

GIT/GITHUB

Phạm Quang Nam

Ngày 9 tháng 1 năm 2023

1 Mở đầu

VERSION CONTROL SYSTEM hay SOURCE MANAGEMENT SYSTEM là một loại phần mềm giúp ta:

- Lưu trữ code, tài liệu khác như .docx, .pdf
- Phân quyền ai chỉnh sửa.
- Lưu trữ các version, cứ đề lên phiên bản mới và bạn có thể lấy lại phiên bản cũ bất cứ lúc nào(roll back), hoặc xóa lịch sử bất cứ chỗ nào cũng được.
- Clone code để thử nghiệm code riêng (rẽ nhánh branch) ở quá trình code và có thể merge khi code hoàn chỉnh.
- Giúp backup lại code nhằm dự phòng mất dữ liệu trên sever...
- Giúp lưu trữ quá trình thay đổi của nơi chứa code.

2 phần mềm làm được điều này gọi là SUBVERSION(LEGACY) Và GIT(MODERN). Trong bài này chúng ta sẽ tìm hiểu về công nghệ GIT.

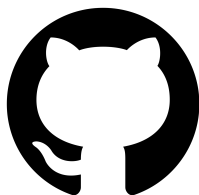
2 Giới thiệu về GIT

Tác giả của GIT là Linus Torvalds(người tạo ra hệ điều hành Linux). GIT(1) là một phần mềm mã nguồn mở(open source), chia sẻ công nghệ, kỹ thuật, code cho mọi người.



Hình 1: Công nghệ GIT.

Hiện tại các không gian lưu trữ code sử dụng công nghệ GIT như : [GITHUB](#)(Microsoft),[GITLAB](#)(Nga),[BITBUCKET](#)(ATLASSIAN/JIRA)



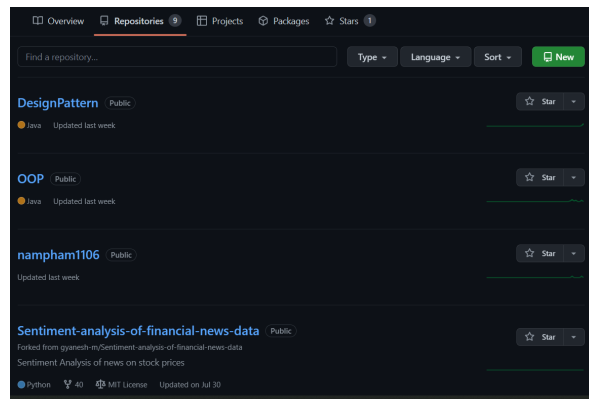
GitLab



Bitbucket

Hình 2: 3 không gian lưu trữ code phổ biến.

Các không gian này sẽ cho phép ta lưu trữ code FREE hoặc mất phí. Code được đặt trong các folder và được gọi là repository - kho chứa code. Mỗi kho (repository) tương ứng với 1 project, 1 dự án hay 1 app. Trong bài này ta sẽ sử dụng một kho lưu trữ code là GITHUB.



Hình 3: Các repository trong một trang GITHUB

3 Thao tác cơ bản với GIT

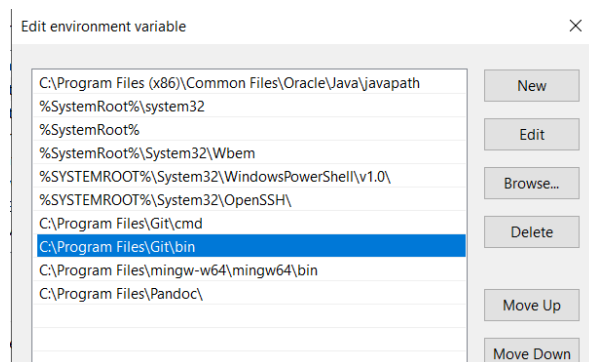
Để sử dụng GIT để upload code lên ta có rất nhiều cách:

1. Xài tool của IDE tích hợp sẵn(hoặc cài Plugin).
2. Xài GUI riêng – tool đồ họa riêng, dùng các thao tác kéo thả(GITHUB DESKTOP/ SOURCE TREE).
3. Xài các lệnh có sẵn, các lệnh này đều được xài chung cho tất cả sever dùng công nghệ GIT.

Sau đây là các bước để upload code sử dụng các lệnh.

3.1 Set up CMD Tool

để tải GIT ta truy cập vào <https://git-scm.com>. Sau đó để thuận tiện cho việc gõ lệnh ở bất kì đâu ta thêm 1 biến môi trường (4) khai báo với Windows.



Hình 4: khai báo biến môi trường với Windows.

3.2 Set up tài khoản ở local để sẵn sàng đồng bộ lên sever

Đầu tiên, chúng ta cần tạo một tài khoản [GITHUB](#). Mở terminal và gõ các câu lệnh sau:

Bạn cần thay thế username, email tương ứng của bạn vào 2 câu lệnh trên. Việc set up tài khoản chỉ cần làm 1 lần duy nhất cho đến khi cài lại Windows hoặc đổi USERNAME/PASSWORD.

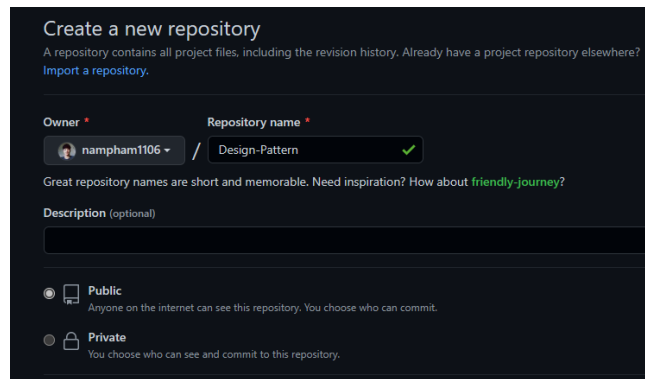
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.2364]
(c) Microsoft Corporation. All rights reserved.

C:\Users\namph>git config --global user.name "namnam1106"
C:\Users\namph>git config --global user.email "nampham11062002@gmail.com"
```

Hình 5: Setup account ở local

3.3 Tạo kho(repository) để chứa code

Bước 1. Tạo 1 kho trống trên [GITHUB](#), tên kho sẽ phải trùng với tên Project ở local(trùng 100% hoa thường và khoảng trắng). VD: tạo một kho tên DesignPattern



Hình 6: tạo kho chứa code

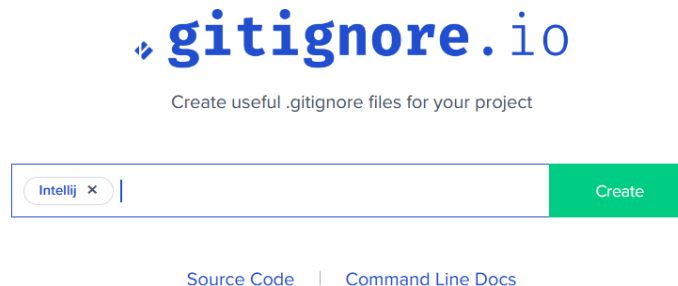
Lưu ý: Kho có thể có nhiều tùy chọn : công khai, khép kín.

Bước 2. Tạo Project chứa code, tạo File .gitignore

Do khi code ta thường sử dụng các IDE sẽ sinh ra các file biên dịch, các folder. Các file này chúng ta không cần thiết phải lưu trữ, đưa lên sever mà chỉ đưa các thư mục chứa code. Do vậy, ta phải thông báo cho GIT những file, folder cần đưa lên và những file cần bỏ lại.

Để giải quyết vấn đề đó ta sử dụng 1 tool có tên là [GITIGNORE](#). GITIGNORE sẽ cung cấp cho ta các đoạn mã tương ứng với IDE mà ta sử dụng nhằm thông báo với GIT những file, folder không cần thiết để đưa lên.

Đầu tiên tạo ta tạo file .gitignore vào Project. Sau đó ta truy cập vào [gitignore.io](#), gõ IDE tương ứng:

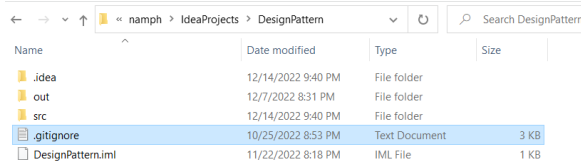


Hình 7: lấy mã trên trang gitignore

Sau đó create, và copy đoạn mã vào file .gitignore mà bạn đã tạo

Lưu ý: Mỗi 1 project ta cần tạo 1 file .gitignore, ta cũng có thể tái sử dụng lại file này nếu project đó sài cùng 1 IDE.

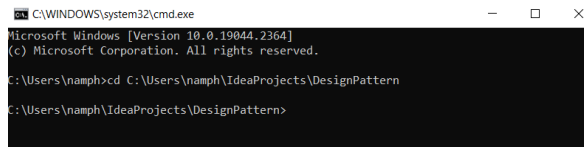
<https://www.overleaf.com/project/6399b123ac40a23e529e619d>



Hình 8: Tạo file .gitignore trong project

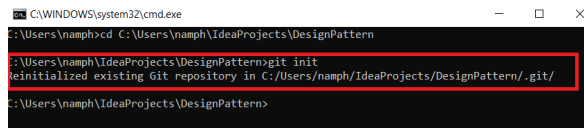
3.4 Đẩy code lên sever

Bước 1. di chuyển đến thư mục chứa project



Hình 9: Mở project

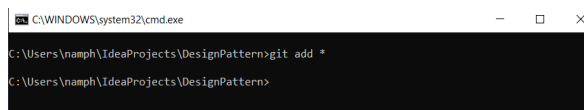
Bước 2. Khởi tạo kho local chuẩn bị đồng bộ, ta gõ lệnh dưới đây 10, lệnh này làm duy nhất một lần cho 1 project.



Hình 10: tạo kho local

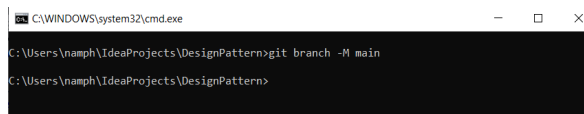
Lưu ý : GIT tự sinh ra cho mình 1 thư mục .GIT trong project. Tuyệt đối không được xóa, sửa file .git này. Nó theo dõi sự thay đổi của code, do đó cho phép mình rollback, history, xử lý xung đột.

Bước 3. gõ lệnh sau, để đưa code lên sever(loại trừ trong file .gitignore)



Hình 11: git add *

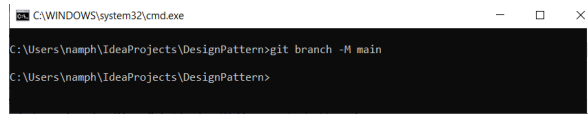
Bước 4. Đưa ra thông báo về việc upload, bằng lệnh sau:



Hình 12: commit code

Việc commit giúp chúng ta dễ quản lý sự thay đổi của code hơn. Do đó khi ta phát hành ứng dụng, ta dễ dàng làm file changelog, ghi lại nhật ký thay đổi của app, để user biết được sự thay đổi so với version trước.

Bước 5. Đưa vào kho ảo chính



```
C:\WINDOWS\system32\cmd.exe
C:\Users\namph\IdeaProjects\DesignPattern>git branch -M main
C:\Users\namph\IdeaProjects\DesignPattern>
```

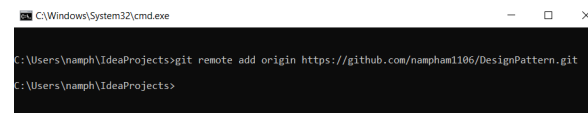
Hình 13: git branch -M main

Git ngầm định thư mục code ở local và ở server có cơ chế ảo "kho ảo tên main kho chính, nhánh chính, branch main. GIT cho phép mình thử nghiệm code từ kho chính bằng cách tạo những kho ảo khác trong cùng thư mục code. Làm được điều này nhờ tệp .git được tạo sau lệnh GIT INIT.

Note: Đọc thêm về nhánh trong GIT/COPY/PASTER PROJECT để thử nghiệm.

Bước 6. Đẩy code lên lần đầu.

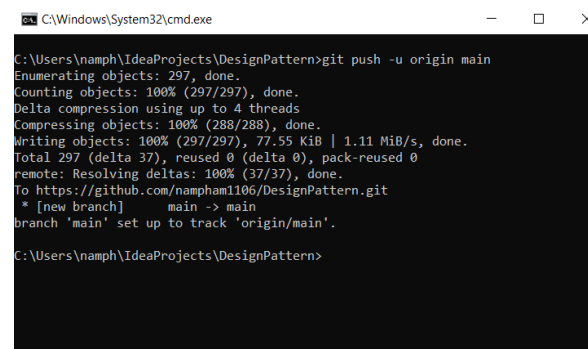
Đầu tiên ta khai báo đường dẫn lên sever(gọi tắt là origin).



```
C:\Windows\System32\cmd.exe
C:\Users\namph\IdeaProjects>git remote add origin https://github.com/nampham1106/DesignPattern.git
C:\Users\namph\IdeaProjects>
```

Hình 14: git remote add origin

khi upload lần đầu ta cần nhập user/password, sau đó upload bằng lệnh sau:



```
C:\Windows\System32\cmd.exe
C:\Users\namph\IdeaProjects\DesignPattern>git push -u origin main
Enumerating objects: 297, done.
Counting objects: 100% (297/297), done.
Delta compression using up to 4 threads
Compressing objects: 100% (288/288), done.
Writing objects: 100% (297/297), 77.55 KiB | 1.11 MiB/s, done.
Total 297 (delta 37), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (37/37), done.
To https://github.com/nampham1106/DesignPattern.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
C:\Users\namph\IdeaProjects\DesignPattern>
```

Hình 15: git push -u origin main

3.5 Cập nhập code thường xuyên

Sau khi upload code lần đầu thành công những lần tiếp theo chúng ta chỉ cần lặp lại những câu lệnh sau:

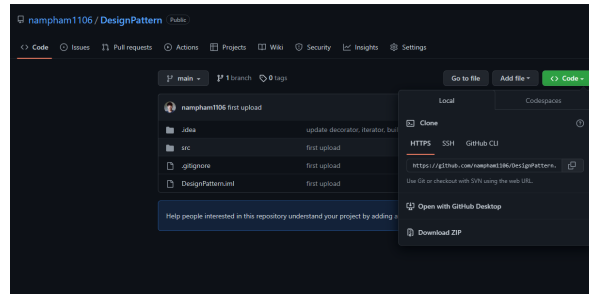
- git add *
- git commit -m "nhập thông báo"
- git push -u origin main

4 GIT trong làm việc nhóm

4.1 Clone project

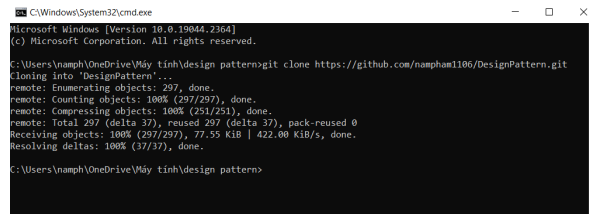
Các thành viên trong nhóm có thể sao chép kho chứa code và đóng góp vào nó. để sao chép ta cần dùng những bước sau:

Bước 1. mở kho cần sao chép và copy đường dẫn



Hình 16: Sao chép địa chỉ kho chứa code

Bước 2. Mở cmd di chuyển tới folder cần sao chép kho về. Gõ lệnh sau để clone code về:

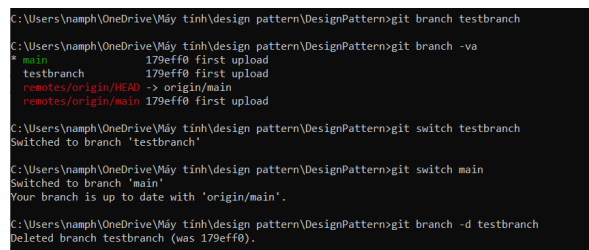


Hình 17: Clone code về local

4.2 Phân nhánh

Trong một repository có thể có nhiều nhánh. Trong đó, nhánh chính(main) chỉ nên giữ làm việc kiểm tra, hoàn thành code. Mỗi tính năng mới hoặc sửa lỗi hoặc câu chuyện người dùng nên được phát triển trong chi nhánh riêng của nó, với nhiều cam kết cần thiết trong chi nhánh đó để có được nó ngay trước khi hợp nhất nó vào nhánh chính. Với các nhánh riêng cho từng tính năng, sự phát triển có thể được thực hiện song song trong khi giảm thiểu các xung đột chỉnh sửa trong các tệp.

Ta có các lệnh sau để tương tác với các branch:



Hình 18: Thêm, liệt kê, switch, xóa 1 branch

Trong đó:

- git branch newbranch -> dùng để tạo 1 branch mới
- git branch -va -> liệt kê các branch đang có
- git switch branchname -> dịch chuyển đến branch cần đến
- git branch -d branchname -> xóa 1 branch

Lưu ý: việc xóa 1 branch chỉ nên thực hiện khi đã gộp vào branch chính

4.3 Điều khiển nhánh

Nếu bạn tạo chi nhánh để chia sẻ với người khác, bạn cần cho Git biết nơi gửi và đặt tên cho nó. Để đẩy branch mới của bạn lên remote repository và đăng ký remote đó Là nơi để đẩy nhánh này trong tương lai, bạn có thể chạy chính xác lệnh Nó cho bạn thấy:

```
C:\Users\namph\IdeaProjects\DesignPattern>git push --set-upstream origin teamone
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'teamone' on GitHub by visiting:
remote:   https://github.com/nampham186/DesignPattern/pull/new/teamone
remote:
To https://github.com/nampham186/DesignPattern.git
 * [new branch]   teamone -> teamone
branch 'teamone' set up to track 'origin/teamone'.
```

Hình 19: remote branch

Bạn có thể chuyển sang một nhánh từ xa mà người khác đã tạo (và do đó chưa tồn tại cục bộ cho bạn) bằng cách kéo kho lưu trữ và sử dụng lệnh. Ví dụ: nếu ai đó tạo một nhánh được đặt tên và đẩy nhánh đó vào điều khiển từ xa, bạn có thể chạy:

```
C:\Users\namph\IdeaProjects\DesignPattern>git pull
Already up to date.

C:\Users\namph\IdeaProjects\DesignPattern>git switch teamtwo
Switched to a new branch 'teamtwo'
branch 'teamtwo' set up to track 'origin/teamtwo'.

C:\Users\namph\IdeaProjects\DesignPattern>
```

Hình 20: switch branch

4.4 Gộp nhánh

Gộp 1 nhánh vào nhánh hiện tại sử dụng câu lệnh dưới đây:

```
C:\Users\namph\IdeaProjects\DesignPattern>git merge teamone
Already up to date.

C:\Users\namph\IdeaProjects\DesignPattern>git branch -va
* main          179eff0 first upload
  teamone       179eff0 first upload
  teamtwo       179eff0 first upload
  remotes/origin/main 179eff0 first upload
  remotes/origin/teamone 179eff0 first upload
  remotes/origin/teamtwo 179eff0 first upload

C:\Users\namph\IdeaProjects\DesignPattern>git branch -d teamone
Deleted branch teamone (was 179eff0).

C:\Users\namph\IdeaProjects\DesignPattern>git branch -va
* main          179eff0 first upload
  teamtwo       179eff0 first upload
  remotes/origin/main 179eff0 first upload
  remotes/origin/teamone 179eff0 first upload
  remotes/origin/teamtwo 179eff0 first upload
```

Hình 21: Merge branch

4.5 Xử lý xung đột

Để xử lý xung đột trong dự án của bạn sử dụng Git, ta có thể thực hiện các bước sau:

1. Xem xung đột nào đang xảy ra: Ta có thể sử dụng lệnh **git diff** để xem những thay đổi mà mình và người khác đã thực hiện trên cùng một tập tin, giúp bạn hiểu rõ hơn về những xung đột nào đang xảy ra.
2. Chỉnh sửa tập tin để khắc phục xung đột: sau khi ta đã xem xung đột đang xảy ra, ta có thể chỉnh sửa tập tin để khắc phục xung đột. Hãy chắc chắn rằng ta đã thực hiện đúng các thay đổi cần thiết và lưu lại tập tin.
3. Thêm tập tin đã chỉnh sửa vào Git: sau khi bạn đã chỉnh sửa xong tập tin, bạn có thể sử dụng lệnh **git add** để thêm tập tin đã chỉnh sửa vào Git.

4. Commit các thay đổi: sau khi bạn đã thêm tập tin đã chỉnh sửa vào Git, bạn có thể commit các thay đổi bằng cách sử dụng lệnh **git commit**. Nhớ để ghi lại nội dung commit cụ thể và mô tả rõ ràng các thay đổi trước khi đẩy code trở lại.

[1]

Tài liệu

[1] Scott Chacon and Ben Straub. Pro git. 2014.