a pull request

Synopsis

Close a pull request

```
gh pr close {<number> | <url> | <branch>} [flags]
```

Options

```
  -d, --delete-branch   Delete the local and remote branch after close
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

➢ CREATING A NEW PR (gh pr create)

Create a pull request

Synopsis

Create a pull request on GitHub.

When the current branch isn’t fully pushed to a git remote, a prompt will ask where to push the branch and offer an option to fork the base repository. Use ‘–head’ to explicitly skip any forking or pushing behavior.

A prompt will also ask for the title and the body of the pull request. Use '–title' and '–body' to skip this, or use '–fill' to autofill these values from git commits.

```
gh pr create [flags]
```

Examples

```
$ gh pr create --title "The bug is fixed" --body "Everything works again"
$ gh pr create --reviewer monalisa,hubot
$ gh pr create --project "Roadmap"
$ gh pr create --base develop --head monalisa:feature
```

Options

```
  -a, --assignee login   Assign people by their login
  -B, --base branch      The branch into which you want your code merged
  -b, --body string      Body for the pull request
  -d, --draft            Mark pull request as a draft
  -f, --fill             Do not prompt for title/body and just use commit info
  -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
  -p, --project name     Add the pull request to projects by name
  -r, --reviewer login   Request reviews from people by their login
  -t, --title string     Title for the pull request
  -w, --web              Open the web browser to create a pull request
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

In use

Interactively

```
# Create a pull request interactively
~/Projects/my-project$ gh pr create
```

With flags

```
# Create a pull request using flags
~/Projects/my-project$ gh pr create --title "Pull request title" --body "Pull request body"
```

In the browser

```
// Quickly navigate to the pull request creation page
~/Projects/my-project$ gh pr create --web
```

Working with forks

This command will automatically create a fork for you if you're in a repository that you don't have permission to push to.

➢ SEE CHANGES TO CODE AS GIVEN IN PR (gh pr diff)

View changes in a pull request

Synopsis

View changes in a pull request

```
gh pr diff [<number> | <url> | <branch>] [flags]
```

Options

```
      --color string   Use color in diff output: {always|never|auto} (default
"auto")
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

➢ LIST ALL PRs IN REPO (gh pr list)

List and filter pull requests in this repository

Synopsis

List and filter pull requests in this repository

```
gh pr list [flags]
```

Examples

```
$ gh pr list --limit 999
$ gh pr list --state closed
$ gh pr list --label "priority 1" --label "bug"
$ gh pr list --web
```

Options

```
  -a, --assignee string   Filter by assignee
  -B, --base string       Filter by base branch
  -l, --label strings     Filter by labels
  -L, --limit int         Maximum number of items to fetch (default 30)
  -s, --state string      Filter by state: {open|closed|merged|all} (default
"open")
  -w, --web               Open the browser to list the pull requests
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

In use

Default behavior

You will see the most recent 30 open items.

```
# Viewing a list of open pull requests
~/Projects/my-project$ gh pr list

Pull requests for owner/repo

#14  Upgrade to Prettier 1.19                          prettier
#14  Extend arrow navigation in lists for MacOS        arrow-nav
#13  Add Support for Windows Automatic Dark Mode       dark-mode
#8   Create and use keyboard shortcut react component   shortcut

~/Projects/my-project$
```

Filtering with flags

You can use flags to filter the list for your specific use cases.

```
# Viewing a list of closed pull requests assigned to a user
~/Projects/my-project$ gh pr list --state closed --assignee user

Pull requests for owner/repo

#13  Upgrade to Electron 7         electron-7
#8   Release Notes Writing Guide   release-notes

~/Projects/my-project$
```

➢ MERGIN PRs (gh pr merge)

Merge a pull request

Synopsis

Merge a pull request on GitHub.

By default, the head branch of the pull request will get deleted on both remote and local repositories. To retain the branch, use '–delete-branch=false'.

```
gh pr merge [<number> | <url> | <branch>] [flags]
```

Options

```
  -d, --delete-branch   Delete the local and remote branch after merge (default
true)
  -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
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

➢ REOPEN A CLOSED PR (gh pr reopen)

Reopen a pull request

Synopsis

Reopen a pull request

```
gh pr reopen {<number> | <url> | <branch>} [flags]
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

➢ REVIEWING PRs (gh pr review)

Add a review to a pull request

Synopsis

Add a review to a pull request.

Without an argument, the pull request that belongs to the current branch is reviewed.

```
gh pr review [<number> | <url> | <branch>] [flags]
```

Examples

```
# approve the pull request of the current branch
$ gh pr review --approve

# leave a review comment for the current branch
$ gh pr review --comment -b "interesting"

# add a review for a specific pull request
$ gh pr review 123

# request changes on a specific pull request
$ gh pr review 123 -r -b "needs more ASCII art"
```

Options

```
  -a, --approve           Approve pull request
  -b, --body string       Specify the body of a review
  -c, --comment           Comment on a pull request
  -r, --request-changes   Request changes on a pull request
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

➢ SEE STATUS OF A PR (gh pr status)

Show status of relevant pull requests

Synopsis

Show status of relevant pull requests

```
gh pr status [flags]
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

In use

```
# Viewing the status of your relevant pull requests
~/Projects/my-project$ gh pr status
```

```
Current branch
  #12 Remove the test feature [user:patch-2]
   - All checks failing - Review required
```

```
Created by you
  You have no open pull requests

Requesting a code review from you
  #13 Fix tests [branch]
  - 3/4 checks failing - Review required
  #15 New feature [branch]
   - Checks passing - Approved

~/Projects/my-project$
```

➢ VIEW PRs (gh pr view)

View a pull request

Synopsis

Display the title, body, and other information about a pull request.

Without an argument, the pull request that belongs to the current branch is displayed.

With '–web', open the pull request in a web browser instead.

```
gh pr view [<number> | <url> | <branch>] [flags]
```

Options

```
  -w, --web   Open a pull request in the browser
```

Options inherited from parent commands

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

In use

In terminal

By default, we will display items in the terminal.

```
# Viewing a pull request in terminal
~/Projects/my-project$ gh pr view 21
```

In the browser

Quickly open an item in the browser using `--web` or `-w`

```
# Viewing a pull request in the browser
~/Projects/my-project$ gh pr view 21 --web
```

With no arguments

We will display the pull request of the branch you're currently on.

```
# Viewing the pull request of the branch you're on
~/Projects/my-project$ gh pr view
```

8) ISSUES

gh issue

Manage issues

Synopsis

Work with GitHub issues

Examples

```
$ gh issue list
$ gh issue create --label bug
$ gh issue view --web
```

Options

```
  -R, --repo OWNER/REPO   Select another repository using the OWNER/REPO format
```

Options inherited from parent commands

```
      --help   Show help for command
```

Mhanje these 2 options will be available in all commands of gh issue ☝

➢ CLOSING ISSUES (gh issue close)

Close issue

Synopsis

Close issue

```
gh issue close {<number> | <url>} [flags]
```

➢ CREATING AN ISSUE (gh issue create)

Create a new issue

Synopsis

Create a new issue

```
gh issue create [flags]
```

Examples

```
$ gh issue create --title "I found a bug" --body "Nothing works"
$ gh issue create --label "bug,help wanted"
$ gh issue create --label bug --label "help wanted"
$ gh issue create --assignee monalisa,hubot
$ gh issue create --project "Roadmap"
```

Options

```
  -a, --assignee login   Assign people by their login
  -b, --body string      Supply a body. Will prompt for one otherwise.
  -l, --label name       Add labels by name
  -m, --milestone name   Add the issue to a milestone by name
  -p, --project name     Add the issue to projects by name
  -t, --title string     Supply a title. Will prompt for one otherwise.
  -w, --web              Open the browser to create an issue
OWNER/REPO format
```

In use

Interactively

```
# Create an issue interactively
~/Projects/my-project$ gh issue create
```

With flags

```
# Create an issue using flags
~/Projects/my-project$ gh issue create --title "Issue title" --body "Issue body"
```

In the browser

```
// Quickly navigate to the issue creation page
~/Projects/my-project$ gh issue create --web
```

➢ LISTING ISSUES (gh issue list)

List and filter issues in this repository

Synopsis

List and filter issues in this repository

```
gh issue list [flags]
```

Examples

```
$ gh issue list -l "help wanted"
$ gh issue list -A monalisa
$ gh issue list --web
$ gh issue list --milestone 'MVP'
```

Options

```
  -a, --assignee string    Filter by assignee
  -A, --author string      Filter by author
  -l, --label strings      Filter by labels
  -L, --limit int          Maximum number of issues to fetch (default 30)
      --mention string     Filter by mention
  -m, --milestone number   Filter by milestone number or `title`
  -s, --state string       Filter by state: {open|closed|all} (default "open")
  -w, --web                Open the browser to list the issue(s)
using the OWNER/REPO format
```

In use

Default behavior

You will see the most recent 30 open items.

```
# Viewing a list of open issues
~/Projects/my-project$ gh issue list
```

```
Issues for owner/repo

#14  Update the remote url if it changed  (bug)
#14  PR commands on a detached head       (enhancement)
#13  Support for GitHub Enterprise        (wontfix)
#8   Add an easier upgrade command        (bug)
```

Filtering with flags

You can use flags to filter the list for your specific use cases.

```
# Viewing a list of closed issues assigned to a user
~/Projects/my-project$ gh issue list --state closed --assignee user
```

➢ REOPEN CLOSED ISSUES (gh issue reopen)

Reopen issue

Synopsis

Reopen issue

```
gh issue reopen {<number> | <url>} [flags]
using the OWNER/REPO format
```

➢ VIEWIN ISSUE STATUS (gh issue status)

Show status of relevant issues

Synopsis

Show status of relevant issues

```
gh issue status [flags]using the OWNER/REPO format
```

In use

```
# Viewing issues relevant to you
~/Projects/my-project$ gh issue status
```

```
Issues assigned to you
  #8509 [Fork] Improve how Desktop handles forks  (epic:fork, meta)

Issues mentioning you
  #8938 [Fork] Add create fork flow entry point at commit warning  (epic:fork)
  #8509 [Fork] Improve how Desktop handles forks  (epic:fork, meta)

Issues opened by you
  #8936 [Fork] Hide PR number badges on branches that have an upstream PR
(epic:fork)
  #6386 Improve no editor detected state on conflicts modal  (enhancement)
```

➢ VIEWIN ISSUES (gh issue view)

View an issue

Synopsis

Display the title, body, and other information about an issue.

With '–web', open the issue in a web browser instead.

```
gh issue view {<number> | <url>} [flags]
```

Options

```
  -w, --web   Open an issue in the browser
repository using the OWNER/REPO format
```

In use

In terminal

By default, we will display items in the terminal.

```
# Viewing an issue in terminal
~/Projects/my-project$ gh issue view 21
```

```
Issue title
opened by user. 0 comments. (label)

  Issue body

View this issue on GitHub: https://github.com/owner/repo/issues/21
~/Projects/my-project$
```

In the browser

Quickly open an item in the browser using `--web` or `-w`

```
# Viewing an issue in the browser
~/Projects/my-project$ gh issue view 21 --web
```