Git:

Quá trình lưu hoạt động của Github

Khi mà thay đổi file : file nào thay đổi theo từng version thì lưu mới , file không thay đổi thì lưu link về file version trước.

GIT SETUP: CẤU HÌNH CHO GIT

1.Cấp độ 1 Lưu cho global (toàn bộ máy tính).

2.Cấp độ 2 cho riêng user sử dụng máy tính này C/user/setting/git/…

3.Cấp độ 3 chính là file .git/config trong từng repository (kho).

YOUR IDENTITY: DANH TÍNH CỦA BẠN ( biết ai commits(thao tác) )

$ git config --global user.name “kien”

$ git config --global user.name “[trungkien2895@gmail.com](mailto:trungkien2895@gmail.com)”

$ git config –global user.password “Tao@12345”

YOUR EDITOR: BẠN CHỈNH SỬA

Vi dụ: muốn set Notepad++ đấy là một Editor mặc định của Của Git

$ git config --global core.editor “’C:\Program Files\Notepad++\ notepad++.exe’ – multiInst -nosession”

Ghi chú: + MultiInst là bật nhiều cửa sổ cùng một lúc.

+ Nosession là không chứa các tab từ trước.

CHECKING YOUR SETTING : KIỂM TRA CÀI ĐẶT CỦA BẠN

$ git config --list: kiểm tra tất cả.

$ git config user.name: kiểm tra theo từng key (có thể thay đổi đây là 1 ví dụ).

Getting Git: lấy hướng dẫn

$ git help: tất cả những git.

$ git clone –help

$ git config –help

SUMMARY : TÓM TẮT (TÓM LƯỢC)

XÓA BỎ USERNAME, EMAIL, PASSWORD KHỎI GIT

$ git config –global –unset user.password

TẠO REMOTE REPOSITORY: KHO LƯU TRỮ TỪ XA

Git: quản lí phiên bản , máy cài trên máy tính của user

Github: chung tâm của git , git ở trên sever máy chủ

KHỞI TẠO VIỆC THEO DÕI REPOSITORY

$ git init //chưa file cấu hình cần thiết giúp cho quá trình thay đổi khác của các thư mục chứa trong thư mục mẹ

**TẠO LOCAL REPOSITORY** VÀ PUSH LÊN REMOTE REPOSITORY

Các bước đẩy nội dung lên github:

$ git int

Tao file cần bất kì: khi $ git status thì trang thái của nó là working directory.

$ git status : kiểm tra xem có file nào đã được sửa đổi hay chưa.

$ git add <file> // thêm vào vùng commit được (vung này là staging area).

$ git add \*: add nhiều file vào staging area cùng 1 lúc.

$ git reset HEAD <file> (hoặc git rm –cached <file> ): xóa file ra khỏi trang thái trờ commit.

$ git commit -m “Tên người đặt”

( lần 2 muốn bỏ qua bước git add -> commit luôn thì: git commit –a –m “ten” );

Mutilfile : add nhiều file vào staging area cùng 1 lúc: $ git add \*.

Local Repository lifecycle (vòng đời của các tập tin):

* Mỗi tập tin trong thư mục làm việc có thể ở một trong hai trạng thái: tracked hoặc untracked.( theo dõi hoặc chưa được theo dõi);
* Tập tin tranked : a) unmodified , b) modifile , c) staged:
* Tất cả tập tin (commit) được theo dõi: unmodified (không thay đổi) và đang được theo dõi tracked.
* 1) Tập tin mới tạo chưa được add theo dõi: tập tin này ở trạng thái untracked.
* 2) Gid add \* tập tin này chuyển thành đã được theo dõi tracked.
* 3) Commit tập tin này chuyển thành trạng thái unmodified.
* 4) Sửa tập tập tin được theo dõis đó thì trạng thái là modified
* 5) commit nó thì nó chuyển thành staged

Sử dụng gitignore:

* Không theo dõi tập tin được chỉ định (Từ chối mộ số file được thêm vào vùng Staged).
* Tạo file .gitignore bằng câu lệnh touch .gitinore.
* Cầu hình các file .gitignore khai báo những file không muốn theo dõi: me.txt , (folder) /libs.

Local Repository Remove:

* $ rm <file> : xóa bỏ file trong vùng working tree.
* $ git rm <file>: xóa khỏi vùng staged.
* $ git rm –f <file>: xóa khỏi vùng staged khi file mới sửa, xóa luôn.
* $ git rm – cached <file>: xóa vẫn dữ lại nội dung , không xóa hẳn
* $ git reset HEAD <file> (hoặc git rm –cached <file> ): xóa file ra khỏi trang thái trờ commit.
* $ git checkout -- <file.txt>: khôi phục lại.

Local Repository rename:

* $ git mv 123.txt abc.txt: đổi tên 123.txt thành abc.txt.

Local Repository rename:

* $ git reset HEAD <file> (hoặc git rm –cached <file> ): xóa file ra khỏi trang thái trờ commit.
* $ git checkout -- <file>: quay về trạng thái trước khi chỉnh sửa.

Git log:

* $ git log: Nhấn vào q để thoát lịch sử commit.
* $ git log -p -2: lấy 2 commit 2 gần nhất.
* $ git log –stat: xem chi tiết lịch sử hơn.
* $ git log -- pretty=online : Danh sách các commit thay đổi hiển thị bằng 1 dòng.
* $ git log –amend: commit bổ xung vào commit.
* Local: Trên máy tính của chúng ta.

Remote: Trên sever github.

**LÀM VIỆC VỚI REMOTE REPOSITORY:**

Chia sẻ code cho người khác:

* Tạo 1 remote repository bằng tài khoản git hub.
* $ git clone <https://github.com/hoanthptvn/git.git>: nhân bản về máy tính.
* Cd git : di chuyển tới thư mục git.
* Nhánh mặc định là nhánh master.
* $ git remote: xem thông tin remote.
* Origin : mặc định là một tên shortname đại diện cho cái remote này. Trên mỗi máy có thể để tên khác nhau.
* $ git remote -v: có 2 đường link, fetch lấy dữ liệu về , push để đẩy dữ liệu lên
* $ git remote show <shortname>: xem thông tin shortname.
* $ git remote rename origin tedu: đổi tên remote origin thành tedu.
* $ git remote rm tedu: xóa tên remote.

Trong trường hợp có một local muốn liên kết với remote repository để chia sẻ cho người khác:

* $ git remote add tedu <đường lẫn>: theo dõi một remote repository với shortname là tedu.
* $ git fetch tedu: cập nhập dữ liệu mới , để chúng ta nhìn xem remote có gì mới hay không.
* $ git pull <shortname> <nhánh của nó>: để lấy dữ liệu xuống.
* $ git push <shortname> <nhánh của nó>: để đẩy dữ liệu lên remote.

Làm việc với tag trong GIT: Đánh dấu commit nào đó quan trọng bằng một tên dễ nhớ nào đó.

* $ git commit: để đánh dấu commit
* $ git tag: danh sách tag;
* Có 2 loại tag chính: + lightweigth tag: chỉ lưu ít thông tin; + Annotated tag: lưu nhiều thông tin hơn cả message nữa.
* Tạo 1 Annotated tag: $ git tag –a <tên tag> -m <message>
* Tạo 1 Lightweight tag: $ git tag <tên tag>
* 2 tag vừa tạo tương ứng với 2 cái commit.
* Chỉ định tag commit thông qua qua mã: $ git tag –a <tên tag> <mã>.
* $ git push < shortname > <tên tag>: push 1 cái lên remote.
* $ git push < shortname > --tags: push hết tag lên remote.
* Tách nhánh bắt đầu từ một tag nào đấy:
* $ git checkout –b <tên nhánh> <tên tag>: tạo 1 nhánh mới gộp những tag từ tag đấy trở đi. Tánh version thành từng nhánh riêng bản này lỗi thì có thể trở về phiên bản trước.
* Delete xóa tag: $ git tag –d <tên tag>:
* Push lên remote: $ git push <shortname> --delete <tên tag>:

Khái niệm và cách làm việc với Branch:

Tính năng mới có test trên nhánh con , nhánh đấy lại liên kết với nhánh master nó lại đồng nhất với nhau. HEAD là một con trỏ.

* $ git branch: liệt kê các nhánh.
* $ git branch <tên nhánh>: tạo mới một nhánh. 2 cái nhánh đang cùng trỏ đến 1 commit cả master và nhánh mới tạo.
* $ git log online: xem thông tin.
* $ git checkout <tên tag>:muốn chuyển sang nhánh đã tồn tại.

Các nhánh merge(hòa hợp) nhanh và xử lý xung đột trong GIT.

* Tạo một nhánh mới add file mới và commit , khi chúng ta thay dổi nhánh thì working directory( thư mục làm việc ) tự động reset nội dung thpeo nhanh con trỏ đấy
* Tao nhánh issue53: $ git branch issue53:
* + trong nhanh issue53 tạo file index.txt có nội dung bất kì.
* + khi chuyển nhánh về master.
* + ở nhánh master chúng ta merge nhánh issue53:
* $ git merge issue53: thì nó tự động commit mới của nhánh issue53 qua nhánh master.
* $ git push –set-upstream origin <tên nhánh>: push toàn bộ nội dung của nhánh đó lên.
* Hai người ở hai máy khác nhau cùng sửa một đoạn code merge cùng nhau:
* Git nó sẽ không biết chọn của ai.
* Phải config merge bằng tay: $ git config –global merge.tool : kiểm tra xem
* $ git config –global merge.tool deiffmerege:
* $ git mergetool: deiffmerege

Quản lý nhánh:

* $ git branch: xem danh sách của nhánh.
* $ git branch –v :xem chi tiết commit mới nhất ở mỗi nhánh.
* $ git branch –merged: xem nhánh nào được merge với nhau chưa.
* $ git branch –no-merged: xem nhanh nào chưa được merge với nhau.
* $ git branch –d <tên branch>: xóa branch đã merge rồi.
* $ git breach –D <tên branch>: xóa branch chưa được merge.

Tìm hiểu Rebasing trong GIT: Tích hợp sự thay đổi từ nhánh này sang nhánh khác trong git có merge và Rebasing.

Rebasing:

* Chúng ta có thể làm một số kịch bản.
* Tạo thêm một nhánh , tiếp đó chúng ta sửa file trong nhánh mới.
* Trở về master chúng ta sửa cùng tên file. Chển đến nhánh vừa tạo
* Sau đó chúng ta Rebasing đến nhánh master: $ git rebase master.

FORK: thư viện ở trên namespace của người ta : lợi ích là đọc code người ta học hỏi người ta , giúp làm việc tạo branch trên git của mình có thể pull request.

Trước hết phải fork về namespace -> tạo branch. : sau đó gửi file thay đổi đến namespace git của họ.

Bảo trì dự án trong Github:

**Tổng hợp lệnh GIT**

1 log,show,diff:

* $ git log: xem lịch sử commit.
* $ git show <số commit>: hiển thị thông tin chi tiết commit.
* $ git diff: kiểm tra thay đổi.

2 checkout,reset:

* $ git reset HEAD <file> (hoặc git rm –cached <file> ): xóa file ra khỏi trang thái trờ commit.
* $ git checkout -- <file>: quay về trạng thái trước khi chỉnh sửa.

3 branching & Merging:

* $ git branch: xem danh sách của nhánh.
* $ git branch –v :xem chi tiết commit mới nhất ở mỗi nhánh.
* $ git branch –merged: xem nhánh nào được merge với nhau chưa.
* $ git branch –no-merged: xem nhanh nào chưa được merge với nhau.
* $ git branch –d <tên branch>: xóa branch đã merge rồi.
* $ git branch –D <tên branch>: xóa branch chưa được merge.
* $ git merge <tên branch>: kéo thay đổi của tên branch vào branch đang được sử dụng.

gitignore:

* $ gitk: Để xem nhánh.
* Gitignore: bỏ qua một số file không muốn nó commit không muốn hiển ra khi gõ git status.
* Module: chalk.

4 git reset:

* $ git reset –sort <to\_commit>: quay lại trạng thái modified: chuẩn bị commit.
* $ git reset –mixed <to\_commit>: quay lại trang thái modified: chưa được chuẩn bị commit.
* $ git reset –hard <to\_commit>:quay lại và làm mất hẳn file.

5 git revert: bỏ commit tùy ý

* $ git revert <to\_commit>: đảo ngược.

6 git remote:

* $ git remote: xem thông tin remote.
* $ git remote -v: có 2 đường link, fetch lấy dữ liệu về , push để đẩy dữ liệu lên
* $ git remote show <shortname>: xem thông tin shortname.
* $ git remote rename origin tedu: đổi tên remote origin thành tedu.
* $ git remote rm tedu: xóa tên remote.

7 git:

* $ git remote add tedu <đường lẫn>: theo dõi một remote repository với shortname là tedu.
* $ git push –u origin master: lần đầu, lần 2 trở đi $git push.

8 git clone & pull: liên kết làm việc 2 máy.

Máy A là chủ và B:

A push code lên:

* $ git push.

A tạo branch:

* $ git checkout –b <tên branch>: tạo branch và trỏ tới luôn.
* $ git push <shortname> <nhánh của nó>: để đẩy dữ liệu, branch lên remote.

A trên remote branch , A muốn B kiểm tra code của mình:

* click new pull request: muốn branch này ghép vào branch master. Đặt tên cho nó đúng nội dung giải thích về pull request bạn muốn làm. ( trả về trạng thái opent)

B pull code về:

* $ git clone <https://github.com/hoanthptvn/git.git>: nhân bản về máy tính.
* $ git pull: tải code về máy.

B xem pull request của A tạo ra: thấy ok comment lại.

A Merge pull request: để gộp vào master.

A xóa branch đấy đi. Tải commit về $ git pull.

2. B muốn tải code về từ A thông qua branch:

* $ git fetch <shortname> <nhánh mà chúng ta muốn tải>:

9 resole confict( giải quyết xung đột ) sử dụng merge :

Khi nào xung đột:

+ Hai bên cùng sửa một file , cùng một dòng nào đấy.

+ A delete file X , B không xóa delete file X mà chỉ sửa X.

Cách giải quyết:

A tạo branch và push code lên vào tạo full request:

A phân quền :

* Vào setting: Collaborators.

A và B cùng sửa 1 file cùng 1 vị trí:

* Tạo brand và cùng push lên remote.
* Tạo pull request.

B Merge pull request: để gộp vào master.

A trỏ về branch master ,$ git pull : tải code về.

A quay lại branch kia:

* $ git cheackout <tên branch A>:
* $ git merge master: merge master vào thằng branch A. nó sẽ tạo ra một commit mới.
* Sửa lỗi trong file xóa những dòng === >>><<<.
* $ git add \*, $ git commit –m “ten’;
* Push nó lên remote;