**GIT AND GITHUB TITV**

[**https://docs.github.com/en/get-started/using-git/about-git**](https://docs.github.com/en/get-started/using-git/about-git)

1. **Định nghĩa git**

Version Control System

Là một hệ thống giúp chúng ta quản lý (kiểm soát) phiên bản (version)

Hệ thống giúp lưu trữ mọi thay đổi của source code

Hỗ trợ nhiều người làm việc cùng lúc

Xem đưa nào thay đổi code

Revert các thay đổi, đưa code về version cũ hơn, không lo mất code

* **Vậy GIT cũng là 1 version control ra đời năm 2005, tác giả là Lunus Torvald**

**Nguyên tắc**

Trong git có 1 kho lưu trữ trung tâm (repository) -> có thể lấy dữ liệu về thực hiện các công việc, cập nhật dữ liệu lên kho trung tâm -> có thể gặp các xung đột -> git sẽ cung cấp các công cụ để giải quyết các xung đột

**Lợi ích**

Đảm bảo không làm hỏng bất cứ thứ gì vĩnh viễn

Giúp chúng ta có thể cộng tác với nhiều người

Phát triển phiên bản mới mà không ảnh hưởng đến phiên bản cũ mà chúng ta đang có

Có thể quay lại phiên bản trước bất kì lúc nào mà bạn muốn

Công bố sản phẩm của mình cho nhiều ng cùng tham gia hoặc xem hoặc tải về

hoặc cùng nhau phát triển

1. **Cài đặt git**

Gõ trên cmd git –version => để kiểm tra có cài đặt git chưa

***Cách cài đặt***

git-csm.com để download

1. **Cách sử dụng**

## **Basic terminal comment**

### **-Mũi tên lên hoặc xuống để thay đổi câu lệnh trước đó hoặc sau đó**

### **-cd : change directory – thay đổi thư mục**

|  |  |
| --- | --- |
| cd .. | quay lại thư mục trước đó (quay lại thư mục cha) |
| cd Youtube |  |
| cd Yotube / Git |  |

### **-dir (ms windown) = ls (Linux, MaxOs) : hiển thị danh sách tập tin bên trong thư mục hiện hành**

|  |  |
| --- | --- |
| dir | Liệt kê toàn bộ file and thư mục trong thư mục hiện hành |

### **-mkdir “folder name” = mkdir folder name : tạo ra một thư mục mới**

|  |  |
| --- | --- |
| mkdir source | Tạo thư mục source |

### **-touch “file name” : tạo một tập tin mới**

|  |  |
| --- | --- |
| touch source | Tạo file source |
| touch ‘file 1.txt’ | Tạo file 1.txt |

**Lưu ý :**

* nếu tên file có khoảng trắng ví dụ như Thu muc 1 nên bỏ trong dấu ngoặc ‘’.
* nếu file đã có sẽ không tạo file mới cũng như không ghi đè (đối với Win)

### **-echo “Vanban” : in/xuất ra câu gì đó**

**> chỉ định hướng xuất dữ liệu**

|  |  |
| --- | --- |
| echo “Text” > “Tenfile” (Overide) | Xuất chữ Text trong file Tenfile. Lưu ý nó sẽ bỏ hết nội dung trong Tenfile và chỉ hiện chữ Text |
| echo “Text” >> “Tenfile” (Overide) | Sẽ không ghi đè, chỉ thêm chữ Text vào thui. |

**Lưu ý :**

* Có thể tạo file mới hoăc vừa ghi vừa tạo file mới

### **-cat** **“file name”** **: hiển thị nội dung của file**

|  |  |
| --- | --- |
| cat “Vidu1.txt” | Xem nội dung trong file Vidu1.txt |

### **-diff file1 file2 : tìm sự khác biệt giữa hai file**

### **-rm “filename” : xóa file**

### **-rm -d “foldername” : xóa thư mục (rỗng)**

### **-rm -r “foldername” xóa thư mục (có nội dung)**

1. **Các lệnh github cơ bản**

**.gitignore :** chỉ định không cho git theo dõi. Đầu tiên tạo file này sau đó để tên những file không muốn theo dõi vào

**git push [link] [tenBranch] :** đẩy dự án từ repo lên github (remote)

**git remote add origin [link]** : giúp thay thế link ở trên thành origin

**git clone [link]** : lấy dự án từ github về repo

**git clone [link] [foldernameLocal]**

**git pull :** đồng bộ trên github về máy

**git branch [tenBranch]** : tạo branch

**git branch** : xem có bao nhiêu branch / đang ở branch nào

**git branch -d [tenBranch] :** xóa branch

**git checkout [tenBranch]** : chuyển nhánh

**git checkout -b [tenBranch] :** vừa tạo branch vừa chuyển

**git reset :** nếu không muốn git add

**git log** : xem lại lịch sử commit, tên, comment

**git log –oneline :** tương tự như trên nhưng gọn hơn

**git checkout [idCommit] :** quay lại bất kì commit phía trước nào

**git checkout [branchname] :** quay lại commit hiện tại

**git init - -bare** : tạo remote repository

**git fetch** **origin**: Bạn đã cập nhật được những thay đổi mới nhất từ GitHub về máy của mình, nhưng những thay đổi đó chưa ảnh hưởng đến mã bạn đang làm việc. Bạn có thể xem các thay đổi đó, và sau đó **quyết định có tích hợp chúng vào nhánh của mình không**.

**Tổng hợp 2 nhánh ở repo**

chuyển về branch master -> **git merge [tenBranchMuonMerge]** : dùng để tổng hợp 2 nhánh lại với nhau ở repo

* *Lưu ý việc này có thể tạo ra conflicts*

**Tùy trường hợp :** Nếu conflicts -> sửa -> git add -> git commit (lưu ý không ghi chú)

## **Tạo branch ở local và đẩy lên github**

* Tạo branch
* git push -u origin [tenBranch] => chỉ dùng lần đầu lần sau chỉ git push

## **Kéo branch ở github về local**

* Tạo branch ở github
* Quay về nhánh master/ main
* git fetch origin
* git checkout -b [staging] origin/[staging]

## **Tổng hợp lại 2 nhánh với nhau ở github (remote)**

* Dùng pull requests
* New pull requests
* Chọn nhánh muốn merge
* Create pull requests
* Create pull requests
* Merge pull requests
* Đặt tên => Confirm requests
* Quay trở về máy -> quay lại branch master/ main **(git checkout main/master)**
* git pull origin tenNhanh

## **Fork dự án người khác về chỉnh sửa và muốn gửi lại cho người khác**

* Pull requests
* New pull request
* Create pull requests
* Create pull requests
* Đợi ngta đồng ý

Đối với tác giả

* Pull request
* Đồng ý -> merge pull requests -> confirm requests

1. **Cấu hình thông tin cho Repository (giúp xác định ai )**

**git config -l :** xem cấu hình

**git config -l --global :** xem cấu hình global

**git config -l --local :** xem cấu hình local

**git config –global user.name “[dienVao]” :** cấu hình user name global

**git config –global user.email “[dienVao]” :** cấu hình user email global

**Lưu ý :**

* global : ảnh hưởng tất cả repo
* Local : chỉ ảnh hưởng 1 repo

1. **Các trạng thái của git**

Working directory -> git add -> staging are -> git commit -> repository (HEAD)

1. **Gitignore**

-Bỏ qua file , thư mực, file log

B1: Tạo file “.gitignore”

B2: C1: Mở bằng Notepad -> ghi tên file /thư mục muốn bỏ theo dõi vào

B2: C2: echo “tenfileMuonBo/” >> .gitignore (Phải ra ngoài thư mục gốc “cd ..”)

**Lưu ý :** / vào cuối nếu là thư mục, >> để tránh ghi đè file

Cần tìm hiểu những file nào không nên đưa lên repository

1. **Central Repositoty (Remote Repository)**

B1: git init –bare “NameRemot”

# **9. Git Large File Storage (LFS)**

Dưới đây là hướng dẫn chi tiết từng bước để bạn có thể sử dụng **Git Large File Storage (LFS)** để quản lý tệp tin lớn trong GitHub và đẩy chúng lên kho lưu trữ.

**Bước 1: Cài đặt Git LFS**

1. **Cài đặt Git LFS**: Nếu bạn chưa cài Git LFS, bạn cần cài đặt nó trước tiên. Tải và cài đặt Git LFS từ trang chính thức: [Git LFS](https://git-lfs.github.com/).
   * **Trên macOS**:
   * brew install git-lfs
   * **Trên Windows**:
     + Tải Git LFS từ trang web và làm theo hướng dẫn: <https://git-lfs.github.com/>.
   * **Trên Linux**:
     + Sử dụng các lệnh sau tùy vào bản phân phối của bạn:
   * sudo apt-get install git-lfs # Debian/Ubuntu
   * sudo yum install git-lfs # CentOS/RHEL
2. **Khởi tạo Git LFS**: Sau khi cài đặt xong, bạn cần khởi tạo Git LFS trong repository của mình:
3. git lfs install

**Bước 2: Theo dõi các tệp tin lớn bằng Git LFS**

1. **Xác định tệp tin lớn**: GitHub cảnh báo rằng tệp SpringMWC/Thuchanh\_Spring\_update.docx của bạn lớn hơn 50 MB. Bạn cần chỉ định Git LFS theo dõi các tệp có đuôi .docx.
2. **Sử dụng Git LFS để theo dõi các tệp lớn**: Để Git LFS theo dõi các tệp .docx, bạn chạy lệnh sau:
3. git lfs track "\*.docx"

Lệnh này sẽ thêm quy tắc vào tệp .gitattributes trong repository của bạn, thông báo cho Git LFS theo dõi các tệp .docx.

1. **Kiểm tra lại tệp .gitattributes**: Git LFS đã thêm quy tắc vào tệp .gitattributes. Bạn có thể kiểm tra nó bằng cách mở tệp này:
2. cat .gitattributes

Nó sẽ có dòng như sau:

\*.docx filter=lfs diff=lfs merge=lfs -text

**Bước 3: Thêm và commit thay đổi**

1. **Thêm .gitattributes vào staging**: Sau khi cấu hình Git LFS theo dõi các tệp .docx, bạn cần thêm thay đổi này vào staging:
2. git add .gitattributes
3. **Commit thay đổi**: Commit thay đổi vào Git với thông điệp thích hợp:
4. git commit -m "Track large .docx files with Git LFS"

**Bước 4: Thêm các tệp lớn vào Git LFS**

1. **Thêm tệp .docx vào staging**: Nếu tệp SpringMWC/Thuchanh\_Spring\_update.docx chưa được thêm vào staging, bạn cần thêm nó:
2. git add SpringMWC/Thuchanh\_Spring\_update.docx
3. **Commit tệp lớn vào Git**: Commit các tệp tin đã được Git LFS theo dõi:
4. git commit -m "Add large .docx file to Git LFS"

**Bước 5: Đẩy thay đổi lên GitHub**

1. **Đẩy các thay đổi lên GitHub**: Cuối cùng, bạn có thể đẩy các thay đổi lên GitHub:
2. git push --all

Git LFS sẽ tự động xử lý việc tải tệp lớn lên GitHub.

**Bước 6: Kiểm tra lại**

1. **Kiểm tra lại trạng thái repository**: Bạn có thể sử dụng lệnh sau để kiểm tra trạng thái hiện tại của repository và đảm bảo mọi thay đổi đã được commit và đẩy lên GitHub:
2. git status
3. **Kiểm tra GitHub**: Sau khi đẩy thành công, bạn có thể vào GitHub để xác minh rằng tệp .docx đã được lưu trữ thông qua Git LFS.

**Tóm tắt:**

* Cài đặt và cấu hình **Git LFS** để theo dõi các tệp tin lớn.
* Thêm và commit các thay đổi liên quan đến Git LFS.
* Đẩy các thay đổi lên GitHub để Git LFS lưu trữ các tệp lớn.

Giải pháp này sẽ giúp bạn quản lý các tệp lớn vượt quá giới hạn 50 MB mà GitHub khuyến nghị, đồng thời giúp repository của bạn được duy trì và quản lý hiệu quả hơn.