Git là hệ thống kiểm soát các phiên bản phân tán mã nguồn mở dùng để hỗ trợ các thao tác trên GitHub ở laptop hoặc máy tính để bàn. Bảng này là tổng hợp các hướng dẫn dòng lệnh Git thường được sử dụng

**MAKE CHANGES**

$ git status:

* Để kiểm tra trạng thái của những file bạn đã thay đổi trong thư mục làm việc (Tất cả các thay đổi cuối cùng từ lần commit cuối cùng: Nhánh có đang up-to-date, Có gì cần pull/push/ commit hay không? Có file nào đã staged, unstaged hay untracked?, Có file mới được tạo/sửa/xóa hay không?)

$ git diff:

* Kiểm tra sự khác nhau, thay đổi giữa thư mục làm việc và commit cuối, giữa index và commit cuối, giữa các nhánh

$ git add [file]:

* [file]: tên file bạn muốn commit
* Dùng để commit những thực hiện thay đổi trong kho chứa (thêm/xóa/sửa file) những thay đổi này chỉ được thực hiện ở máy cá nhân

$ git diff --staged:

* --staged: phiên bản được commit cuối cùng
* Kiểm tra sự thay đổi của index (staging) với phiên bản cũ (commit cuối)

$ git reset [file]:

* [file]: tên file bạn muốn hoàn nguyên
* Trả lại thư mục đã git add, không thay đổi nội dung

$ git commit -m "[descriptive message]":

* -m: đã sửa đổi (modified)
* [descriptive message]: tóm tắt nội dung thay đổi bằng lời nhắn
* Commit thay đổi cùng với một thông báo tùy chỉnh

**CONFIGURE TOOLING**

$ git config --global user.name "[name]":

* git config: Để đặt user name và email của bạn trong file cấu hình chính
* --global: được sử dụng để áp dụng cho tất cả các dự án
* user.name [name]: tên để sử dụng khi commit
* Đặt tên mà bạn muốn đính kèm khi commit

$ git config --global user.email "[email address]":

* git config: Để đặt user name và email của bạn trong file cấu hình chính
* --global: được sử dụng để áp dụng cho tất cả các dự án
* user.email [email address]: địa chỉ email bạn muốn đính kèm
* Đặt email mà bạn muốn đính kèm khi commit

$ git config --global color.ui auto:

* git config: Để đặt user name và email của bạn trong file cấu hình chính
* --global: được sử dụng để áp dụng cho tất cả các dự án
* Tô màu kết quả xuất ra

**CREATE REPOSITORIES**

$ git init [project-name]:

* [project-name]: tên của dự án
* Khởi tạo 1 kho chứa trên máy cá nhân với tên cụ thể tại thư mục hiện tại

$ git clone [url]:

* [url]: đường dẫn của kho mà bạn muốn tải về
* Sao chép một kho chứa đã tồn tại với đường dẫn cụ thể

**GROUP CHANGES**

$ git branch:

* Hiện thị tất cả các nhánh ở kho chứa hiện tại

$ git branch [branch-name]:

* [branch-name]: tên của nhánh bạn muốn tạo ra
* Tạo 1 nhánh mới với tên cụ thể

$ git checkout [branch-name]:

* [branch-name]: tên của nhánh bạn muốn chuyển sang
* Chuyển sang nhánh cụ thể khác và cập nhật các danh mục hiện có ở nhánh đó

$ git merge [branch-name]:

* [branch-name]: tên của nhánh bạn muốn hợp nhất
* Hợp nhất một nhánh cụ thể khác với nhánh hiện tại

$ git branch -d [branch-name]:

* -d: bị xóa (deleted)
* [branch-name]: tên của nhánh bạn muốn xóa
* Xóa một nhánh cụ thể

**REFACTOR FILENAMES**

$ git rm [file]:

* [file]: tên file bạn muốn xóa
* Xóa 1 file cụ thể từ kho chứa và máy cá nhân

$ git rm --cached [file]:

* --cached: file trên máy cá nhân sẽ không bị xóa
* [file]: tên file bạn muốn xóa
* Chỉ xóa file trên kho chứa

$ git mv [file-original] [file-renamed]:

* [file-original]: file bạn muốn di chuyển hoặc đặt tên
* [file-renamed]: địa chỉ thư mục bạn muốn chuyển file hoặc tên mới cho file
* Di chuyển hoặc đặt lại tên cho file/thư mục

**SUPPRESS TRACKING**

$ git ls-files --other --ignored --exclude-standard:

* --other: những file không bị theo dõi
* --ignored: những file bị phớt lờ
* --exclude-standard: thêm file .gitignore .git/info/exclude vào từng thư mục trong kho chứa ngoại trừ những file chung của người dùng
* Hiển thị thông tin về file

**SAVE FRAGMENTS**

$ git stash:

* Lưu lại các thay đổi chưa được commit mà bạn muốn chuyển sang 1 branch khác để làm việc

$ git stash pop:

* Lấy lại thay đổi và xoá nội dung thay đổi lưu trong stack đi

$ git stash list:

* Xem lại các lần lưu thay đổi khi chưa commit

$ git stash drop:

* Lấy lại thay đổi và xoá nội dung thay đổi lưu trong stack đi

**REVIEW HISTORY**

$ git log:

* Hiển thị lịch sử thay đổi cho nhánh hiện tại

$ git log --follow [file]:

* --follow: cụ thể file
* [file]: Tên file bạn muốn kiểm tra lịch sử
* Liệt kê lịch sử thay đổi cho 1 file với tên cụ thể

$ git diff [first-branch]...[second-branch]:

* [first-branch]: nhánh thứ nhất muốn so sánh
* [second-branch]: nhánh khác muốn so sánh
* So sánh sự khác nhau giữa 2 hoặc nhiều nhánh khác nhau

$ git show [commit]:

* Hiển thị thông tin cụ thể của 1 commit và nội dung thay đổi

**REDO COMMITS**

$ git reset [commit]:

* Hoàn tác đến thời điểm trước 1 commit

$ git reset --hard [commit]:

* Hoàn tác lại commit gần nhất, không giữ thay đổi ở máy cá nhân

**SYNCHRONIZE CHANGES**

$ git fetch [bookmark]:

* Tải tất cả các lịch sử từ một bookmark

$ git merge [bookmark]/[branch]:

* Hợp nhất một boomark từ một nhánh khác với nhánh cục bộ

$ git push [alias] [branch]:

* Tải lên tất cả các commit từ nhánh cục bộ lên Github

$ git pull:

* Lấy thông tin từ mã nguồn trên Github về và cập nhật vào máy cá nhân