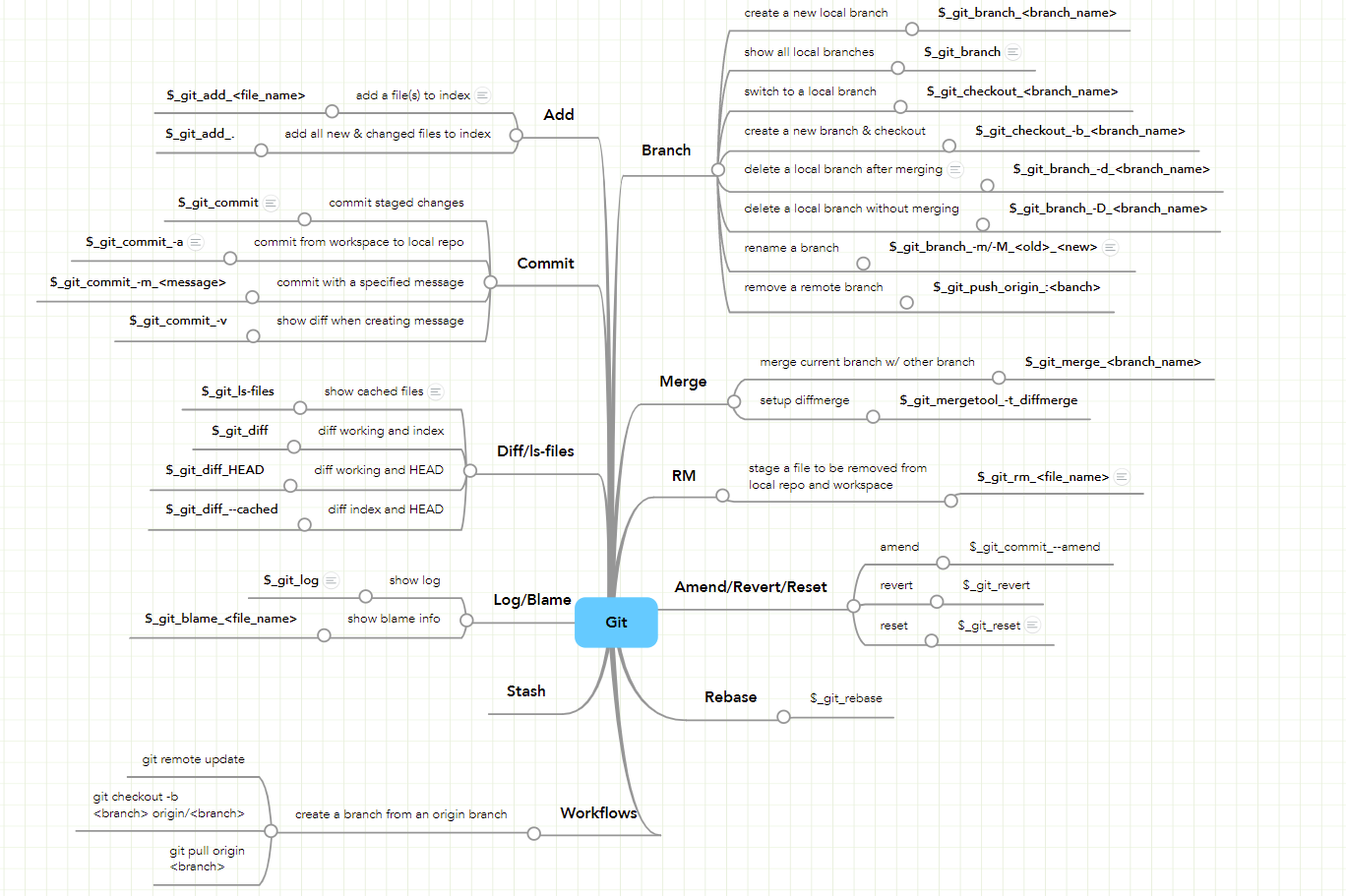
**Git**

[](https://www.blogger.com/blog/post/edit/5020789110855227593/4004407712605480515)

𝐌Ứ𝐂 ĐỘ 𝐂Ơ 𝐁Ả𝐍

Đây là 9 câu lệnh GIT cơ bản và được sử dụng thường xuyên nhất. Cũng là thứ mà bạn cần nắm vững.

1, 𝟏, 𝐠𝐢𝐭 𝐜𝐨𝐧𝐟𝐢𝐠

Git config là câu lệnh mà chúng ta phải thực thi đầu tiên cài đặt git lên máy. Câu lệnh này sẽ giúp các bạn thiết lập tên và email cá nhân của bạn, những thông tin này sẽ đính kèm trong mọi commit của bạn, đều này sẽ rất hữu ích khi chúng ta muốn biết đoạn code nào đó đã được ai triển khai để có thể thảo luận trong trường hợp chúng ta không hiểu rõ đoạn code đấy sử dụng cho mục đích gì.

$ git config --global user.name "Your name"

$ git config --global user.email "Your email"

𝟐, 𝐠𝐢𝐭 𝐯𝐞𝐫𝐬𝐢𝐨𝐧

Câu lệnh này dùng để kiểm tra phiên bản git đang sử dụng trên máy.

$ git version

𝟑, 𝐠𝐢𝐭 𝐢𝐧𝐢𝐭

Đây là câu lệnh đầu tiên khi chúng ta bắt đầu một dự án mới, câu lệnh này sẽ giúp chúng ta tạo một repository mới, sau đó nó sẽ được sử dụng để lưu trữ và quản lý mã nguồn trong repository này.

$ git init

// Hoặc bạn có thể đặt tên cho repo với lệnh

$ git init <your repository name>

𝟒, 𝐠𝐢𝐭 𝐜𝐥𝐨𝐧𝐞

Câu lệnh này giúp download một repository đã tồn tại sẵn trên khô lưu trữu (github, gitlab v.v) về máy.

git clone <your project URL>

𝟓, 𝐠𝐢𝐭 𝐚𝐝𝐝 .

Git add là câu lệnh giúp thêm tất cả các file code mới hoặc các file code được chỉnh sửa vào repository.

$ git add your\_file\_name - Thêm một file (thêm mới hoặc chỉnh sửa) vào staging area

$ git add \* - Thêm tất cả các file (thêm mới hoặc chỉnh sửa) vào staging area

𝟔, 𝐠𝐢𝐭 𝐜𝐨𝐦𝐦𝐢𝐭

Đây là câu lệnh được sử dụng phổ biến nhất, câu lệnh này giúp lưu các thay đổi ở các file trong vùng staging area xuống repository.

Có thể hiểu git add dùng để thêm thêm các file được thay đổi hoặc thêm mới vào vùng staging area, và chúng sẽ sẵn sàng để commit và sau đó những thay đổi này sẽ được lưu xuống repository.

$ git commit -m “your useful commit message”

𝟕, 𝐠𝐢𝐭 𝐬𝐭𝐚𝐭𝐮𝐬

Câu lệnh này cho phép bạn xem tình trạng hiện tại của mã nguồn như có bao nhiêu file được thêm mới hoặc chỉnh sửa. Những file nào đang nằm trong vùng staging area hoặc đang nằm ngoài staging area.

𝟖, 𝐠𝐢𝐭 𝐛𝐫𝐚𝐧𝐜𝐡

Trong một Git repository luôn luôn tồn tại nhiều nhánh riêng biệt dùng để triển khai một tính năng nào đó độc lập với các nhánh khác.

Các lệnh branch các bạn có thể sử dụng:

$ git branch

> Dùng để hiển thị tất cả các branch đang có.

$ git branch

> Dùng để tạo một branch mới.

$ git branch -d <branch\_name>

> Xoá branch.

𝟗, 𝐠𝐢𝐭 𝐜𝐡𝐞𝐜𝐤𝐨𝐮𝐭

Để di chuyển qua lại giữa các branch, chúng ta có thể sử dụng git checkout để đạt được điều này.

git checkout <branch\_name>

Ngoài ra các bạn có thể vừa chuyển qua một branch mới và tiện thể khởi tạo nếu chưa tồn tại với câu lệnh.

$ git checkout -b <your\_new\_branch\_name>

𝐁/ 𝐌Ứ𝐂 ĐỘ 𝐓𝐑𝐔𝐍𝐆 𝐁Ì𝐍𝐇

Sau các lệnh GIT cơ bản thường xuyên được sử dụng, chúng ta sẽ tìm hiểu các lệnh ở mức độ trung bình, cường độ sử dụng ích hơn.

𝟏𝟎, 𝐠𝐢𝐭 𝐫𝐞𝐦𝐨𝐭𝐞

Repository được các bạn khởi tạo với câu lệnh git init chỉ đang tồn tại trên máy local của các bạn. Nếu muốn lưu trữ repository này lên một dich vụ lưu trữ git từ xa nào đó chẳng hạn như gitlab, github thì các bạn cần phải sử dụng git remote để kết nối giữa chúng.

$ git remote add <shortname> <url>

> Ví dụ

$ git remote add origin

𝟏𝟏, 𝐠𝐢𝐭 𝐩𝐮𝐬𝐡

Khi đã kết nối giữa local và dịch vụ lưu trữ git, chúng ta cần sử dụng lệnh git push để đồng bộ những thay đổi được commit trên local lên dich vụ lưu trữ.

$ git push -u <short\_name> <your\_branch\_name>

> Ví dụ

$ git push -u origin feature\_branch

Ngoài ra trước khi sử dụng git push các bạn nên cấu hình origin và upstream.

$ git push --set-upstream <short\_name> <branch\_name>

> Ví dụ

$ git push --set-upstream origin feature\_branch

𝟏𝟐, 𝐠𝐢𝐭 𝐟𝐞𝐭𝐜𝐡

Git được sử dụng để làm việc nhóm, quản lý mã nguồn. Ngoài những commit của bạn thì còn vô số commit khác của các thành viên khác trong team. Sử dụng git fetch sẽ giúp chúng ta cập nhật tất cả những thông tin mới như commit, branch, v.v.

$ git fetch

𝟏𝟑, 𝐠𝐢𝐭 𝐩𝐮𝐥𝐥

Câu lệnh này sẽ download tất cả những nội dung (không chỉ là metadata như git fetch) từ dịch vụ lưu trữ xuống local repository.

$ git pull <remote\_url>

𝟏𝟒, 𝐠𝐢𝐭 𝐬𝐭𝐚𝐬𝐡

Git stash cho phép chúng ta lưu trữ các file được chỉnh sửa trong vùng nhớ tạm.

$ git stash

- Nếu muốn xem tất cả các stash các bạn có thể sử dụng lệnh:

$ git stash list

- Nếu bạn muốn áp dụng các chỉnh sửa trong một stash nào đó lên branch hiện tại đang sử dụng.

$ git stash apply

or

$ git stash pop

𝟏𝟓, 𝐠𝐢𝐭 𝐥𝐨𝐠

Với câu lệnh git log các bạn có thể xem tất cả những commit trước đó được sắp xếp theo thứ tự commit gần nhất cho đến những commit cũ hơn.

$ git log

𝟏𝟔, 𝐠𝐢𝐭 𝐬𝐡𝐨𝐫𝐭𝐥𝐨𝐠

Nếu chỉ muốn xem git log với nội dung được tóm tắt ngắn gọn thì các bạn có thể sử dụng git shortlog.

$ git shortlog

𝟏𝟕, 𝐠𝐢𝐭 𝐬𝐡𝐨𝐰

Lệnh này dùng để xem thông tin chi tiết của một commit bất kỳ.

$ git show <your\_commit\_hash>

𝟏𝟖, 𝐠𝐢𝐭 𝐫𝐦

Đôi lúc các bạn muốn xoá một file từ code base, trong trường hợp này các bạn có thể sử dụng git rm.

$ git rm <your\_file\_name>

𝟏𝟗, 𝐠𝐢𝐭 𝐦𝐞𝐫𝐠𝐞

Git merge cho phép các bạn kết mã nguồn và những thay đổi trên một branch khác vào branch hiện tại.

$ git merge <branch\_name>

Câu lệnh này sẽ kết hợp những thay đổi trên branch có tên là <branch\_name> vào branch hiện tại.

𝐂/ 𝐌Ứ𝐂 ĐỘ 𝐍Â𝐍𝐆 𝐂𝐀𝐎

Những câu lệnh ở mức độ nâng cao thường ít được sử dụng, và yêu cầu các bạn phải có kiến thức đủ tốt về git trước khi sử dụng. Hãy sử dụng chúng thật cẩn thận nhé.

𝟐𝟎, 𝐠𝐢𝐭 𝐫𝐞𝐛𝐚𝐬𝐞

Git rebase tương tự như git merge, nó sẽ kết hợp 1 branch vào branch hiện tại với một ngoại lệ, git rebase sẽ ghi lại tất cả các lịch sử commit.

Bạn nên sử dụng lệnh Git rebase khi bạn có nhiều branch riêng dùng để hợp nhất thành một branch duy nhất. Và nó sẽ làm cho lịch sử commit trở nên tuyến tính và dễ truy vết hơn.

$ git rebase <base>

𝟐𝟏, 𝐠𝐢𝐭 𝐛𝐢𝐬𝐞𝐜𝐭

Git bitsect giúp bạn tìm ra những bad commit.

- Để bắt đầu sử dụng $ git bisect start

- Cho git bisect biết về một commit tốt $ git bisect good a123

- Cho git bisect biết về một commit xấu $ git bisect bad z123

𝟐𝟐, 𝐠𝐢𝐭 𝐜𝐡𝐞𝐫𝐫𝐲-𝐩𝐢𝐜𝐤

Git cherry-pick là một lệnh hữu ích. Đó là một lệnh cho phép bạn chọn bất kỳ commit nào từ một branch bất kỳ và áp dụng nó vào một branch hiện tại.

$ git cherry-pick <commit-hash>

𝟐𝟑, 𝐠𝐢𝐭 𝐚𝐫𝐜𝐡𝐢𝐯𝐞

Lệnh Git archive sẽ kết hợp nhiều tệp thành một tệp duy nhất. Nó giống như một tiện ích zip, vì vậy nó có nghĩa là bạn có thể giải nén các tệp lưu trữ để lấy các tệp riêng lẻ.

$ git archive --format zip HEAD > archive-HEAD.zip

𝟐𝟒, 𝐠𝐢𝐭 𝐩𝐮𝐥𝐥 –𝐫𝐞𝐛𝐚𝐬𝐞

Nếu bạn muốn download content từ dịch vụ lưu trữ và dùng rebase thay vì merge thì có thể sử dụng

$ git pull --rebase

𝟐𝟓, 𝐠𝐢𝐭 𝐛𝐥𝐚𝐦𝐞

Nếu bạn cần kiểm tra nội dung của bất kỳ tệp nào, bạn cần sử dụng git blame. Nó giúp bạn xác định ai đã thực hiện các thay đổi đối với tệp.

$ git blame <your\_file\_name

𝟐𝟔, 𝐠𝐢𝐭 𝐭𝐚𝐠

Trong Git, các thẻ tag rất hữu ích và bạn có thể sử dụng chúng để quản lý bản phát hành. Bạn có thể coi thẻ Git giống như một nhánh sẽ không thay đổi. Nó quan trọng hơn đáng kể nếu bạn đang phát hành công khai.

$ git tag -a v1.0.0

𝟐𝟕, 𝐠𝐢𝐭 𝐯𝐞𝐫𝐢𝐟𝐲-𝐜𝐨𝐦𝐦𝐢𝐭

Lệnh git verify-commit sẽ kiểm tra chữ ký gpg. GPG hoặc “GNU Privacy Guard” là công cụ được sử dụng trong các tệp ký tên và chứa các chữ ký của chúng.

$ git verify-commit <commit>

𝟐𝟖, 𝐠𝐢𝐭 𝐯𝐞𝐫𝐢𝐟𝐲-𝐭𝐚𝐠

Tương tự git verify commit, các bạn có thể kiểm tra trên tag với lệnh

$ git verify-tag <tag>

𝟐𝟗, 𝐠𝐢𝐭 𝐝𝐢𝐟𝐟

Nếu các bạn muốn so sánh một file code nào thay đổi những gì trước khi commit thì các bạn có thể sử dụng

$ git diff HEAD <filename>

Để kiểm tra sự khác nhau giữa mã nguồn hiện tại đã được thay đổi so với local repo

$ git diff HEAD <filename>

So sánh 2 branch

𝟑𝟎, 𝐠𝐢𝐭 𝐜𝐢𝐭𝐨𝐨𝐥

Git citool là một giải pháp thay thế đồ họa của Git commit.

$ git citool

𝟑𝟏, 𝐠𝐢𝐭 𝐦𝐯

Đổi tên git file từ tên cũ sang tên mới.

$ git mv <old-file-name> <new-file-name>

𝟑𝟐, 𝐠𝐢𝐭 𝐜𝐥𝐞𝐚𝐧

Bạn có thể xoá sạch các nội dung được thay đổi với các untracked files (chưa được theo dõi) với lệnh git clean.

$ git clean

𝟑𝟑, 𝐠𝐢𝐭 𝐡𝐞𝐥𝐩

Giúp bạn xem tất cả các thông tin cần thiết để sử dụng git.

$ git help <git\_command>

𝟑𝟒, 𝐠𝐢𝐭 𝐰𝐡𝐚𝐭𝐜𝐡𝐚𝐧𝐠𝐞𝐝

Lệnh này thực hiện tương tự như git log nhưng ở dạng thô. Và đó là do nguyên nhân lịch sử.

$ git whatchanged