Nguyen Anh Tien Follow
Published Nov 27th, 2015 7:35 am

19 Bí Kíp Bạn Có Thể Dùng Khi Phạm Sai Lầm Với Git

Git

Nov 27th, 2015 7:35 am

6057 69 4 Report

Marked as Trending on 2017-10-24 14:40:13

Bài viết được dịch từ bài Gitでやらかした時に使える19個の奥義 của tác giả muran001 trên Qiita.

Những nội dung sau rất nguy hiểm, nếu bạn làm theo, xin hãy chịu trách nhiệm về bản thân mình.

Nếu có gì sai sót, mong bạn hãy chỉ ra để tôi được biết

Ở trên local

Có hiệu quả khi làm việc trên môi trường phát triển của bản thân, không làm ảnh hưởng đến người khác.
Khi làm việc trên remote sẽ phát sinh ảnh hưởng đến người khác nên rất nguy hiểm.

Sự cố 1： Khi trong commit message lỡ có những từ cấm kị khiến bạn chỉ muốn chết đi cho xong

Để sửa lại nội dung commit thì rất đơn giản. Cũng có thể thêm được cả file vào trong commit

```
$ git commit --amend
# Sau lệnh này ta có thể thay đổi nội dung commit message một cách tuỳ ý
```

Sự cố 2： Khi lỡ tay commit dưới tên của một user khác

+38

Ở đây chúng ta đang nói về việc thay đổi tác giả (Author)
Hãy đổi Commiter ở `.gitconfig` hoặc `.git/config`

Khi muốn đổi tên user hoặc email của HEAD

```
# Hãy sửa lại user.name và user.email
$ git commit --amend -m "nội dung commit message" --author="user.name
<user.email>"
```

Trường hợp muốn sửa lại commit từ lâu rồi

```
$ git rebase -i <commit>

====
# Sau lệnh này sẽ mở ra editor nên hãy sửa lại như sau rồi lưu lại

# (trước khi sửa) các commit cũ từ trên xuống dưới
pick aa11bbc commit message 1
pick b2c3c4d commit message 2
pick 4e56fgh commit message 3
. . .

# (Sau khi sửa) Đổi commit cần sửa sang edit
edit aa11bbc commit message 1
pick b2c3c4d commit message 2
pick 4e56fgh commit message 3
. . .
====

# Sau đó thì sửa tương tự
$ git commit --amend -m "commit message" --author="user.name <user.email>"

# Trở lại như cũ
$ git rebase --continue
```

(NGUY HIỂM) Viết lại toàn bộ lịch sử commit

```
git filter-branch --commit-filter '
        if [ "$GIT_COMMITTER_NAME" = "<Old Name>" ];
        then
                GIT_COMMITTER_NAME="<New Name>";
                GIT_AUTHOR_NAME="<New Name>";
                GIT_COMMITTER_EMAIL="<New Email>";
                GIT_AUTHOR_EMAIL="<New Email>";
                git commit-tree "$@";
        else
                git commit-tree "$@";
        fi' HEAD
```

Sự cố 3 : Khi đã lỡ tay commit một số file thừa nhưng sau khi nghĩ lại thì nên ignore thì hơn

Trường hợp commit xoá

Nếu có những phần thừa thì ta xoá đi rồi thêm vào ignore

```
# Đầu tiên là xoá các file đã commit khỏi repository
$ git rm --cached <tên file>

# Sau đó là thêm vào gitignore
$ echo '<tên file>' >> .gitignore
```

Trường hợp muốn coi như là chưa từng có file đó

Khi mà bạn muốn: "Hả, file nào cơ, là gì có file nào như thế ?"

```
# Trường hợp chỉ xoá khỏi lịch sử commit  (để lại ở working tree)
$ git filter-branch -f --index-filter 'git rm --cached -rf --ignore-unmatch <tên
file>' HEAD

# Trường hợp xoá khỏi lịch sử commit và cả ở working tree
$ git filter-branch -f --index-filter 'git rm -rf --ignore-unmatch <tên file>'
HEAD
```

```
$ git branch -m <tên branch sau khi đổi>
```

Sự cố 5： Khi trong phút nông nổi bạn lỡ commit và giờ muốn commit đó biến mất

Không để cho ai biết （= người khác vẫn chưa pull về）

Sửa lại phần lịch sử commit ngay trước đó

```
# Tuỳ vào từng trường hợp mà ta có 3 cách sau để đưa lịch sử commit về như cũ

# 1. Chỉ đưa HEAD về như cũ
$ git reset --soft HEAD~

# 2. Đưa HEAD và index về như cũ
$ git reset HEAD~

# 3. Đưa cả index, working tree về 1 commit trước đó
$ git reset --hard HEAD~
```

Trường hợp đã bị ai đó phát hiện （= người khác đã pull về）

Nếu thay đổi lịch sử commit sẽ dẫn đến tình trạng lộn xộn nên ta cầm dùng commit xoá để giải quyết

```
$ git revert <commit>
```

Sự cố 6： Khi muốn tổng hợp những commit vụn vặt trước đây thành một commit

Sử dụng `rebase` + `squash` (or `fixup`) để tổng hợp lại các commit cũ

```
# Ví dụ: git rebase -i HEAD~3
$ git rebase -i HEAD~<số lượng commit>

====
# Sau lệnh này sẽ mở ra editor nên hãy sửa lại như sau rồi lưu lại

# (trước khi sửa) các commit cũ từ trên xuống dưới
pick aa11bbc commit message 1
pick b2c3c4d commit message 2
pick 4e56fgh commit message 3
. . .

# (sau khi sửa) các commit trước là squash sẽ được tổng hợp vào 1 commit ở đằng trước
pick aa11bbc commit message 1
squash b2c3c4d commit message 2
squash 4e56fgh commit message 3
. . .
====
```

Thay vì dùng `squash` có thể dùng `fixup` nhưng khi đó commit message của những commit được fixup sẽ bị mất

Sự cố 7 : Khi bạn muốn chia commit to thành những commit nhỏ hơn một cách thông minh

Ở mục 6, tuy tổng hợp lại là tốt nhưng khi tổng hợp quá nhiều cũng sẽ trở thành vấn đề
Rất tiện khi mà những commit chức năng bị lẫn vào những commit sửa bug

```
# Đầu tiên là đưa HEAD và index về 1 commit trước đó
# Nói cách khác là ta coi như là chưa từng có commit to đùng kia và chỉ giữ lại trạng thái của working tree
$ git reset HEAD~

# Dùng lệnh sau đây để thêm (y) hoặc không thêm (n) các phần nhỏ (=hunk) vào index
# Nếu các phần nhỏ đó vẫn chưa đủ nhỏ thì dùng (s) để tiếp tục chia nhỏ hơn  (Chi tiết của git add -p thì hãy hỏi anh Google nhé)
$ git add -p

# Khi các phần cần thay đổi đã có thì ta commit
$ git commit -m "commit message"

# Sau đó ta lặp đi lặp lại các bước như trên cho phần còn lại
```

Sự cố 8 : Khi lỡ tay commit nhầm sang một branch khác

Có những lúc ta sơ suất lỡ tay commit thẳng vào master trong khi thực ra là muốn commit vào một branch khác

```
# Đầu tiên là tạo một branch khác chứa trạng thái mà ta đã commit
$ git branch other-branch

# Đưa HEAD, index của master về 1 commit trước đó
$ git reset --hard HEAD~

# Check out sang branch có commit trước đó
$ git checkout other-branch
```

Sự cố 9 : Khi muốn xoá hoàn toàn các file không cần thiết

Có những lúc ta muốn 1 phát xoá tất tần tật những file tự động sinh ra hoặc những file tự động back up. Nếu không phải là những file cần git quản lý hoặc những file sẽ commit thì xoá đi thôi. Ơ mà cái này cũng ko phải là sự cố nhỉ...

```
$ git add <file cần commit>
$ git commit -m "commit message"

# Dùng git stash để sơ tán cả những file untracked
$ git stash -u

# drop các thứ đã stash để xoá đi
$ git stash drop
```

Tiện đây thì trong trường hợp chỉ là những file mà git không quản lý thì chỉ như sau là OK

```
# Kiểm tra lại các file sẽ xoá cho chắc
$ git clean -n

# Tiến hành xoá bỏ
$ git clean -f
```

Sự cố 10： Khi tạm thời cần tạm dừng công việc hiện tại và chuyển sang một branch khác

Tuy mình không xấu nhưng có những lúc cần phải tạm dừng công việc để làm việc khác

```
# Tạm thời lưu lại các phần công việc còn đang làm dở
$ git stash -u

# Chuyển sang một branch khác và làm việc
$ git checkout -b other-branch
~làm việc, làm việc, làm việc~
$ git add <các file cần thiết>
$ git commit -m "commit message"

# Trở về branch cũ
$ git checkout origin-branch

# Lấy lại các nội dung công việc đang làm dở trước đó
$ git stash pop
```

```
# Đầu tiên là xem lại toàn bộ lịch sử commit
$ git reflog

# Từ đó chọn commit muốn phục hồi và khôi phục lại
# ví dụ) git reset --hard HEAD@{2}
$ git reset --hard <commit>
```

Sự cố 12： Khi lỡ tay xoá mất branch và muốn lấy lại

Nếu vẫn còn reflog thì không có sao. Vẫn có thể hồi phục lại được
Nếu commit lên một branch không tên, sau đó check out sang một branch khác bị xoá mất thì cũng tương tự

Tương tự giống như sự cố 13 dưới đây

```
# Đầu tiên là xem lại toàn bộ lịch sử commit
$ git reflog

# Từ các commit này, chọn rồi tạo branch mới
# ví dụ) git branch new-branch HEAD@{2}
$ git branch <tên branch> <commit>
```

Sự cố 13: Khi muốn thay đổi một phần nội dung commit

Khi có những typo nhỏ và muốn chỉnh sửa thì ko cần thiết phải tạo commit mới để sửa mà hãy tiến hành sửa lại luôn

```
# git rebase -i <commit>

====
# Sau lệnh này sẽ mở ra editor nên hãy sửa lại như sau rồi lưu lại

# (trước khi sửa) các commit cũ từ trên xuống dưới
pick aa11bbc commit message 1
pick b2c3c4d commit message 2
pick 4e56fgh commit message 3
. . .

# (Sau khi sửa) Đổi commit cần sửa sang edit
edit aa11bbc commit message 1
pick b2c3c4d commit message 2
pick 4e56fgh commit message 3
. . .
====

# Commit cần sửa sẽ được khai triển working tree nên ta chỉ cần sửa rồi sau đó
add lại
$ git add <tên file>

# Sửa lại commit
$ git commit --amend

# Hoàn tất rebase để trở lại như cũ
$ git rebase --continue

# Khi muốn huỷ bỏ rebase
$ git rebase --abort
```

Sự cố 14: Khi có conflict trong quá trình rebase

Khi không merge mà tiến hành rebase thì nếu có conflict, chắc hẳn là hoảng loạn rồi

```
$ git rebase branch2

~~ Phát sinh conflict ~~

# Sẽ trở thành branch không tên nền cần phải chú ý
$ git branch

* (no branch, rebasing branch2)
  branch1
  branch2
  master

# Hãy cố gắng giải quyết các conflict

# Sau khi hoàn thành, ta add rồi tiến hành rebase
$ git add <tên file>

# Thay đổi commit
$ git commit --amend

$ git rebase --continue
```

Sự cố 15: Khi đã merge nhưng lại muốn trở lại như lúc trước

Chỉ cho trường hợp mà người khác vẫn chưa pull code về

```
# tiến hành merge
$ git checkout <tên brach nguồn>
$ git merge <tên branch muốn merge>

# Sau khi merge, nhưng lại muốn trở lại như trước thì làm như sau
# Nếu chỉ định là ORIG_HEAD thì có thể trở lại trạng thái trước khi merge
$ git reset --hard ORIG_HEAD
```

Ở trên Remote

Sự cố 16: Khi pull từ remote về thì có quá nhiều conflict nên tạm thời muốn trở

```
# Lấy từ code mới remote
$ git pull origin master

# Phát sinh conflict

# Suy nghĩ lại thì trong pull(fetch + merge) muốn bỏ phần merge đi
$ git reset --hard ORIG_HEAD
```

Sự cố 17: Lỡ tay push force lên master trên remote

Việc `push -f` lên origin/master là việc bị cấm
Tuy nhiên thì thỉnh thoảng vẫn có một số người làm...

Ta sẽ sửa lại như sau

Trường hợp chỉ cần push về 1 commit trước đó

```
# Cái này ko được làm đâu nhé, push force
$ git commit -m "commit message"
$ git push -f origin master

# Nếu là một công ty nghiêm túc thì đến đây chắc chắn sẽ nổi giận đấy

# Nếu muốn đưa về 1 commit tước đó thì làm như sau
$ git push -f origin HEAD~:master
```

Nếu trên local cũng muốn như vậy thì tiến hành reset cũng tốt

```
$ git reset HEAD~
$ git push -f origin master
```

Nếu muốn xoá đi vài commit trước đó

```
# git commit -m "commit message
$ git push -f origin master

# Khi bị mắng và muốn trở về như cũ thì
$ git rebase -i <commit muốn rebase về>

====
# Sau lệnh này sẽ mở ra editor nên hãy sửa lại như sau rồi lưu lại

# (trước khi sửa) các commit cũ từ trên xuống dưới
pick aa11bbc commit message 1
pick b2c3c4d commit message 2
. . .

# (Sau khi sửa) Xoá các commit không cần thiết đi
pick b2c3c4d commit message 2

. . .
====

# push force lại một lần nữa
$ git push -f origin master
```

Sự cố 18: Sau khi release phát hiện thấy có bug và muốn trở về như cũ

Do phát sinh thiệt hại nên những thứ đã merge và maste cần phải chuyển tạm thời về như cũ
Tuy nhiên không thể revert từng commit 1 nếu có nhiều commit

```
# git log --oneline --graph

====
*   1x3y5z7 - Merge pull request #4 from develop
|\
| * a2b45c7 - (develop) commit 3
* | dbc65f4 - commit 2
* | f0b0a91 - commit 1
# Chỉ định điểm muốn revert về (Cơ bản 1 là OK)
====

# Nếu merge develop vào master thì chỉ định master sẽ là 1、develop là 2 rồi revert
# Ví dụ) git revert -m 1 1x3y5z7
$ git revert -m 1 commit merge>
```

Nếu vẫn chưa có ai kéo phần code này về thì có thể làm được cả những việc sau nữa（Cái này nguy hiểm）

```
# Trở về trạng thái trước khi merge
$ git reset ORIG_HEAD

# PUSH FORCE
$ git push -f origin master
```

Sự cố 19：Khi sau khi chuyển lại và đã commit thêm một số commit mới nhưng không thể merge lại được

Ở sự cố 18, khi đã xoá bỏ merge, tiến hành chỉnh sửa nhưng merge lại không được

commit revert không có nghĩa là sẽ ko còn commit merge nữa mà là làm ngược lại những việc mà commit merge làm

Thế nên, khi merge lại thì cần phải revert lại commit revert

```
# Thêm commit vào develop branch
$ git checkout develop
$ git commit -m "commit message"

# Checkout sang master
$ git checkout master

# revert lại commit revert
$ git revert <commit revert>

# merge develop
$ git merge develop
```

Khi muốn thử những điều trên ở một môi trường an toàn thì

Đúng là nếu chưa quen mà tự nhiên lại thực hành luôn thì rất vất vả.

Những ai đã có sản phẩm thì có thể clone về và dùng cái đó để test thử

Nếu thế thì cho dù có bị thất bại thì cũng không gây ảnh hưởng gì đến môi trường làm việc chính.

Những ai chưa có sản phẩm có thể lên Github, tìm hiếm 1 repo nào đó nổi tiếng, clone về rồi thực hành

Tuy nhiên cần chú ý 1 điều là những cái này chỉ được thử ở trên môi trường local thôi nhé

Nếu muốn thử cả ở trên remote nữa thì có thể chuẩn bị sẵn môi trường test trên Github, clone về rồi thử

Hình dung cơ bản như sau

```
# git clone <URL của repository>

# Thử nghiệm trên local

# Kiểm tra remote hiện tại
$ git remote -v

# Xoá remote đã đăng ký
$ git remote rm origin

# Tạo remote đã được chuẩn bị sẵn cho test ở Github và thay đổi lại remote
$ git remote add origin <URL của repository mới>

# Sau đó là thử nghiệm trêm remote
```

Have problems with Git? **Ask on Viblo »**

Related

Git Merging vs. Rebasing

Nguyễn Huy Hùng

164 0 0 5

[Git] - Merging vs. Rebasing

Truong Bui

90 1 0 0

Basic Tricks

SonNT

25 0 0 0

Rubocop và file thay đổi

More from Nguyen Anh Tien

Elixir và Unicode, Phần 2: Làm việc với Unicode Strings

Nguyen Anh Tien

130 5 1 5

Elixir và Unicode, Phần 1: Unicode và UTF-8 là gì ?

Nguyen Anh Tien

369 8 2 6

[SLIDE] Introduce To Some Common Cyber-Attack Vectors

Nguyen Anh Tien

166 5 1 6

Tất cả những thứ bạn cần biết về HTTP security headers

Nguyen Anh Tien

1307 20 1 10

Comments

David Black

Nov 27th, 2015 2:07 pm

rất hữu ích.

0 share

Hoàng văn Anh

Dec 8th, 2015 1:43 pm

đúng là chả hiểu gì cả

0 share

+38

Cảm ơn tác giả!!!

0 share

Nguyễn Xuân Quỳnh

Sep 25th, 11:23 am

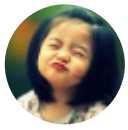
Nhiều bí kíp hay quá. Em mới luyện đến tầng 5 là suýt tẩu hỏa nhập ma rồi anh ạ. @@

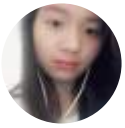
0 share

FACEBOOK

HOT AUTHORS





RESOURCES

Posts

Questions

Videos

Tags

Terms

RSS Feed

Browser extension

Atom plugin

+38

© 2017 **Viblo**. All rights reserved.