Bạn còn có thể sử dụng thêm một số tùy chọn xem log sau để tối ưu hơn quy trình đọc log.

* --since, --after: Xem các lần commit kể từ ngày nhất định.
* --until: Xem các lần commit trước từ ngày nhất định.
* --author: Xem các lần commit của một người nào đó.
* --grep: Lọc các chuỗi trong log và in ra.

Ví dụ:

**$** git log --author=contact@thachpham.com --pretty="%s"

Hihi

First commit on Github

Cái --pretty là gì mình sẽ nói ở dưới.

Lọc log với --pretty

Tham số --pretty rất có ích nếu bạn muốn lọc xem một đối tượng nào đó trong lịch sử commit, ví dụ như chỉ xem lời nhắn commit hoặc chỉ xem email của người commit.

Cách sử dụng tham số --pretty là bạn phải viết kèm các tag của nó như sau:

**$** git log --pretty="%tag"

Các %tag phải dược đặt trong cặp dấu ngoặc kép và bạn có thể sử dụng nhiều %tag khác nhau.

Danh sách các %tag:

* %H –  Commit hash
* %h – Abbreviated commit hash
* %T – Tree hash
* %t – Abbreviated tree hash
* %P – Parent hashes
* %p – Abbreviated parent hashes
* %an – Author name
* %ae – Author e-mail
* %ad – Author date (format respects the –date=option)
* %ar – Author date, relative
* %cn – Committer name
* %ce – Committer email
* %cd – Committer date
* %cr – Committer date, relative
* %s – Subject

Ví dụ:

**$** git log --pretty="%an - %s"

Thach Pham - Hihi

Thach Pham - First commit on Github

Thủ thuật xem log đẹp hơn

Bạn có thể thấy mỗi khi gõ git log mà không có các tham số kèm theo thì nó hiện ra rất khó nhìn. Tuy nhiên có một cách để bạn có thể làm gọn cái log mặc định, và thêm màu sắc cho nó để bạn dễ nhìn hơn.

Trước tiên bạn copy lệnh này để thiết lập.

git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

Bây giờ để xem log, bạn chỉ cần gõ git lg để xem. Bạn có thể kết hợp nó với các tham số ở trên. Cái quan trọng nhất là với cách xem này, bạn có thể thấy **HEAD** (con trỏ của branch hiện tại) mà sau này khi bạn tìm hiểu tới branch bạn sẽ thấy nó rất có ích.

### Undo Commit

Nếu bạn cần xóa bỏ lần commit trước và cần undo để commit lại thì có thể sử dụng tham số --amend trong lệnh git commit.

**$** git log --pretty="%s"

Hihi

First commit on Github

**$** git commit --amend -m "Hehe"

[master 3682e56] Hehe

Date: Tue Apr 21 17:16:37 2015 -0700

2 files changed, 1 insertion(+), 1 deletion(-)

create mode 100644 faq.html

**$** git log --pretty="%s"

Hehe

First commit on Github

Lưu ý rằng undo nghĩa là bạn quay trở lại bước commit lần trước, do vậy nếu cần bổ sung tập tin nào vào để commit thì hãy đưa tập tin đó vào Staging Area trước.

### Bỏ tập tin ra khỏi Staging Area

Nếu 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 thì có thể sử dụng lệnh git reset HEAD tên\_file. HEAD là cái gì thì tới phần phân nhánh (Branch) mình sẽ giải thích.

### Lời kết

Trong bài này bạn đã biết cách thao tác để xem lại lịch sử của những lần commit để theo dõi các lần commit tốt hơn, cũng như cách bạn thao tác phục hồi liên quan đến việc Commit. Tuy nhiên, bạn có thấy là mình đã có nói qua về việc quản lý phiên bản trong Git chưa? Chưa có đúng không? Được rồi, vậy thì ở bài sau mình sẽ hướng dẫn các bạn làm quen với tính năng Tagging (đánh dấu) của Git và đây là cách để chúng ta quản lý từng phiên bản trong mã nguồn.