**DÙNG GITHUB:**

**Nội dung:**

1. [Thiết lập chứng thực cá nhân](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Thiet_lap_chung_thuc_ca_nhan)
2. [Tạo một kho chứa Git](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Tao_mot_kho_chua_Git)
3. [Sao chép một kho chứa đã tồn tại](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Sao_chep_mot_kho_chua_da_ton_tai)
4. [Nhánh trong git](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Nhanh_trong_git)
5. [Chuyển nhánh](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Chuyen_nhanh)
6. [Cập nhật thay đổi](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Cap_nhat_thay_doi)
7. [Cập nhật lên server](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Cap_nhat_len_server)
8. [Gộp nhánh](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Gop_nhanh)
9. [Xem lại lịch sử commit](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Xem_lai_lich_su_commit)
10. [Xem thay đổi trước khi push](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Xem_thay_doi_truoc_khi_push)
11. [Gộp commit](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Gop_commit)
12. [Pull từ remote repository](https://jobs.hybrid-technologies.vn/blog/nhung-lenh-git-co-ban-can-nho/#Pull_tu_remote_repository)

**Phần chứng thực theo cmd:**

Thiết lập chứng thực cá nhân:

$ git config --global user.name "User Name"

Ví dụ: git config –-global user.name “dangvanthuan”

$ git config --global user.email [username@gmail.com](mailto:username@gmail.com)

Ví dụ: git config –global user.email “dangvanthuan26@gmail.com”

Lưu ý: –global được sử dụng để áp dụng cho tất cả các projects. Nếu không sử dụng –global thì settings sẽ chỉ dùng cho riêng project đó.

Tạo một kho chứa Git

$ git init

Nếu như bạn muốn theo dõi một dự án cũ trong Git, bạn cần ở trong thư mục của dự án đó. Lệnh này sẽ tạo một thư mục mới có tên .git, thư mục này chứa tất cả các tập tin cần thiết cho kho chứa.

#### ****Sao chép một kho chứa đã tồn tại****

$ git clone https://github.com/user/repository.git

Câu lệnh trên sẽ tạo một thư mục mới có tên giống trên của Repo.

#### ****Nhánh trong git****

Khi sử dụng Git, bạn có thể tạo ra nhiều nhánh (branch) khác nhau. Câu lệnh Git này dùng để kiểm tra branch hiện tại:

$ git branch

Để tạo mới một branch:

$ git branch <name\_branch>

Để chuyển và tạo mới:

$ git branch -b <name\_branch>

#### ****Chuyển nhánh****

Trước khi muốn thay đổi source code, điều đầu tiên mà bạn cần phải làm là checkout một nhánh. Để checkout một nhánh, bạn dùng câu lệnh Git sau:

$ git checkout <name\_branch>

#### ****Cập nhật thay đổi****

Sau khi bạn thay đổi source code: thêm mới, sửa, xoá files,… Bạn cần phải cập nhật lên Staging Area. Để cập nhật hết các files:

$ git add .

Sau lệnh add, bạn cần sử dụng câu lệnh Commit để đây thông tin thay đổi lên Local Respository:

$ git commit -m "Message"

#### ****Cập nhật lên server****

Sau câu lệnh Commit, thông tin mới chỉ được cập nhật lên Local Repository. Nếu muốn cập nhật lên server thì bạn phải sử dụng câu lệnh push:

$ git push origin <name\_branch>

Ngoài ra, nếu chưa tồn tại remote trên server thì bạn cần phải add mới một remote trước rồi mới push:

$ git remote add origin <remote\_url>

$ git push origin <name\_branch>

#### ****Gộp nhánh****

Sau một thời gian cập nhật các file và push lên git trên branch mới, bây giờ mình cần ghép (merge) code lại vào nhánh gốc (master). Trước tiên, cần phải checkout ra khỏi branch hiện tại cần gộp để vào branch master, sau đó thì dùng lệnh merge để ghép branch mới vào master:

$ git checkout master

$ git merge <new\_branch>

#### ****Xem lại lịch sử commit****

$ git log

Lệnh git log sẽ cho bạn biết về người commit, ngày giờ, message của những lần commit đó.

#### ****Xem thay đổi trước khi push****

$ git diff

Lệnh này giúp bạn biết những gì đã được thay đổi giữa nhánh hiện tại và nhánh trước nó.

#### ****Gộp commit****

$ git rebase -i HEAD~

Sau dấu ~ là số commit muốn gộp. Sau khi gõ lệnh này một cửa sổ trình soạn thảo hiện ra. Thay đổi ký tự pick của dòng các dòng sau dòng đầu thành s rồi lưu lại/kết thúc. Khi đó, trình soạn thảo để chỉnh sửa giải thích commit thiết lập cho commit sau khi đã tổng hợp sẽ được hiển thị, nên hãy chỉnh sửa lưu lại/kết thúc.

#### ****Pull từ remote repository****

$ git pull origin master

Lệnh trên sẽ gộp những thay đổi mới kéo về từ máy chủ từ xa với nhánh hiện tại trên máy local.

Phần 1: Cài đặt và tạo liên kết chứng thực GITHUB local và GITHUB online

* Địa chỉ tải: <https://git-scm.com/downloads>
* Graphical user interface, application, website

  Description automatically generated
* Cài đặt:
  + Graphical user interface, text, application, email

    Description automatically generated
  + Chọn NEXT
  + Graphical user interface, text, application, email

    Description automatically generated
  + Chọn NEXT tiếp theo
  + Graphical user interface, text, application, email

    Description automatically generated
  + Chọn NEXT
  + Graphical user interface, text, application, email

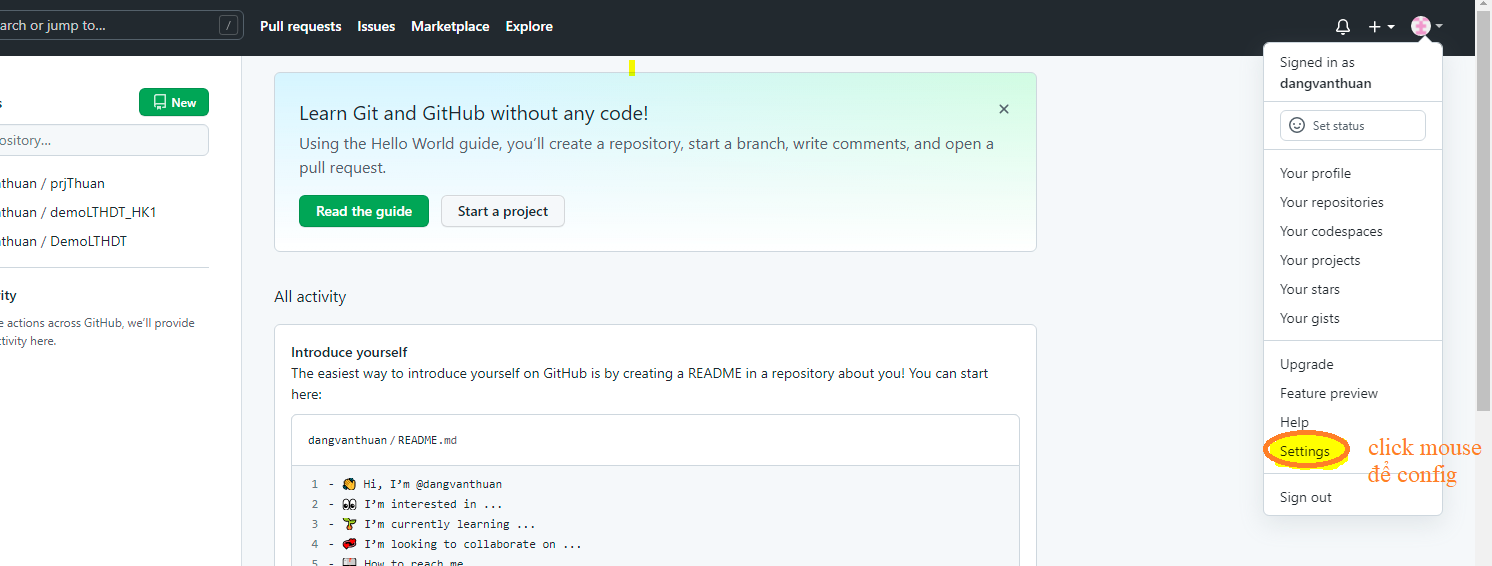
    Description automatically generated
  + OK thành công
* Tạo tài khoản GITHUB online:
  + Graphical user interface, website

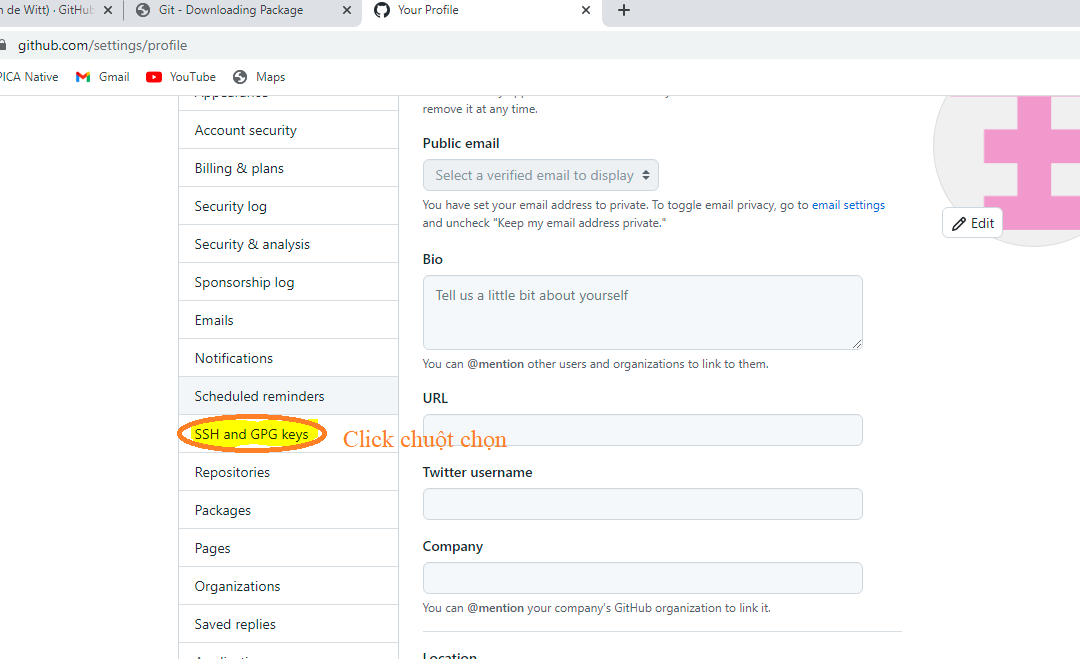
    Description automatically generated
  + Click sign up for GitHub
  + Graphical user interface, application

    Description automatically generated
  + Click chọn continue

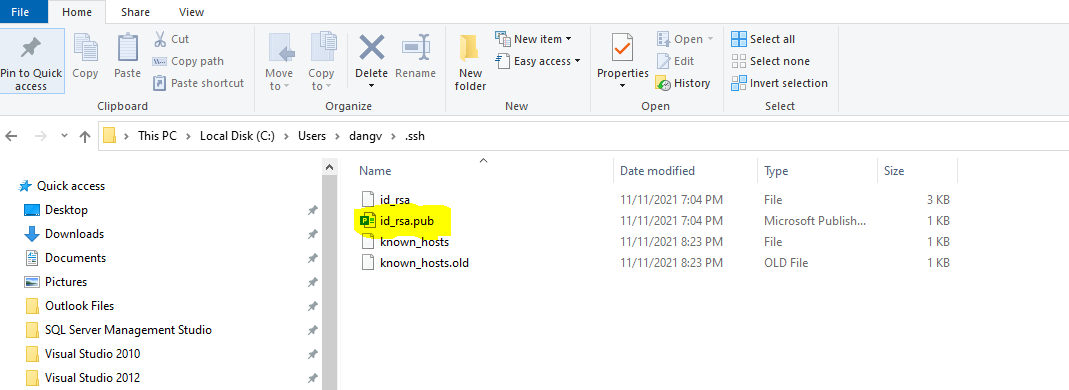
Graphical user interface, text, application

Description automatically generated



* Chọn theo mục:
* 
  + Graphical user interface, text, application

    Description automatically generated
  + HÌNH ####
  + Setting: copy SSH từ máy tính của bạn vào:
    - Mở my computer:



* **Nếu máy tính đang ẩn file phần mở rộng ta cấu hình view option folder:**
  + Graphical user interface, text, application

    Description automatically generated
* **sao cho thấy được đuôi flie:**
  + **Id\_rsa.pub**
  + **Mở file này với wordpad lên và chép phần nộp dung dáng vào** HÌNH ####

**Thực hiện liên kết xong giữa máy tính và tài khoản online của github**

* Phần 2: Tạo thư mục chứa project file code
  + Kích chuột phải chuột vào thư mục vừa tạo chọn:
* Text

  Description automatically generated

**Step 1: Tạo Repository để lưu code offline**

- CMD: git init

**step 2: xem trạng thái Repository**

- CMD: git status

**step 3: Tạo nhánh mới có tên develop**

- CMD: git checkout -b develop

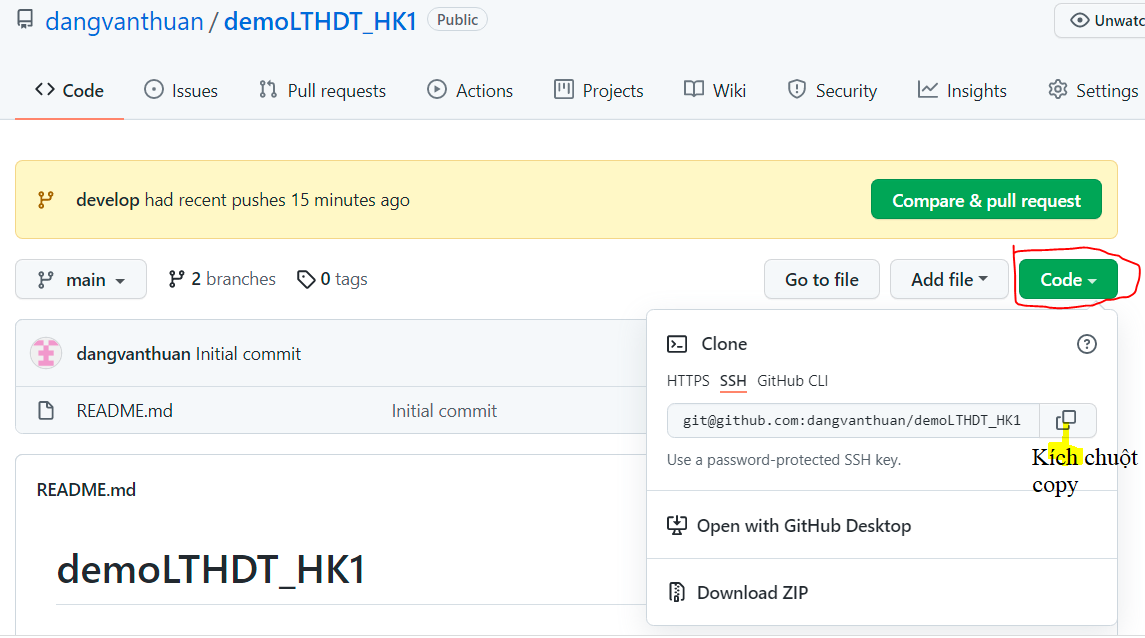
**step 4: Đưa file vào môi trường github:**

- CMD: git add <tên file> // Đưa một file vào môi trường github

- CMD: git add . // Đưa tất cả các file vào môi trường github

- CMD: git commit -m “lần xác nhận” // vd: chuổi xác nhận ‘origin’ xác nhận file vào môi trường github nhớ close các file

**step 5: Ðưa lên online github:**



- CMD: git remote add dautien [git@github.com:dangvanthuan/demoLTHDT\_HK1.git//](mailto:git@github.com:dangvanthuan/demoLTHDT_HK1.git//) chép code liên kết:

**step 6: xem kết quả:**

- CMD: git remote -v

$ git remote -v

dautien git@github.com:dangvanthuan/demoLTHDT\_HK1.git (fetch)

dautien git@github.com:dangvanthuan/demoLTHDT\_HK1.git (push)

**kết thúc:**

$ git push dautien develop

The authenticity of host 'github.com (20.205.243.166)' can't be established.

RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.

This key is not known by any other names

Are you sure you want to continue connecting (yes/no/[fingerprint])? y

Please type 'yes', 'no' or the fingerprint: yes

Warning: Permanently added 'github.com' (RSA) to the list of known hosts.

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 4 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (4/4), 336 bytes | 112.00 KiB/s, done.

Total 4 (delta 0), reused 0 (delta 0), pack-reused 0

remote:

remote: Create a pull request for 'develop' on GitHub by visiting:

remote: https://github.com/dangvanthuan/demoLTHDT\_HK1/pull/new/develop

remote:

To github.com:dangvanthuan/demoLTHDT\_HK1.git

\* [new branch] develop -> develop

**step 7: Tạo nhánh cho repository:**

- CMD: git checkout -b <tên nhánh> hoặc nhánh con theo <nhanh1/nhanh2>

dangv@PAUL MINGW64 /d/GitProjectCode/code (develop)

$ git checkout -b develop2/develop3

Switched to a new branch 'develop2/develop3'

- CMD: git checkout <tên nhánh> // về nhánh có tên chỉ định

Step 8: xóa nhánh :

-CMD: git branch -d <tên nhánh>

Các cú pháp câu lệnh

dangv@PAUL MINGW64 /c/baitap (baitapv1)

$ git status -?

error: unknown switch `?'

usage: git status [<options>] [--] <pathspec>...

-v, --verbose be verbose

-s, --short show status concisely

-b, --branch show branch information

--show-stash show stash information

--ahead-behind compute full ahead/behind values

--porcelain[=<version>]

machine-readable output

--long show status in long format (default)

-z, --null terminate entries with NUL

-u, --untracked-files[=<mode>]

show untracked files, optional modes: all, normal, no. (Default: all)

--ignored[=<mode>] show ignored files, optional modes: traditional, matching, no. (Default: traditional)

--ignore-submodules[=<when>]

ignore changes to submodules, optional when: all, dirty, untracked. (Default: all)

--column[=<style>] list untracked files in columns

--no-renames do not detect renames

-M, --find-renames[=<n>]

detect renames, optionally set similarity index

--show-ignored-directory

(DEPRECATED: use --ignore=matching instead) Only show directories that match an ignore pattern name.

--no-lock-index (DEPRECATED: use `git --no-optional-locks status` instead) Do not lock the index

dangv@PAUL MINGW64 /c/baitap (baitapv1)

$ git init -?

error: unknown switch `?'

usage: git init [-q | --quiet] [--bare] [--template=<template-directory>] [--shared[=<permissions>]] [<directory>]

--template <template-directory>

directory from which templates will be used

--bare create a bare repository

--shared[=<permissions>]

specify that the git repository is to be shared amongst several users

-q, --quiet be quiet

--separate-git-dir <gitdir>

separate git dir from working tree

-b, --initial-branch <name>

override the name of the initial branch

--object-format <hash>

specify the hash algorithm to use

dangv@PAUL MINGW64 /c/baitap (baitapv1)

$ git checkout -?

error: unknown switch `?'

usage: git checkout [<options>] <branch>

or: git checkout [<options>] [<branch>] -- <file>...

-b <branch> create and checkout a new branch

-B <branch> create/reset and checkout a branch

-l create reflog for new branch

--guess second guess 'git checkout <no-such-branch>' (default)

--overlay use overlay mode (default)

-q, --quiet suppress progress reporting

--recurse-submodules[=<checkout>]

control recursive updating of submodules

--progress force progress reporting

-m, --merge perform a 3-way merge with the new branch

--conflict <style> conflict style (merge or diff3)

-d, --detach detach HEAD at named commit

-t, --track set upstream info for new branch

-f, --force force checkout (throw away local modifications)

--orphan <new-branch>

new unparented branch

--overwrite-ignore update ignored files (default)

--ignore-other-worktrees

do not check if another worktree is holding the given ref

-2, --ours checkout our version for unmerged files

-3, --theirs checkout their version for unmerged files

-p, --patch select hunks interactively

--ignore-skip-worktree-bits

do not limit pathspecs to sparse entries only

--pathspec-from-file <file>

read pathspec from file

--pathspec-file-nul with --pathspec-from-file, pathspec elements are separated with NUL character

dangv@PAUL MINGW64 /c/baitap (baitapv1)

$ git branch -?

error: unknown switch `?'

usage: git branch [<options>] [-r | -a] [--merged] [--no-merged]

or: git branch [<options>] [-l] [-f] <branch-name> [<start-point>]

or: git branch [<options>] [-r] (-d | -D) <branch-name>...

or: git branch [<options>] (-m | -M) [<old-branch>] <new-branch>

or: git branch [<options>] (-c | -C) [<old-branch>] <new-branch>

or: git branch [<options>] [-r | -a] [--points-at]

or: git branch [<options>] [-r | -a] [--format]

Generic options

-v, --verbose show hash and subject, give twice for upstream branch

-q, --quiet suppress informational messages

-t, --track set up tracking mode (see git-pull(1))

-u, --set-upstream-to <upstream>

change the upstream info

--unset-upstream unset the upstream info

--color[=<when>] use colored output

-r, --remotes act on remote-tracking branches

--contains <commit> print only branches that contain the commit

--no-contains <commit>

print only branches that don't contain the commit

--abbrev[=<n>] use <n> digits to display object names

Specific git-branch actions:

-a, --all list both remote-tracking and local branches

-d, --delete delete fully merged branch

-D delete branch (even if not merged)

-m, --move move/rename a branch and its reflog

-M move/rename a branch, even if target exists

-c, --copy copy a branch and its reflog

-C copy a branch, even if target exists

-l, --list list branch names

--show-current show current branch name

--create-reflog create the branch's reflog

--edit-description edit the description for the branch

-f, --force force creation, move/rename, deletion

--merged <commit> print only branches that are merged

--no-merged <commit> print only branches that are not merged

--column[=<style>] list branches in columns

--sort <key> field name to sort on

--points-at <object> print only branches of the object

-i, --ignore-case sorting and filtering are case insensitive

--format <format> format to use for the output

dangv@PAUL MINGW64 /c/baitap (baitapv1)

$ git add -?

error: unknown switch `?'

usage: git add [<options>] [--] <pathspec>...

-n, --dry-run dry run

-v, --verbose be verbose

-i, --interactive interactive picking

-p, --patch select hunks interactively

-e, --edit edit current diff and apply

-f, --force allow adding otherwise ignored files

-u, --update update tracked files

--renormalize renormalize EOL of tracked files (implies -u)

-N, --intent-to-add record only the fact that the path will be added later

-A, --all add changes from all tracked and untracked files

--ignore-removal ignore paths removed in the working tree (same as --no-all)

--refresh don't add, only refresh the index

--ignore-errors just skip files which cannot be added because of errors

--ignore-missing check if - even missing - files are ignored in dry run

--chmod (+|-)x override the executable bit of the listed files

--pathspec-from-file <file>

read pathspec from file

--pathspec-file-nul with --pathspec-from-file, pathspec elements are separated with NUL character

dangv@PAUL MINGW64 /c/baitap (baitapv1)

$ git commit -?

error: unknown switch `?'

usage: git commit [<options>] [--] <pathspec>...

-q, --quiet suppress summary after successful commit

-v, --verbose show diff in commit message template

Commit message options

-F, --file <file> read message from file

--author <author> override author for commit

--date <date> override date for commit

-m, --message <message>

commit message

-c, --reedit-message <commit>

reuse and edit message from specified commit

-C, --reuse-message <commit>

reuse message from specified commit

--fixup [(amend|reword):]commit

use autosquash formatted message to fixup or amend/reword specified commit

--squash <commit> use autosquash formatted message to squash specified commit

--reset-author the commit is authored by me now (used with -C/-c/--amend)

--trailer <trailer> add custom trailer(s)

-s, --signoff add a Signed-off-by trailer

-t, --template <file>

use specified template file

-e, --edit force edit of commit

--cleanup <mode> how to strip spaces and #comments from message

--status include status in commit message template

-S, --gpg-sign[=<key-id>]

GPG sign commit

Commit contents options

-a, --all commit all changed files

-i, --include add specified files to index for commit

--interactive interactively add files

-p, --patch interactively add changes

-o, --only commit only specified files

-n, --no-verify bypass pre-commit and commit-msg hooks

--dry-run show what would be committed

--short show status concisely

--branch show branch information

--ahead-behind compute full ahead/behind values

--porcelain machine-readable output

--long show status in long format (default)

-z, --null terminate entries with NUL

--amend amend previous commit

--no-post-rewrite bypass post-rewrite hook

-u, --untracked-files[=<mode>]

show untracked files, optional modes: all, normal, no. (Default: all)

--pathspec-from-file <file>

read pathspec from file

--pathspec-file-nul with --pathspec-from-file, pathspec elements are separated with NUL character