**GIT**

**I) Bài học**

\*lưu ý\* :

-cẩn thận “git push origin master”  vì nó push thẳng lên master, k có Pull Request => Nên, ko đc push master, chỉ push branch của mình th

**1) git init :**

tạo 1 folder project mới chưa có gì -> cd \*that-folder\* -> git init

 -> để cho git quản lý folder này (nó tạo ra folder .gitinit hay tên gì đó quên ròi để qly nma thường sẽ bị ẩn đi)

- nếu là lần đầu chạy git sẽ phải xác minh mail, name

**2) git status :**

- coi đang ở branch nào

- có thay đổi nào chưa add / commit

**3) git add :**

- syntax : git add .

(add tất cả)

- syntax : git add Cat.js Mouse.js

(add chỉ 2 file Cat.js Mouse.js)

đưa những sự thay đổi từ Working Directory vào 1 Staging area

**4) git commit:**

- syntax : git commit -m 'abc'

đóng Staging Area hiện tại và dán tên "abc" nếu muốn

**5) git log**

- coi các commit hiện tại đang có

**6) git show \*id commit\***

- show nội dung chi tiết của 1 commit

**7) git diff**

- show chi tiết sự thay đổi của các file đang bị modified (coi bằng git status)

- enter cho tới khi xuất hiện "(END)"

- nhấn "q" để thoát ra

**8) thuật ngữ :**

- working directory : chứa các sự thay đổi chưa đc git add

- staging area : chứa các sự thay đổi đã đươc git add, chưa đc git commit

- git repository : chứa các commit

- ngoài lề : trong terminal, ghi "code ." nó sẽ mở thử mục hiện tại bằng vscode(xanh)

**9) git checkout --**

- syntax : git checkout -- cat.js

(undo lại thay đổi trong file cat.js chưa đc git add

hay

còn trong Working directory, chưa lên Staging Area)

**10)  git reset**

- syntax : git reset HEAD cat.js

(hnhu tùy vào branch thì ghi HEAD hay tên branch)

- để đẩy các sự thay đổi đã Git Add, chưa Git commit xuống 1 bậc

từ Staging Area -> Working directory

**11) git branch**

- coi mình đang ở branch nào trong local chứ ko phải trong remote github)

**12) git checkout -b**

- syntax : git checkout -b feature/dog-class

- tạo và chuyển vào branch mới tên "feature/dog-class"

- thường khi muốn phát triển tính năng mới thì checkout với cách đặt tên như v rồi mới code

**13) git merge**

- nếu muốn merge B vào A : A <--- B

thì phải :

git checkout A

git merge B

- nó sẽ đưa các commit mới của B vào A

**14) git checkout**

- chuyển qua branch đã có

**15) git branch -D \*tên branch\***

- xóa branch

**16) git reset --soft \*ID commit\***

- quay về cái commit với ID truyền vào (That-Commit)

- Và những thay đổi trong các commit sau That-Commit thì unpack, chuyển từ Git Repo xuống Staging Area

**17) git reset --mixed \*ID commit\***

- quay về cái commit với ID truyền vào (That-Commit)

- Và những thay đổi trong các commit sau That-Commit thì unpack, chuyển từ Git Repo xuống thẳng Working Directory

**18) git reset --hard \*ID commit\***

- quay về cái commit với ID truyền vào (That-Commit)

- Và những thay đổi trong các commit sau That-Commit thì bị mất hoàn toàn

**19) git revert \*ID commit\***

- đảo dấu cộng trừ trong commit đó

- tự tạo 1 commit mới tên : Revert"tên commit" để thực hiện sự thay đổi trên

-> nguy hiểm, dễ conflict nên tránh dùng

**20) .gitignore**

- terminal : npm init

-> tạo file package.json -> lưu thông tin về project

- terminal : npm install chalk --save

-> để install thư viện chalk, "--save" để lưu thông tin vào file package.json

-> nó sẽ tạo ra thư mục chưa thứ viện

-> khi gửi code cho mn thì ko nên gửi luôn cả thư mục trên vì nặng. Với cả mình đã lưu thông tin thư viện cần install vào package.json rồi. Nên ngkhac khi nhận code chỉ cần gõ : "npm install" sẽ tự install tất cả thư viện đã ghi trong package.json

-> phải ignore folder thư viện trên

- tạo file .gitignore -> ghi tên folder hoặc file muôn ignore vào từng dòng

- khi git status sẽ ko còn thấy folder / file đó nữa

- nên ignore khi folder/ file chưa commit

**21) git remote add origin \*link github repo\***

- để kết nối prj hiện tại với github repo

- đứng ở nhánh nào cũng đc, như nhau

**22) git remote -v**

- để xem prj hiện tại đang trỏ tới link github nào

**23) git push -u origin master**

- đẩy nhánh master lên

- dùng khi lần đầu push code (tạo upstream branch cho nhánh master)

- chỉ đẩy các commit của nhánh master lên thôi

**24) git push : A->B**

a) lần đầu đẩy lên server phải tạo upstream nên :

**git push -u origin master (?)**

A : local repo

B : Remote repo

**b) git push**

đẩy data lên upstream branch của branch đang ở htai

A : local branch

B : Remote upstream branch

**c) git push origin \*tên branch\***

đẩy  that branch trên lên remote (kể cả đang đứng ở branch khác)

A : local \*tên branch\*

B : remote repo

**d) git push origin --all**

đẩy tất cả nhánh ở local lên remote

**e)  git branch -a :**

ktra các nhánh ở remote

**f) git fetch:**

tìm nạp nhánh ở remote mà ko làm thay đổi commit ở local

**g) git pull:**

là tìm nạp + merge. Nên nó dúng fetch nma tự động merge vô local nữa

**25) git config --global credential.helper store**

- để lưu vào ổ đĩa tài khoản github vào 1 file (~/.git.credentials)

- ko rcm, vì người khác dùng laptop bạn sẽ mở ra thấy đc thông tin dễ dàng

**26) git config --global credential.helper "cache --timeout=18000"**

- nó sẽ lưu thông tin đăng nhập vào RAM (khó đọc hơn đĩa)

- sau 5 tiếng sẽ mất

**27) git clone \*link\***

- clone nguyên prj về máy

- tạo 1 folder trong folder hiện tại, với tên là tên prj và chứa prj trong đó

(trong laptop t đang test là ở folder 2-commit)

**\*28) Quy trình làm nhóm Pull Request**

**a) git checkout -b \*tên branch là task\***

- khi bắt đầu 1 task thì :  Pull code về -> checkout ra 1 branch mới -> code

**b) git push origin \*tên branch trên\***

- code xong -> add, commit -> push branch đó lên github

**c) tạo 1 cái Pull Request trên Github**

- vào github, vào Pull Request hoặc mò ở đâu đó để có nút Create Pull Request ĐỐI VỚI ĐÚNG BRANCH MÌNH PUSH LÊN

**d) review code**

- review xem code ổn chưa

d.1) review code online (github)

- vào Pull Request

- chọn request muốn review

- vào tab commit ở taskbar dưới r coi code diff

- oke thì qua tab Conversation ở taskbar dưới r Merge

d.2) fetch branch into local to test offline (optional)

- git fetch origin  \*branch muốn test\*

- git checkout \*branch đó\*

- xong r chạy test ok chưa

- ok thì cmt "ok hay Approve" cho cái Pull Request, rồi bấm Merge y như bước d.1

d.3) approve the pull request

**e) merge code**

- review ok thì merge code vô master

**\*29) Xử lý Conflicts**

- Khi nào xảy ra Conflict?

+) Thay đổi cùng file + cùng dòng

+) A xóa file X, B chỉnh sửa file X

-> vd: "1-commit" code RUN, Pull Request | "2-commit" code Jump, Pull Request

Xong admin merge Pull Request Jump.

Khi Merge Pull Request RUN bị báo Conflict

Thì lúc này đứa ở "1-commit" chịu trách nhiệm resolve

- Cách 1 :

+) Dùng "git rebase"

đầu tiên là về checkout master, pull code về(có code JUMP mới)

rồi, checkout lại branch mà code Run đó (QUAN TRỌNG)

rồi, ghi "git rebase master" -> để rebase lại từ master vào branch htai

+) Resolve conflict

rồi, nó hiện 2 code Run và Jump -> chọn chỉnh sửa lấy cũ giữ mới hay ngược lại, hay lấy cả hai -> nchung là sửa lại cho oke

rồi, git status để ktra -> git add \*tên file\* để cho git biết là mình resolve conflict rồi

rồi, ghi "git rebase --continue" -> nếu bị chạy mh vim (# xanh) -> tra gg Exit Vim

+) Push lên lại với "-f"

rồi, ghi "git push origin \*tên branch\* -f" (nhớ phải có -f vì hnhu nó sửa lại thông tin commit cũ nên sẽ ghi đè lên commit cũ)

rồi, admin check lại xem hết conflict thì bấm Merge

- Cách 2 :

+) Merge updated master to feature branch

đứa A chịu trách nhiệm conflict thì vào code checkout ra Master -> git pull code mới về (lúc này origin master đã có code mới của B)

rồi, checkout branch mà đang đang code conflict đó

rồi, git merge master

+) Resolve conflict

xử lý conflict, giữ code mới hay cũ hay cả 2

+) commit and push

git status kiểm tra sửa chưa

git add .

git commit -m '...'

git push origin \*branch conflict\*

rồi, lên github merge dô lại bthuong

29) nếu tkhoan member ko đc cho push lên :

- chủ repo vào tab Settings

- vào Collaborators ở cột trái

- r thêm người đó vào

- ngkia vào accept invite

**II) Câu Hỏi :**

1) undo add / commit làm sao?

2) tất cả thay đổi đều ở trong working tree

vậy working tree/ staging area/ git repo là  chỉ có 1 đối với 1 device (độc lập với branch)

hay mỗi branch sẽ có working tree/ staging area/ git repo riêng

3) code rồi mà quên checkout -> code ở sai nhánh thì làm sao

4)

- checkout -b branch Con. Code, commit 1 ở branch con. Xong checkout branch Cha, code và commit 1 cái mới ở branch Cha. Thì khi merge Con vào Cha sẽ như nào.

- Nếu hiểu Merge Cha vào Ông Nội thì Ông Nội sẽ tới commit gần nhất khi checkout đứa Con đầu tiên, hay, tới commit gần nhất của branch Cha? ( t nghĩ là học tới Conflict rồi hẵng test cái này)

Ông nội merge Con vào đc ko?

Con merge Ông nội vào đc ko?

5) nếu xóa branch thì branch con có bị xóa theo ko

6) Có 1 vde là, khi làm tới bài 5

hiện tại master -> checkout ra feature.. -> checkout ra timhieubranch

- khi git log thì hiện hết cho tới commit mới nhất của timhieubranch

- merge Feature vào Master

- git log chỉ thấy tới commit hiện tại của Feature và Master

- phải checkout vào timhieubranch mới thấy commit mới nhất cửa timhieubranch

- timhieubranch đc checkout trỏ tới commit timhieu đầu tiên

7) git push | git pull | git fetch

8) git push origin \*branch\*

- là nó push cái  cái branch của mình lên github / hay nó push code lên branch có trên github

- vậy vd cái branch đó đã có trên github r, và code 1 cái khác thì sao?

- thử code trong branch A nhma push lên branch B (chưa có/ đã có) trên github

9) thử commit 2 cái trong 1 branch, rồi push branch đó lên xem thử khi tạo Pull Request nó hiện tên commit nào, hay tên branch

**III) Trả lời:**

1) undo : git reset

2) theo mình hiểu thì :

working tree/ staging area/ git repo là chỉ có 1

branch sẽ chỉ chưa các sự thay đổi đã đc commit

vd: khi mình ở branch nào mà commit thì commit sẽ đc đưa vào branch đó (và đồng thời có trong git repo của máy mình ?)

3) phải để code ở status Chưa commit (ở Staging Area trở xuống). (dùng git reset nếu lỡ commit | dùng gitk để coi id commit)

Sau đó checkout branch mới, rồi commit ở branch mới -> thì code mới sẽ đc lưu ở branch mới, branch cũ ko lưu code mới

Còn nếu muốn để code ở 1 branch cũ mà khác vị trí commit hiện tại với branch hiện tại đang dùng thì ko thể checkout qua đc.

6) git log chỉ show các commit của branch hiện tại (ko có commit mới của cha từ khi checkout -b branch hiện tại) thôi. Còn các commit mới của con của branch thì ko show

5) xóa branch cha sẽ ko bị xóa branch con

9) Trong màn hình Create Pull Request, có 2 ô : 1 ô name, 1 ô description

1 PR có nhiều commits

Khi bấm danh sách các Pull Request  thì cái tên hiển thị là nội dung của ô Name

nó sẽ cập nhật commit liên tục của branch mỗi khi push branch đó lên, kể cả khi đã đc Create PR

----------------

lát tạo thêm 1 commit trong timhieubranch rồi tạo thay đổi mới xem có checkout qua feature đc nữa k -> ko -> các branch ở vị trí cùng commit htai thì mới checkout qua nhau kể cả có code mới mà ko bị Aborting đc thôi

hiện tại đang:

- code mới trong timhieubranch

- checkout qua feature

- git add trong feature

- checkout qua timhieubranch

- git commit trong timhieubranch

-> kết quả : code mới đc lưu vào timhieubranch -> commit sẽ đưa staging area vào branch mà nó đc commit

—------------------------------------------------------------------------------------------------------

**Case Study**

1. **Already having src code at local -> wanna push to github**

* Create repo on github WITHOUT ANY FILES (gitignore, readme,..)
* At local terminal :
* cd to your project
* git init -b master
* git add .
* git commit -m ‘asdasdada’
* git remote add origin \*your link github repo\*
* git remote -v //(to check)
* git push origin master