framgia

Hướng dẫn về GIT

I. GIT là gì

- Git là hệ thống quản lý phiên bản (version control system)
- Trên git ta có thể lưu các trạng thái của file dưới dạng lịch sử cập nhật, vì thế có thể đưa file đã chỉnh sửa về trạng thái cũ hoặc có thể kiểm tra xem file được chỉnh sửa chỗ nào.

II. Công cụ làm việc với GIT

- GIT BASH
- GIT CMD (git command promt) ví dụ: Cygwin Terminal
- GIT GUI
- GIT SERVER: github.com, bitbucket.org, gitlab.com

III. Đặc tính nổi bật của GIT

- Quản lý source code tập trung trên Git Server
- Chia sẻ code nhanh chóng và an toàn hơn: thay vì phải copy paste, dùng usb chuyển gửi code giữa các thành viên với nhau, thì với git chỉ cần đơn giản là đẩy code lên git server (github, bitbucket), sau đó các thành viên khác tự kéo về máy.
- Git lưu các trạng thái của file dưới dạng lịch sử cập nhật, vì thế có thể đưa file đã chỉnh sửa về trạng thái cũ.
- Các git server đều có chế độ so sánh code mới thay đổi với code cũ giúp ta thuận tiện cho việc kiểm soát sự thay đổi của các file trong source code, bên cạnh đó ta có thể share bản so sánh đấy với cách thành viên để họ biết code thay đổi những chỗ nào.

IV. Quy trình làm việc với GIT của mỗi dự án

1. Chuẩn bị:

- Giả định đã tạo một project trên github(bitbucket) (team-url)
- Fork team-url về tài khoản của mình (self-url)
- Clone team-url về máy, khi này trong máy đã có 1 remote mặc định là 'origin' - gọi là team-remote

git clone [url_project]
git remote -v (kiểm tra các remote hiện có)

- Dùng git remote add để thêm self-url, tạo thêm một remote đến tài khoản github của mình (đặt tên tuỳ ý) gọi là self-remote

git remote add [shortname] [url] (tạo thêm nhánh)

2. Quy trình làm task

2.1 Pull code mới nhất trên server về máy:

Git pull [team-remote] master

2.2 Từ branch master, dùng git checkout -b branchname để tạo branch mới và bắt đầu code - commit bao nhiêu tuỳ ý

git status (Check sự thay đổi so với code trên master)
git add . (add hết tất cả các file thay đổi)
git add [name_file] (add từng file)
git commit -m "name_commit"

2.3 Khi code xong, dùng git log để kiểm tra xem mình đã commit bao nhiêu lần

2.4 Nếu có nhiều commit cho task vừa rồi, dùng git rebase -i để gộp tất cả commit lại còn một commit

2.5 Dùng git checkout master để sang nhánh master, sau đó git pull team-Remote master để kéo code mới nhất từ team-url về máy của mình.

2.6 Checkout sang branch làm task, dùng git rebase master để cập nhật code cho branch task.

2.7 Nếu xảy ra conflict thì sửa conflict (xem ở dưới), nếu không còn conflict, dùng *git push self-remote branchname* đẩy code lên self-url.

2.8 Lên self-url, Tạo pull request cho code vừa đẩy cho team-url, sau đó copy đường dẫn pull request gửi cho team để nhờ check code.

2.9 Trường hợp có comment, quay lại branch task trên máy sửa code rồi làm lại bước **2.2**. Trường hợp pull request được chấp nhận và merge, thì xoá branch và chờ nhận task mới.

V. Một số trường hợp hay gặp khi dùng GIT

1. Sửa conflict:

- Khi xảy ra conflict khi rebase master, terminal sẽ hiện những dòng thông báo có chữ: CONFLICT(..): merge conflict in 'filename'
- Dùng code-editter mở từng file conflict lên, tìm đến những dòng <<<<< HEAD [code trên master] ==== [code trên branch task] >>>>>> branch name
- Xoá những dòng <<<<< HEAD, ======, >>>>>> branchname, sau đó xoá [code trên master] hoặc [code trên branch], hay giữ lại cả 2 tuỳ trường hợp.
- Sau khi sửa hết conflict, dùng terminal: git add, sau đó git rebase --continue để tiếp tục rebase master.
- Nếu huỷ bỏ quá trình rebase, dùng lệnh git rebase --abort.

2. Trường hợp mình gửi pull request nhưng lại có pull request khác được merge:

- git checkout master
- git pull origin master
- git checkout #task
- git rebase master (cập nhật lại code vừa được merge)
- Sửa conflict (xem trường hợp 1 ở trên)
- git push self-remote branch-name -f (thêm -f để đẩy đè lên code cũ)

3. Đổi tên commit:

- git commit --amend
- Sửa tên commit trong terminal editer rồi thoát ra.

4. Đổi tên branch:

- git branch -m new-branch-name

5. Gộp nhiều commit:

- git rebase -i HEAD~n (trong đó n là số commit muốn gộp theo thứu tự từ trên xuống)
- Khi terminal mở text-editer, đổi những chữ 'pick' đầu dòng thứ 2 - n thành chữ cái 'f', khi này các commit dòng 2 - n sẽ được gộp vào commit dòng đầu tiên.
- Thoát text-editer và git log để kiểm tra lại.

6. Trường hợp code xong, muốn gộp code mới vào commit trước đó mà không muốn dùng git rebase -I:

- git add .
- git commit --amend --no-edit

7. Trường hợp muốn trở lại commit trước đó, xoá hết sự thay đổi file của những commit gần nhất:

- git reset --hard HEAD~n (n là số commit muốn xoá)

Hoặc

- git log --oneline
- Tìm commit cần rebase, xác định dãy số index của commit đó
- git reset --hard [index_commit]

8. Trường hợp muốn trở lại commit trước đó, nhưng vẫn giữ code hiện tại:

✓ Cách 1:

- git checkout -b newbranch (tạo branch mới để giữ code hiện tại)
- Quay lại branch vừa rồi, dùng git reset --hard HEAD~n

✓ Cách 2:

- git reset HEAD~n (n là số commit muốn tách)
- git log để kiểm tra commit
- git status để xem code (khi này sẽ hiện rất nhiều dòng file name màu đỏ => đây là những file unstracked)
- Với những untracked files này, mình có thể git add một số file muốn giữ, rồi tạo commit mới theo ý. Hoặc git add . sau đó checkout sang branch khác rồi commit bên branch đó, hay checkout -b sang một branch mới commit đẩy code này sang đó.

9. Trường hợp commit nhầm lên branch master:

- git log để kiểm tra bị nhầm bao nhiêu commit
- git reset HEAD~n (n là số commit nhầm)
- git add .
- git checkout sang branch task cần commit
- git commit

10. Task cũ chưa được merge, muốn làm tiếp task mới:

- git checkout master
- git checkout -b newbranch

11. Task cũ chưa được merge, muốn làm task mới, nhưng task mới cần dùng code của task cũ:

- git checkout branchtask1 (hoặc ko cần nếu đang ở task cũ)
- git checkout -b branchtask2
- git commit --amend (đổi tên commit của task 1, thành tên commit mới)
- Bắt đầu làm task 2. Khi này mọi thứ đều bình thường nếu task 1 được merge mà ko có lỗi. Nếu task 1 phải sửa lỗi, thì hoặc là sửa trên cả 2 task, hoặc là sửa task 1 rồi sau đó sửa task 2 khi conflict (vì task 2 lấy code cũ của task 1).

12. Trường hợp push -f nhầm lên origin/master:

- git checkout master
- git reset HEAD~1 (quay lại 1 commit)
- git push origin master -f (đẩy master của cũ ban đầu vừa reset về trên máy mình lên thay thế master mới vừa push nhầm trên origin)
- git add .
- git checkout -b newbranch
- git commit -m ‘message’ (commit code sang branch mới)

13. Trường hợp xoá nhầm branch master trên máy:

- git checkout sang branch vừa làm
- git checkout -b master (tạo branch master mới)

- git log để mở lịch sử commit, sau đó tìm tới commit được merge gần nhất
- git rebase --hard HEAD~n đến commit vừa tìm
- git pull origin master để cập nhật lại code cho master

14. Trường hợp git pull origin master bị conflict (có thể do commit nhầm lên master trước đó):

- git merge --abort
- git log kiểm tra có commit nhầm lên master hay không, nếu có thì xử lý.
- git reset --hard HEAD~n để xoá một vài commit trên master
- git pull origin master