Git Practice Course

Làm sao để bạn quản lý dự án của mình một cách hiệu quả?

Mục tiêu khóa học

- Nắm bắt được các khái niệm cơ bản về quản lý phiên bản
- Nắm được kiến thức cơ bản về Git
- Các lệnh thực hành thường sử dụng với Git
- Quy trình làm việc với Git
- Sử dụng Github, giới thiệu về BitBucket
- Sử dụng Git hiệu quả với công cụ Source Tree

Giới thiệu

- Tác giả: Nguyễn Văn Nghĩa
  - SĐT/Zalo: 0986.951.957
  - Facebook:
    - Cá nhân: https://www.facebook.com/nghia12a1
    - Nhóm: https://www.facebook.com/groups/laptrinhdientu
  - Blog: https://laptrinhcnhung.blogspot.com/

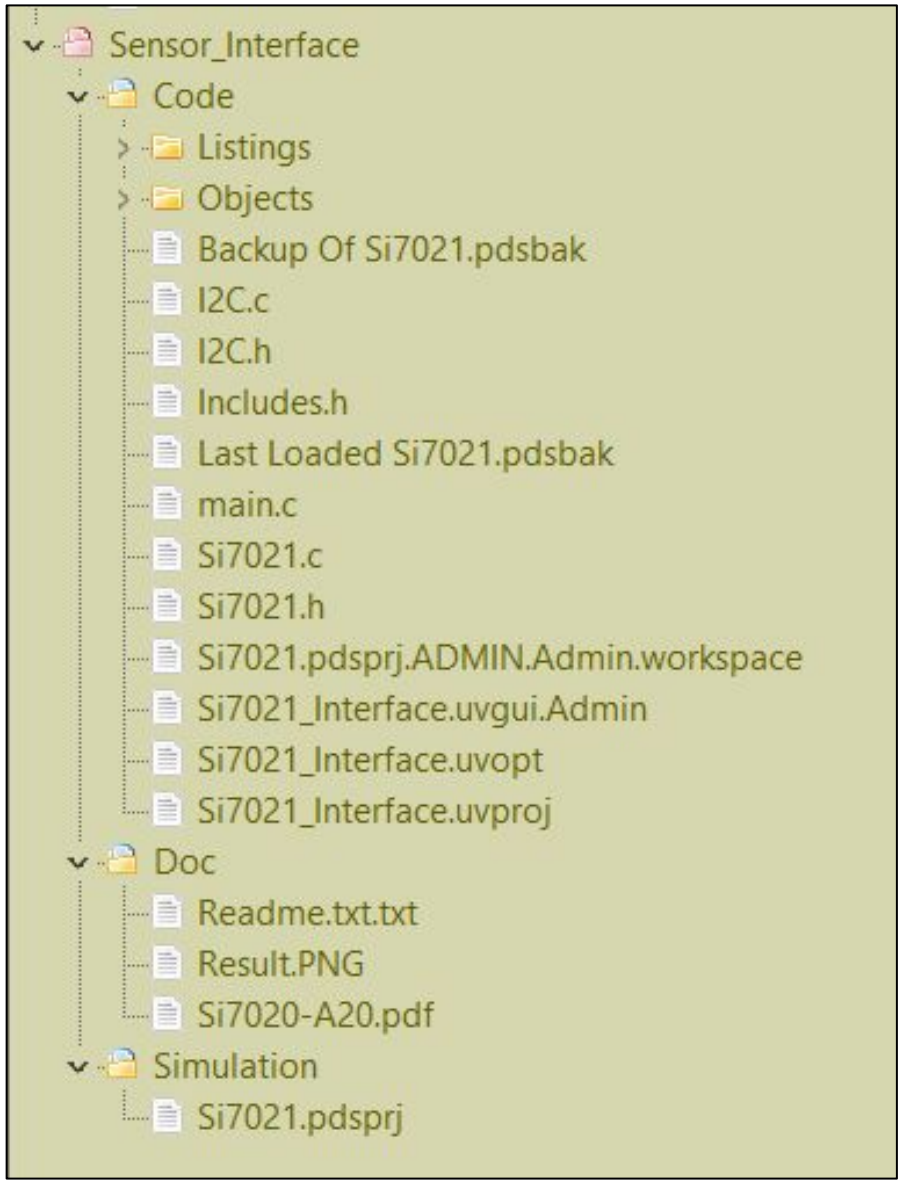
- Tham khảo:
  - Open Source của SUN
  - Quy trình làm việc với Git tại Fsoft

Nội dung

1. Giới thiệu và cài đặt Git
2. Khái niệm cơ bản trong Git
3. Branch trong Git
4. Quản lý dự án với Git-Github
5. Git trong dự án thực tế?

Nội dung thực hành

- Quản lý dự án đơn giản với Sourcetree và Github



1. Giới thiệu và Cài đặt Git

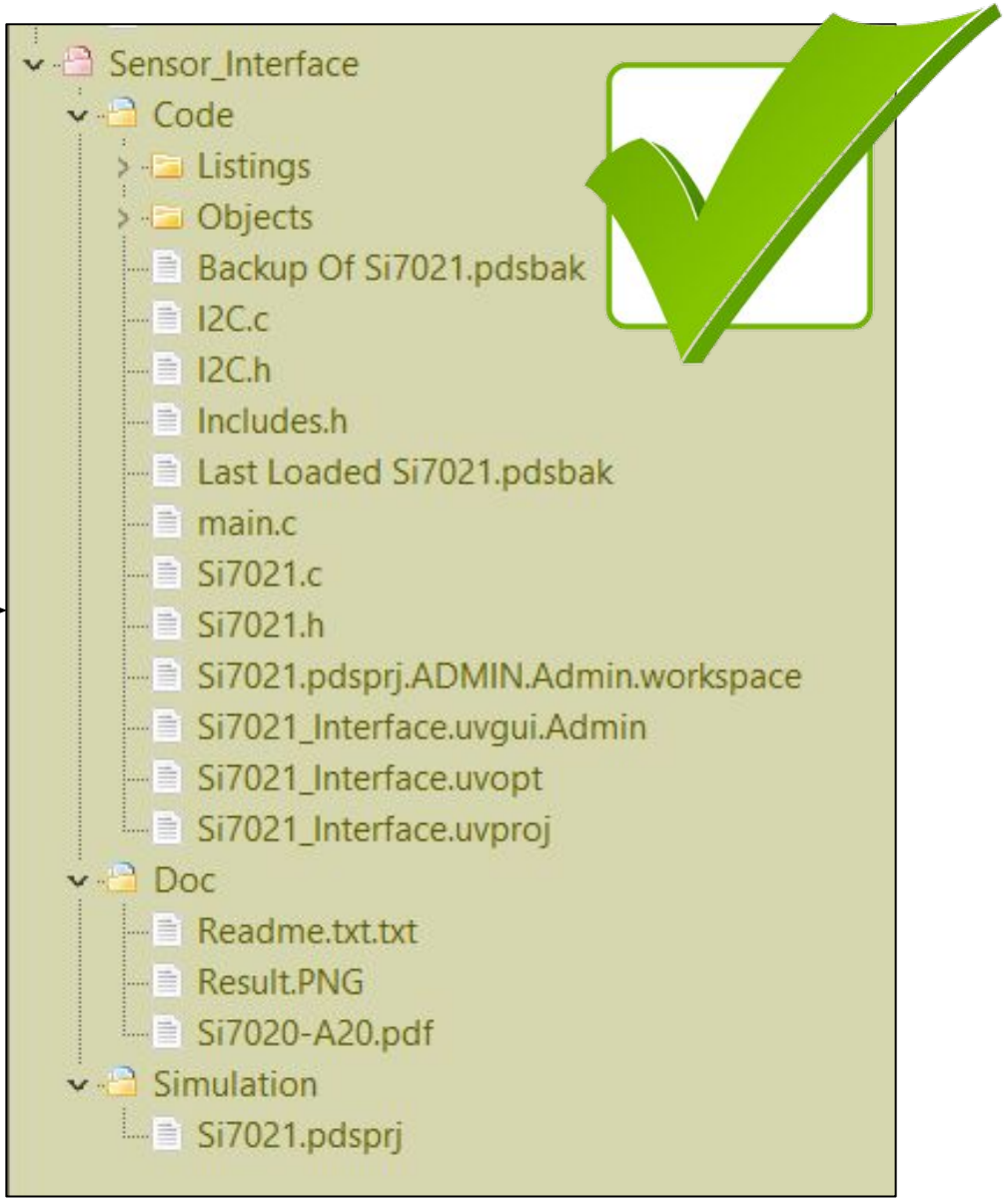
Giới thiệu về Quản lý phiên bản?

Tình huống – No VCS

- Code lưu trên máy nhóm trưởng!

Teamwork ngon lành

Code OK!

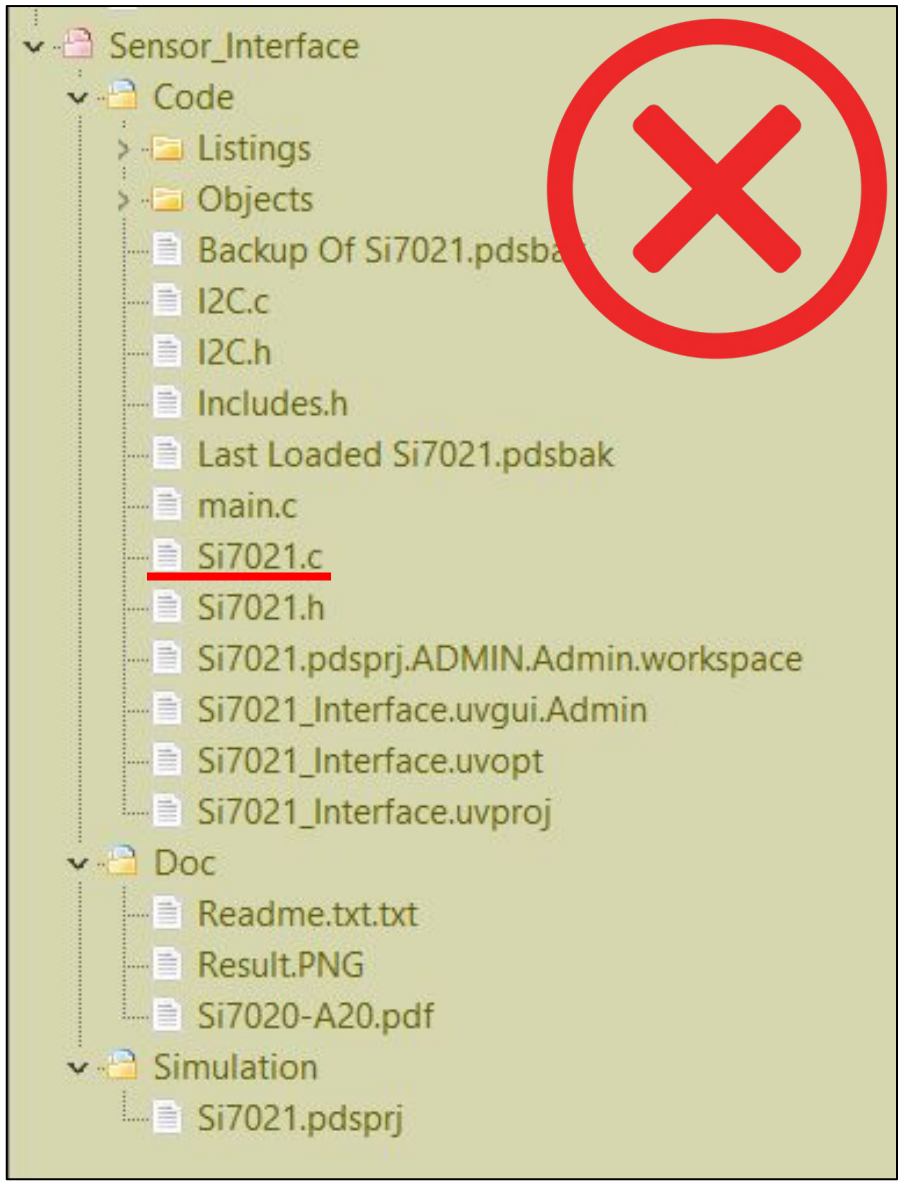


Tình huống – Xóa/Sửa nhầm file

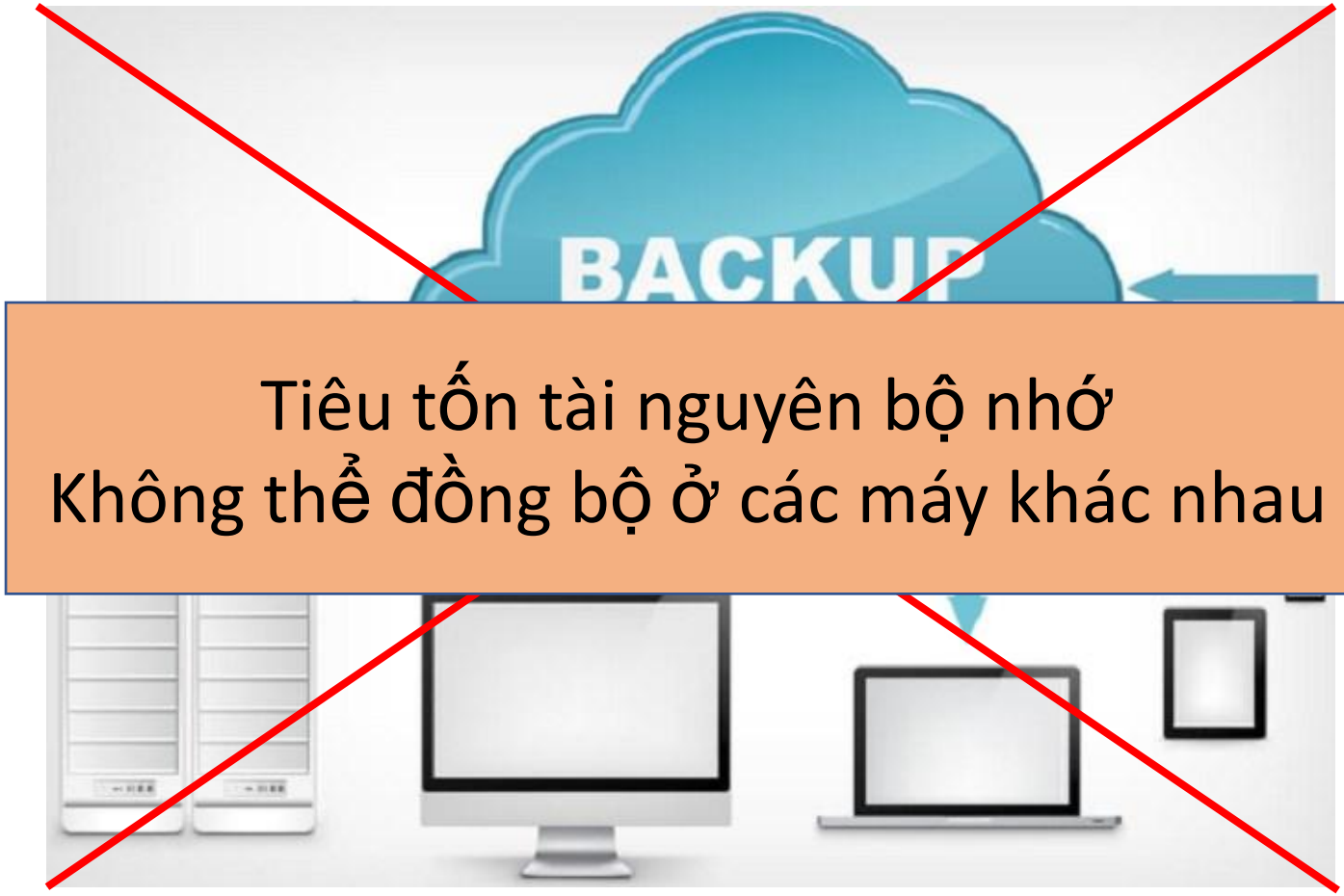
- Nhóm trưởng sửa/xóa nhầm file

 Code không chạy được

 Ảnh hưởng đến kết quả cả nhóm



Giải quyết?



Tiêu tốn tài nguyên bộ nhớ
Không thể đồng bộ ở các máy khác nhau

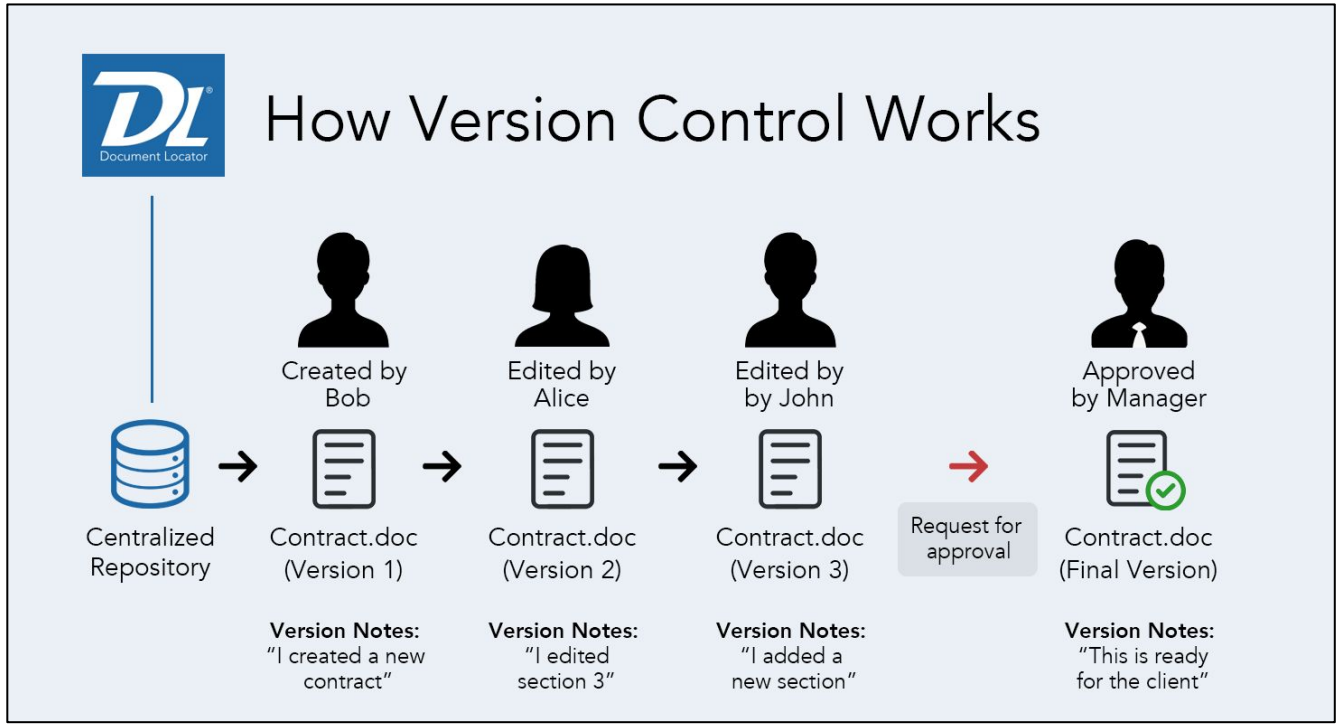
Tại sao cần quản lý phiên bản?

- Trong các dự án:
  - Code sẽ đi qua nhiều phiên bản, nhiều sự thay đổi trên từng phiên bản
  - Có thể có rất nhiều file, folder
  - Nhiều người làm việc trên cùng một file/folder
  - Các file có thể bị xóa/sửa không như ý (do lỡ tay/mất điện/virus)
  - Phiên bản mới bị fail, cần dùng lại phiên bản cũ
  - …

 Cần có một hệ thống quản lý phiên bản

VCS – Version Control System

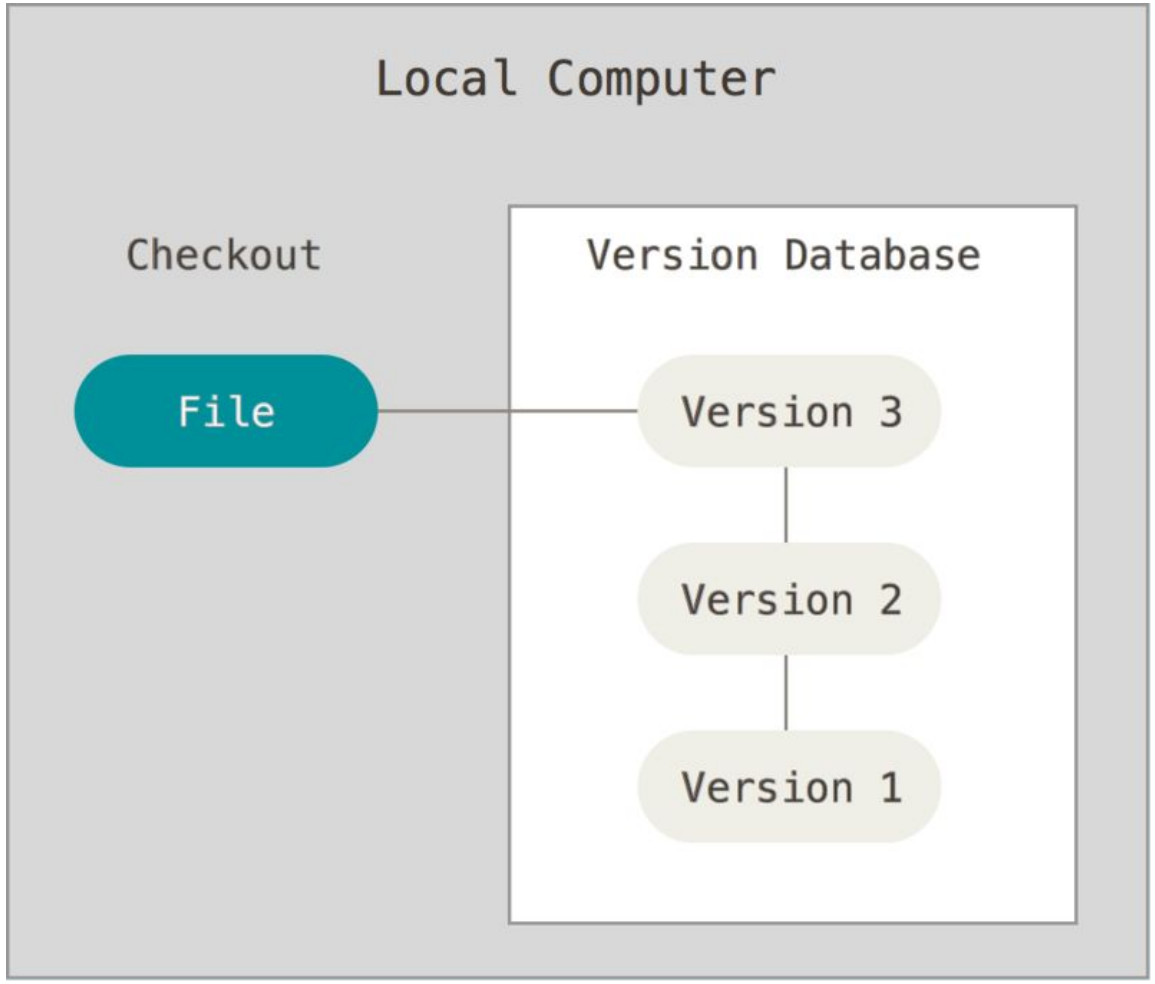
- Ghi lại sự thay đổi các file
- So sánh các versions, quản lý các versions
- Giảm thiểu sự mất mát file



Local Version Control System

- Sử dụng với mô hình máy cá nhân
- Người dùng sao chép các phiên bản khác nhau của dự án ra các thư mục khác nhau

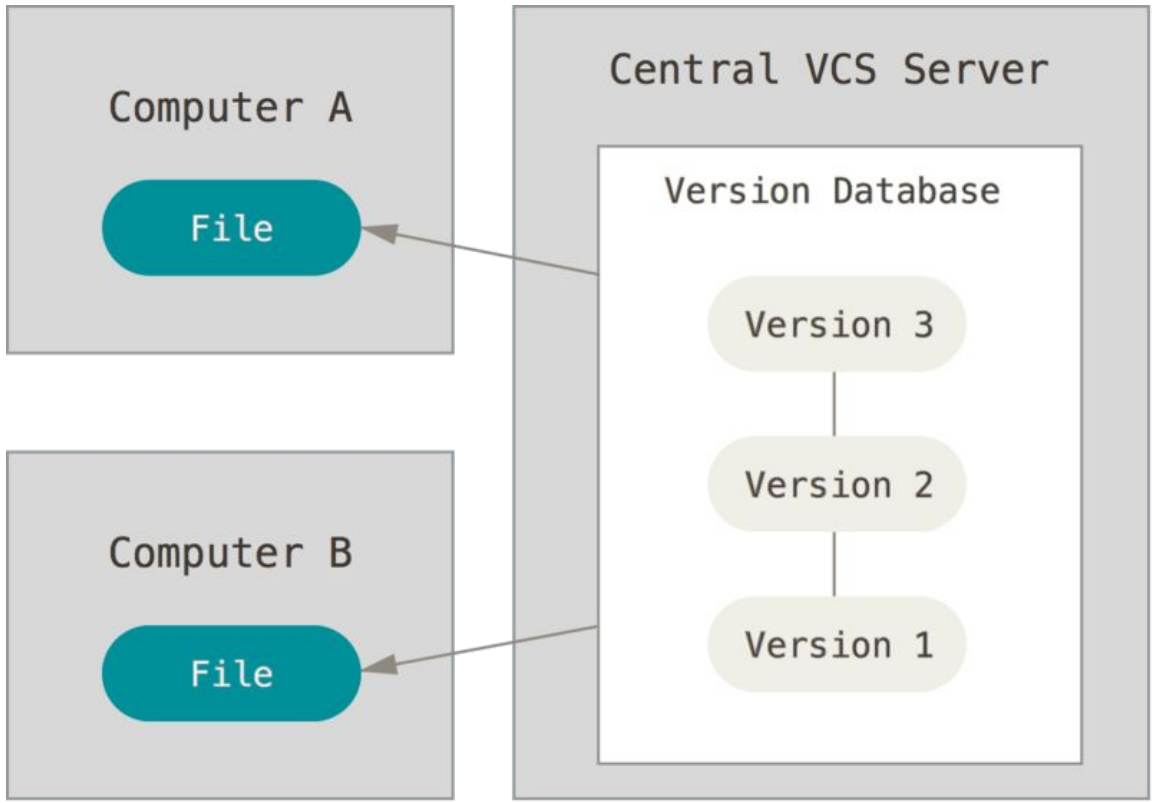
- Dễ gây nhầm lẫn vì có quá nhiều phiên bản trong máy
- Gây tốn bộ nhớ máy tính



Centralized Version Control System – SVN

- Một máy chủ lưu trữ các phiên bản của dự án + một số máy khách
- Các máy khách thao tác trên từng các files

- Nếu máy chủ bị hỏng có thể gây mất dữ liệu
- Máy chủ bảo trì có thể khiến công việc bị ngưng hoạt động

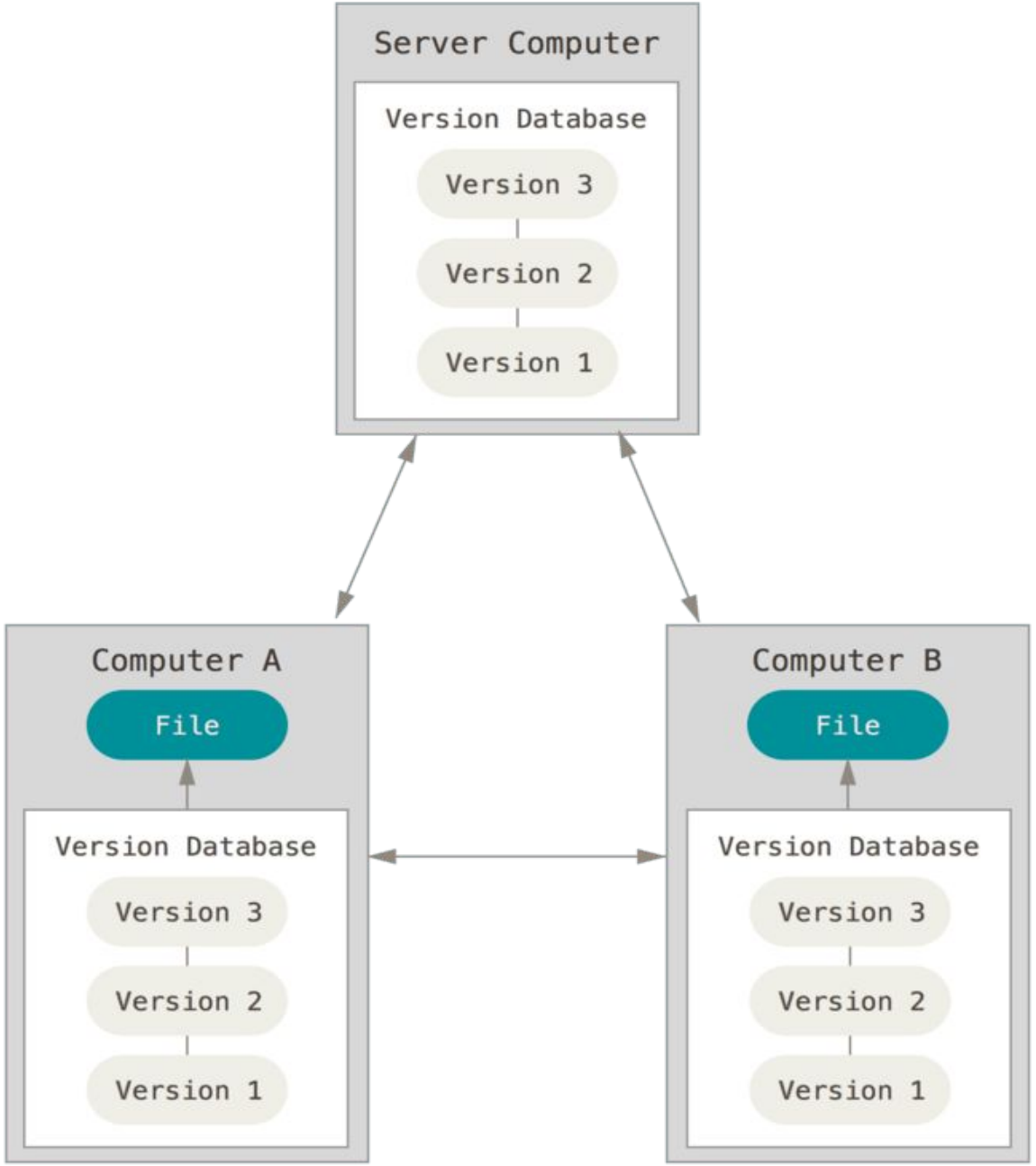


Distributed Version Control System

- Một máy chủ Server phản ánh đầy đủ nhất về dự án
- Các máy khách cũng sẽ lưu các phiên bản trên máy cá nhân

 Không bị mất dữ liệu khi server sập!

 Các thành viên có thể nắm bắt được toàn dự án



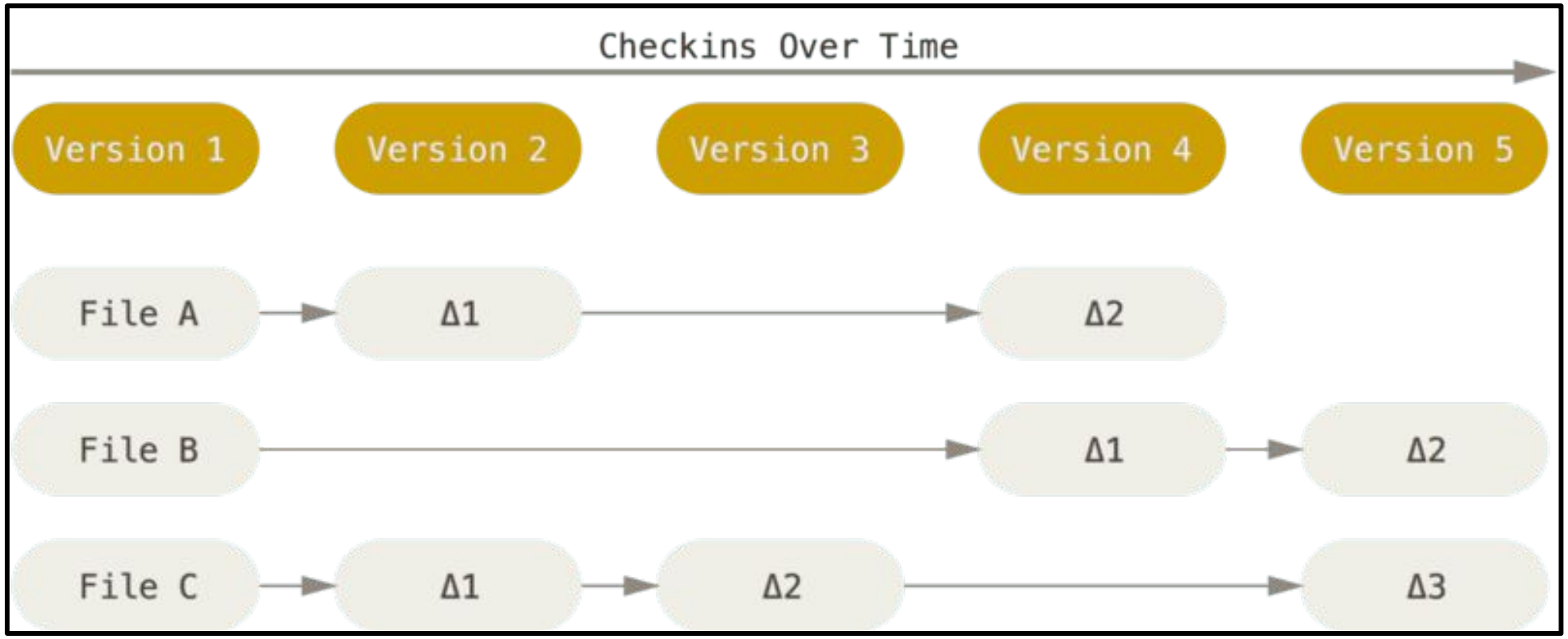
Giới thiệu và cài đặt Git

Tổng quan về Git

- Git là một công cụ quản lý phiên bản phân tán (Distributed VCS)
- Git được phát triển vào năm 2005 bởi cộng đồng Linux với các đặc điểm:
  - Tốc độ nhanh
  - Thiết kế đơn giản
  - Hỗ trợ hàng ngàn nhánh (branch) khác nhau
  - Có thể xử lý các dự án lớn
  - Hoạt động theo mô hình phân tán (Distributed VCS)

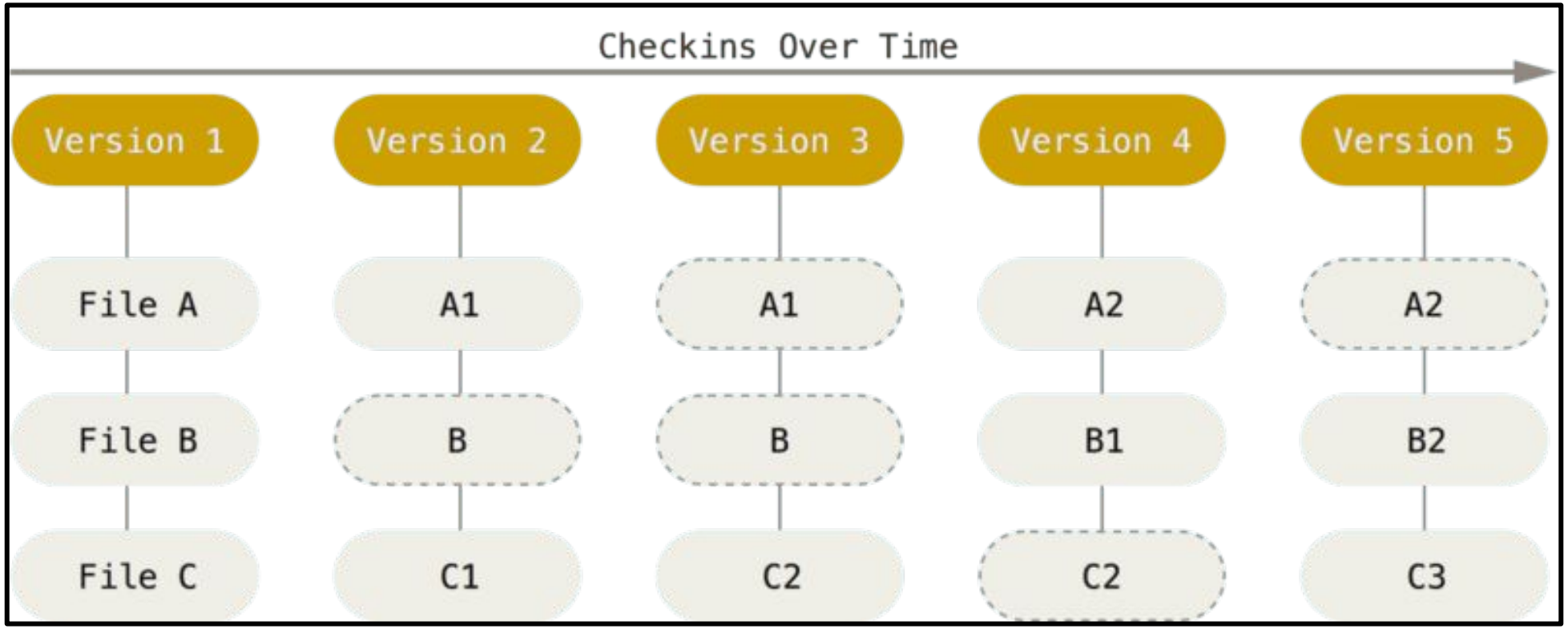
Đặc điểm của Git – Snapshot, Not differences

Mô hình khác: quản lý sự thay đổi của các file trong dự án



Đặc điểm của Git – Snapshot, Not differences

Git: Quản lý tất cả các file trong dự án



Đặc điểm của Git – Nearly Every Operation is Local

THAO TÁC TRÊN MÁY CÁ NHÂN - LOCAL
 TĂNG TỐC ĐỘ

KHÔNG PHỤ THUỘC VÀO INTERNET

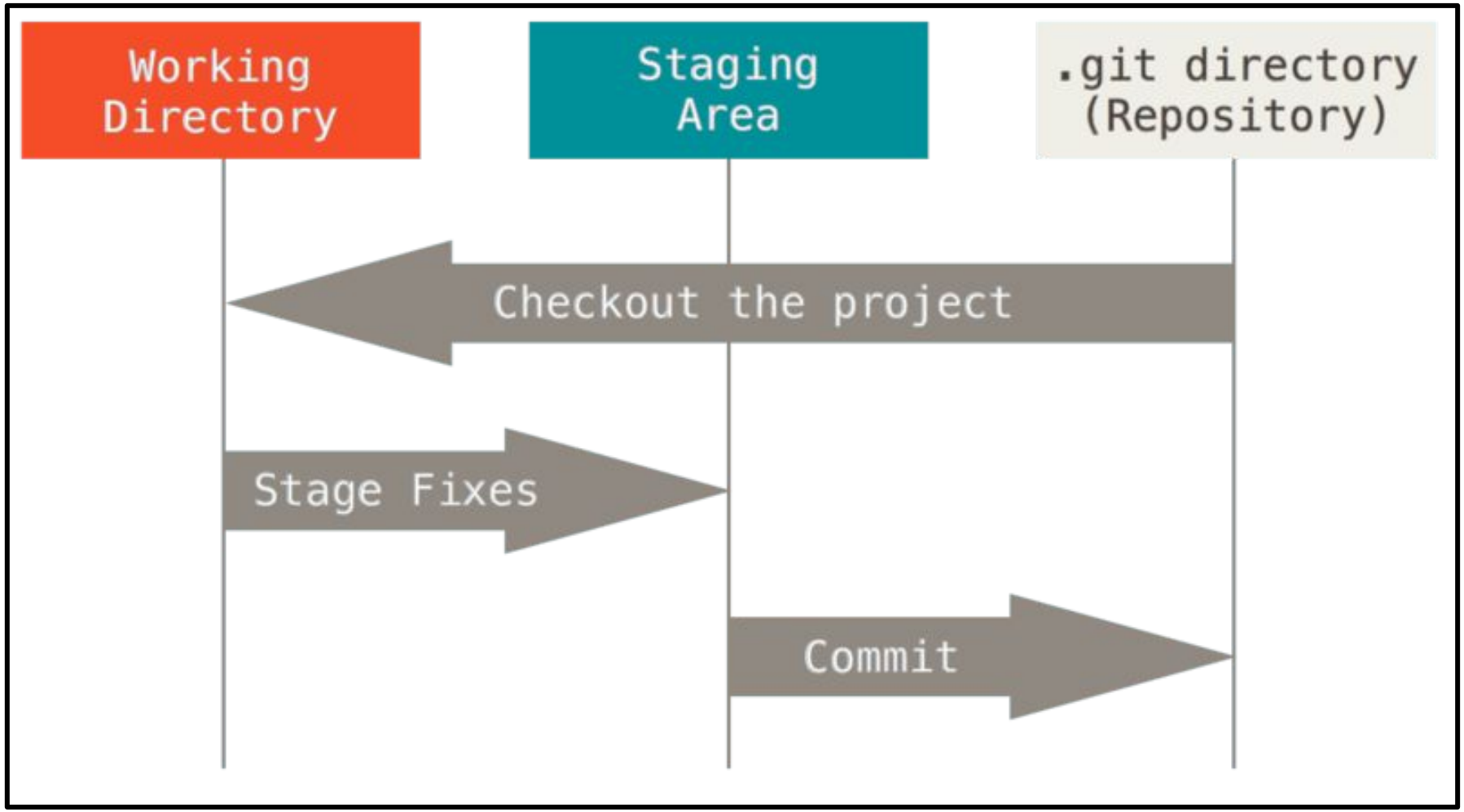
Đặc điểm của Git – Integrity – Tính toàn vẹn

- Mọi thứ trong Git được kiểm tra trước khi nó được lưu trữ và sau đó được tham chiếu bởi thuật toán Check-sum
  - Không thể thay đổi nội dung không có trên Git
  - Đảm bảo tính toàn vẹn dữ liệu khi đồng bộ

Đặc điểm của Git – Git thường chỉ thêm dữ liệu

- Bất cứ thao tác nào trên Git (chỉnh sửa, xóa files) sẽ được coi như một Version mới.
  - Không sợ bị mất file
  - Có thể thoải mái thử nghiệm

Đặc điểm của Git – Ba trạng thái hoạt động



Tải & Cài đặt Git

- Link tải Git trên Linux: https://git-scm.com/download/linux
- Link tải Git trên MacOS: https://git-scm.com/download/mac
- Link tải Git trên Windows: https://git-scm.com/download/win

Cấu hình Git cho lần đầu sử dụng

- Cấu hình Định danh (Tên + Địa chỉ)

```
$ git config --global user.name “Your name”
$ git config --global user.email abc@gmail.com
```

- Cấu hình Editor (Trình soạn thảo)

```
$ git config --global core.editor
   "'C:/Program Files/Notepad++/notepad++.exe’
   -multiInst -nosession"
```

Cấu hình Git cho lần đầu sử dụng

- Kiểm tra cấu hình

```
$ git config --list
```

- Mở trợ giúp

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

- Trợ giúp với một lệnh cụ thể Sử dụng tùy chọn **-h**

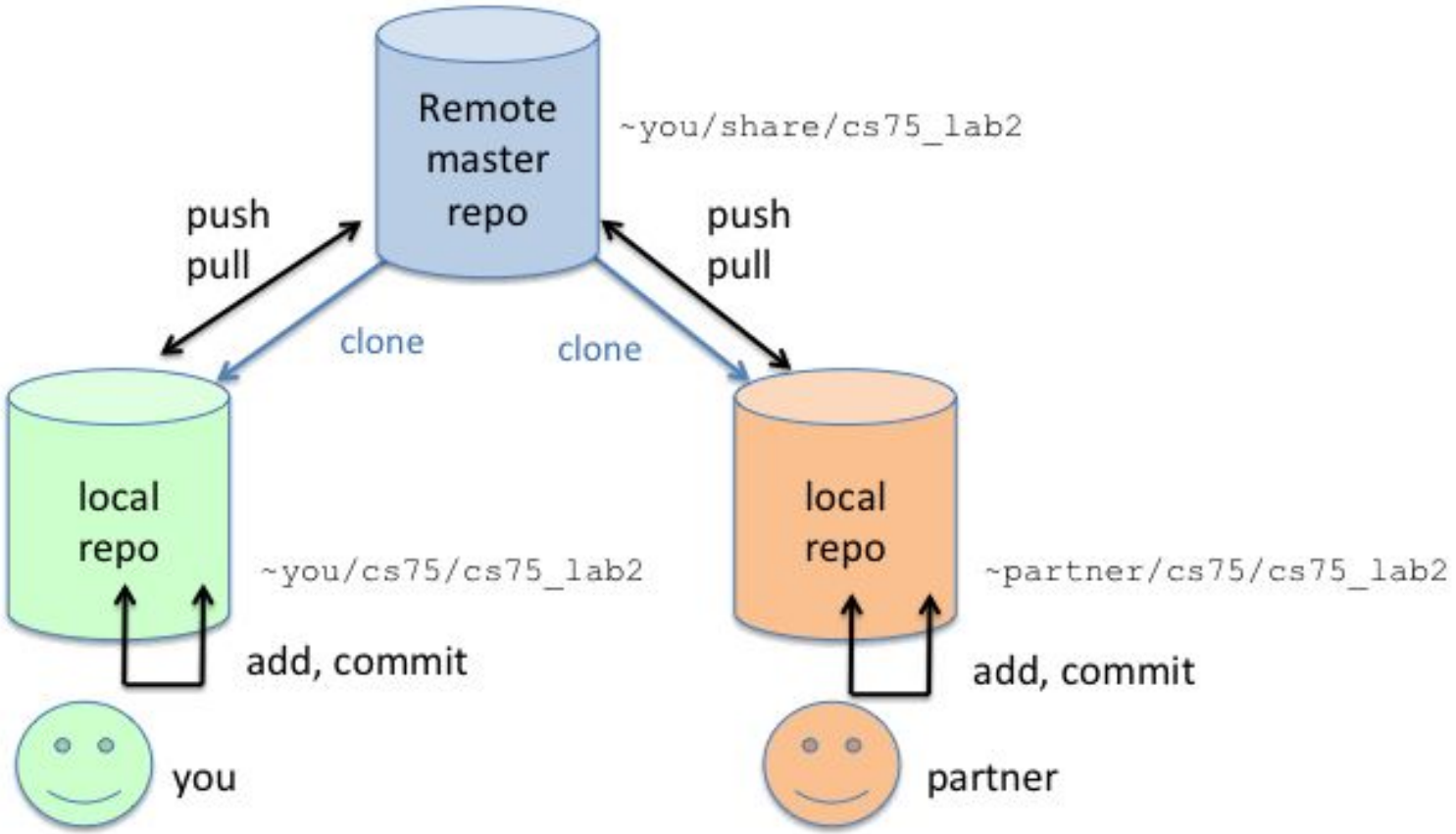
```
$ git add -h
```

2. Khái niệm cơ bản trong Git

Repository – Kho lưu trữ

Repository – Kho lưu trữ

- Nơi lưu trữ các phiên bản của dự án, có thể truy cập đẩy code lên hoặc lấy code về



Tạo một Repository

- Trỏ đến folder cần tạo Repo gọi lệnh **git init**
- Hoặc Chuột phải vào vị trị cần tạo Repo chuột phải chọn **git bash here**

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

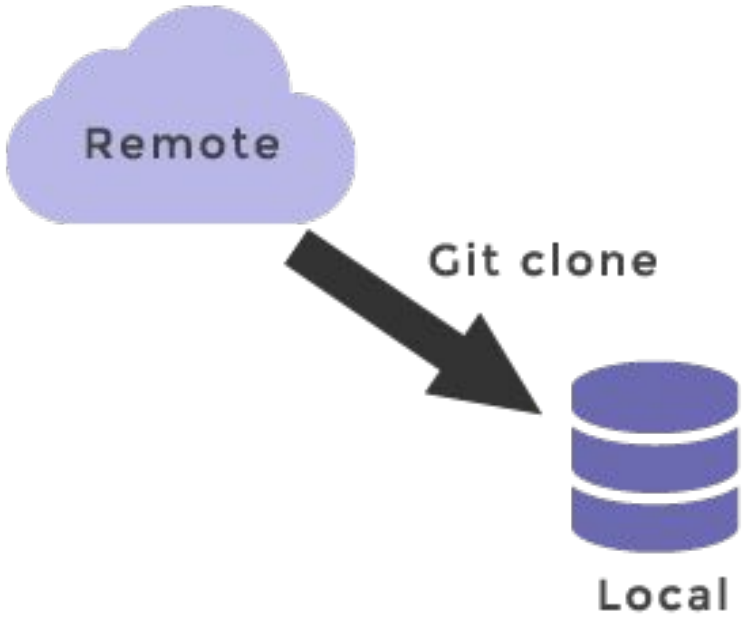
- Một thư mục con (.git) được tạo ra chứa tất cả file repo cần thiết – Git repository Skeleton.

Sao chép một Repo (Clone)

- Mỗi repo sẽ có một đường dẫn <url> để người dùng truy cập

```
$ git clone <url>
```

- Repo sẽ được sao chép về thư mục đang trỏ tới

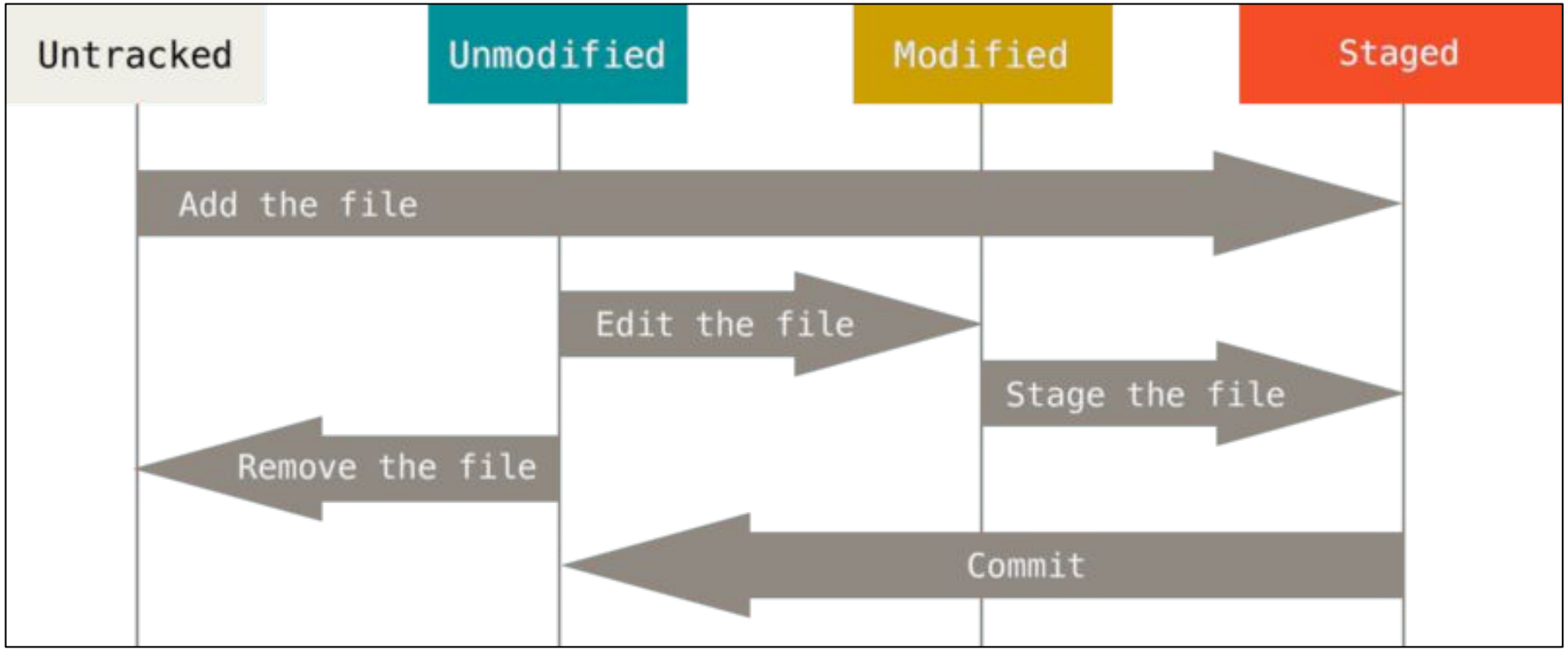



```
$ git clone https://github.com/nghia12a1-t-ara/8051.git
```

Chu kỳ làm việc của files

Kiểm tra sự thay đổi của các files

$ git status



Ví dụ: Tạo và thêm một file

- Tạo và thêm một file

$ **echo >** LED.c ⟶ $ **git add** LED.c

- Kiểm tra sự thay đổi bằng câu lệnh **git status**
- Kết quả:

```
Admin@Admin MINGW64 /d/Teaching_Documents/Git/Repo_Examples (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   LED.c
```

Commit sự thay đổi

Commit để xác nhận các file ở trạng thái **Staged** sẽ chuyển sang trạng thái **Unmodified**

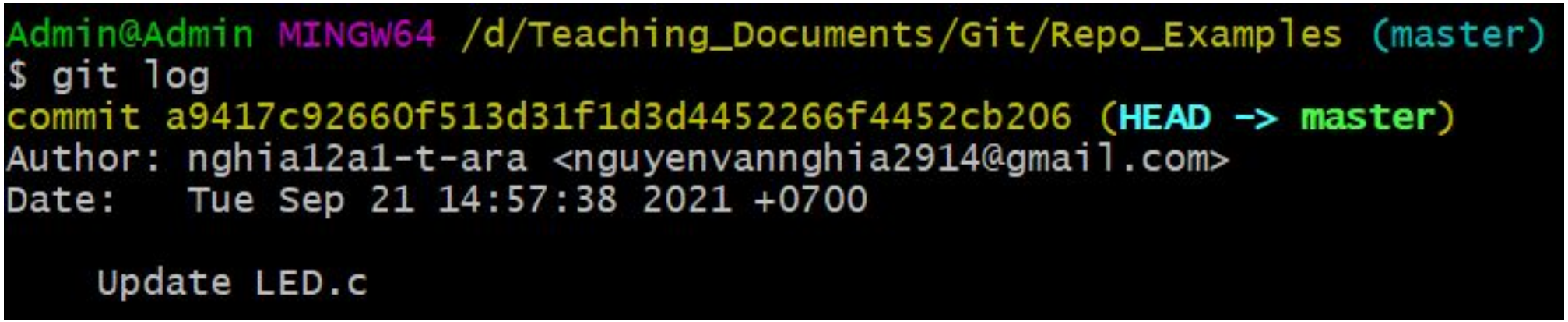
$ **git commit –m** “Your Message”

```
Admin@Admin MINGW64 /d/Teaching_Documents/Git/Repo_Examples (master)
$ git commit -m "Update LED.c"
[master (root-commit) a9417c9] Update LED.c
 1 file changed, 1 insertion(+)
 create mode 100644 LED.c
```

Kiểm tra lịch sử Commit

Có thể xem lại lịch sử Commit

```
$ git log
```



Unstaged một file Staged

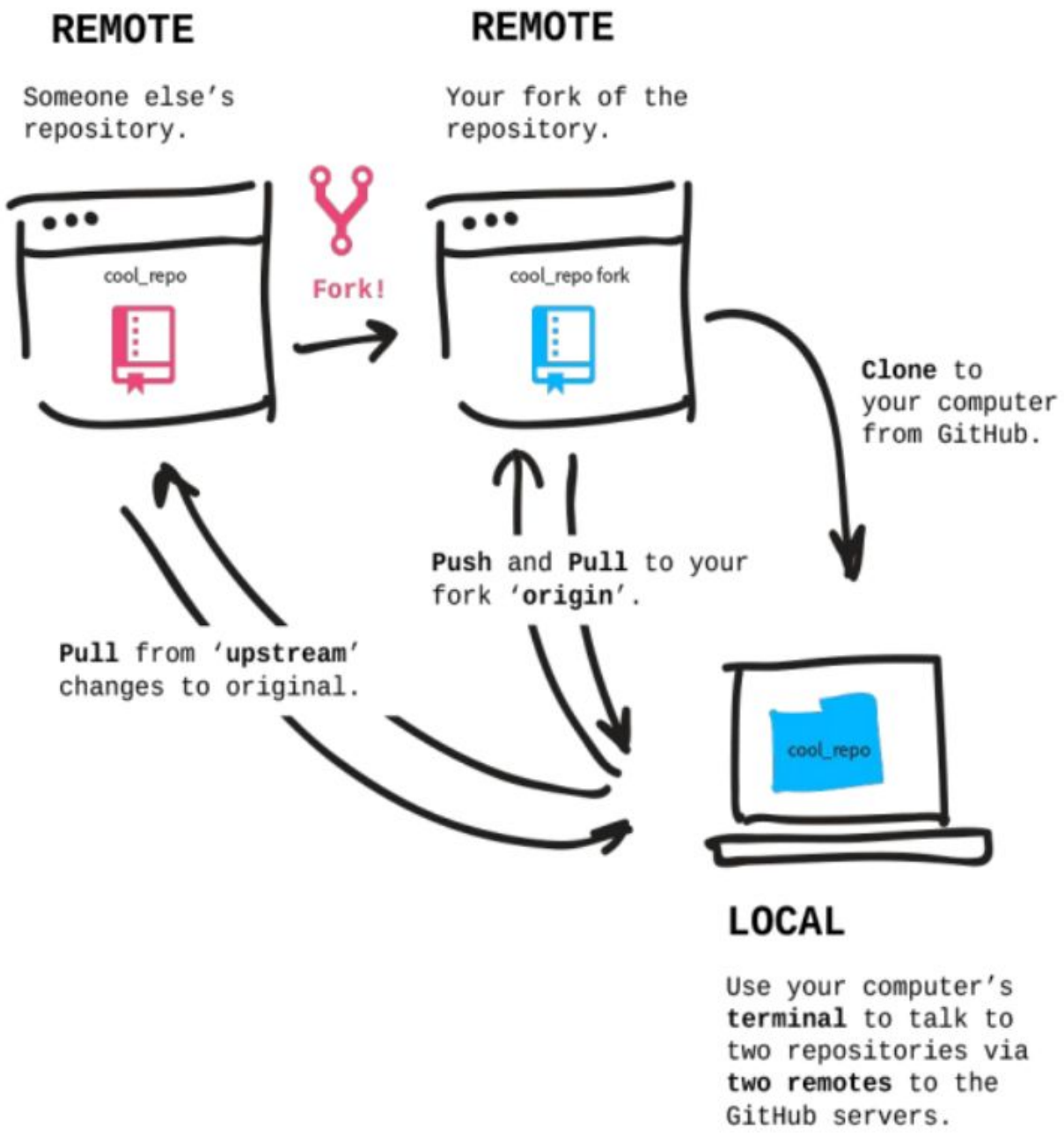
- Sử dụng khi cần chỉnh sửa nội dung file đã commit

```
$ git restore “File”
$ git reset HEAD -- path/to/file
```

Làm việc với Remote

Remote Repository là gì?

- Các phiên bản của dự án được lưu trữ trên Server
- Bạn có thể có một trong những Remote này



Làm việc với Remote

- Chỉ ra Remote của bạn đang làm việc

```
$ git remote –v
```

- Thêm một remote vào Repository

```
$ git remote add “remote name” <url>
```

Làm việc với Remote

- Cập nhật trạng thái mới nhất của Repo

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

- Lấy code trên Repo về

```
$ git pull <remote>
```

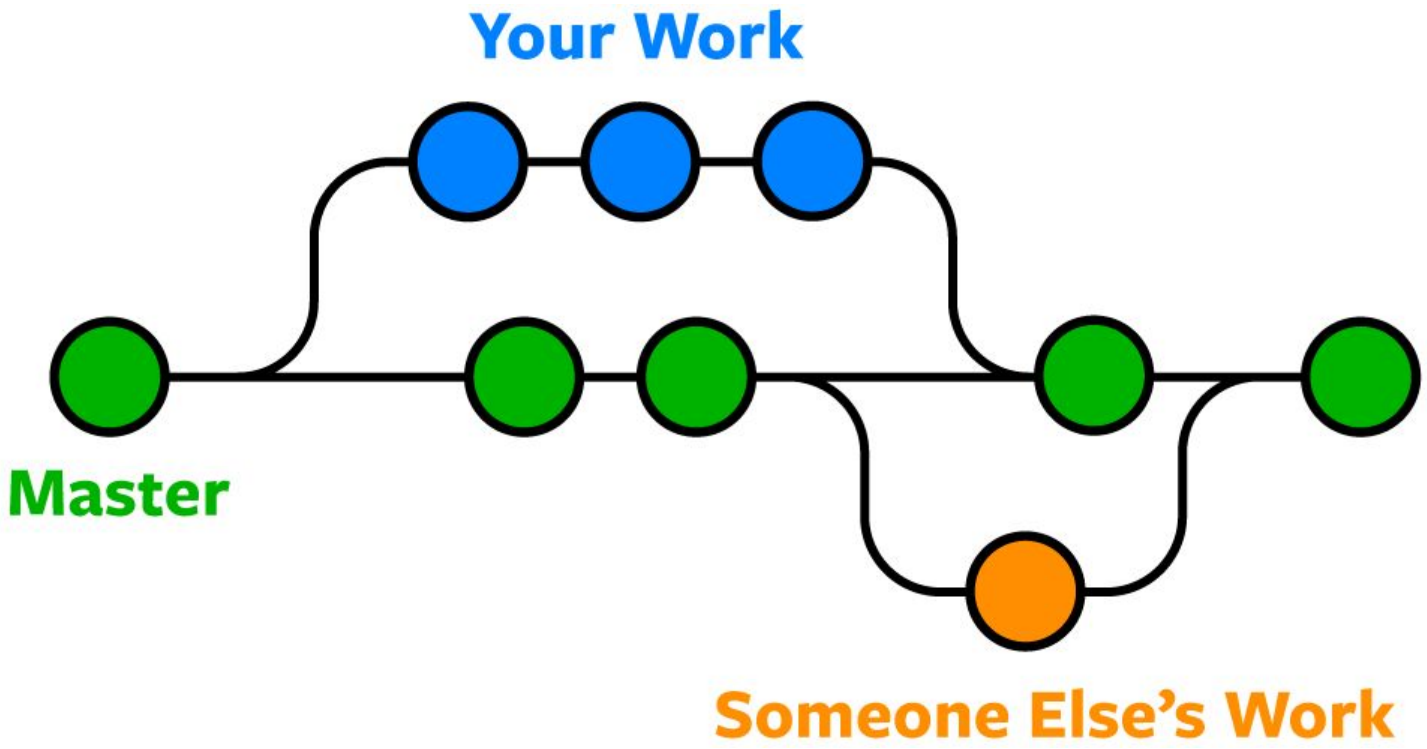
- Đẩy code đã commit lên Repo

```
$ git push <remote> <branch>
```

3. Branch trong Git

Branch là gì?

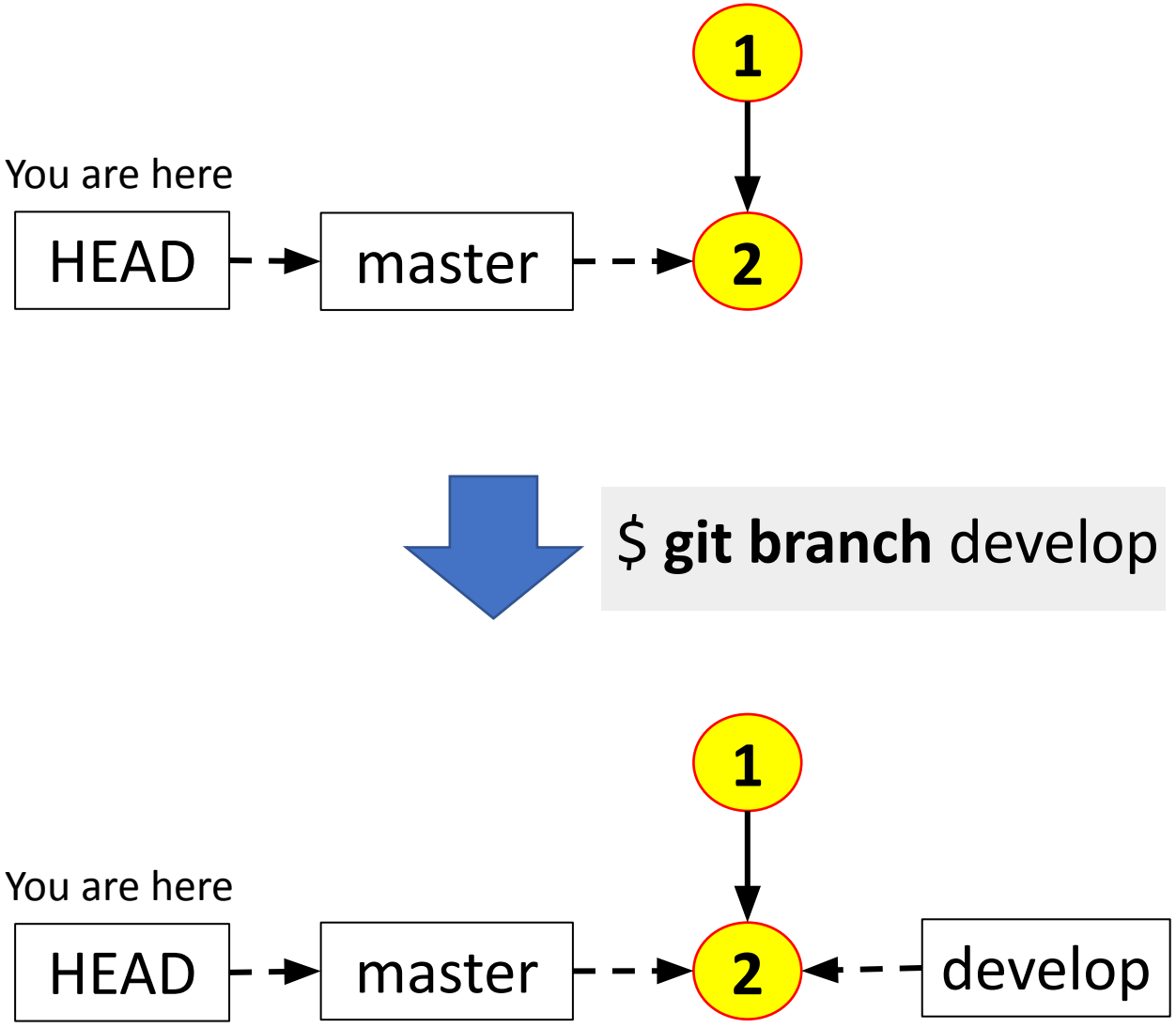
- Là các nhánh làm việc trong dự án
- Mỗi người / nhóm / task có thể làm việc trên một nhánh



Tạo Branch

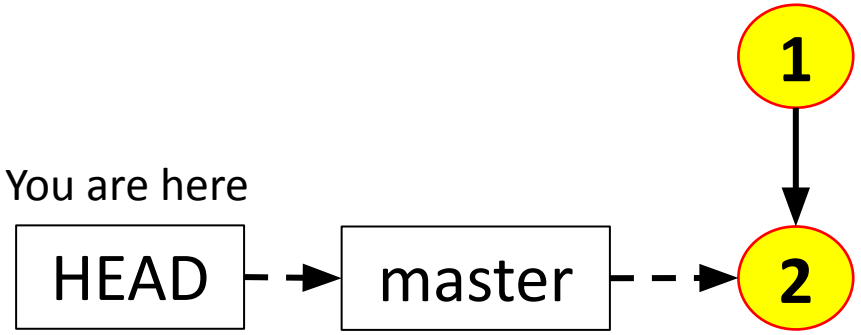
- Là Tạo một Branch mới

```
$ git branch <branch name>
```



Checkout đến Branch

- Nhảy đến một Branch để làm việc

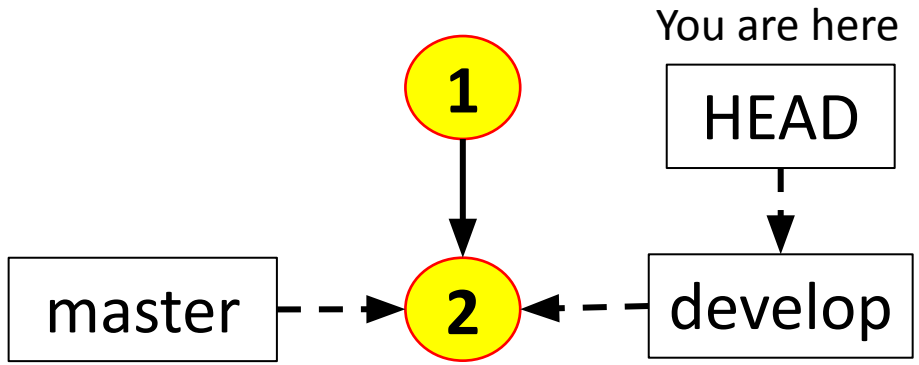


$ **git checkout** develop

$ **git checkout** <branch name>

Tạo Branch và checkout

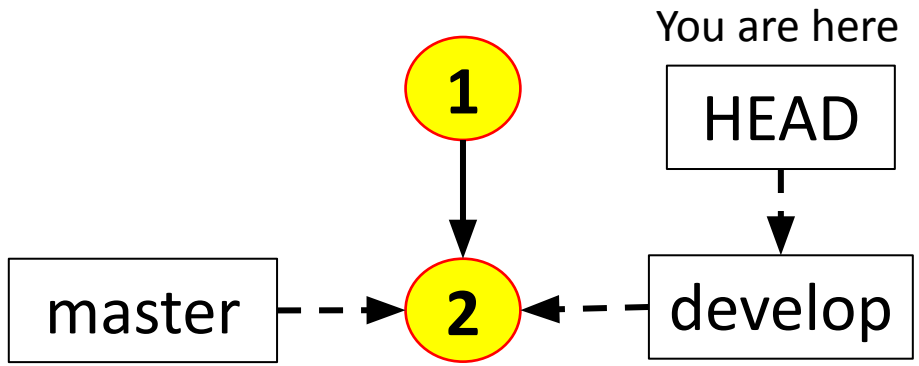
$ **git checkout –b** <branch name>



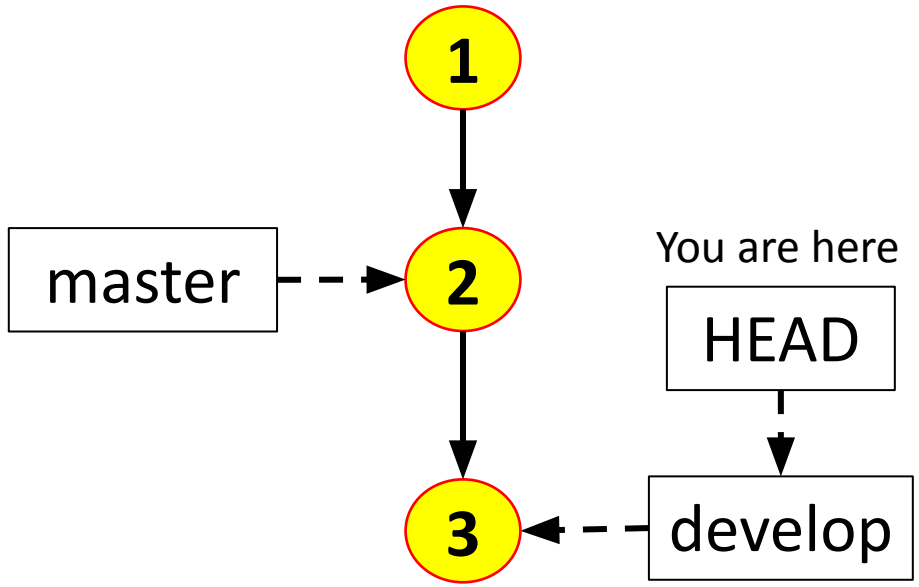
Commit trên Branch mới

- Commit trên Branch mới sẽ tạo thêm một node trên branch này

```
$ git commit -m “message”
```




```
$ git commit -m “led.c”
```



Làm việc với Branch

- Liệt kê các tag đang có

$ **git checkout –b** <branch name>

```
Admin@Admin MINGW64 /d/Teaching_D
$ git checkout -b task1
Switched to a new branch 'task1'
```

- Kiểm tra các branch và bạn đang ở đâu?

$ **git branch –a**

```
Admin@Admin MINGW64 /d/T
$ git branch -a
  master
* task1
```

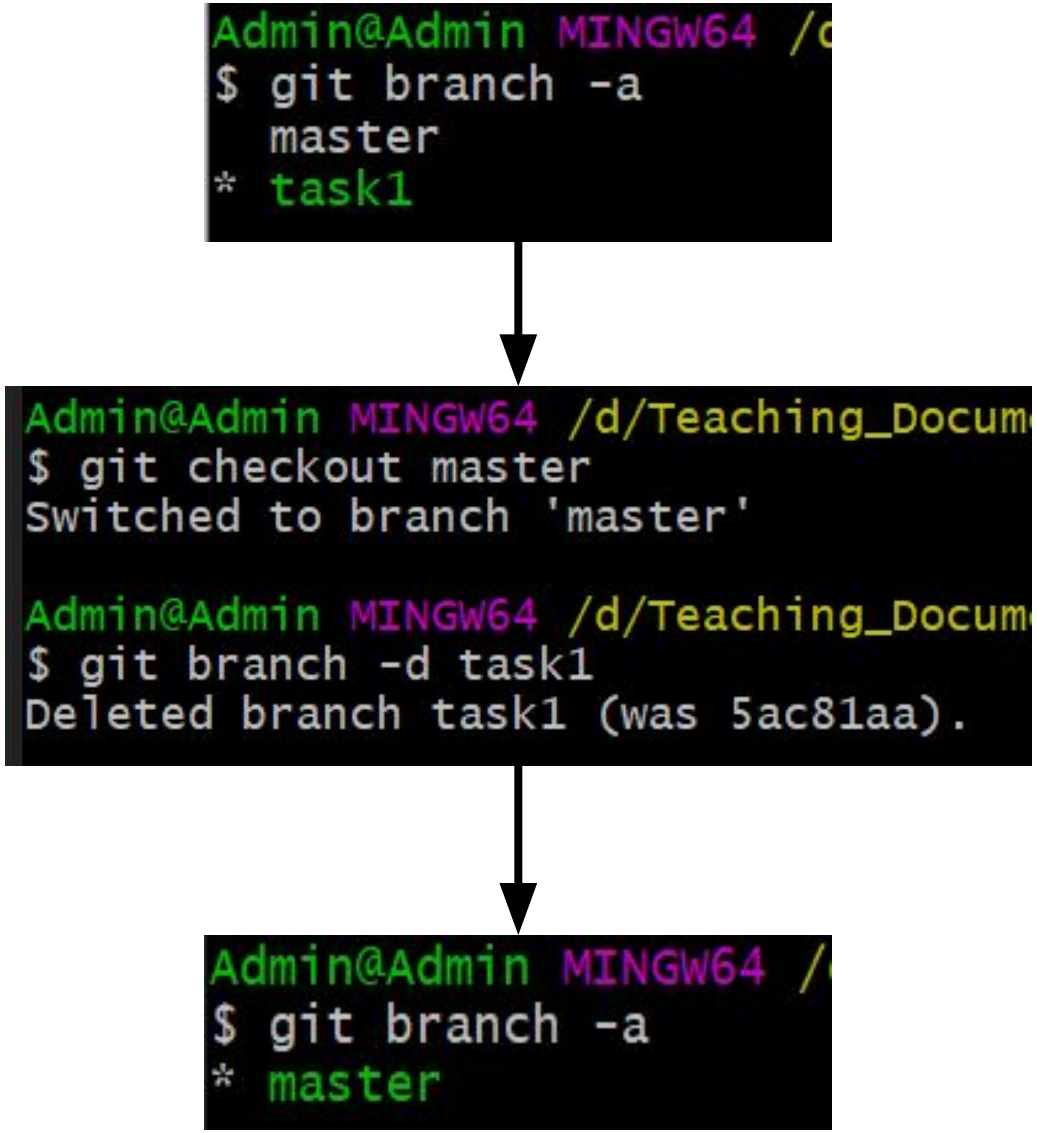
$ **git branch –v**

```
Admin@Admin MINGW64 /d/Teaching
$ git branch -v
  master 5ac81aa update LED.c
* task1  5ac81aa update LED.c
```

Xóa Branch

- Checkout đến một Branch khác và xóa branch cần xóa

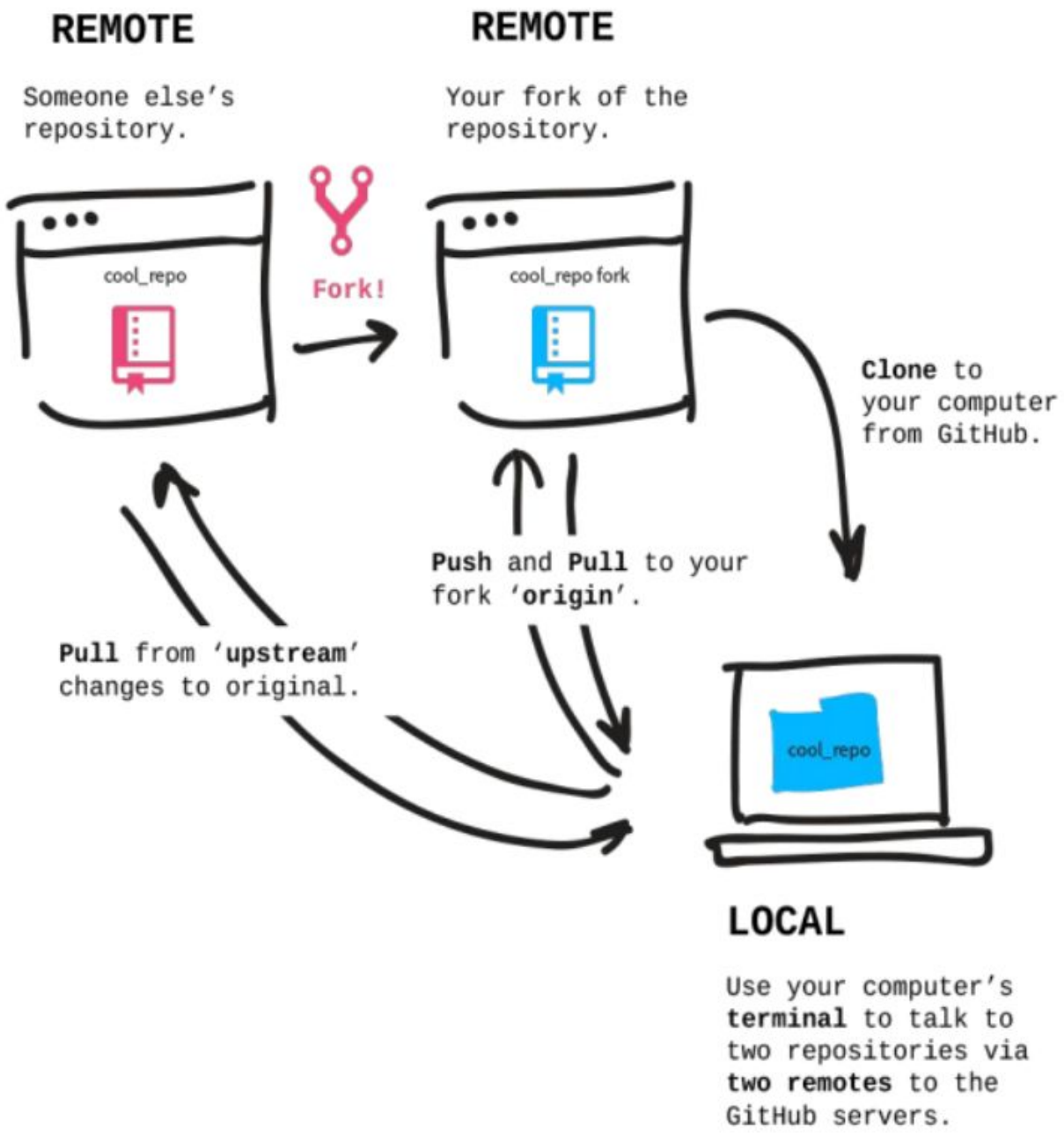
$ **git branch -d** <branch name>



Push Branch

- Đẩy Branch Local lên Remote Repository

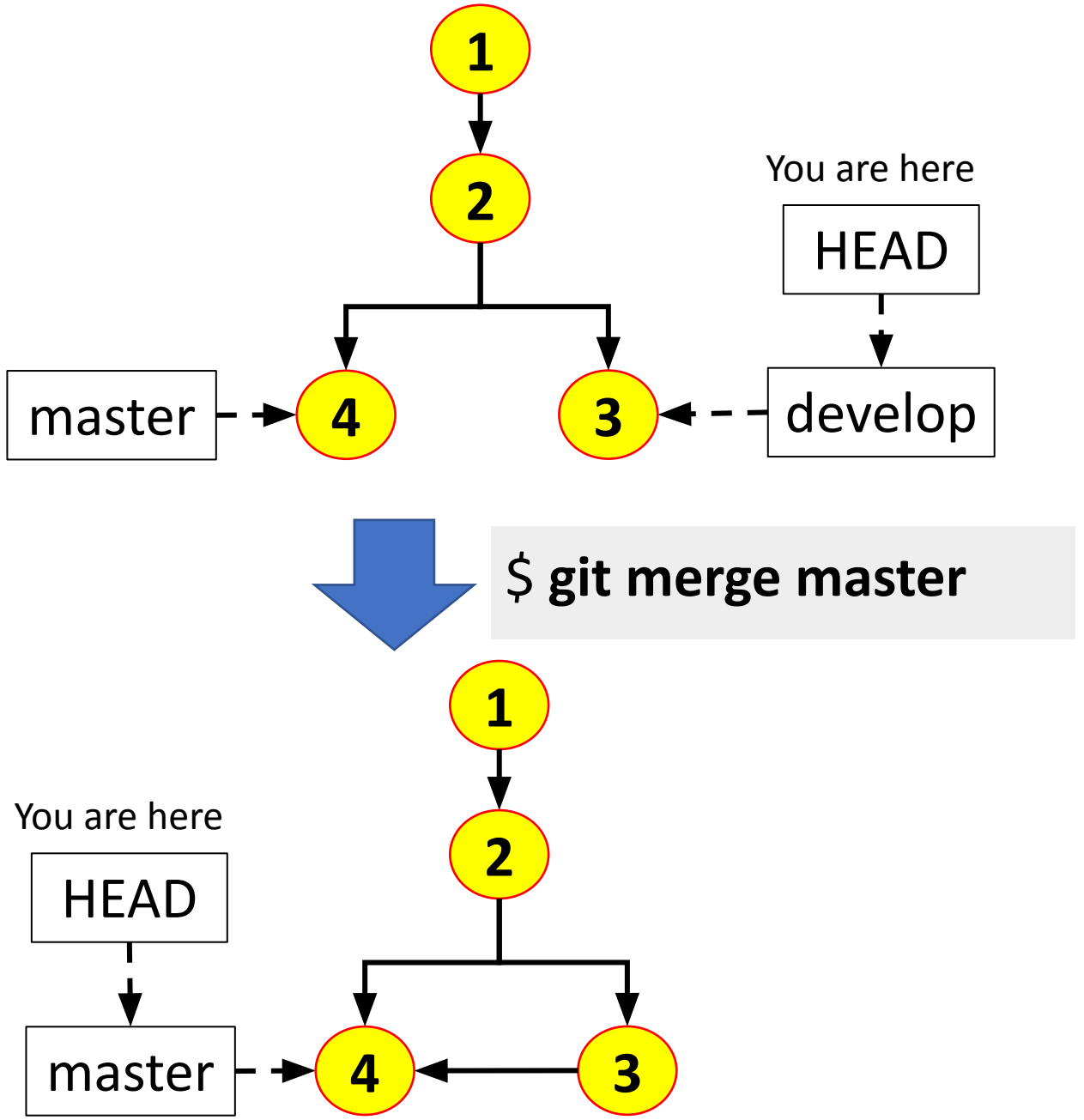
```
$ git push --set-upstream
<remote> <branch name>
```



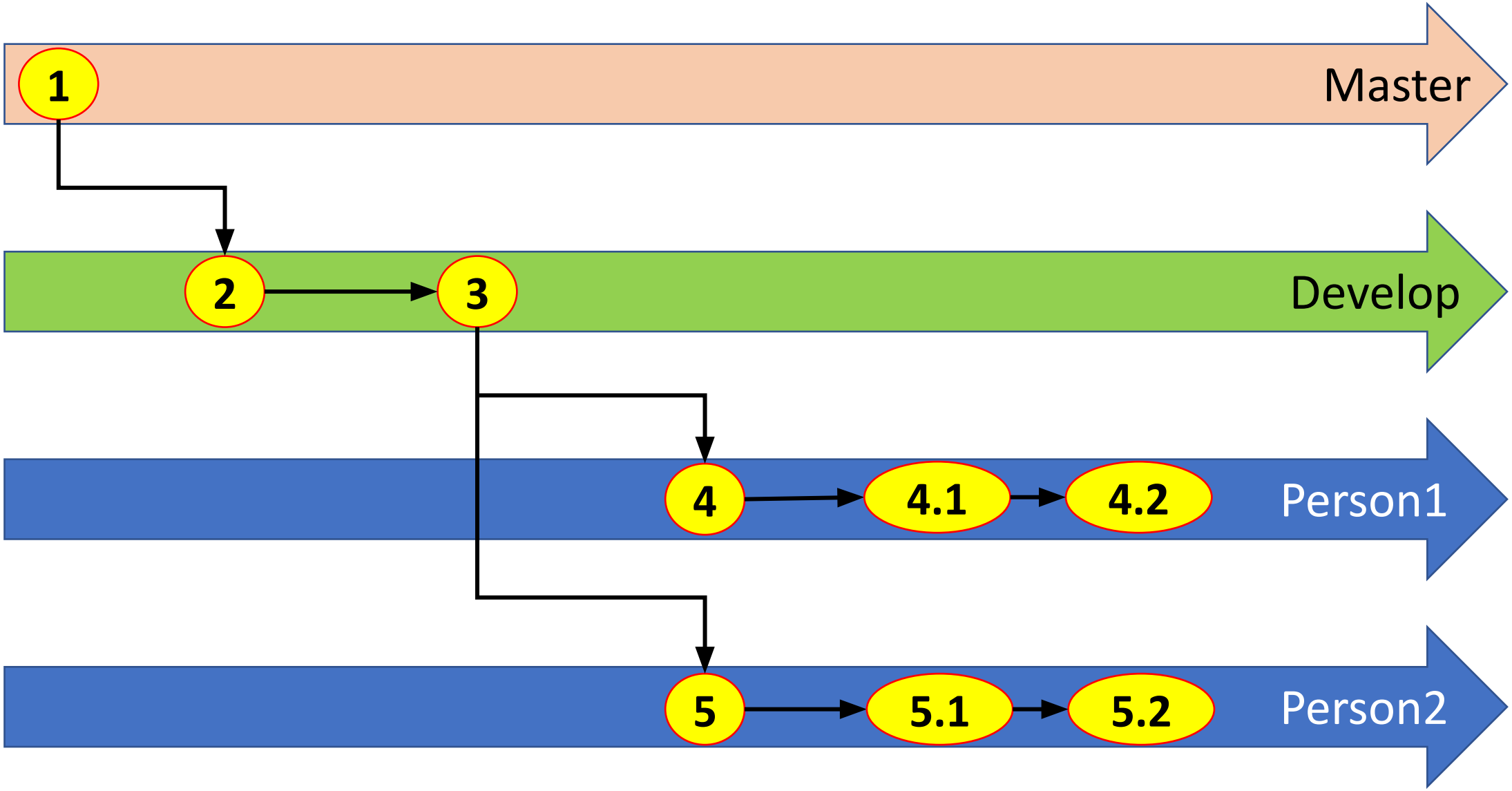
Merge Branch

- Khi hoàn thành các nhánh của dự án, chúng ta cần ghép các nhánh lại nhánh ban đầu

```
$ git merge <branch name>
```

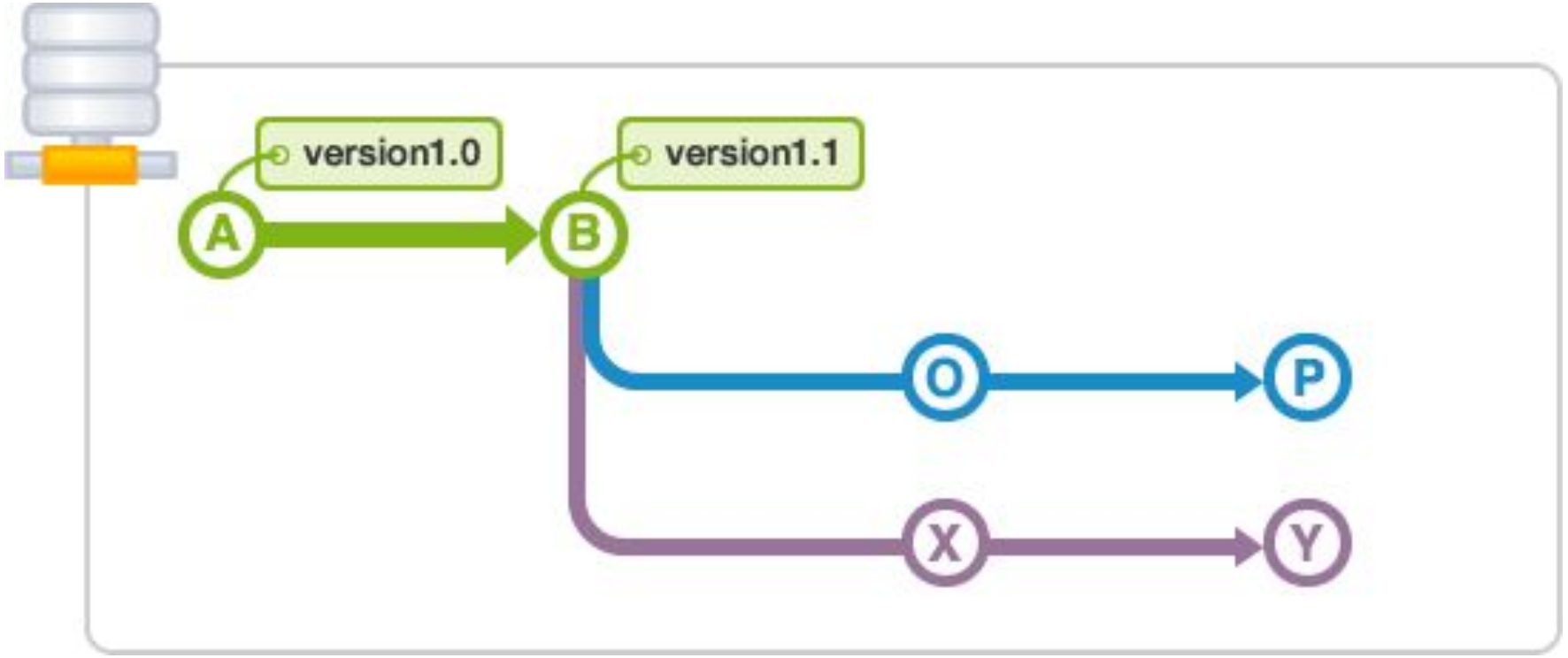


Branch Workflow



Tagging

Việc đánh tag cho giúp lưu lại lịch sử các phiên bản



Tagging

- Liệt kê các tag đang có

```
$ git tag
$ git tag --list
```

- Đánh Tag cho node của bạn

```
$ git tag –a <tag name> -m “message”
```

```
$ git push <remote name> --tags
```

- Checkout đến Tags

```
$ git checkout tags/<tag name> -b <branch>
```

Tagging

- Fetch Tag trên Server

```
$ git fetch --all --tags
```

- Kiểm tra trạng thái của branch

```
$ git log --oneline --graph
```

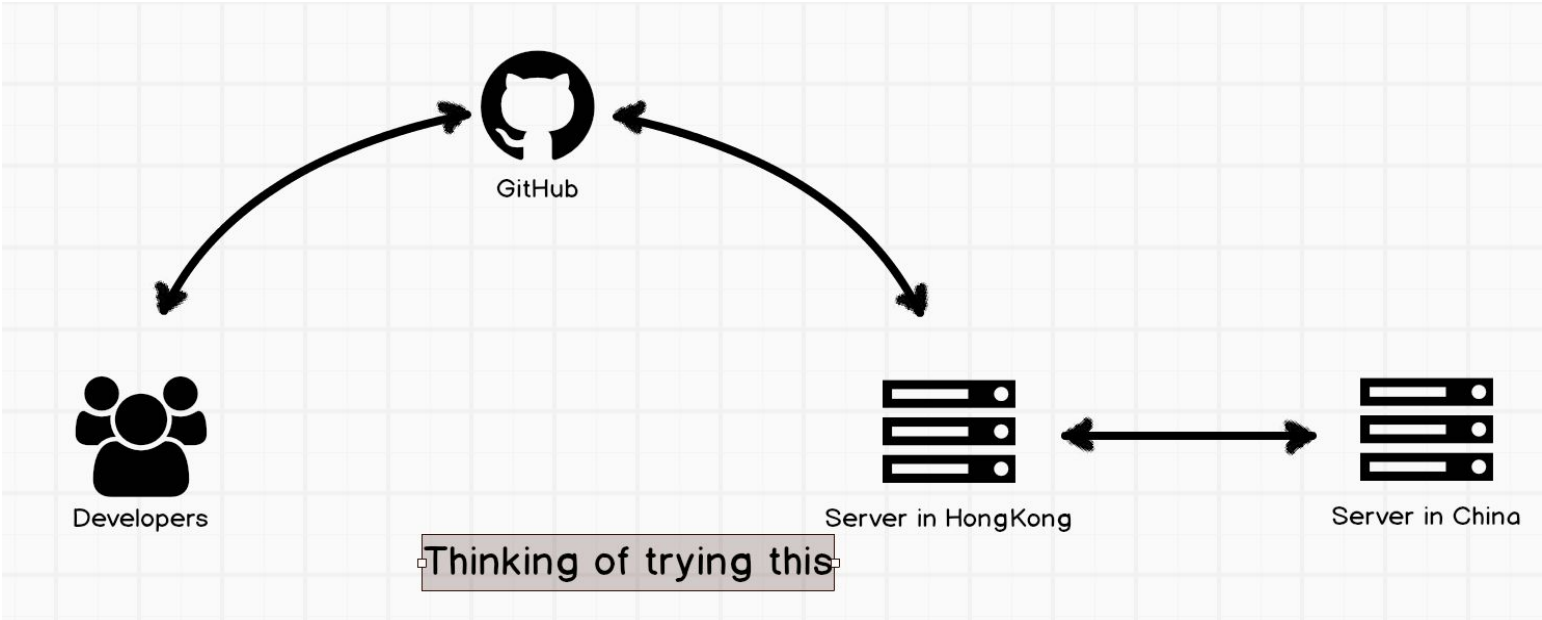
- Checkout đến Tags mới nhất

```
$ git checkout $tag -b latest
```

4. Quản lý dự án với Git–Github

Github là gì?

- GitHub là một dịch vụ lưu trữ trên web dành cho các dự án có sử dụng hệ thống kiểm soát Git revision.



Tạo tài khoản Github

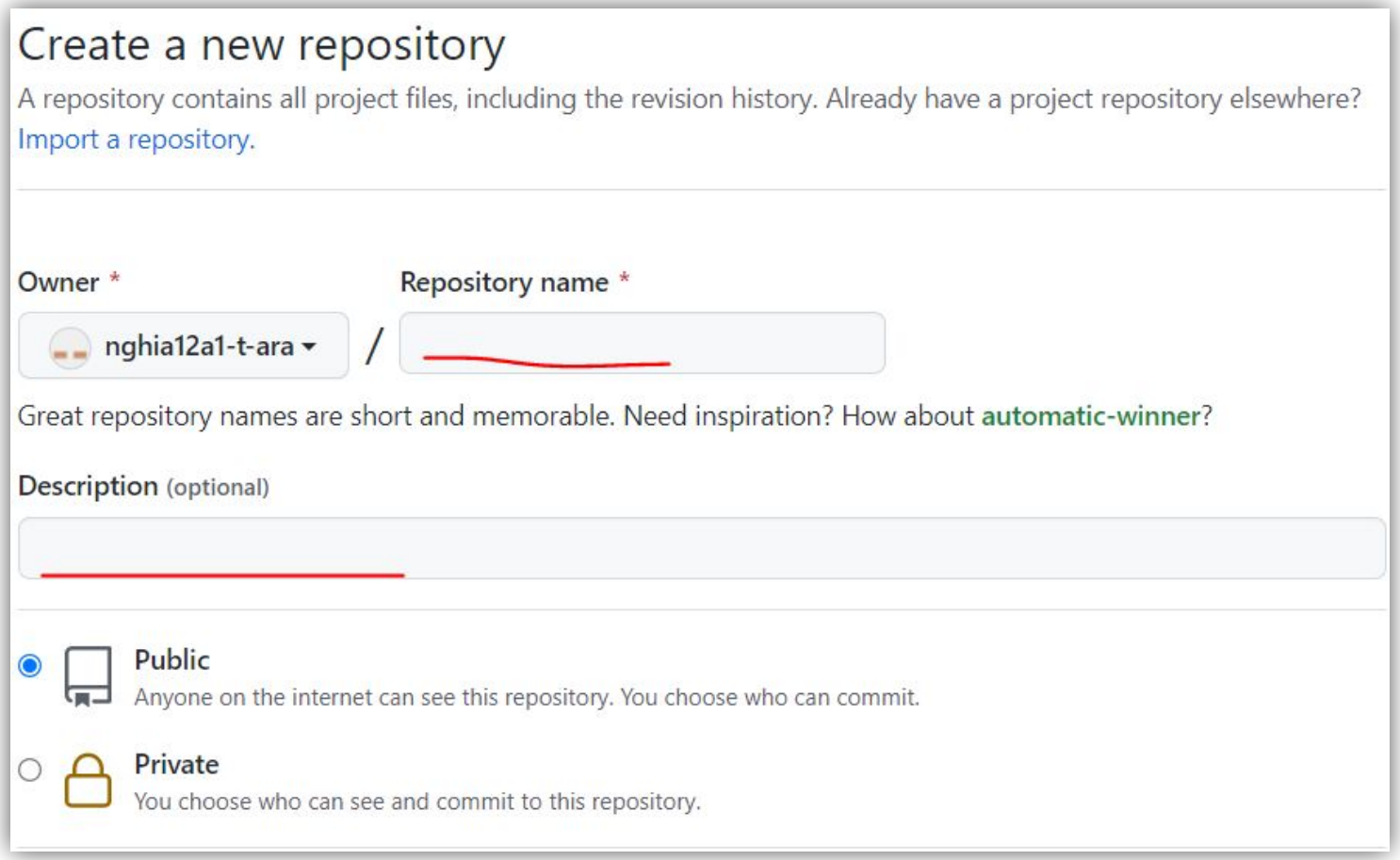
github.com

- GitHub là một dịch vụ lưu trữ trên web dành cho các dự án có sử dụng hệ thống kiểm soát Git revision.

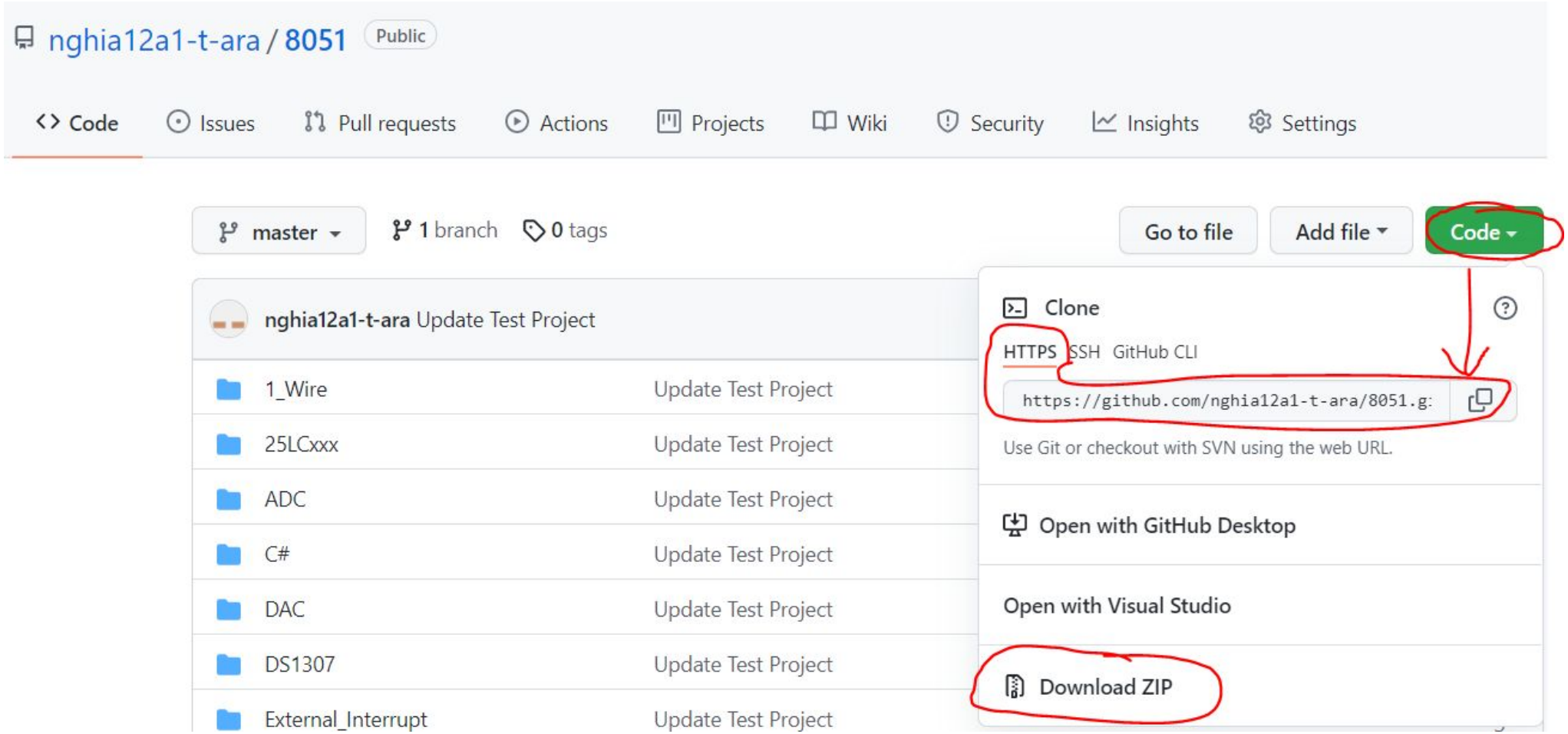
Tạo một Remote Repo

- Tạo một Server (Remote Repo) để lưu trữ Project

Vào tùy chọn chọn Your Repositories
 Chọn New



Clone & Download từ Github



Thực hành với Github

- Tạo một Remote Repo trên Github
- Clone Repo
- Thay đổi Code
- Đẩy Code lên Remote Repo

5. Git trong dự án thực tế?

NOT Github

Các công ty lớn cấm sử dụng Github vì vấn đề bảo mật

BitBucket

- Server lưu trữ giống như Github
- Trello Boards
- Tích hợp Jira
- Tính bảo mật tốt hơn Github

BitBucket vs Github

- Mặc dù có tính năng bảo mật của BitBucket tốt hơn, nhưng số lượng lập trình viên sử dụng Github ở Việt Nam vẫn lên đến hơn 70%

Tính năng	Bitbucket	GitHub
VCS được hỗ trợ	Mercurial, Git	Git
Public repositories	Miễn phí, không giới hạn số lượng	Miễn phí, không giới hạn số lượng
Private repositories	Miễn phí cho nhóm 5 người trở xuống	Từ $7/ tháng, không giới hạn người dùng
Tích hợp	Jira, Crucible, Jenkins, Bamboo	Asana, Zendesk, CloudBees, Travis, CodeClimate, AWS, Windows Azure, Google Cloud, and Heroku
Host lưu trữ dự án phổ biến	Adium, Mailchimp, Opera, Python, Django	Bootstrap, Node,js, jQuery, Rails, Homebrew
Tính năng mở rộng nổi bật	Spoon, Jira integration, External authentication via Github, Twitter, Facebook, Google	Xác nhận 2 lớp, Github Pages, Github Gists

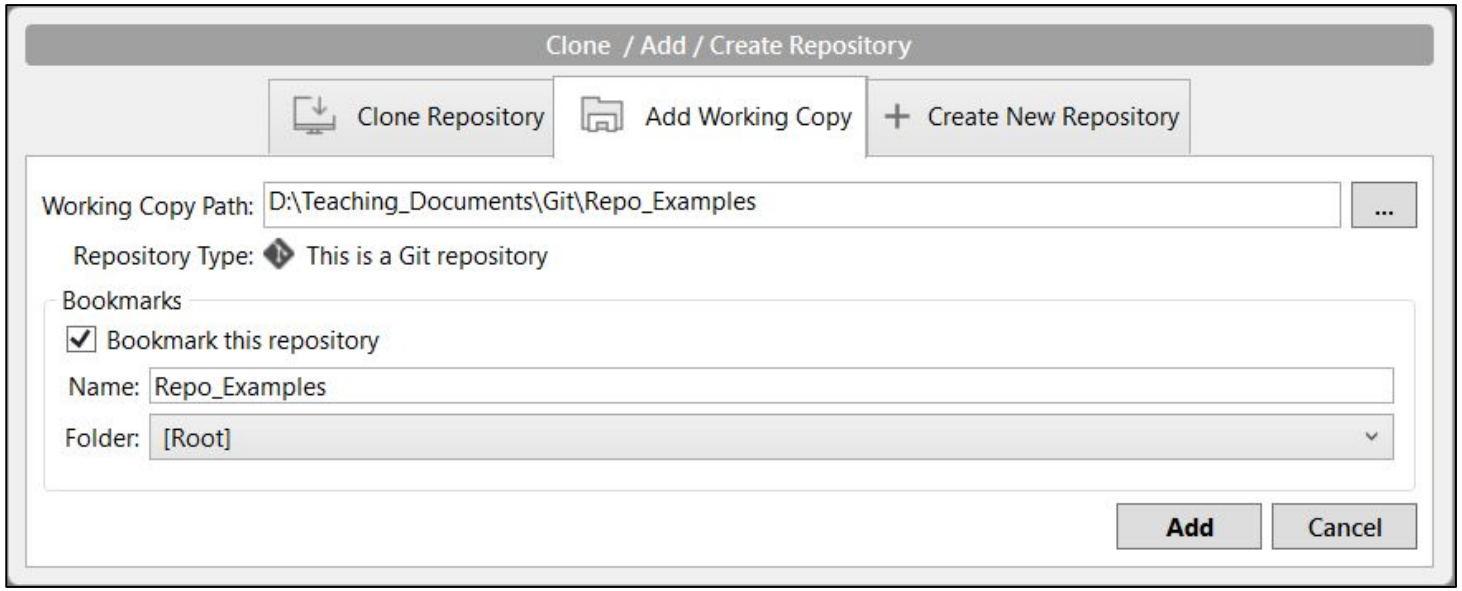
Tool for Git – Source Tree

- Tự động các công việc thay vì sử dụng Command
- Giao diện dễ quan sát & Nắm được hệ thống
- Hữu dụng cho người mới học

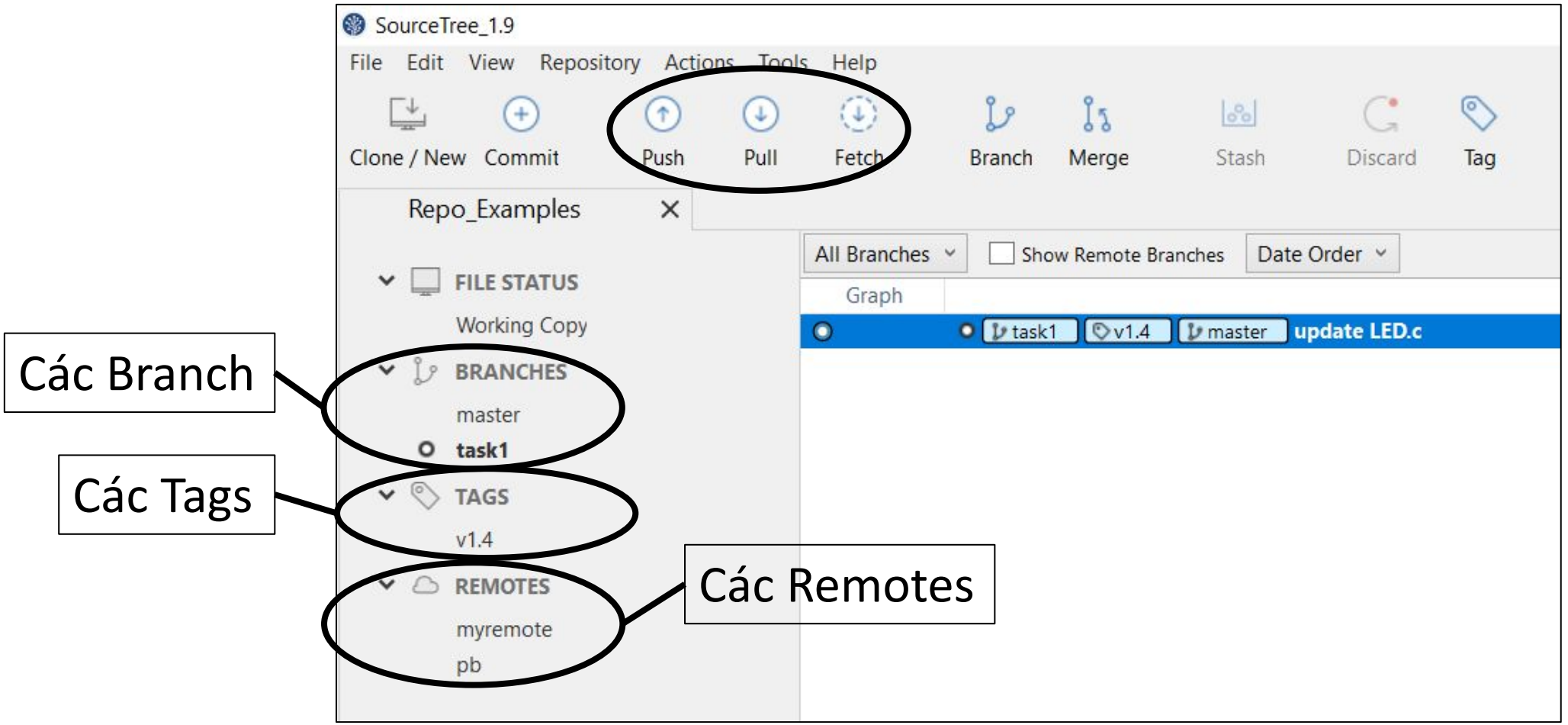
bitbucket.org

Add một Repository

- File Clone/New Chọn đến đường dẫn của Repo (chứa file .git) Chọn **Add**



Giao diện Source Tree



Thực hành với Source Tree

- Tạo Branch, ….
- Thao tác trên Branch
- Staged Committ Push
- Tag
- Merge