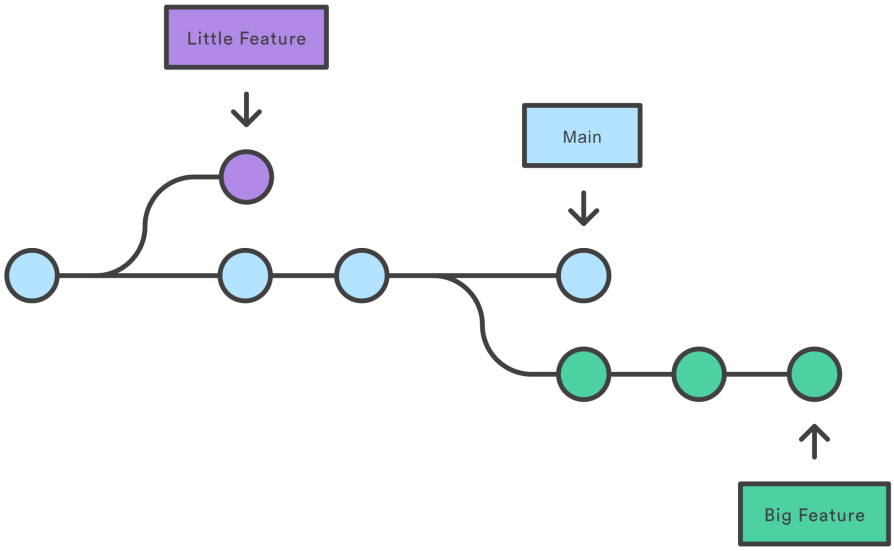
Một số khái niệm liên quan đến git

- **Git**: có thể hiểu là phần mềm quản lý phiên bản cho mã nguồn (soure code). Chúng ta sử dụng git trong quá trình phát triển phần mềm để có thể sao lưu các phiên bản khác nhau của source code. Nếu một lập trình viên độc lập mà sử dụng git thì họ có thể dễ dàng quản lý các phiên bản khác nhau của source code, giúp dễ dàng lấy lại các source code cũ hoặc khôi phục source code về hiện trạng ban đầu trước khi xảy ra một sự cố nào đó.
- **Github**: là một nền tảng lưu trữ mã nguồn dành cho git, giúp cho nhiều người dùng có thể cùng cộng tác với nhau khi làm việc trên cùng một project. Github sẽ là nơi trung tâm lưu trữ mọi thay đổi chung của các developers tham gia cùng cộng tác.
- **Repository (repo)**: một repository là một project được quản lý bởi git nơi chứa các tập tin, thư mục (chủ yếu là code) liên quan đến dự án mà các developer đang tham gia thực hiện. Một repository có thể được khởi tạo trên máy local hoặc được khởi tạo trực tiếp trên các nền tảng như github rồi được tải về máy local (hoặc ngược lại).
- **Commit**: trong quá trình viết code tạo ra ứng dụng, có rất nhiều tập tin mới được tạo ra/được điều chỉnh/bị xóa. Sau khi dev cảm thấy những thay đổi này đã ổn, ví dụ một tính năng mới đã được thêm và hoạt động tạm ổn, một bug đã được fix và chạy được, một số cấu hình project đã được thực hiện thành công... thì dev sẽ muốn lưu lại những thay đổi này để trong tương lai nếu lỡ tạo ra sai sót gì đó thì dev có thể khôi phục lại hiện trạng cũ. Lúc này dev sẽ tạo ra một commit, một commit có thể hiểu là một bản 'backup', một bản 'snapshot' để lưu lại toàn bộ trạng thái hiện hành của project. Khi tạo một commit, dev phải nhập mô tả chi tiết (message) cho commit đó để sau này khi có quá nhiều commit thì dev sẽ dựa vào message đó để biết được mình đã làm gì ở từng commit. Mỗi commit sẽ có một mã hash xem như là id, dev sẽ dựa vào id này để đưa project trở về bản commit đó.
- **Log**: khi dev thực hiện việc "xem log" nghĩa là dev muốn xem lại danh sách các commit đã được tạo ra trong project hiện hành. Khi xem log thì dev sẽ thấy các commit liệt kê theo thứ tự từ mới đến cũ, mỗi commit có id (mã hash), thời gian tạo và thông tin message do người tạo commit đã nhập.
- **Branch**: có thể được hiểu như là 'dòng thời gian' (xem ảnh bên dưới), mặc định mỗi repo luôn có một branch thường được đặt tên là main hoặc master. Đôi lúc chúng ta có những ý tưởng mới muốn áp dụng thử vào project đang hoạt động ổn nhưng lại không muốn ảnh hưởng đến hướng phát triển chính đang có trong project, khi đó ta sẽ tạo ra các nhánh mới. Mỗi nhánh

khi đó có thể xem là một dòng thời gian độc lập, không ảnh hưởng đến các nhánh khác kể từ vị trí tách nhánh. Một nhánh sau khi đã dùng xong có thể được xóa và bỏ đi hoàn toàn hoặc được gộp lại với các nhánh khác để đưa các thay đổi của nhánh này vào nhánh kia.



- **Remote**: Khi một repository được tạo ra ở máy tính của dev (máy local) thì repo đó chỉ dùng cho một dev duy nhất, không chia sẻ hay cộng tác gì với các dev khác. Để chia sẻ/cộng tác với các dev khác, ta phải đưa code ở local repo lên các dịch vụ hosting ví dụ như Github, Bitbucket, Gitlab (hoặc các dịch vụ tương đương khác). Giả sử dev muốn đưa code ở local lên đồng thời github, gitlab và bitbucket thì họ phải tạo 3 tài khoản ở 3 dịch vụ trên rồi tạo các repository trên 3 dịch vụ này, sau đó họ phải "kết nối" các repo vừa tạo trên 3 nền tảng kia với repo ở máy local. Khi một kết nối này được tạo ra, nó được gọi là một "remote". Với github, người ta thường đặt tên remote này là origin. Như vậy khi nói "push code lên origin" tức là đưa code ở local repo lên github. Khi đưa code lên một máy remote thì ta phải chỉ định rõ tên của remote và tên nhánh nơi mà ta muốn đưa code lên, ví dụ: `git push origin main` nghĩa là đưa code lên máy server đang được đại diện bởi từ khóa 'origin' và đưa lên nhánh tên là main của máy đó.
- **Ba khu vực làm việc của git**: các tập tin, thư mục trong một project có thể được nằm trong 3 khu vực khác nhau bao gồm: working area, staging và commit history. Working area là nơi chứa các tập tin thư mục vừa được tạo mới, đã có sẵn nhưng vừa được chỉnh sửa, hoặc đã có sẵn và vừa bị xóa. Staging area là nơi chứa các tập tin vừa được chuyển từ working area qua để làm ứng viên đưa vào commit history. Các tập tin nếu không đưa vào staging thì sẽ không

được đưa vào commit history tự động. Một khi đã được đưa vào commit history rồi thì chúng đã được 'backup' và có thể được khôi phục trở lại trong tương lai.

Download và cài đặt git trên Windows

- Windows **64** bit: `https://github.com/git-for-windows/git/releases/download/v2.37.3.windows.1/Git-2.37.3-64-bit.exe`
- Windows 32 bit: `https://github.com/git-for-windows/git/releases/download/v2.37.3.windows.1/Git-2.37.3-32-bit.exe`

Thiết lập thông tin ban đầu

- Thiết lập username: `git config --global user.name "Mai Van Manh"`
- Thiết lập email: `git config --global user.email "mvmanh@gmail.com"`

Bắt đầu làm việc với một repository:

- Tạo một repo trên local: `cd <thư mục làm việc>` sau đó gõ `git init .`
- Clone một repo từ github: `git clone <url_trên_github>`
- Tạo một folder để làm trung tâm, nơi host dữ liệu của repo: `git init --bare .`

Các câu lệnh git thường dùng:

- Kiểm tra log (danh sách các commit): `git log`
- Kiểm tra log chỉ hiển thị 1 dòng dữ liệu: `git log --oneline`
- Kiểm tra tình trạng tập tin/folder: `git status`
- Xem danh sách các nhánh (branch): `git branch`
- Xem danh sách các remote (repo kết nối từ xa): `git remote`

Bắt đầu với một git repository mới:

- Khởi tạo repo mới: `git init .`
- Tạo một tập tin mới: `echo "hello" > hello.txt`
- Kiểm tra trạng thái của project: `git status` (lúc này hello.txt có màu đỏ, ở working area)
- Đưa tập tin hello.txt vào staging: `git add hello.txt` hoặc `git add .`
- Kiểm tra trạng thái của project: `git status` (lúc này hello.txt có màu xanh lá cây)

- Tạo commit để lưu lại tập tin này: `git commit -m "ghi nội dung mô tả ở đây"`
- Kiểm tra lại commit đã tạo: `git log` hoặc `git log --oneline`

Khôi phục repo về hiện trạng ban đầu

- Giả sử ứng dụng đang chạy rất ổn với nhiều tính năng hay. Khi đó mọi thứ đã được lưu lại và project đang có 5 commits. Trong ngày tiếp theo dev bắt đầu làm việc bằng cách: tạo nhiều tập tin, thư mục mới; xóa nhiều tập tin, thư mục cũ đang có; thay đổi nhiều tập tin, thư mục cũ đang có. Sau đó dev phát hiện ra rằng mình đã làm cho mọi thứ xáo trộn lên và muốn đưa project về hiện trạng của commit thứ 5 nhưng lại không biết cách làm thế nào.
- Khi đó dev sẽ gõ lệnh: `git clean -fd` để xóa các tập tin, thư mục mới mà dev đã tự thêm vào. Sau đó dev tiếp tục gõ lệnh `git checkout .` hoặc `git restore .` để đưa các tập tin cũ mà đã lỡ xóa hoặc lỡ thay đổi nội dung về hiện trạng ban đầu.
- Như vậy câu lệnh cần gõ là: `git checkout . && git clean -fd`

Xem lại hiện trạng cũ của project

- Giả sử project đã được phát triển qua một thời gian và đang có 10 commits. Mỗi commit tương ứng với một tính năng được bổ sung dần dần trong quá trình code. Lúc này dev muốn xem lại hiện trạng của app tương ứng với một commit cụ thể nào đó thì dev cần phải biết hashcode của commit đó. Dev gõ: `git log --oneline` để xem danh sách commit và hash.
- Tiếp theo dev gõ lệnh: `git checkout <hash_code>` trong đó `<hash_code>` là mã hash của commit mà dev muốn đưa project trở về. Sau khi chạy xong lệnh này thì toàn bộ source code của project sẽ được đưa trở về hiện trạng vào thời điểm mà commit đó được tạo ra. Lúc này dev có thể tự do build và chạy ứng dụng để xem.
- Nếu muốn quay trở lại commit trên cùng (hiện trạng mới nhất) thì chỉ cần gõ lệnh: `git branch <tên_nhánh>` trong đó `<tên_nhánh>` là tên của nhánh đang chứa các commit này. Mặc định tên nhánh thường là main hoặc master. Muốn biết danh sách nhánh đang có trong project thì gõ lệnh: `git branch`. Tuy nhiên cần lưu ý là cách này chỉ hoạt động nếu ta chỉ xem code chứ không chỉnh sửa gì, nếu lỡ chỉnh sửa gì đó thì ta phải hủy các thay đổi này trước khi gọi git brach. Để hủy các thay đổi thì gõ lệnh `git restore .`