Tags: GIT

Tổng hợp 35 lệnh GIT cơ bản

Deft May 19, 2021

Mục lục [ẩn]

Bài viết liên quan

Sự khác nhau giữa Git và GitHub

Cách thêm SSH key vào Gitlab

Cách thêm SSH key vào Github

SSH là gì? Cơ chế hoạt động của SSH

Cách download Git trên Linux

Cách download Git trên MacOS

Cách download Git trên Windows

Git là gì? Developer nào cũng cần biết!

Tổng hợp 35 lệnh GIT cơ bản

Là một developer thì chắc hẳn chúng ta cần phải trang bị kiến thức sử dụng GIT để quản lý mã nguồn và làm việc nhóm. Việc sử dụng GIT thuần thục sẽ giúp chúng ta quản lý mã nguồn và làm việc nhóm hiệu quả hơn.

Trong bài viết này, chúng ta sẽ cùng nhau tìm hiểu một số các lệnh GIT từ cơ bản, trung bình đến các lệnh GIT nâng cao.

Lệnh GIT mức độ cơ bản

Trong phần này chúng ta sẽ tìm hiểu những lệnh GIT cơ bản và được sử dụng thường xuyên nhất.

git config

Git config là câu lệnh mà chúng ta phải thực thi đầu tiên sau khi 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
```

git version

Nhìn vào câu lệnh chắc hẳn chúng ta cũng đoán ra được câu lệnh này dùng để kiểm tra phiên bản git đang sử dụng trên máy.

```
$ git version
```

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>
```

git clone

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

```
git clone <your project URL>
```

git add

Git add là câu lệnh giúp chúng ta thêm tất cả các file code mới 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
```

git commit

Đây là câu lệnh được sử dụng phổ biến nhất, câu lệnh này sẽ giúp chúng ta 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"
```

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.

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.

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>
```

Lệnh GIT 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.

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 trữu 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
https://dev.azure.com/aCompiler/_git/DemoPr
```

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
```

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
```

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>
```

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
```

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
```

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
```

git show

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

```
$ git show <your_commit_hash>
```

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>
```

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.

Lệnh GIT 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. Và hãy sử dụng chúng thật cẩn thận nhé.

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>
```

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
```

git cherry-pick

Git cherry-pick là một lệnh hữu ích. Đó là một lệnhcho 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>
```

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
```

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
```

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
```

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
```

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>
```

git verify-tag

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

```
$ git verify-tag <tag>
```

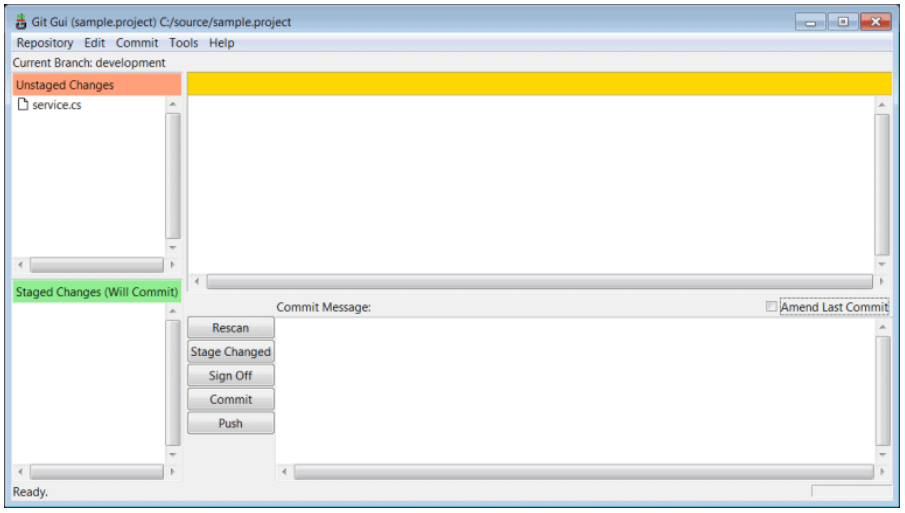
git diff

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
```

git citool

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




```
$ git citool
```

git mv

Đổi tên git file từ tên cũ sang tên mới.s

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

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
```

git help

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

```
$ git help <git_command>
```

git whatchanged

ệnh này thực hiện tương tự như git log nhưng ở dạng th[illegible] đó là do nguyên nhân lịch sử.

```
$ git whatchanged
```

Nguồn

https://dzone.com/articles/top-35-git-commands-with-examples-and-bonus

PREV ARTICLE NEXT ARTICLE

4

Article Rating

Subscribe Login

Be the First to Comment!

0 COMMENTS

Tham gia thảo luận tại group

Trang web liên kết

Baeldung
Tin Tuc So 24h
Gametechz.com

Comment gần đây

Deft on Cách đếm số lượng khoảng trắng trong Java String

Meow on Cách đếm số lượng khoảng trắng trong Java String

Danh on Một số câu hỏi về Spring, Spring Boot được nhà tuyển dụng lựa chọn

codethu on Cách trở thành lập trình viên giỏi? Những điểm cần khắc phục

Deft on Tổng hợp bài tập lập trình hướng đối tượng trong java