|  |
| --- |
| **GIT** |
| **1. Khái niệm** |
| - **Git** là một phần mềm , **hệ thống quản lý mã nguồn phân tán.**  - Với Git bạn có thể lưu lại lịch sử và trạng thái của các file trong dự án, rollback về một lịch sử bất kì mà không cần phải backup  - Để sử dụng Git ta cần kết hợp với một dịch vụ lưu trữ mã nguồn trực tiếp như Github. Từ đây các thành viên sẽ đẩy dữ liệu từ máy tính cá nhân lên Github |
| **2. Cài đặt** |
| - Down tại: <https://git-scm.com/download/win>  - Sư dụng git tại một folder nào đó: **Click phải** 🡺 **Git bash here**  - Update lên bản mới nhất:  Dùng dòng lệnh sau: git clone https://github.com/git/git |
| **3. Cấu hình** |
| - Trước khi sử dụng git thì cần phải cấu hình 2 thông số là **username** và **email** để có thể thực hiện các thao tác trên github   * **username**: git config --global user.name “Tên của bạn” Vd: git config --global user.name “Binh Tran” * **email**: git config --global user.email “Email của bạn”   Vd: git config --global user.name “[tranngocbaobinh@gmail.com](mailto:tranngocbaobinh@gmail.com)” |
| **4. Thao tác trong giao diện CLI** |
| |  |  |  | | --- | --- | --- | | **Lệnh** | **Ý Nghĩa** | **Ví dụ** | | **cd** | Change directory : Truy cập vào folder | **cd workspace**  *// Truy cập vào folder có tên workspace*  **cd /d**  *// Truy cập vào ổ đĩa D* | | **cd..** | Quay lại thư mục gốc |  | | **dir**  **ls** | Directory : Liệt kê các thư mục |  | | **mkdir** | Make directory: Tạo thư mục mới | **mkdir newfolder**  // Tạo thư mục mới tên newfolder | | **rmdir** | Remove directory: Xoá thư mục | **rmdir newfolder** hoặc **rd newfolder**  // Xoá thư mục mới tên newfolder | | **rename** | Đổi tên thư mục | **rename newfolder myfolder** | |
| **5. Git- Tạo Repository** |
| - **Repository** : Còn gọi là **Repo** - Kho , đây là nới chứa tất cả mã nguồn của dự án được quản lý với Git  - Có 2 loại Reposity:   * **Local repository** : Là repo được cài trên máy tính , repo này sẽ được đồng bộ với Remote reposotory bằng các lệnh của Git * **Remote repository:** Là repo được cài trên server chuyên dụng - Github   - Để khởi tạo một Repo:   1. Truy cập vào folder dự án: cd /d/workspace 2. Dùng lệnh **git init** : **git init**   - Lúc này ta sẽ đứng ngay file master là file gốc của dự án |
| **6. Git - Branch** |
| - **Branch** ( Cành cây ): Tương ứng với nhiệm vụ ( Task ) mà ta sẽ tạo ra một Brand dựa trên Repo để làm việc trên đó. Các branch này sẽ hoạt động riêng lẻ và không ảnh hưởng nhau.  - Branch là những phân nhánh ghi luồng thay đổi lịch sử, có thể tiến hành nhiều thay đổi trên cùng một repository.  - **Branch master :**  Đây là Branch đóng vai trò cập nhật dữ liệu và đồng bộ với remote repository  Khi giải quyết các task chỉ nên làm trên branch nhánh sau đó merge lại với branch master  **- Cú pháp tạo branch :**   |  |  | | --- | --- | | **Lệnh** | **Ý Nghĩa** | | **git branch** | Liệt kê tất cả branch hiện có | | **git branch TenBranch** | Tạo một Branch mới có tên là TenBranch | | **git checkout TenBranch** | Chuyển qua làm việc từ brand master sang brand TenBranch | |
| **6. Git - Commit : Lưu thay đổi trên branch** |
| - Sau khi làm việc xong trên một branch nào đó nếu bạn chuyển qua branch khác mà không commit tất cả những gì đã làm sẽ không được lưu lại  - Những thay đổi được commit trong branch này sẽ không tồn tại trên branch khác.  - **Cú pháp :**  **git commit -m “message”**  // Nhớ thêm thông tin vào phần message nêu rõ công việc bạn đang commit lên là gì.  vd: git commit -m “sua phan dang nhap”  - Trước khi commit phải sử dụng lệnh **git add <file\_name>** để khai báo là sẽ commit file này, còn nếu không sử dụng **git add** bạn sẽ commit tất cả các file trong branch |
| **7. Git - Merge : Hợp nhất 2 branch với nhau** |
| - **Merge Branch** tức là bạn gộp 2 brand lại với nhau, thao tác này thường dùng để merge branch khác vào branch master trước khi push lên Remote Repository, hoặc merger 2 branch thành 1 để giải quyết 1 task nào đó.  **- Cú pháp :** Đứng tại brand hiện tại muốn gộp một branch khác ta dùng cú pháp sau:  **git merge <branch\_name>**  **- Xử lý xung đột :** Xung đột xảy ra khi merger 2 brand cùng xử lý trên 1 file duy nhất.  **Vd:**  - Tại **branch1** ta có file **demo.txt** có nội dung: “ Đây là branch 1”  - Tại **branch2** ta có file **demo.txt** có nội dung: “ Đây là branch 2”  Đứng tại branch1 ta hợp nhất với branch2:  **git merge branch2**  Sẽ nhận được thông báo như sau :    Và trong file demo.txt sẽ có nội dung được gộp từ file ở branch1 và branch2 như sau:  <<<<<< HEAD  Đây là branch 1  ========== 🡺 Các ký hiệu kia đùng để chỉ rõ nội dung của từng Branch, lúc này bạn có  Đây là branch2 thể xoá các ký hiệu này và sửa lại nội dung mong muốn và **COMMIT** lại  >>>>>> brand2 |
| **8. Xoá branch** |
| - Dùng sau khi đã merge branch vào branch master để tránh xung đột mã nguồn và dễ quản lý hơn.  - Với mỗi branch được tạo ra sẽ có 2 trạng thái:   * **not fully merged** : branch chưa được merge vào branch master * **fully merged** : branch đã được gộp vào branch master   - Ở trạng thái **fully merged** ta xoá branch bằng cú pháp sau :  **git branch -d <branch\_name>**  - Ở trạng thái **not** **fully merged** ta xoá branch bằng cú pháp sau :  **git branch -D <branch\_name>**  🡺 Trường hợp này nên cân nhắc vì sẽ mất đi các commit đã thực hiện trên đó |
| **9. 3 Trạng thái Committed - Staged - Modified** |
| - Mỗi tập tin trong Git được quản lý dựa trên 3 trạng thái sau:   * **Committed** : Là trạng thái dữ liệu đã được lưu an toàn trong database   // committed = sửa + add + commit   * **Staged :** Là trạng thái dữ liệu đã được đánh dấu để chuẩn bị commit   // staged = sửa + add   * **Modifed :** Là trạng thái dữ liệu đã được chỉnh sữa nhưng chưa được được đánh dấu hay commit   // modifed = sửa  - Vì 3 trạng thái đó nên có thể chia một dự án sử dụng Git thành **3 khu vực làm việc khác nhau :**   * **Working directory: Khu vực làm việc,** là một bản sao của dự án gốc ( Remote repository) * **Staging area** : **Khu vực tổ chức** , chứa thông tin về trạng thái của 1 file trong dự án * **Git directory ( repository ) : Thư mục lưu trữ siêu dữ liệu (metadata)** , là một thư mục ẩn bới window, tiếp nhận và lưu trữ các commit từ stage area   🡺 **Dùng git status để xem trạng thái của git** |
| **10. Git - Amend : Thay đổi commit cuối cùng** |
| - Trong trường hợp sau khi commit mà bạn quên một file chưa add thì Amend ( cải thiện ) sẽ giúp bạn làm được nó mà không cần phải tạo ra một commit mới.  - **Cú pháp** : Thêm dòng này sau khi đã add file bổ sung vào staged  **git commit --amend**  **Vd:**  // Thực hiện commit  // Quên file này nên lại đưa nó lên stage  // Cho nó vào commit trước |
| **11. Git - restore --stage : Loại bỏ tập tin đã đưa vào stage** |
| - Dùng để loại bỏ một tập tin lỡ đưa lên stage mà không muốn commit nó:  **Cú pháp:**  **git restore --stage <file\_name>** |
| **12. Git - Stash : Trở về trạng thái ban đầu** |
| - **git stash** : đưa dữ liệu về **trạng thái ban đầu** có tất cả file nằm trong working directory.  - **TRẠNG THÁI BAN ĐẦU :** là nội dung ban đầu của file ( so với commit cuối cùng hoặc pull từ repository)  - Lệnh git stash có tác dụng với tất cả dữ liệu đang hoạt động trong working directory với điều kiện là dữ liệu đó **từng được committed**   |  |  | | --- | --- | | **Lệnh** | **Ý Nghĩa** | | **git stash** | Đưa file về trạng thái lần **commit cuối cùng** | | **git stash list** | Liệt kê các lần đã sử dụng **stash** | | **git stash clear** | Xoá lịch sử đã stash | | **git stash apply** | Không stash nữa ( quay lại file trước khi stash ) |   . |
| **13. Git - Rebase : Một cái gì đó rất là vl** |
|  |
| **14. Remote Repository** |
| - Sử dụng github dùng làm server để cài đặt git remote trên đó  - Phiên bản repository trên remote là phiên bản chính của dự án, các thành viên sẽ thực hiện các thao tác như pull và push để lấy dự án về local repository   * **Add remote repository :**   - Để thêm một remote repo vào local repo thì sử dụng cú pháp sau:  **git remote add <short\_name> <url>**  **Trong đó:**  - **short\_name** : tên đặt cho remote repo  - **url** : đường dẫn đến repo đó , có đuôi là .git   * **Xem remote đã thêm**   **git remote hoặc git remote -v**   * **Thay đổi link của remote**   - Trường hợp khi bạn gõ nhầm link và cần đổi lại link:  **git remote set-url <short\_name> <new\_url>**   * **Xoá remote repository**   **git remote rm <short\_name>**   * **Đổi tên remote**   **git remote rename <old\_name> <new\_name>**   * **Push lên remote**   **git push --set-upstream <remote> <branch>**  **Trong đó:**  - **remote** : tên của remote repo  - **branch** : branch mà bạn muốn đẩy lên  🡺 Sau khi lệnh này thực hiện thành công thì git bash sẽ yêu cần nhập username và password cho tài khoản github.  🡺 Tại **branch hiện tại** sau mỗi lần commit bạn lại dùng lệnh **git push** để đưa commit lên remote tại branch hiện tại trên remote  🡺 Nếu muốn chuyển qua branch khác ta lại phải sử dụng lệnh  **git push --set-upstream <remote> <branch>**   * **Lưu thông tin đăng nhập với Credential store**   **git config credential.helper store**  🡺 Nhập lệnh này trước khi thực hiện thao tác push thì chỉ lần đầu tiên có yêu cầu đăng nhập , những lần sau sẽ không có   * **Pull từ remote**   **git pull** |
| **Tổng hợp các lệnh git cơ bản** |
| |  |  | | --- | --- | | **Lệnh** | **Ý Nghĩa** | | **git init** | Tạo local repo | | | **git clone <url>** | Lấy repo trên github về | | | **git pull** | Đồng bộ từ remote về local | | | **git add và git add .** | Đưa file vào Staged | | | **git commit** | Lưu file và Commit | | | **git push** |  | | | **git log** | Liệt kê các commit | | | **git log -decorate -graph -oneline** |  | | |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |