Git là gì?

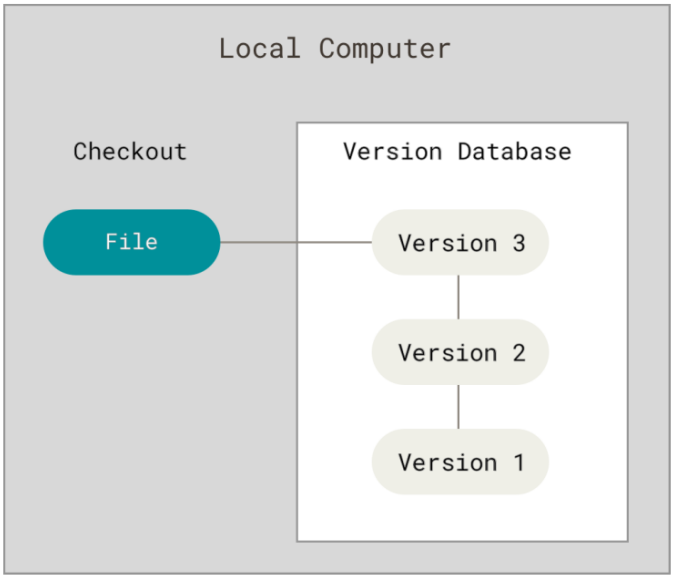
Để hiểu được Git thì trước hết phải hiểu được các hệ thống CVS là gì.

VS - version control là một hệ thống ghi lại các thay đổi của 1 file hay một chuỗi file theo thời gian để bạn có thể gọi lại một phiên bản cụ thể nào đó về sau

1. Local Version Control Systems

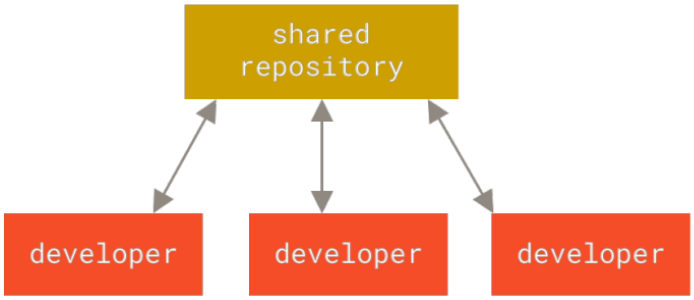
Phương pháp quản lí viên bản phổ biến của nhiều người là copy file sang thư mục khác. Phương pháp này dễ và phổ biến nhưng cũng cực kì dễ bị lỗi. Việc quên mình lưu vào thư mục nào hoặc ghi đè thư mục xảy ra như cơm bữa.

Để đối phó với vấn đề này thì lập trình viên đã phát triển 1 hệ thống VCS mà có cơ sở dữ liệu đơn giản lưu trữ tất cả thay đổi của file



2. Centralized Version Control Systems

Vấn đề lớn tiếp theo mà nhiều người mắc phải đó là họ cần phối hợp với dev ở hệ thống khác. Để giải quyết vấn đề này, CVCS ra đời. Hệ thống này có một sever chính chứa tất cả các phiên bản của file, và các khách hàng check out file từ tại địa điểm trung tâm đó.

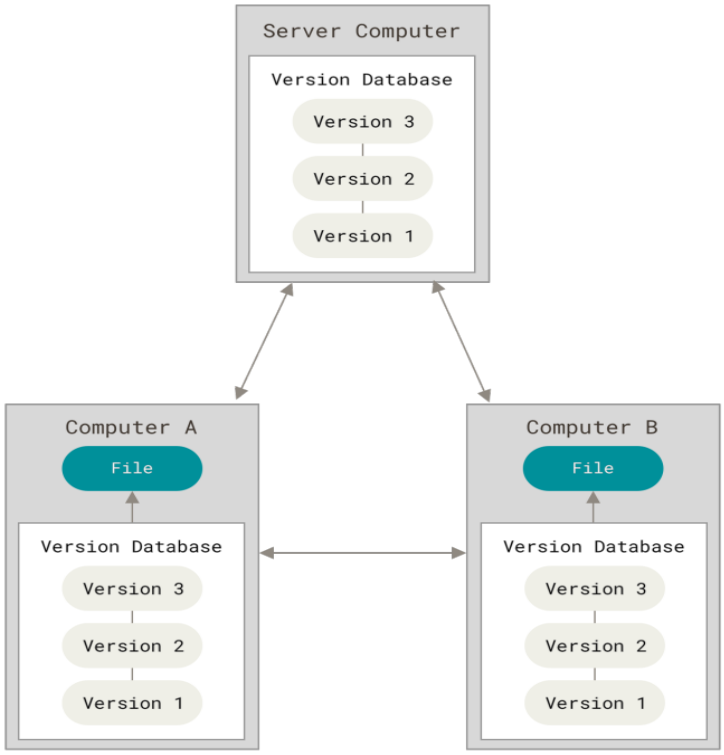


Thiết kế này mang lại nhiều lợi ích hơn so với **Local Version Control Systems.** Ít nhất là người này vẫn biết được người kia đang làm gì. Admin có thể điều khiển ai có thể làm việc gì.

Thế nhưng hệ thống này có một vài điểm bất lợi nghiêm trọng. Rõ ràng nhất là nếu điểm trung tâm sập thì không ai có thể phối hợp được với ai nữa. Nếu ổ cứng ở điểm trung tâm hỏng mà không có backup thì coi như là mất trắng mọi thứ.

3. Distributed Version Control Systems

Đây là lúc mà DVCS toả sáng. Trong hệ thống này, khách không chỉ check out snapshot mới nhất của file mà có thể nhân bản kho lưu trữ. Vì vậy nếu sever chết thì vẫn còn backup phía máy khách. Mỗi nhân bản đều là bản backup đầy đủ của toàn bộ dữ liệu.



Vậy có thể hiểu đơn giản Git là một dạng của DVCS. Thế nhưng điểm khiến Git trở nên khác biệt là gì. Đó là cách Git nghĩ về dữ liệu. Cơ bản mà nói, hầu hết các hệ thống khác lưu trữ thông tin dưới dạng danh sách các thay đổi của file. Các hệ thộng khác như CVS, Subversion, Perforce xem thông tin mà nó chứa là các bộ file và thay đổi của nó qua thời gian.

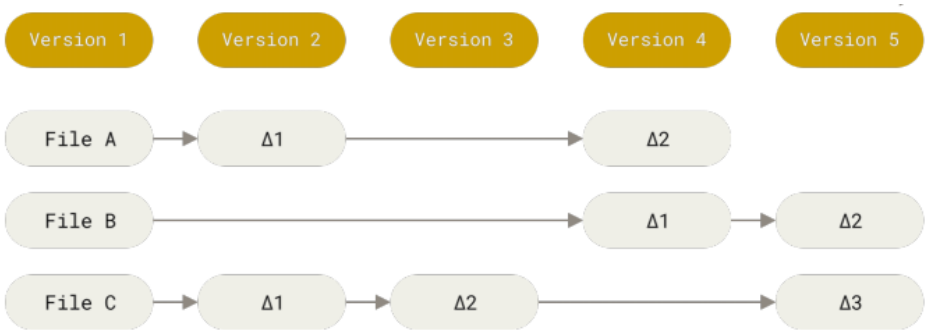


Figure 4. Storing data as changes to a base version of each file

Git lại xem dữ liệu của nó như các chuỗi ‘snapshot’ của một hệ thống tệp thu nhỏ. Mỗi khi bạn ‘commit’, Git chụp ảnh lại toàn bộ file của bạn tại thời điểm đấy và lưu một tham chiếu đến ‘snapshot’ đấy. Nếu file chưa được thay đổi, Git không lưu file lại một lần nữa mà chỉ là một đường dẫn đến file y hệt trước đó mà đã được lưu.

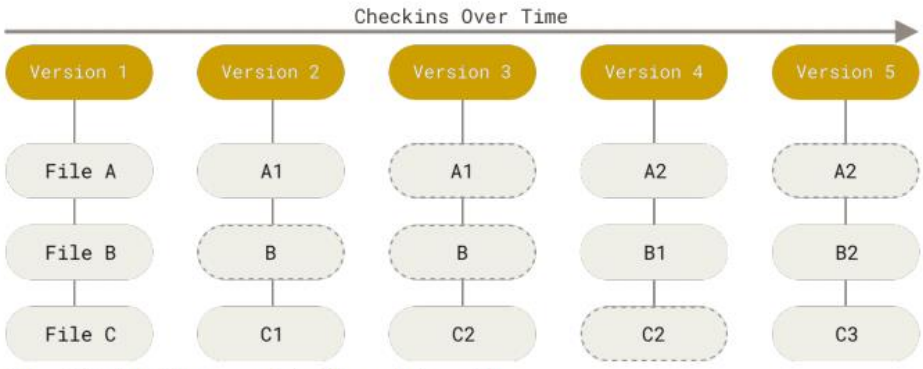


Figure 5. Storing data as snapshots of the project over time

Gần như mọi hoạt động của git chỉ cần các file cục bộ để hoạt động, bạn không cần bất cứ thông tin nào từ các máy tính khác. Bởi vì toàn bộ lịch sử của dự án đều nằm trên máy bạn, gần như mọi hoạt động đều xảy ra gần như đồng thời.

Tất cả mọi thứ trong Git đều được checksum trước khi lưu trữ sau đó được tham chiếu bởi checksum đó. Điều này nghĩa là không thể thay đổi nội dung file mà Git không biết.

Git sử dụng SHA-1 hash cho checksum của nó. Checksum này xuất hiện mọi nơi khi sử dụng Git. Git lưu trữ mọi thứ không phải bằng tên file mà là giá trị hash của file.

4. Ba trạng thái

Git có 3 trạng thái chính đó là: modified, staged, and committed

+ Modified nghĩa là file đã bị thay đổi nhưng vẫn chưa được commit vào cơ sở dữ liệu

+ Staged nghĩa là file thay đổi đã được đánh dấu để được vào lần commit snapshot tiếp theo

+ Committed nghĩa là dữ liệu đã được lưu an toàn trong cơ sở dữ liệu cục bộ

The working tree là 1 lần checkout của 1 phiên bản project. Những file này được lấy từ cơ sở dữ liệu trong đường dẫn Git xuống ổ đĩa để sử dụng và sửa đổi

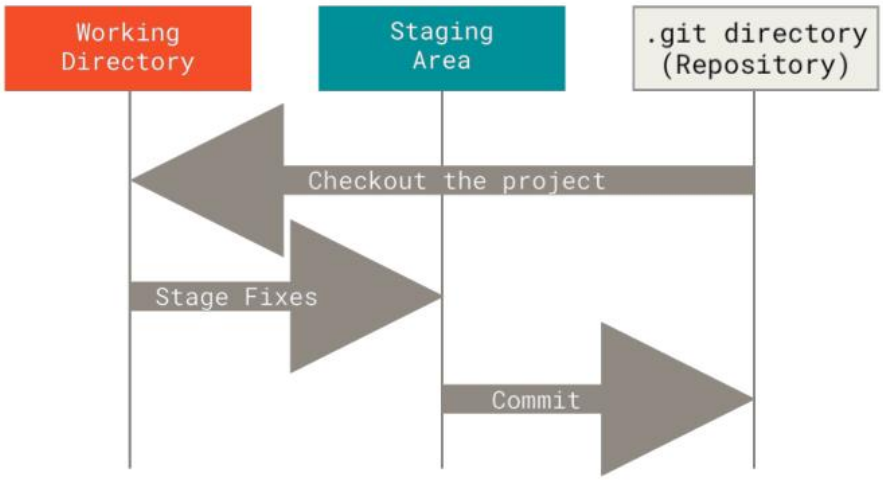


Figure 6. Working tree, staging area, and Git directory

The staging area là file nằm trong đường dẫn Git mà chứa thông tin về những gì sẽ được đưa vào lần commit tiếp theo.

The Git directory là nơi Git chứa metadata của cơ sở dữ liệu của đối tượng cho dự án. Đây là phần quan trọng nhất của Git, và nó là phần được copy khi mà nhân bản kho lưu trữ từ máy tính khác.

Chu trình làm việc cơ bản của Git như thế này:

1. Bạn thay đổi file trong working tree

2. Bạn stage những thay đổi mà bạn muốn cho lần commit tiếp theo, nghĩa là chỉ thêm những thay đổi đấy vào staging area

3. Khi bạn commit thì những file nằm trong staging area sẽ được lưu vĩnh viễn trong đường dẫn Git

Git cơ bản

Khi khởi động Git lần đầu thì cần phải thiết lập một số thông số cho môi trường làm việc

- Thiết lập nhận dạng người dùng

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

- Thiết lập editor

```
$ git config --global core.editor emacs
```

- Thiết lập tên của nhánh mặc định

```
$ git config --global init.defaultBranch main
```

- Khởi tạo kho dữ liệu ở một đường dẫn nhất định

Chuyển địa chỉ sang đường dẫn nào đó

```
$ cd C:/Users/user/my_project
```

Khởi tạo Git trong thư mục đường dẫn đấy

```
$ git init
```

- Nhân bản một kho dữ liệu đã tồn tại

```
$ git clone https://github.com/libgit2/libgit2
```

- Hiển thị trạng thái file

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

Trạng thái ở đây nghĩa là đang ở nhánh gốc/master, các file bạn đang theo dõi không bị sửa đổi và không có file nào chưa được theo dõi.

```
$ echo 'My Project' > README
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README

nothing added to commit but untracked files present (use "git add" to track)
```

Nếu thêm 1 file README vào thì file đó sẽ được hiển thị là chưa được theo dõi. Chưa được theo dõi nghĩa là file của bạn chưa từng có snapshot.

- Theo dõi file mới

```
$ git add README
```

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)

    new file:   README
```

Khi chạy git status lần nữa thì có thể thấy file đã được stage

Git add còn có thể được sử dụng để stage file đã bị sửa đổi. Thử thay đổi file CONTRIBUTING.md

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

```
$ git add CONTRIBUTING.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README
    modified:   CONTRIBUTING.md
```

Sau khi chạy git add thì file đã được stage.

- Hoàn tác

Khi bạn commit nhưng nhận ra mình add thiếu file hoặc làm hỏng file commit

```
$ git commit --amend
```

```
$ git commit -m 'Initial commit'
$ git add forgotten_file
$ git commit --amend
```

- Unstage file

```
$ git reset HEAD CONTRIBUTING.md
Unstaged changes after reset:
M   CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    README.md -> README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

- Hoàn tác sửa đổi file

```
$ git restore --staged CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:    README.md -> README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   CONTRIBUTING.md
```

- Hiển thị kho lưu trữ từ xa

```
$ git remote -v
origin  https://github.com/schacon/ticgit (fetch)
origin  https://github.com/schacon/ticgit (push)
```

- Thêm kho lưu trữ từ xa

```
$ git remote
origin
$ git remote add pb https://github.com/paulboone/ticgit
$ git remote -v
origin  https://github.com/schacon/ticgit (fetch)
origin  https://github.com/schacon/ticgit (push)
pb  https://github.com/paulboone/ticgit (fetch)
pb  https://github.com/paulboone/ticgit (push)
```

- Lấy dữ liệu từ kho

```
$ git fetch <remote>
```

- Đẩy dữ liệu lên kho

```
$ git push origin master
```

- Thêm tag

Tag được chú thích chứa đầy đủ thông tin về object, checksum, tên của người tag, email, thời điểm, tin nhắn và có thể được ký và xác minh

```
$ git tag -a v1.4 -m "my version 1.4"
$ git tag
v0.1
v1.3
v1.4
```

- Chia sẻ tag

```
$ git push origin v1.5
Counting objects: 14, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (14/14), 2.05 KiB | 0 bytes/s, done.
Total 14 (delta 3), reused 0 (delta 0)
To git@github.com:schacon/simplegit.git
 * [new tag]         v1.5 -> v1.5
```

Git Phân Nhánh

- Nhánh trong Git

Nhánh trong Git cơ bản là một con trỏ tới các commit. Tên nhánh mặc định trong Git là master. Khi bạn bắt đầu commit, bạn được cấp 1 nhánh master trỏ tới lần commit mới nhất. Và mỗi khi bạn commit, con trỏ nhánh master tự động tiến lên trước.

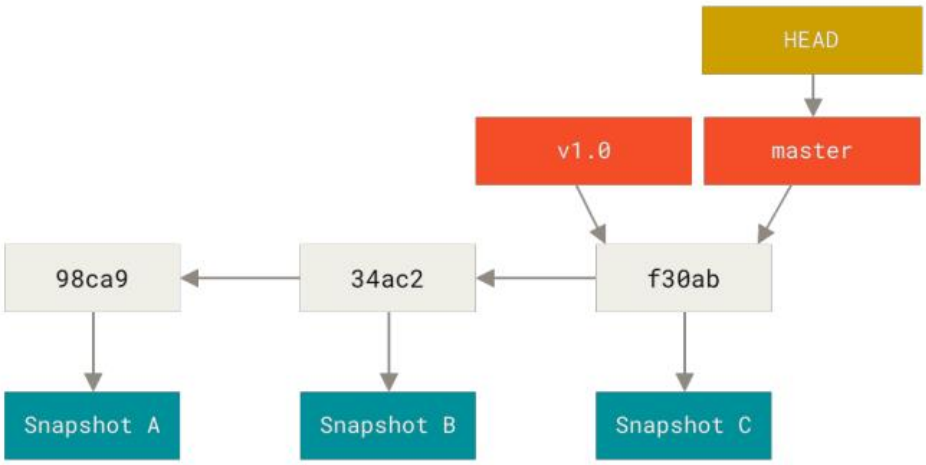


Figure 11. A branch and its commit history

Tạo nhánh mới

Tạo nhánh mới trong Git nghĩa là tạo một con trỏ mới để di chuyển xung quanh. Tạo 1 nhánh testing mới bằng lệnh git branch, lệnh này chỉ tạo nhánh mới nhưng không tự động chuyển đến nó

```
$ git branch testing
```

Git sử dụng con trỏ Head để trỏ đến địa chỉ mà bạn đang ở. Trong trường hợp này là nhánh master. Git sử dụng con trỏ này để biết được vị trí hiện tại của bạn

Figure 13. HEAD pointing to a branch

Nhảy sang nhánh khác

```
$ git checkout testing
```

Head sẽ nhảy sang nhánh testing

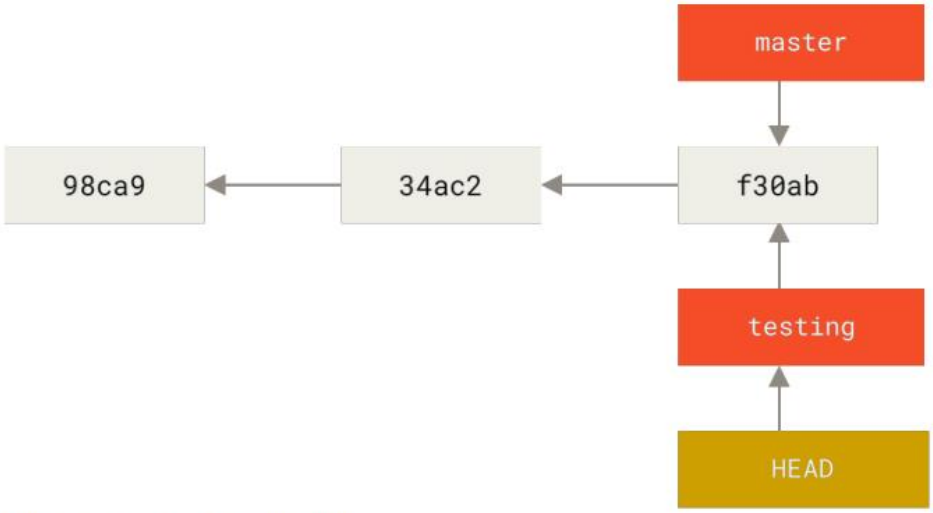


Figure 14. HEAD points to the current branch

```
$ vim test.rb
$ git commit -a -m 'made a change'
```

Khi commit, HEAD và testing sẽ di chuyển lên trước

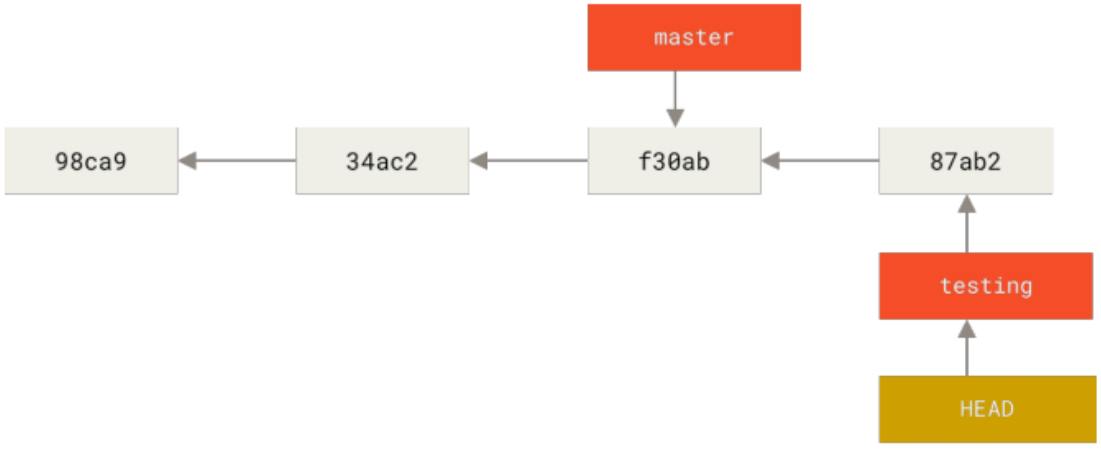
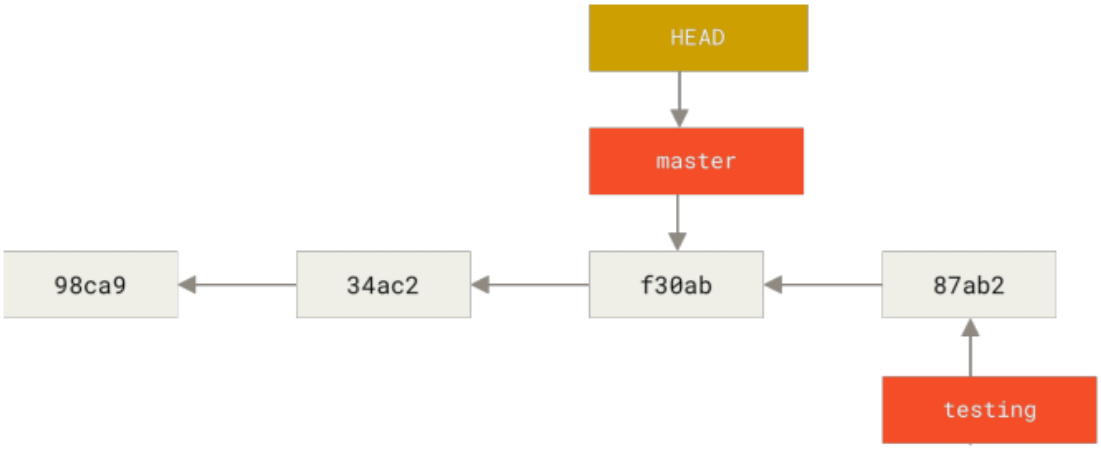


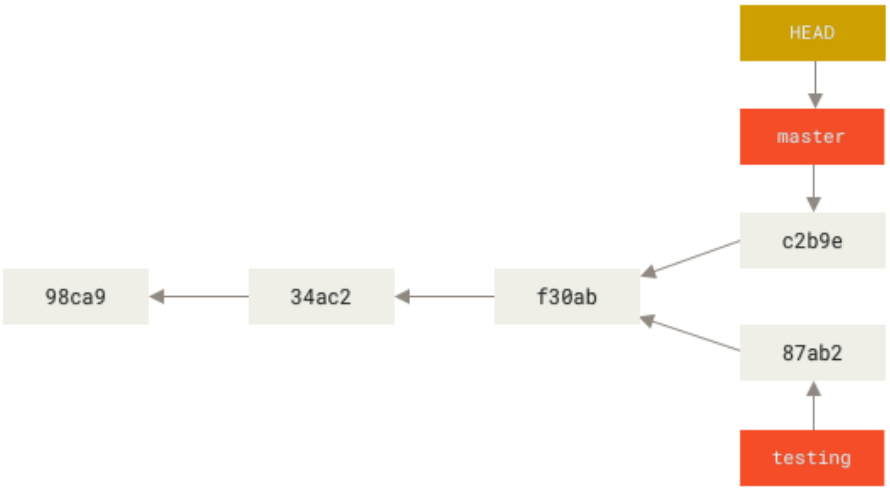
Figure 15. The HEAD branch moves forward when a commit is made

Thử chuyển sang nhánh master

```
$ git checkout master
```

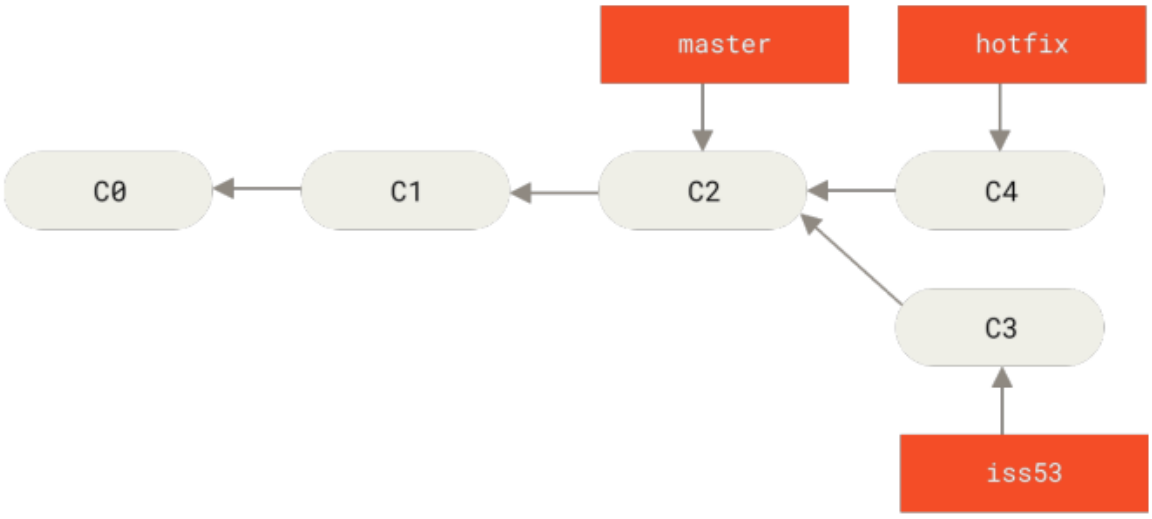


Lệnh này chuyển con trỏ HEAD về nhánh master và hoàn tác file trong thư mục làm việc trở về thời điểm snapshot mà con trỏ master trỏ đến. Bất cứ thay đổi nào từ thời điểm này sẽ phân nhánh khỏi phiên bản cũ.



- Gộp cơ bản

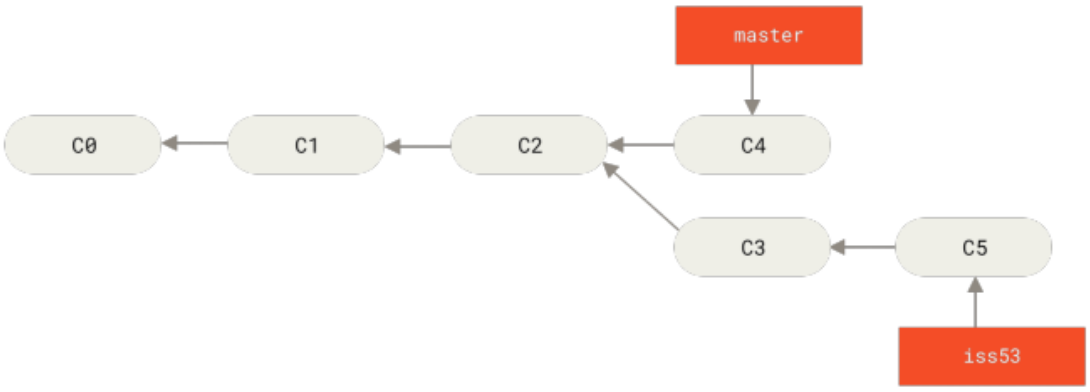
Giả sử có các nhánh master, hotfix, iss53



Gọi lệnh gộp nhánh hotfix vào master

```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```

Cụm Fast-forward trong khi gộp, bởi vì nhánh hotfix C4 nằm ngay trước nhánh master C2 vậy nên Git sẽ chỉ đơn giản chuyển con trỏ master lên trước.



Gộp nhánh iss53 vào master

```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
index.html |    1 +
1 file changed, 1 insertion(+)
```

Lúc này sẽ gộp bằng phương pháp recursive, bởi vì lịch sử phát triển đã bị phân nhánh, nên lúc này git sẽ tạo commit mới và gộp các nhánh cần gộp vào.

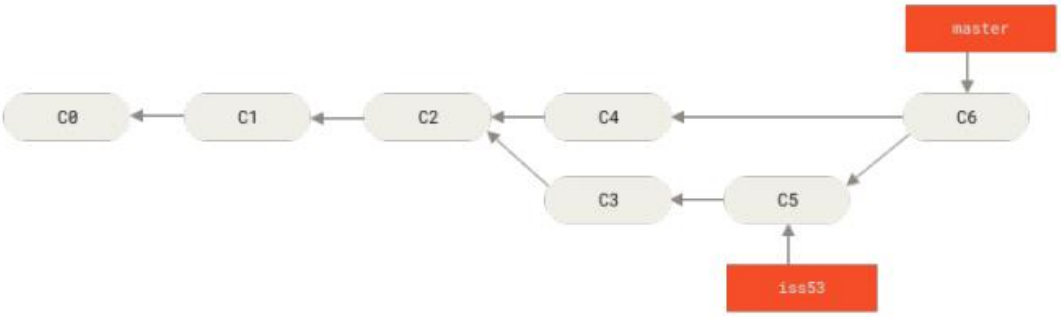


Figure 25. A merge commit

- Quản lí nhánh

```
$ git branch -v
  iss53   93b412c Fix javascript issue
* master  7a98805 Merge branch 'iss53'
  testing 782fd34 Add scott to the author list in the readme
```

Dấu * thể hiện con trỏ HEAD đang ở nhánh nào

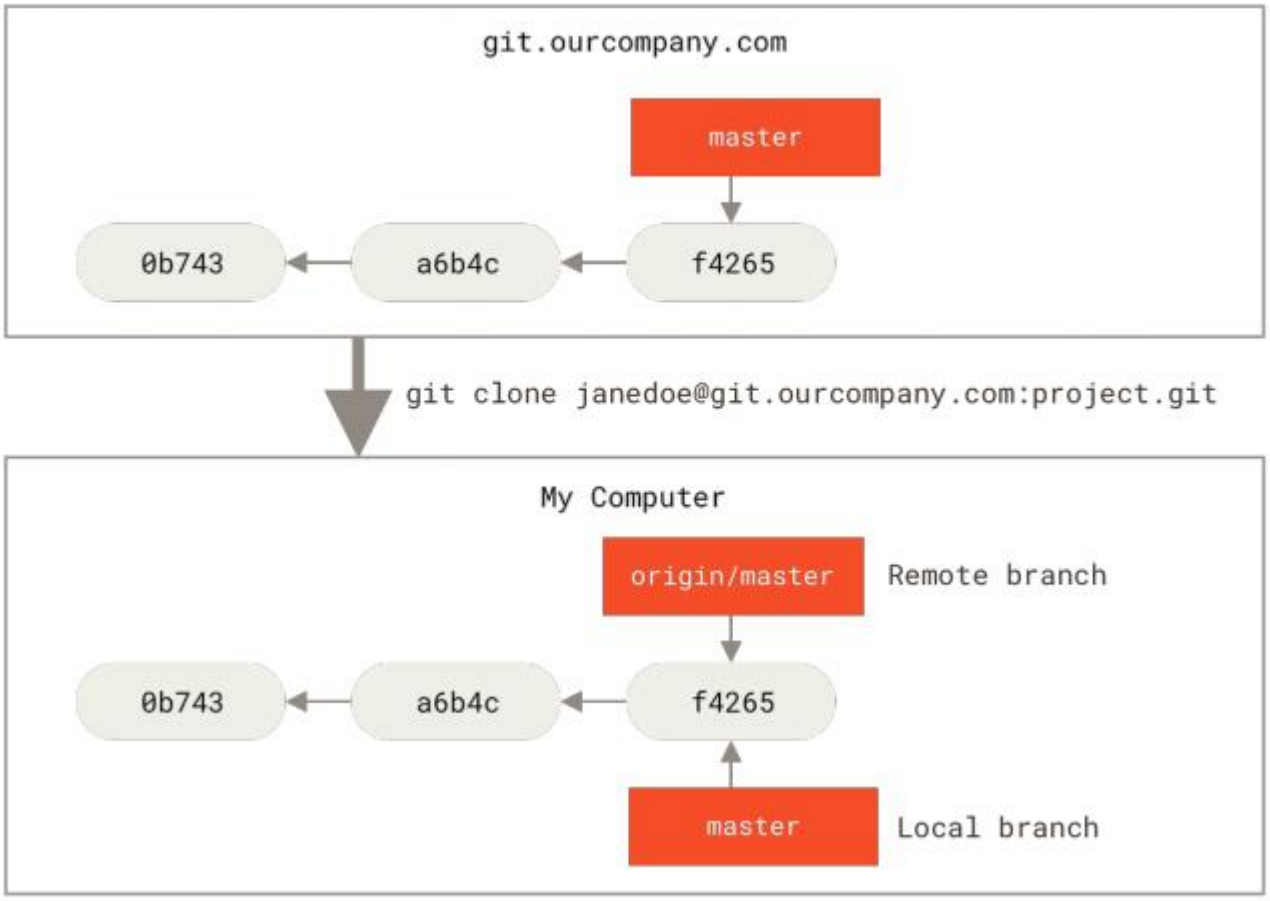
```
$ git branch --merged
  iss53
* master
```

Thêm –merged để biết nhánh nào đang gộp vào nhánh nào

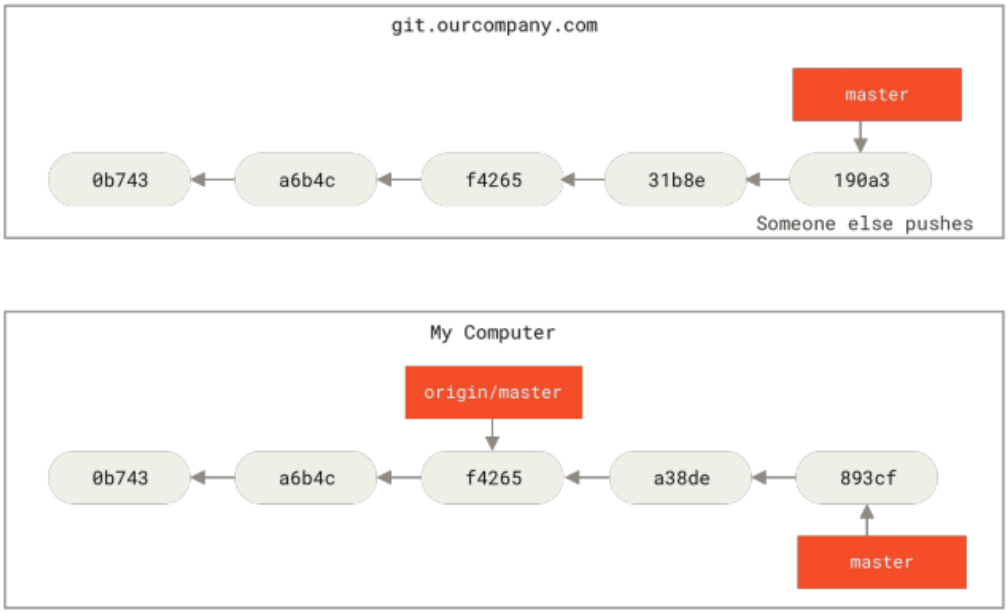
- Nhánh Remote

Nhánh Remote- tracking là tham chiếu tới trạng thái của nhánh Remote. Bạn không thể di chuyển nhánh này, Git sẽ di chuyển nó khi bạn thực hiện kết nối để có thể thể hiện được chính xác trạng thái của kho lưu trữ từ xa.

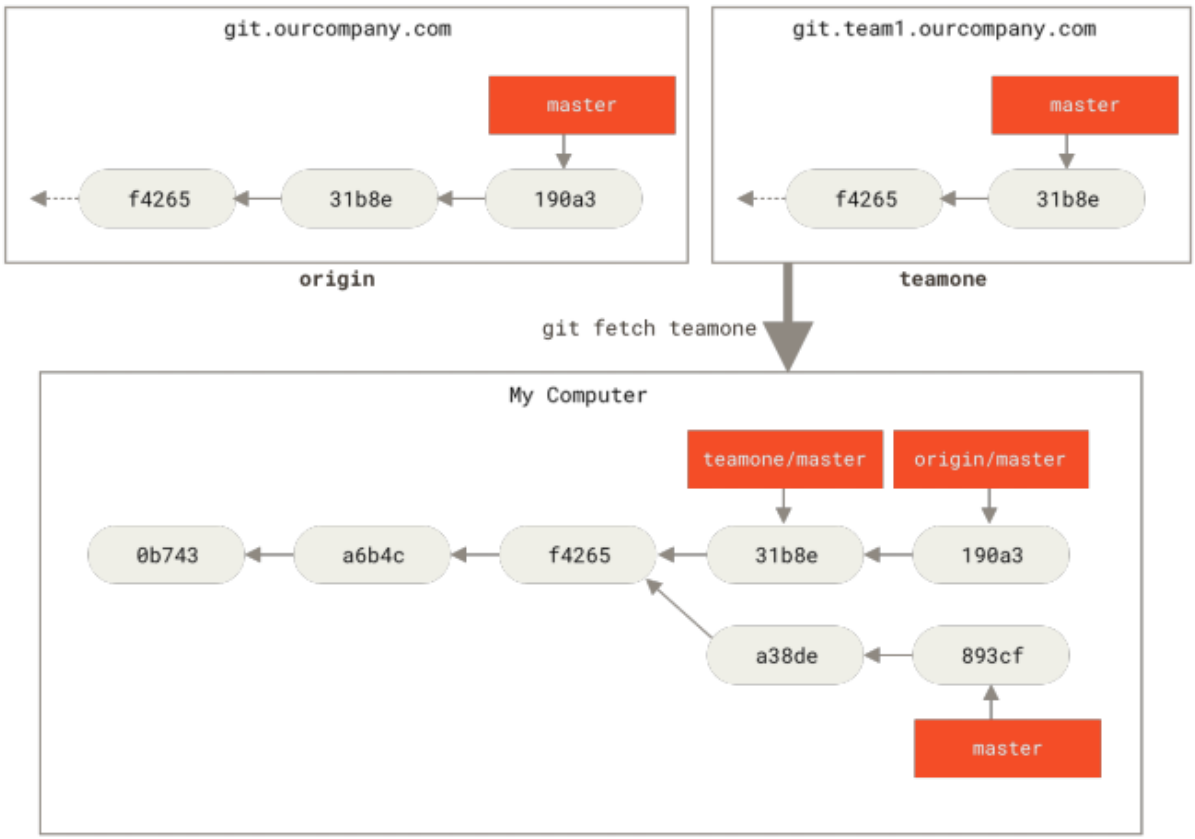
Giả sử bạn có một Git sever ở git.ourcompany.com, nếu bạn gọi git clone thì Git sẽ tự động đặt tên nhánh là Origin, kéo toàn bộ dữ liệu xuống và tạo con trỏ master trỏ đến, và đặt tên nó là origin/master



Nếu bạn làm việc với nhánh master và trong lúc đó, ai đó đẩy dữ liệu lên remote thì lịch sử dữ liệu của bạn sẽ khác với remote

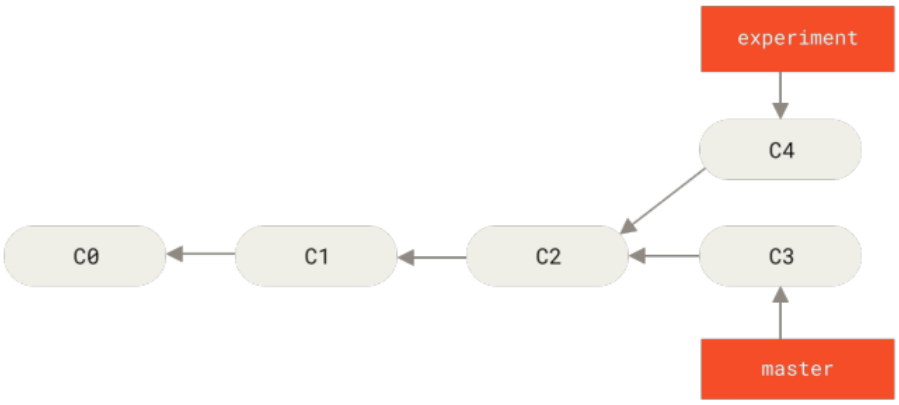


Giả sử có một sever Git khác được dùng để phát triển bởi một team khác, tạm gọi là git.team1.ourcompany.com. Gọi lệnh git remote add để thêm remote mới. Gọi lệnh git fetch để update dữ liệu mà bạn chưa có. Bởi vì sever đấy chỉ có một phần dữ liệu của origin sever nên Git sẽ chỉ tạo nhánh remote- tracking mới để trỏ tới điểm commit master của teamone

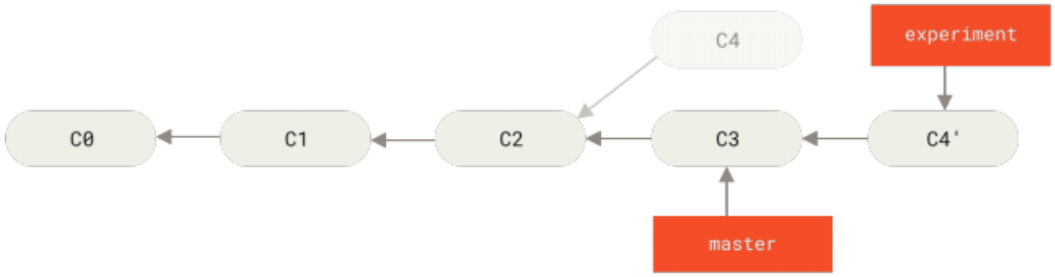


- Rebase

Rebase nghĩa là tạo một bản patch của những thay đổi trên C4 và áp dụng nó phía trên C3. Có thể hiểu là lấy những thay đổi đã được thực hiện trên 1 nhánh và áp dụng nó trên 1 nhánh khác. Mục đích của rebase là để có lịch sử rõ ràng hơn, nếu xem xét log của nhánh được rebase thì giống như mọi thứ xảy ra theo chuỗi chứ trong thực tế thì nó xảy ra song song

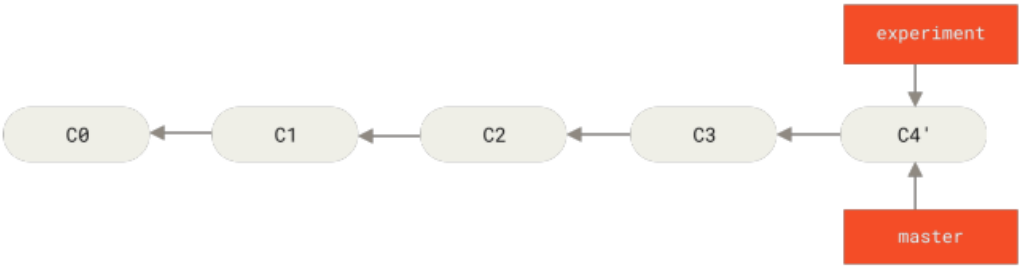



```
$ git checkout experiment
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: added staged command
```



Sau đó quay lại nhánh master để gộp

```
$ git checkout master
$ git merge experiment
```



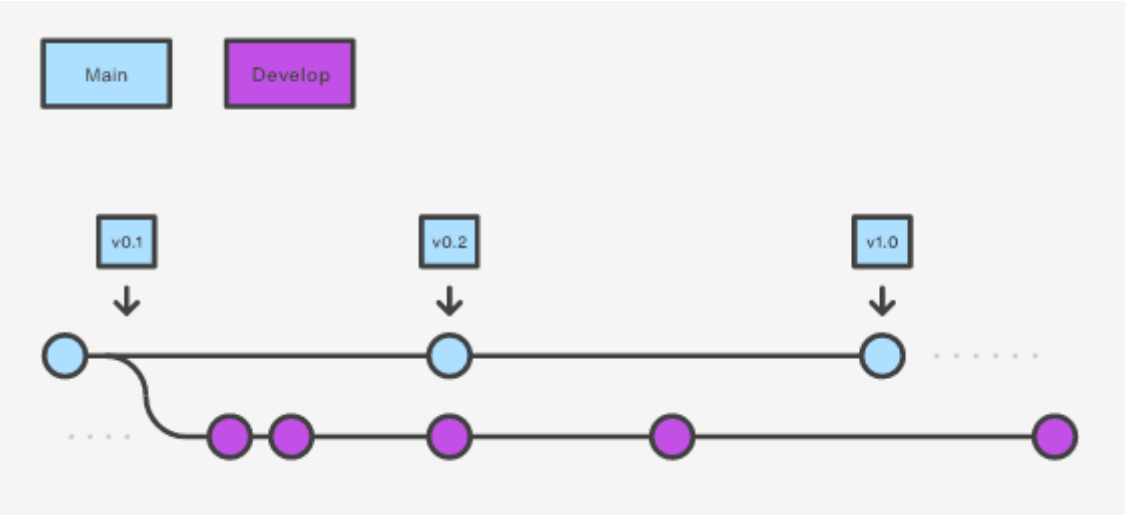
Gitflow

Gitflow là một mô hình phân nhánh gồm các nhánh tính năng và nhiều nhánh chính, Gitflow có nhiều nhánh kéo dài với commit lớn. Khi sử dụng mô hình này, dev sẽ tạo nhánh tính năng và trì hoãn việc gộp nó vào cành chính cho đến khi tính năng đấy hoàn thành. Những nhánh tính năng này cần sự phối hợp tốt mới có thể gộp và có thể đi chệch hướng khỏi cành chính.

- ## Cách hoạt động

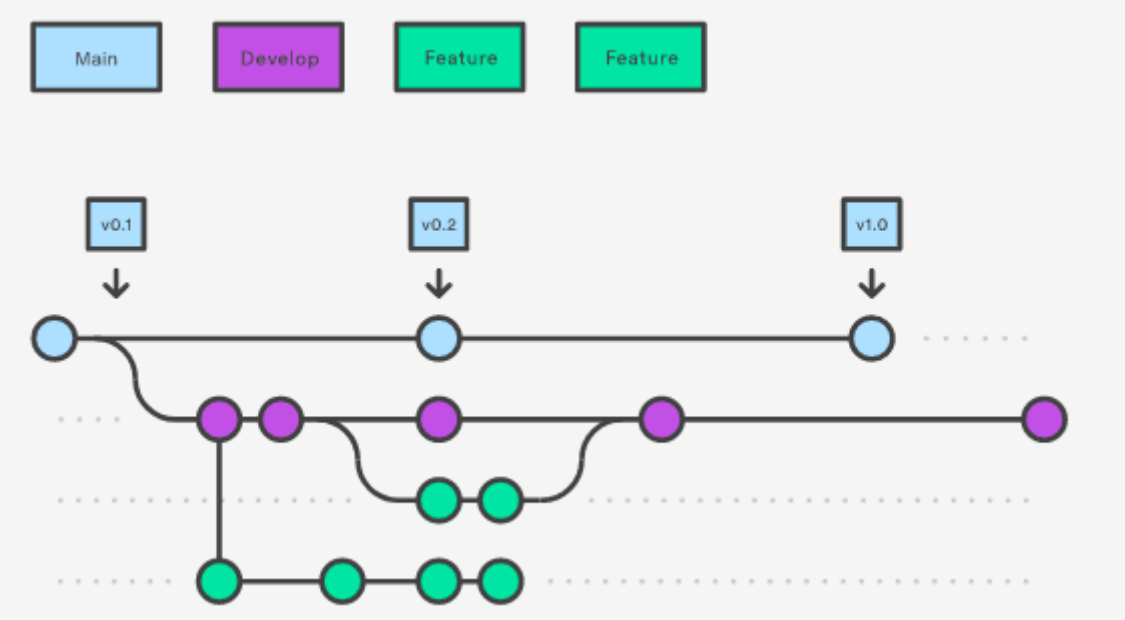
Nhánh develop và nhánh chính

Thay vì 1 nhánh chính, workflow này sử dụng 2 nhánh để theo dõi lịch sử project. Nhánh main lưu trữ những bản release chính và nhánh develop là nhánh để nhập các tính năng phát triển.



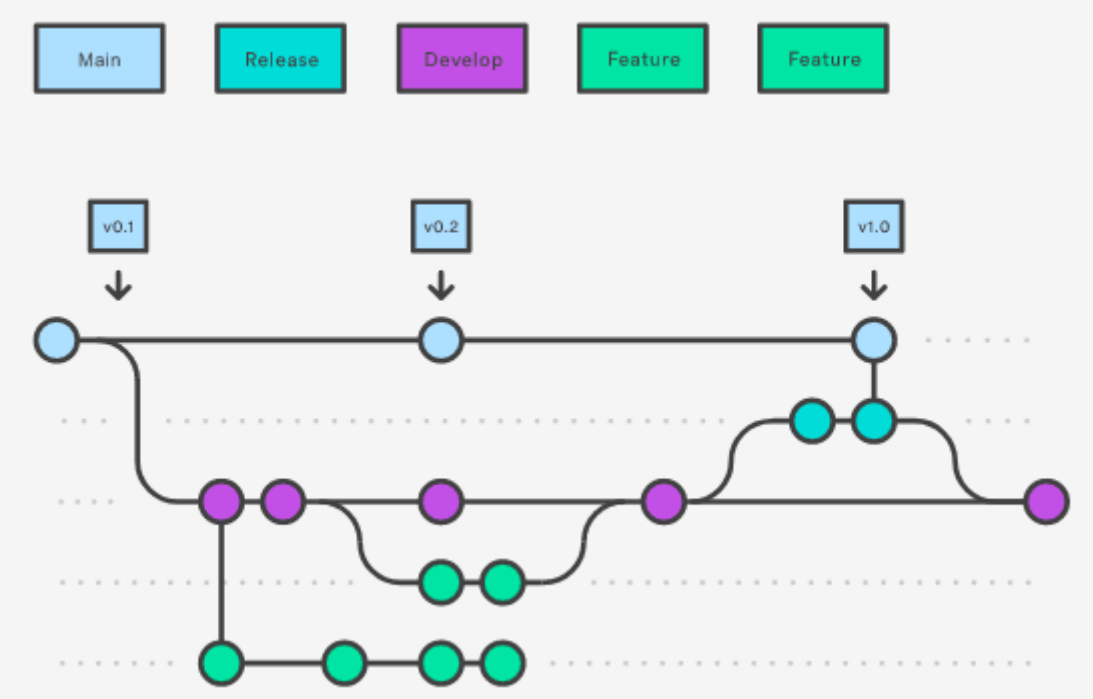
Nhánh feature

Mỗi tính năng mới nên nằm trên các nhánh riêng, có thể được đẩy lên kho lưu trữ trung tâm để backup và phối hợp. Thay vì phân nhánh ở main, nhánh feature xem nhánh develop là nhánh mẹ. Khi tính năng hoàn tất thì nhập vào nhánh develop.



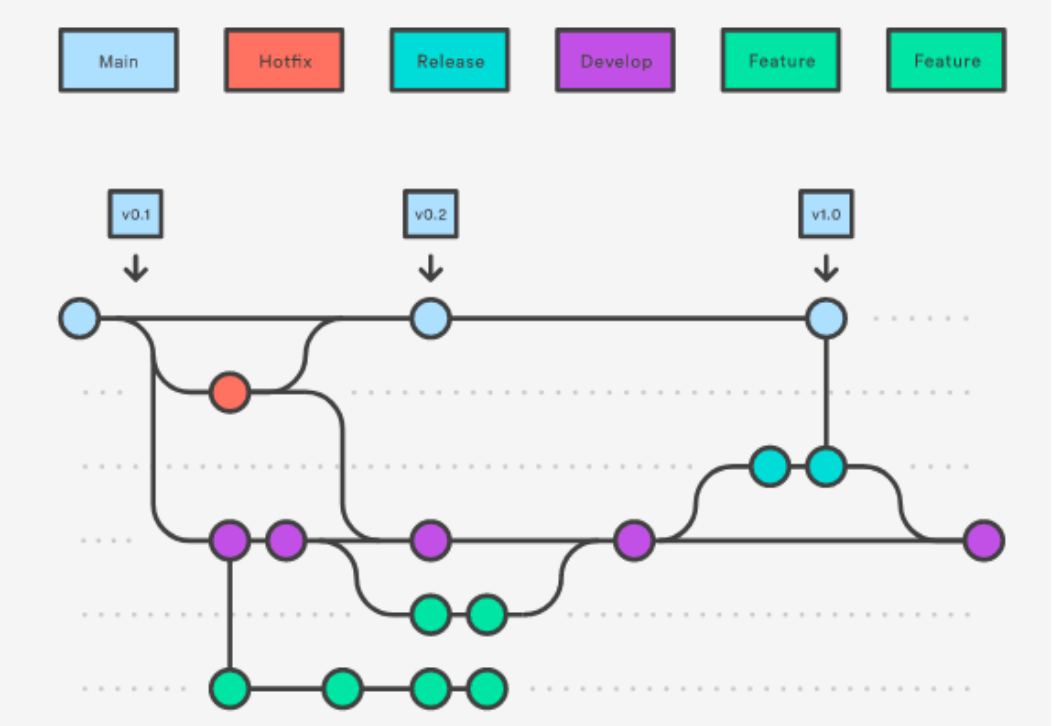
Nhánh release

Khi nhánh develop đã có đủ tính năng để release, bạn tạo 1 bản release từ nhánh develop. Tạo nhánh này dẫn đến 1 chu kì release mới nên sẽ không có tính năng mới được phát triển sau thời điểm này, chỉ có sửa lỗi và các phần phụ nhỏ khác. Khi sẵn sàng xuất xưởng, nhánh release được gộp vào nhánh chính và đánh dấu phiên bản. Thêm vào đó, nhánh này nên được gộp lại vào nhánh develop.



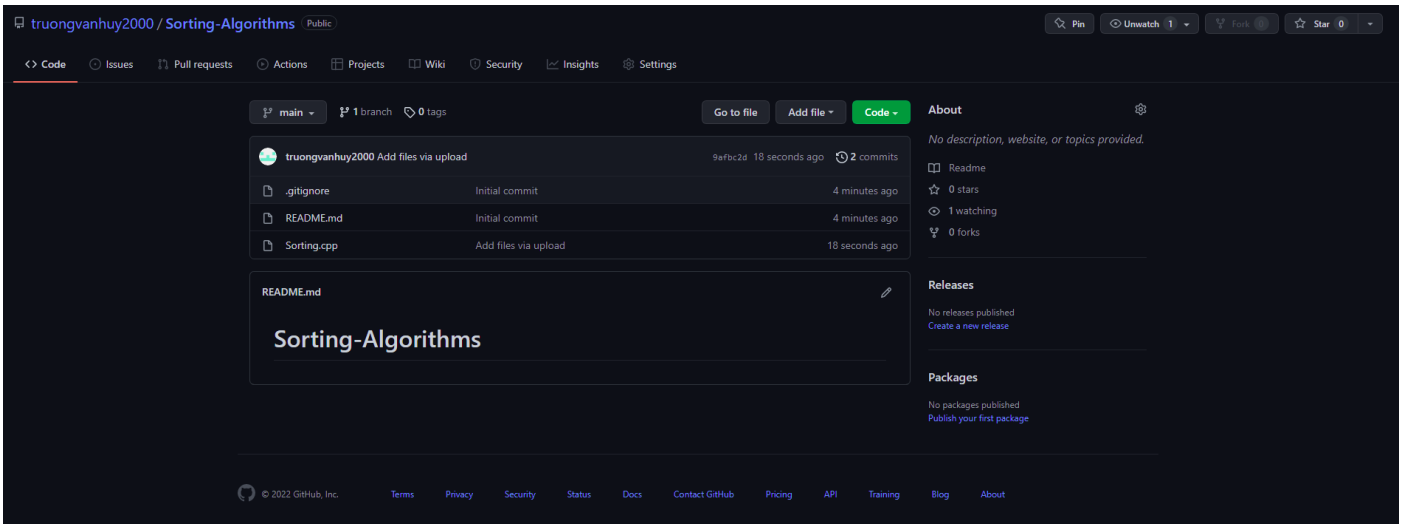
Nhánh hotfix

Nhánh hot fix thường được dùng để vá nhanh bản phát hành sản xuất. Nhánh hotfix dựa trên nhánh main thay vì develop. Khi mà đã fix được lỗi thì nên được gộp vào nhánh main và nhánh develop, và nhánh main nên được đánh dấu phiên bản



Thực hành với dự án tự tạo

Dự án Sorting Algorithm đã được tạo trên Github: https://github.com/truongvanhuy2000/Sorting-Algorithms



File code C++ thuật toán sắp xếp

52 lines (52 sloc) | 1.14 KB

Raw Blame

```cpp
#include <iostream>
#include <vector>
#include <iterator>
using namespace std;
#define I_MAX 10
class BubbleSorting
{
    public:
    void BubbleSort(vector<int>& array)
    {
        for(int i = 0; i < array.size(); i++)
        {
            for(int y = 0; y < array.size() - 1 - i; y++)
            {
                if(array[y] > array[y+1])
                {
                    swap(array[y], array[y+1]);
                }
            }
        }
        cout<< "Bubble Sort"<< endl;
    }
};
class Sorting: public BubbleSorting
{};
void NhapMang(vector<int>& array)
{
    int i_input_value = 0;
    for(int i = 0; i < I_MAX; i++)
    {
        cout<< "nhap so thu "<< i+1<< endl;
        cin>> i_input_value;
        array.push_back(i_input_value);
    }
}
void PrintMang(vector<int>& array)
{
    cout<< "mang da sap xep la: "<< endl;
    for(int i = 0; i < array.size(); i++)
    {
        cout<< array[i]<< endl;
    }
}
int main()
{
    Sorting c_sort;
    vector<int> v_mang = {1, 8, 5, 4, 7, 12, 2, 15, 6};
    c_sort.BubbleSort(v_mang);
    PrintMang(v_mang);
    //NhapMang(v_mang);
    return 0;
}
```

Chuyển đến thư mục GitWork và sau đó clone kho lưu trữ

```
Trương Văn Huy@TruongVanHuy MINGW64 /
$ cd GitWork

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork
$ git clone https://github.com/truongvanhuy2000/Sorting-Algorithms
Cloning into 'Sorting-Algorithms'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.
```

Chuyển đến thư mục làm việc

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork
$ cd Sorting-Algorithms
```

Tạo các nhánh develop, feature

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch develop

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch feature

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch -v
  develop 9afbc2d Add files via upload
  feature 9afbc2d Add files via upload
* main    9afbc2d Add files via upload
```

Chuyển sang nhánh feature để phát triển thêm Selection Sorting

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git checkout feature
Switched to branch 'feature'

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ code.
bash: code.: command not found

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ code .
```

Check git status thì thấy file Sorting.cpp đã bị thay đổi

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git status
On branch feature
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Sorting.cpp

no changes added to commit (use "git add" and/or "git commit -a")
```

Gọi lệnh git commit -a

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git commit -a
hint: Waiting for your editor to close the file...
[main 2022-04-04T17:10:43.287Z] window#load: attempt to load window (id: 1)
[main 2022-04-04T17:10:43.322Z] update#setState idle
[main 2022-04-04T17:10:43.365Z] ExtensionHostStarterWorker created
[main 2022-04-04T17:10:44.281Z] window#load: window reported ready (id: 1)
[main 2022-04-04T17:10:44.497Z] Starting extension host with pid 15076 (fork() took 12 ms).
[main 2022-04-04T17:10:44.498Z] ExtensionHostStarterWorker.start() took 14 ms.
[main 2022-04-04T17:10:49.020Z] Waiting for extension host with pid 15076 to exit.
[main 2022-04-04T17:10:49.032Z] Extension host with pid 15076 exited with code: 0, signal: null.
[feature c3c3e7d]       modified:   Sorting.cpp
 1 file changed, 21 insertions(+)
```

Chuyển về nhánh develop để gộp tính năng mới

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git checkout develop
Switched to branch 'develop'

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git merge feature
Updating 9afbc2d..c3c3e7d
Fast-forward
 Sorting.cpp | 21 ++++++++++++++++++++
 1 file changed, 21 insertions(+)
```

Tạo nhánh release để tung ra bản cập nhật 0.1

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git branch release_v0.1

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git checkout release_v0.1
Switched to branch 'release_v0.1'
```

Cập nhật nội dung release và sau đó gộp vào main, đặt tag 0.1

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (release_v0.1)
$ code .

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (release_v0.1)
$ git commit -a -m'version 0.1'
[release_v0.1 7c28aa2] version 0.1
 1 file changed, 1 insertion(+), 1 deletion(-)

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (release_v0.1)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git merge release_v0.1
Updating 9afbc2d..7c28aa2
Fast-forward
 README.md   |  3 ++-
 Sorting.cpp | 30 ++++++++++++++++++++++++-----
 2 files changed, 27 insertions(+), 6 deletions(-)

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git tag v0.1
```

Đẩy cả nhánh develop và release_v0.1 lên kho lưu trữ

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/truongvanhuy2000/Sorting-Algorithms/pull/new/develop
remote:
To https://github.com/truongvanhuy2000/Sorting-Algorithms
 * [new branch]      develop -> develop

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git push origin release_v0.1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'release_v0.1' on GitHub by visiting:
remote:      https://github.com/truongvanhuy2000/Sorting-Algorithms/pull/new/release_v0.1
remote:
To https://github.com/truongvanhuy2000/Sorting-Algorithms
 * [new branch]      release_v0.1 -> release_v0.1
```

Giả sử trong thời gian này ai đó cập nhật thêm dữ liệu tên kho, gọi lệnh git fetch để cập nhật dữ liệu mới

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git fetch origin
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 52.07 KiB | 205.00 KiB/s, done.
From https://github.com/truongvanhuy2000/Sorting-Algorithms
   7c28aa2..127ed66  main       -> origin/main

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git checkout main
Switched to branch 'main'
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

Có thể thấy nhánh main hiện đang nằm sau origin/main 1 commit, gọi lệnh merge để gộp sau đó có thể tiếp tục công việc

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git merge origin
Updating 7c28aa2..127ed66
Fast-forward
 channels4_profile.jpg | Bin 0 -> 53016 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 channels4_profile.jpg
```

Xoá các nhánh cũ

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch --d develop
Deleted branch develop (was a80f8af).

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch
  feature
* main
  release_v0.1

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch --d feature
Deleted branch feature (was c3c3e7d).

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch --d realease_v0.1
error: branch 'realease_v0.1' not found.

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch --d release_v0.1
Deleted branch release_v0.1 (was 7c28aa2).
```

Tiếp tục làm việc

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch develop

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git branch feature

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git checkout develop
Switched to branch 'develop'

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ code .

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git checkout feature
Switched to branch 'feature'

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ code .
```

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git status
On branch feature
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Sorting.cpp

no changes added to commit (use "git add" and/or "git commit -a")

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ code .

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git status
On branch feature
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Sorting.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .vscode/
        Sorting.exe

no changes added to commit (use "git add" and/or "git commit -a")
```

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git add Sorting.exe

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git add .vscode/
warning: LF will be replaced by CRLF in .vscode/tasks.json.
The file will have its original line endings in your working directory

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git commit -a -m 'add insertion'
[feature ea5dff0] add insertion
 2 files changed, 28 insertions(+)
 create mode 100644 .vscode/tasks.json
 create mode 100644 Sorting.exe

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (feature)
$ git checkout develop
Switched to branch 'develop'

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git merge feature
Updating 127ed66..ea5dff0
Fast-forward
 .vscode/tasks.json |  28 ++++++++++++++++++++++++++
 Sorting.cpp        |  22 ++++++++++++++++++++-
 Sorting.exe        | Bin 0 -> 159889 bytes
 3 files changed, 49 insertions(+), 1 deletion(-)
 create mode 100644 .vscode/tasks.json
 create mode 100644 Sorting.exe
```

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git push origin develop
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 48.31 KiB | 1.61 MiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/truongvanhuy2000/Sorting-Algorithms
   a80f8af..ea5dff0  develop -> develop
```

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git push origin feature
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/truongvanhuy2000/Sorting-Algorithms/pull/new/feature
remote:
To https://github.com/truongvanhuy2000/Sorting-Algorithms
 * [new branch]      feature -> feature

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git merge develop
Updating 127ed66..ea5dff0
Fast-forward
 .vscode/tasks.json |  28 ++++++++++++++++++++++++++
 Sorting.cpp        |  22 ++++++++++++++++++++-
 Sorting.exe        | Bin 0 -> 159889 bytes
 3 files changed, 49 insertions(+), 1 deletion(-)
 create mode 100644 .vscode/tasks.json
 create mode 100644 Sorting.exe
```

```
Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git tag -a v1.4 -m "version 0.2"

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git push orgigin main v1.4
fatal: 'orgigin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git push origin main v1.4
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 160 bytes | 160.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/truongvanhuy2000/Sorting-Algorithms
   127ed66..ea5dff0  main -> main
 * [new tag]         v1.4 -> v1.4

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git tag -d v1.4
Deleted tag 'v1.4' (was 4c14741)

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git push origin --delete v1.4
To https://github.com/truongvanhuy2000/Sorting-Algorithms
 - [deleted]         v1.4

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git tag -a v0.2 -m 'version 0.2'

Trương Văn Huy@TruongVanHuy MINGW64 /GitWork/Sorting-Algorithms (main)
$ git push origin main v0.2
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 159 bytes | 159.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/truongvanhuy2000/Sorting-Algorithms
 * [new tag]         v0.2 -> v0.2
```

Phiên bản 0.2 đã được đẩy lên Github

Nguồn tham khảo : Pro Git - Scott Chacon, Ben Straub: https://git-scm.com/book/en/v2

https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow