Git là gì? Các lệnh git cơ bản mà mọi lập trình viên nên biết

Git là gì?

`Git` là một hệ thống quản lý phiên bản phân tán **(Distributed Version Control System – *DVCS*)**, nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. **Git** cung cấp cho mỗi lập trình viên kho lưu trữ (**repository**) riêng chứa toàn bộ lịch sử thay đổi.

Version Control System – VCS là gì?

VCS là viết tắt của **Version Control System** là **hệ thống kiểm soát các phiên bản phân tán mã nguồn mở**. Các VCS sẽ lưu trữ tất cả các file trong toàn bộ dự án và ghi lại toàn bộ lịch sử thay đổi của file. Mỗi sự thay đổi được lưu lại sẽ được và thành một version (phiên bản).

VCS nghĩa là hệ thống giúp lập trình viên có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (**clone**) từ một kho chứa mã nguồn (**repository**), mỗi thay đổi vào mã nguồn trên local sẽ có thể ủy thác (**commit**) rồi đưa lên server nơi đặt kho chứa chính.

Và một máy tính khác nếu họ có quyền truy cập cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia.

Lập trình viên có thể xem lại danh sách các sự thay đổi của file như xem một dòng thời gian của các phiên bản. Mỗi phiên bản bao gồm: nội dung file bị thay đổi, ngày giờ sửa đổi, người thay đổi là ai, lý do thay đổi hay tên phiên bản…

VCS có tác dụng như thế nào?

1. Lưu lại lịch sử các version của bất kỳ thay đổi nào của dự án. Giúp xem lại các sự thay đổi hoặc khôi phục (revert) lại sau này.
2. Việc chia sẻ code trở nên dễ dàng hơn, lập trình viên có thể để public cho bất kỳ ai, hoặc private chỉ cho một số người có thẩm quyền có thể truy cập và lấy code về.

Vốn là một VCS nên Git cũng ghi nhớ lại toàn bộ lịch sử thay đổi của source code trong dự án. Lập trình sửa file, thêm dòng code tại đâu, xóa dòng code ở hàng nào…đều được Git ghi nhận và lưu trữ lại.

Git hoạt động như thế nào?

Sự khác biệt chính giữa Git và bất kỳ VCS nào khác (bao gồm Subversion…) là cách Git nghĩ về dữ liệu của nó.

Về mặt khái niệm, hầu hết các hệ thống khác đều lưu trữ thông tin dưới dạng danh sách các thay đổi dựa trên file. Các hệ thống này (CVS, Subversion, Perforce, Bazaar, v.v.) coi thông tin chúng lưu giữ dưới dạng một tập hợp các file và những thay đổi được thực hiện đối với mỗi file theo thời gian.

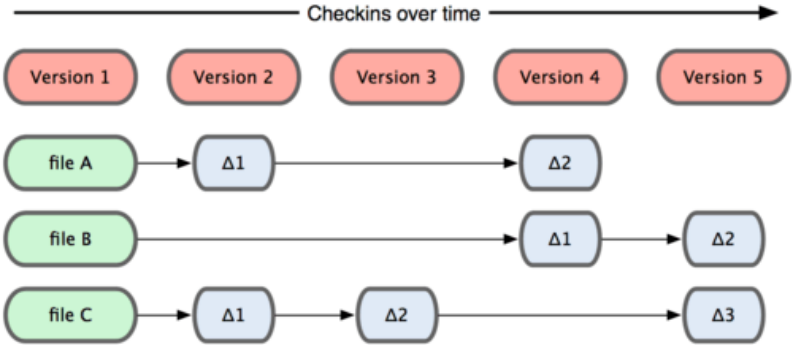


Image Credit: Stackoverflow

Git không nghĩ đến hoặc lưu trữ dữ liệu của mình theo cách này. Thay vào đó, Git coi thông tin được lưu trữ là một tập hợp các snapshot – ảnh chụp toàn bộ nội dung tất cả các file tại thời điểm.

Mỗi khi bạn "commit", Git sẽ "chụp" và tạo ra một snapshot cùng một tham chiếu tới snapshot đó. Để hiệu quả, nếu các tệp không thay đổi, Git sẽ không lưu trữ lại file — chỉ là một liên kết đến tệp giống file trước đó mà nó đã lưu trữ. Git nghĩ về dữ liệu của nó giống như dưới đây:

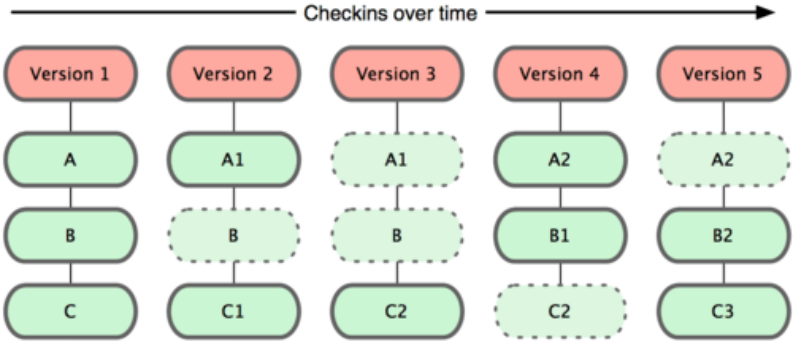


Image Credit: Stackoverflow

Đây là điểm khác biệt quan trọng giữa Git và gần như tất cả các VCS khác. Nó khiến Git phải xem xét lại hầu hết mọi khía cạnh của kiểm soát phiên bản mà hầu hết các hệ thống khác đã sao chép từ thế hệ trước. Điều này làm cho Git giống như một hệ thống tệp nhỏ với một số công cụ cực kỳ mạnh mẽ được xây dựng trên nó, thay vì chỉ đơn giản là một VCS.

Git có lợi ích gì?

Các dự án thực tế thường có nhiều lập trình viên làm việc song song. Vì vậy, một hệ thống kiểm soát phiên bản như Git là cần thiết để đảm bảo không có xung đột code giữa các lập trình viên.

Ngoài ra, các yêu cầu trong các dự án như vậy thay đổi thường xuyên. Vì vậy, một hệ thống kiểm soát phiên bản cho phép các nhà phát triển revert và quay lại phiên bản cũ hơn của code.

Cuối cùng, đôi khi một số dự án đang được chạy song song liên quan đến cùng một cơ sở code. Trong trường hợp như vậy, khái niệm phân nhánh trong Git là rất quan trọng.

- Dễ sử dụng, thao tác nhanh, gọn, lẹ và rất an toàn.
- Sễ dàng kết hợp các phân nhánh (branch), có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều.
- Chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa là bạn có thể làm việc ở mọi lúc mọi nơi.
- Deployment sản phẩm của bạn một cách không thể nào dễ dàng hơn.

Các thuật ngữ Git quan trọng

1. Branch

Các **Branch** (nhánh) đại diện cho các **phiên bản cụ thể** của một kho lưu trữ tách ra từ project chính của bạn.

Branch cho phép bạn theo dõi các thay đổi thử nghiệm bạn thực hiện đối với kho lưu trữ và có thể hoàn nguyên về các phiên bản cũ hơn.

2. Commit

Một commit đại diện cho một thời điểm cụ thể trong lịch sử dự án của bạn. Sử dụng lệnh commit kết hợp với lệnh **git add** để cho git biết những thay đổi bạn muốn lưu vào local repository.

3. Checkout

Sử dụng lệnh `git checkout` để chuyển giữa các branch. Chỉ cần nhập git checkout theo sau là tên của branch bạn muốn chuyển đến hoặc nhập git checkout master để trở về branch chính (master branch).

4. Fetch

Lệnh `git fetch` tìm nạp các bản sao và tải xuống tất cả các tệp branch vào máy tính của bạn. Sử dụng nó để lưu các thay đổi mới nhất vào kho lưu trữ của bạn. Nó có thể tìm nạp nhiều branch cùng một lúc.

5. Fork

Một fork là một bản sao của một kho lưu trữ (repository). Các lập trình viên thường tận dụng lợi ích của fork để thử nghiệm các thay đổi mà không ảnh hưởng đến dự án chính.

6. Head

Các commit ở đầu của một branch được gọi là head. Nó đại diện cho commit mới nhất của repository mà bạn hiện đang làm việc.

7. Index

Bất cứ khi nào bạn thêm, xóa hoặc thay đổi một file, nó vẫn nằm trong chỉ mục cho đến khi bạn sẵn sàng commit các thay đổi. Nó như là khu vực tổ chức (stagging area) cho Git. Sử dụng lệnh `git status` để xem nội dung của index của bạn.

Stagging là một bước trước khi commit trong git.

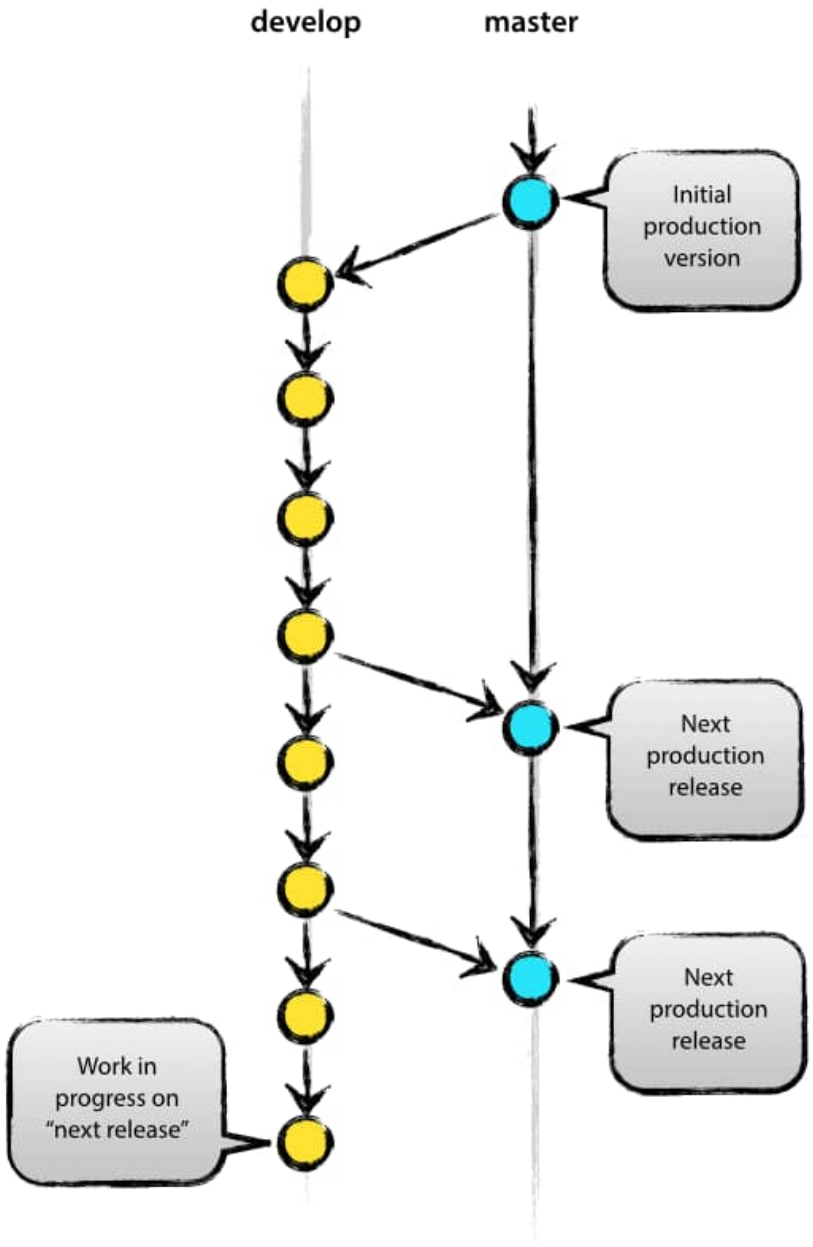
Một commit trong git được thực hiện theo hai bước: Stagging và commit thực tế. Miễn là những thay đổi nằm trong khu vực tổ chức (stagging area), git cho phép bạn chỉnh sửa nó theo ý muốn (thay thế các tệp được phân đoạn bằng các phiên bản khác của các tệp được phân loại, loại bỏ các thay đổi khỏi phân đoạn, v.v.)

Xem tiếp...

Những thay đổi được tô sáng bằng màu xanh lá cây đã sẵn sàng để được commit trong khi những thay đổi màu đỏ thì chưa.

8. Master

Master là nhánh chính của tất cả các repository của bạn. Nó nên bao gồm những thay đổi và commit gần đây nhất.

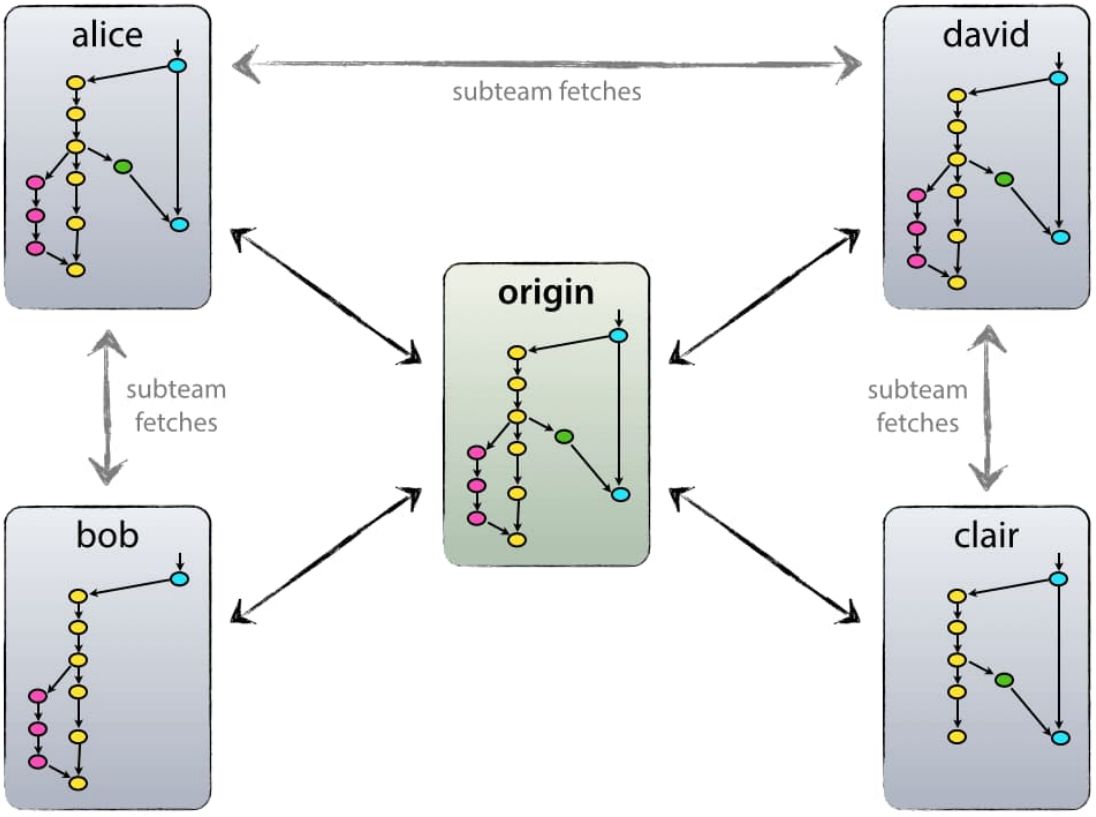


Nhánh Master

9. Merge

Lệnh `git merge` kết hợp với các yêu cầu kéo (pull requests) để thêm các thay đổi từ nhánh này sang nhánh khác.

10. Origin



Origin là phiên bản mặc định của repository. Origin cũng đóng vai trò là bí danh hệ thống để liên lạc với nhánh chính.

Lệnh `git push origin master` để đẩy các thay đổi cục bộ đến nhánh chính.

11. Pull

Pull requests thể hiện các đề xuất thay đổi cho nhánh chính. Nếu bạn làm việc với một nhóm, bạn có thể tạo các pull request để yêu cầu người bảo trì kho lưu trữ xem xét các thay đổi và hợp nhất chúng.

Lệnh `git pull` được sử dụng để thêm các thay đổi vào nhánh chính.

12. Push

Lệnh `git push` được sử dụng để cập nhật các nhánh từ xa với những thay đổi mới nhất mà bạn đã `commit`.

13. Rebase

Lệnh `git rebase` cho phép bạn phân tách, di chuyển hoặc thoát khỏi các commit. Nó cũng có thể được sử dụng để kết hợp hai nhánh khác nhau.

14. Remote

Một Remote (kho lưu trữ từ xa) là một bản sao của một chi nhánh. Remote giao tiếp ngược dòng với nhánh gốc (origin branch) của chúng và các Remote khác trong kho lưu trữ.

15. Repository

Kho lưu trữ Git chứa tất cả các tệp dự án của bạn bao gồm các branch, tags và commit.

16. Stash

Lệnh git stash sẽ loại bỏ các thay đổi khỏi chỉ mục của bạn và xóa stashes chúng đi sau.

Nó có ích nếu bạn muốn tạm dừng những gì bạn đang làm và làm việc khác trong một khoảng thời gian. Bạn không thể đặt stash nhiều hơn một bộ thay đổi ở cùng một thời điểm.

17. Tags

Tags cung cấp cho bạn một cách để theo dõi các commit quan trọng. Các tags nhẹ chỉ đơn giản đóng vai trò là con trỏ trong khi các tags chú thích được lưu trữ dưới dạng các đối tượng đầy đủ.

19. Upstream

Trong ngữ cảnh của Git, upstream đề cập đến nơi bạn push các thay đổi của mình, thường là nhánh chính (master branch).

Xem Git docs reference để biết thêm chi tiết về thuật ngữ liên quan đến Git.

Các lệnh git cơ bản

1) git config

Tác dụng : Để set user name và email của bạn trong main configuration file.
Cách xài : Để kiểm tra tên và kiểu email trong cấu hình dùng `git config -- global user.name` và `git config -- global user.email` . Để set email hoặc tên mới `git config -- global user.name = “Hải Nguyễn”` và `git config -- global user.email = “hainguyen@gmail.com”`

2) git init

Tác dụng : Khởi tạo 1 git repository 1 project mới hoặc đã có.

Cách xài: `git init` trong thư mục gốc của dự án.

3) git clone

Tác dụng: Copy 1 git repository từ remote source.

Cách xài: `git clone <:clone git url:>`

4) git status

Tác dụng: Để check trạng thái của những file bạn đã thay đổi trong thư mục làm việc. VD: Tất cả các thay đổi cuối cùng từ lần commit cuối cùng.

Cách xài: `git status` trong thư mục làm việc.

5) git add

Tác dụng: Thêm thay đổi đến stage/index trong thư mục làm việc.

Cách xài: `git add`

6) git commit

Tác dụng: commit nghĩa là một action để Git lưu lại một snapshot của các sự thay đổi trong thư mục làm việc. Và các tập tin, thư mục được thay đổi đã phải nằm trong Staging Area. Mỗi lần commit nó sẽ được lưu lại lịch sử chỉnh sửa của code kèm theo tên và địa chỉ email của người commit. Ngoài ra trong Git bạn cũng có thể khôi phục lại tập tin trong lịch sử commit của nó để chia cho một branch khác, vì vậy bạn sẽ dễ dàng khôi phục lại các thay đổi trước đó.

Cách dùng: `git commit -m ”Đây là message, bạn dùng để note những thay đổi để sau này dễ dò lại”`

> **5 tip về GitHub cho lập trình viên**

7) git push/git pull

Tác dụng: Push hoặc Pull các thay đổi đến remote. Nếu bạn đã added và committed các thay đổi và bạn muốn đẩy nó lên hoặc remote của bạn đã update và bạn apply tất cả thay đổi đó trên code của mình.

Cách dùng: `git pull <:remote:> <:branch:>` and `git push <:remote:> <:branch:>`

8) git branch

Tác dụng: liệt kê tất cả các branch (nhánh).

Cách dùng: `git branch` hoặc `git branch -a`

9) git checkout

Tác dụng: Chuyển sang branch khác

Cách dùng: `git checkout <: branch:>` hoặc `** _ git checkout -b <: branch:>` nếu bạn muốn tạo và chuyển sang một chi nhánh mới.

10) git stash

Tác dụng: Lưu thay đổi mà bạn không muốn commit ngay lập tức.

Cách dùng: `git stash` trong thư mục làm việc của bạn.

11) git merge

Tác dụng: Merge 2 branch lại với nahu.

Cách dùng: Chuyển tới branch bạn muốn merge rồi dùng `git merge <:branch_ban_muon_merge:>`

12) git reset

Tác dụng: Bạn đã đưa một tập tin nào đó vào Staging Area nhưng bây giờ bạn muốn loại bỏ nó ra khỏi đây để không phải bị commit theo.

Cách dùng: `git reset HEAD tên_file`

13) git remote

Tác dụng: Để check remote/source bạn có hoặc add thêm remote

Cách dùng: `git remote` để kiểm tra và liệt kê. Và `git remote add <: remote_url:>` để thêm.

14) git add

Tác dụng: Để đưa một tập tin vào Staging Area

Cách dùng: `git add tên_file` hoặc muốn thêm hết file của thư mục thì `git add all`

Lời khuyên khi thao tác thường xuyên với Git trong công việc

1. Git Cheet Sheets

Bạn không thể nào nhớ được hết các lệnh, lúc này bạn nên sử dụng các Git Cheet Sheets để dễ dàng tìm được lệnh Git bạn cần:

- **https://rogerdudler.github.io/git-guide/**
- **https://git-scm.com/docs/gittutorial**
- **https://gitsheet.wtf/**
- **http://ndpsoftware.com/git-cheatsheet.html**
- **https://gitexplorer.com/**

2. Nên commit thương xuyên

Tách nhỏ commit của bạn và commit thường xuyên nhất có thể. Điều này giúp các thành viên trong nhóm dễ dàng tích hợp công việc của họ hơn mà không gặp phải xung đột hợp nhất.

3. Test rồi mới commit

Không bao giờ commit nếu chưa hoàn tất process. Cần phải test các thay đổi của bạn trước khi chia sẻ chúng với người khác.

4. Viết ghi chú khi commit

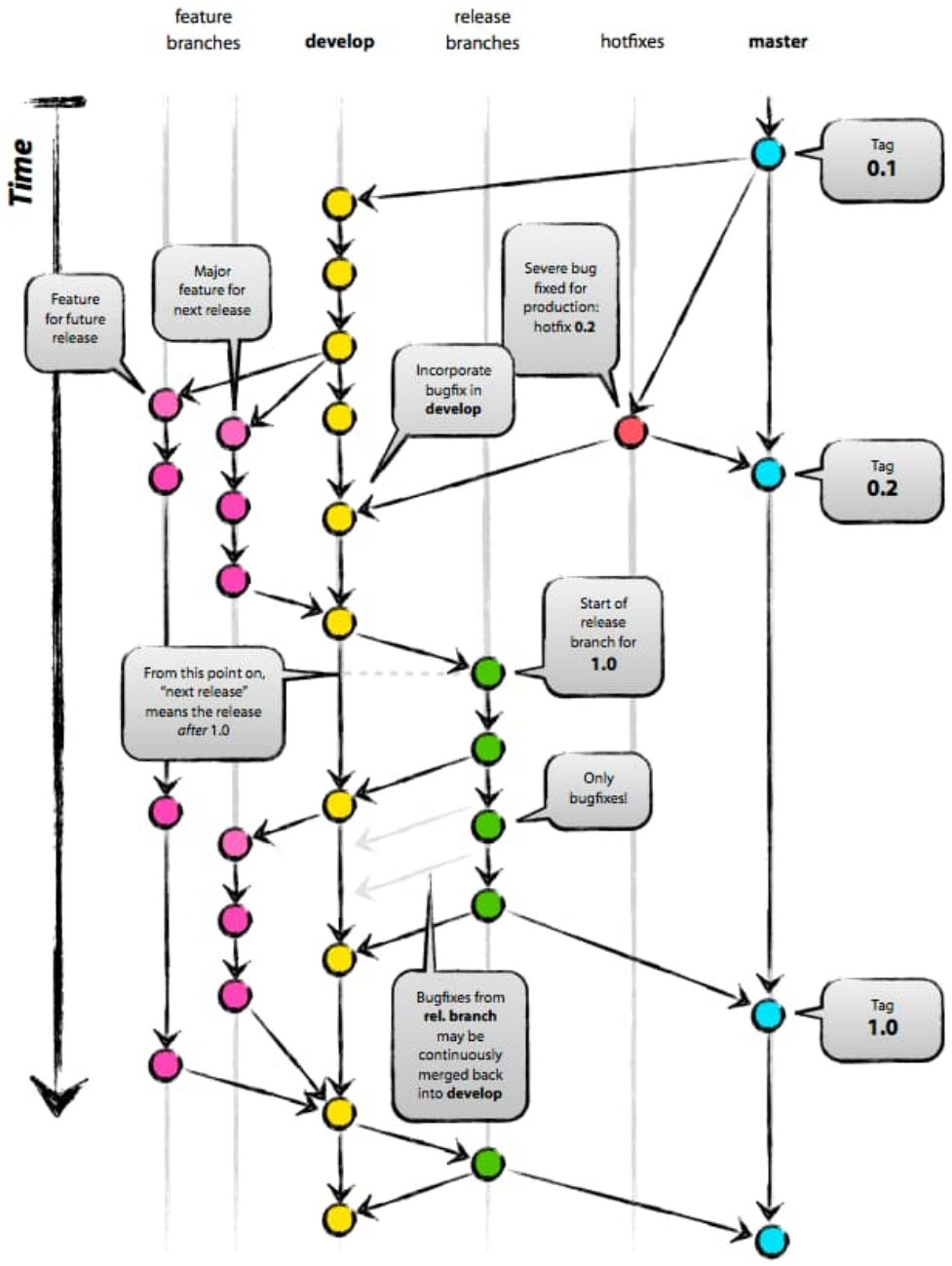
Viết ghi chú khi commit để cho các thành viên khác trong nhóm biết loại thay đổi bạn đã thực hiện. Hãy mô tả càng nhiều càng tốt.

5. Thử nghiệm Branch khác

Tận dụng lợi thế của các branch để giúp bạn theo dõi các dòng phát triển khác nhau.

6. Theo một Git Workflow

Bạn nên chọn theo một Git Workflow để đảm bảo cả nhóm của bạn đều cùng thực hiện như nhau.



Hy vọng với bài viết này bạn sẽ có thêm những thông tin hữu ích về GIT là gì? Và Các lệnh git cơ bản mà mọi lập trình viên nên biết. TopDev Blog sẽ tổng hợp thêm các kiến thức hữu ích hơn tới các bạn đọc. Mong các bạn luôn ủng hộ và yêu thương chúng mình nha

Đừng quên, xem các vị trí **tuyển dụng it tại HCM, Hà Nội, Đà Nẵng** hấp dẫn tại đây nhé!

TopDev Blog

https://topdev.vn

Thích chia sẻ về Development, Growthacking, Automation Marketing, bóng rổ...

Việc làm HOT cho Fresher

Việc làm theo ngành nghề

Việc làm IT theo loại hình