


# Giới thiệu git cơ bản

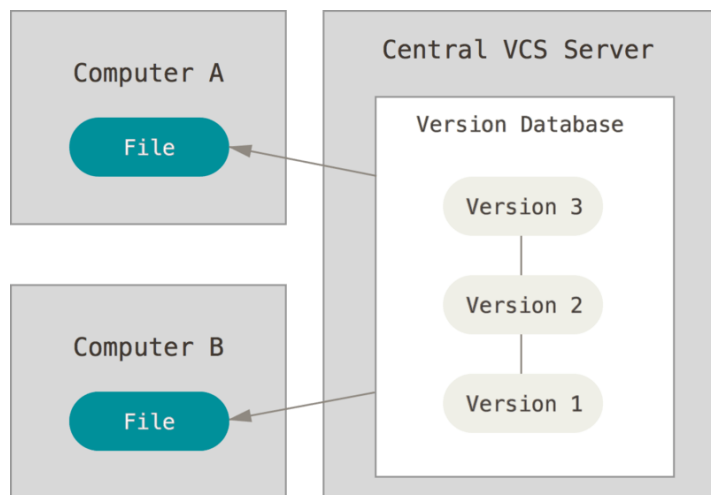
Nguyễn Đức Hà – D1

- 
- Khái niệm quản lý version và git
  - Các lệnh git cơ bản
  - Một số bài toán thường gặp
  - Git convention và work-flow

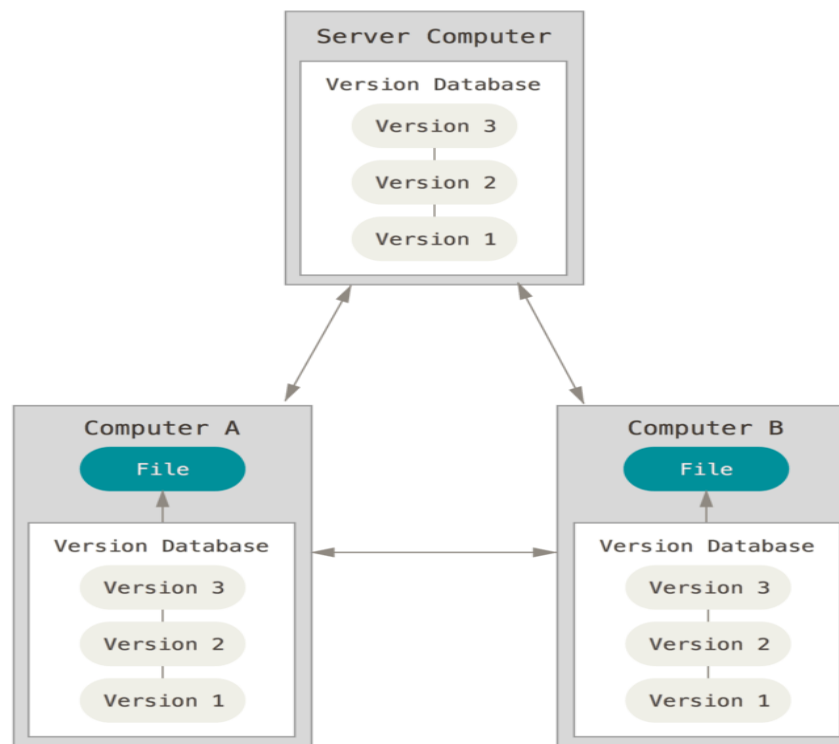


# Khái niệm quản lý version và git

Centralized version control system

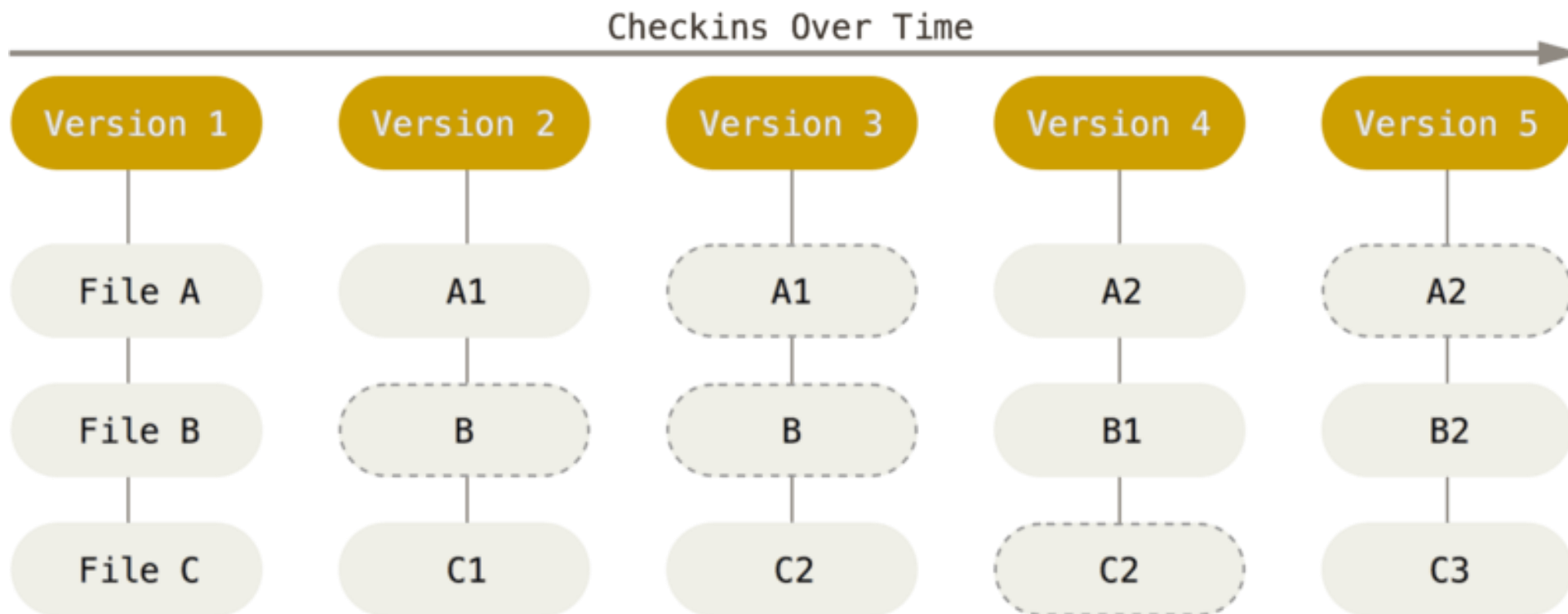


Distributed version control system



# Git là gì?

- Quản lý phiên bản theo dạng snapshot



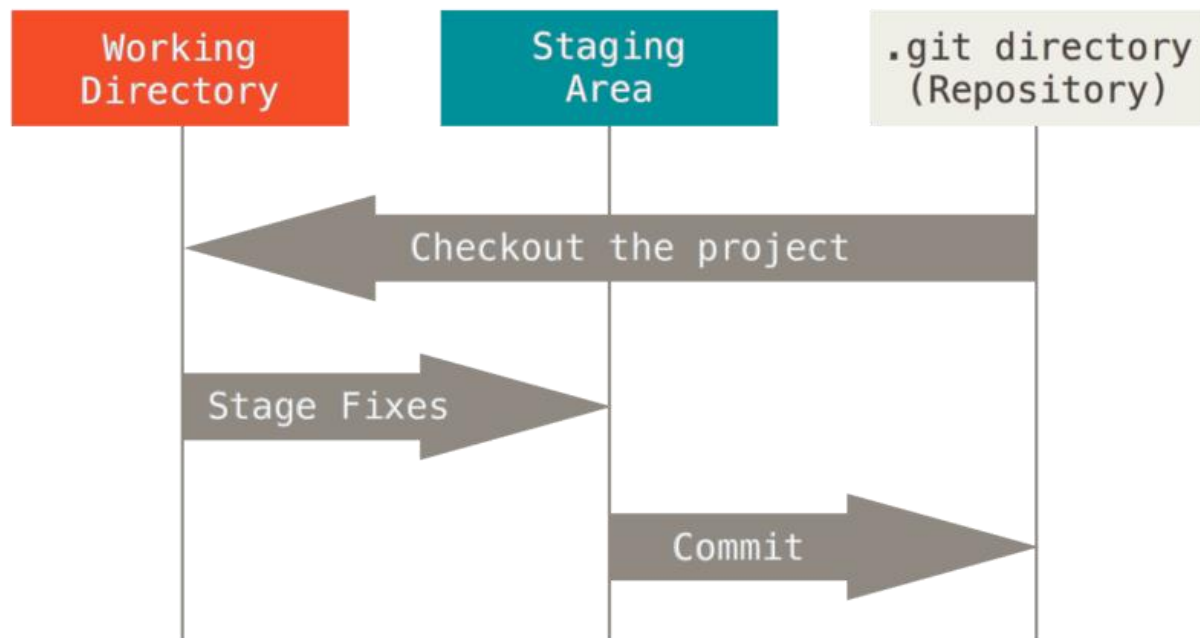
# Git là gì?

- Hầu hết thao tác đều có thể làm ở local
- Git có tính toàn vẹn: mỗi commit có mã hash. Đã commit thì gần như không thể bị mất code.



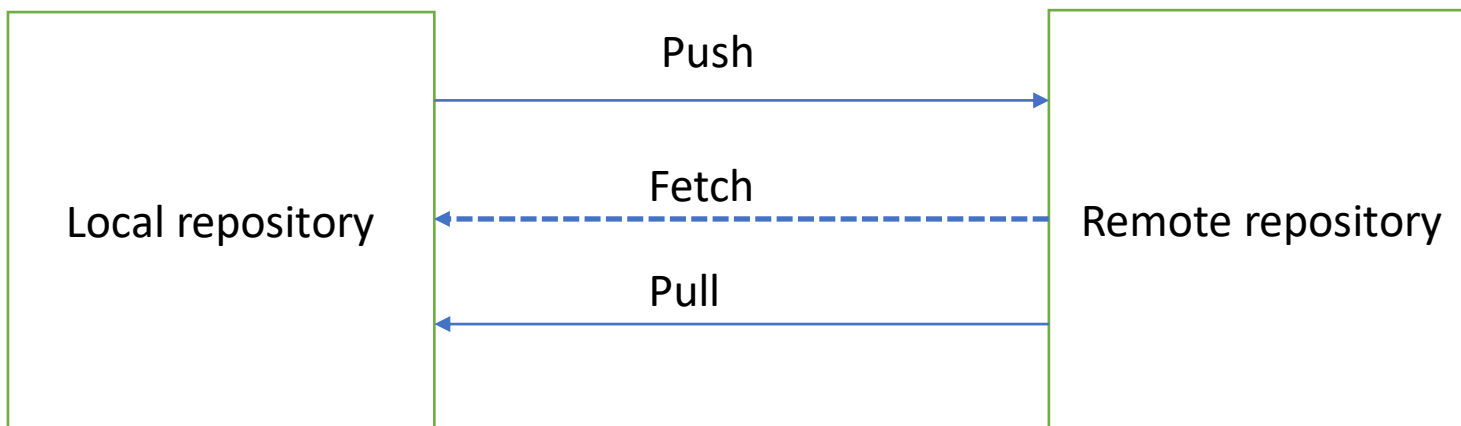
# Git là gì?

- 3 section chính của git project



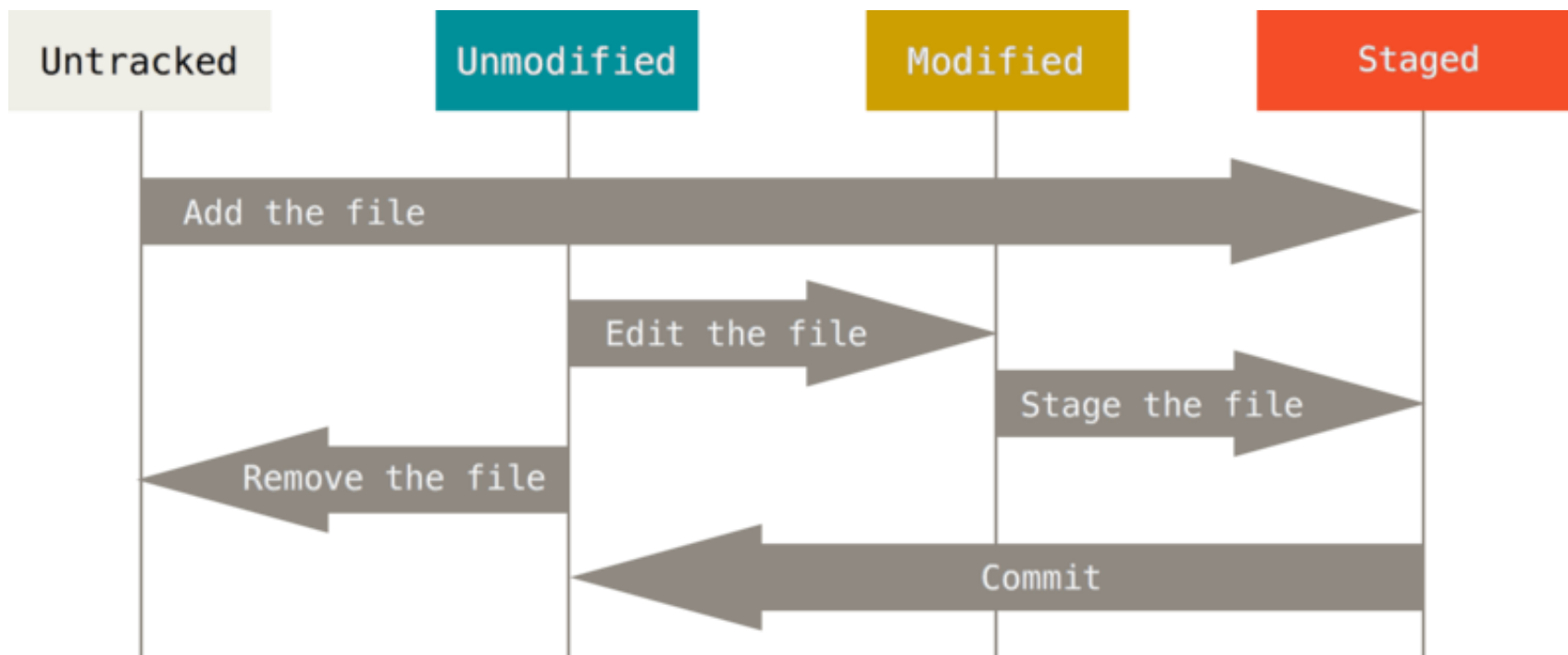
# Repository

- Local repository và remote repository
- Chứa tất cả dữ liệu và lịch sử commit



# Repository

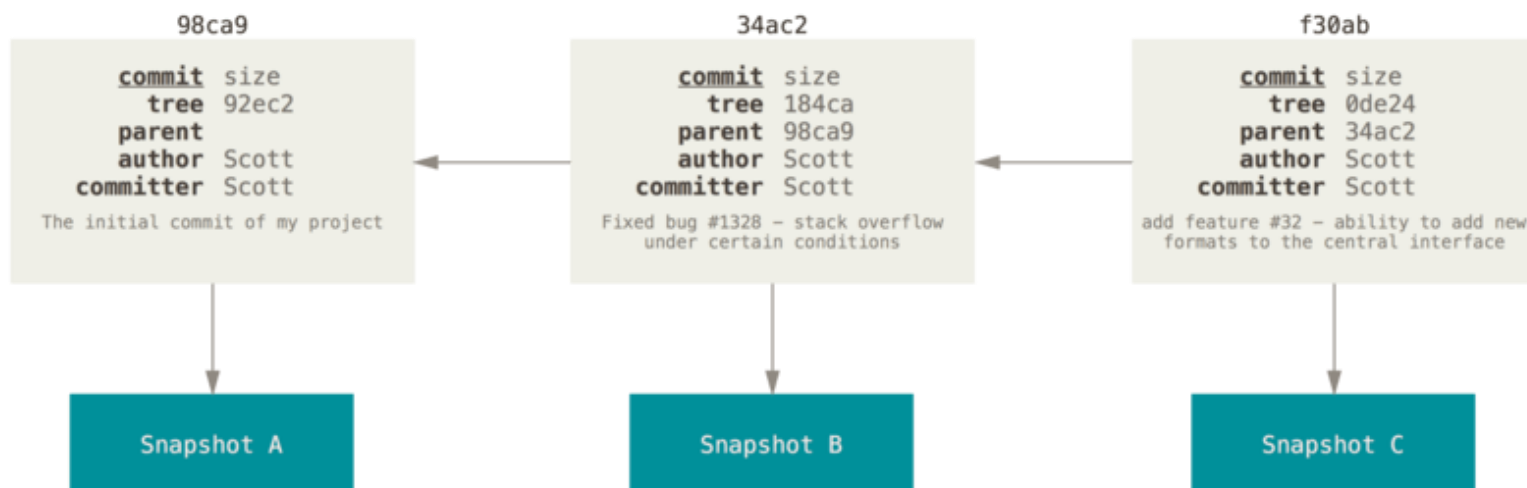
- Vòng đời của file trong repository





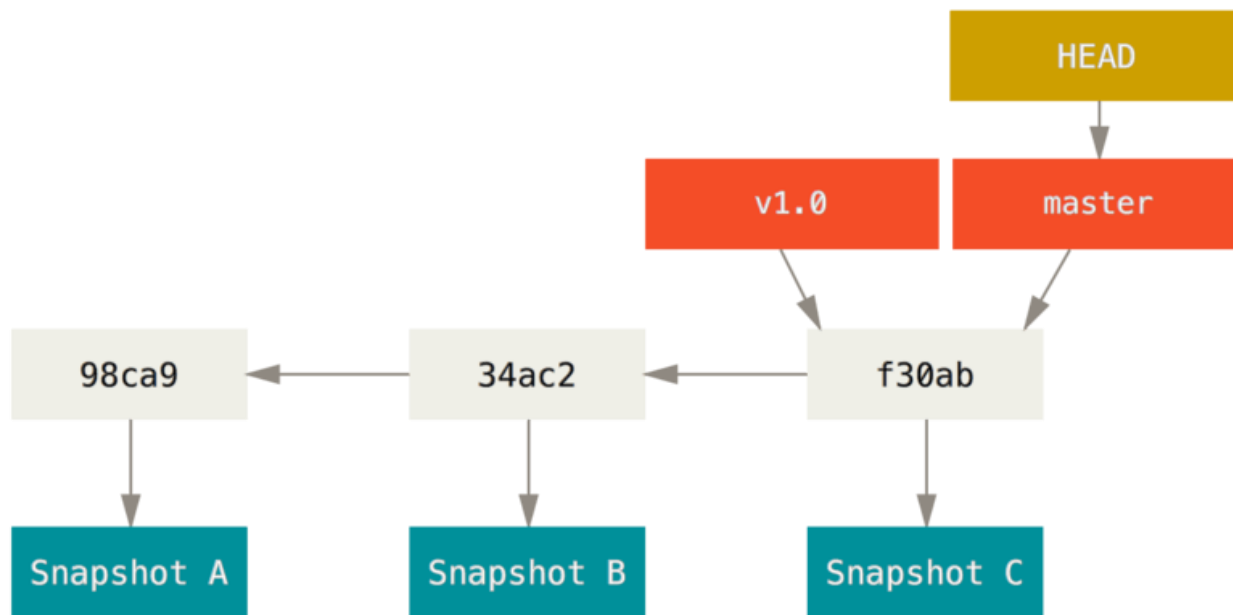
# Branch

- Lịch sử git là một chuỗi các commit, commit sau trở vào commit trước, mỗi commit lại trở đến snapshot tương ứng.



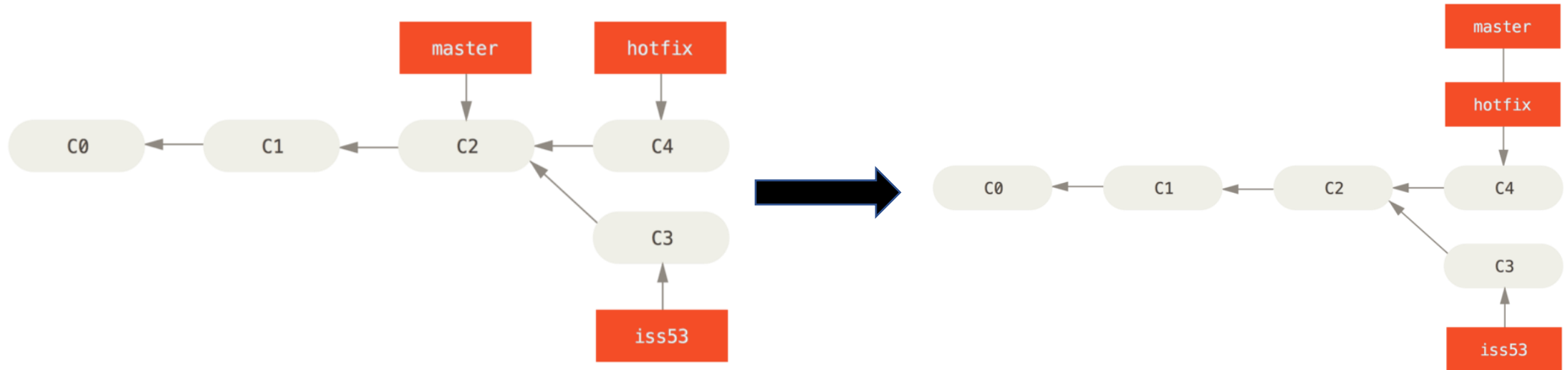
# Branch

- Branch là một con trỏ, trỏ đến một trong những commit này.
- HEAD là con trỏ đặc biệt, trỏ đến branch local hiện tại



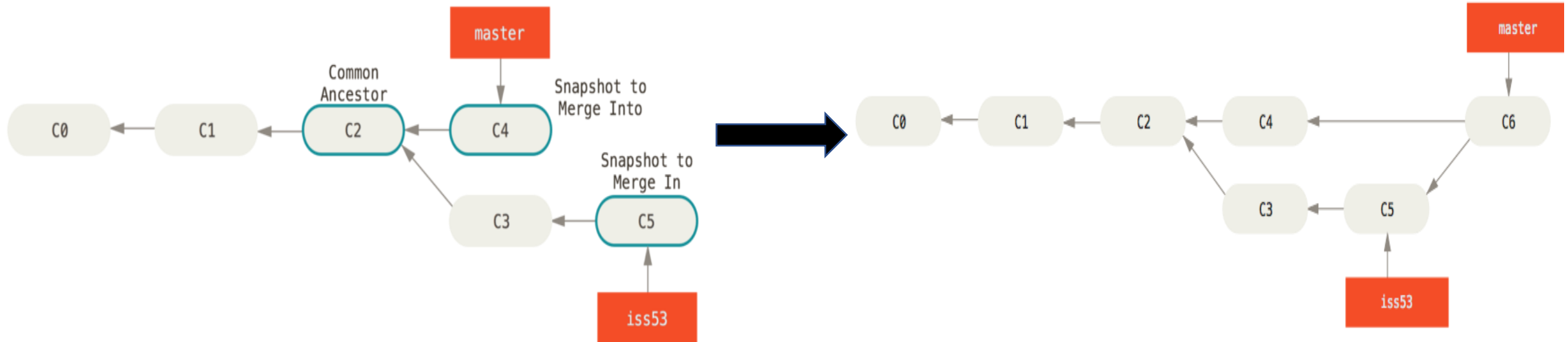
# Branch

- Merge
  - Fast-forward



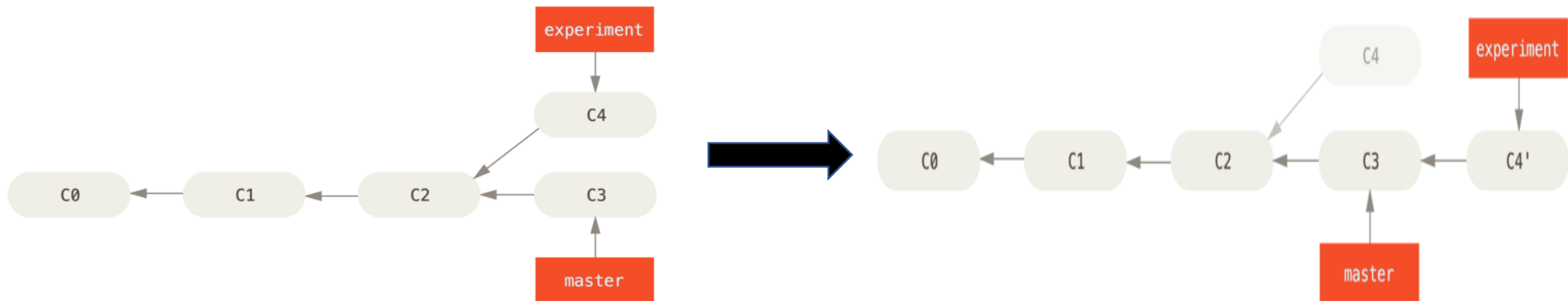
# Branch

- Merge
  - Recursive



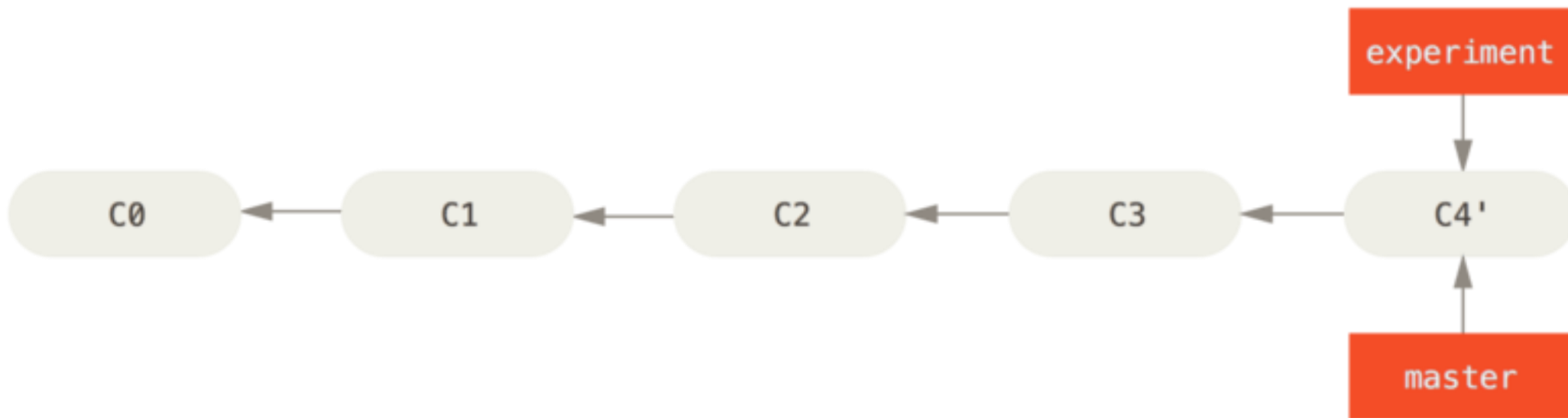
# Branch

- Rebase
  - Rebase *experiment* onto *master*



# Branch

- Rebase
  - Sau khi rebase xong thì merge fast-forward từ experiment vào master



# Các lệnh git cơ bản

- Khởi tạo git

git init

```
→ git_demo git init
Initialized empty Git repository in /Users/hand/Desktop/git_demo/.git/
→ git_demo git:(master)
```

- Khởi tạo git từ remote

git clone <url>

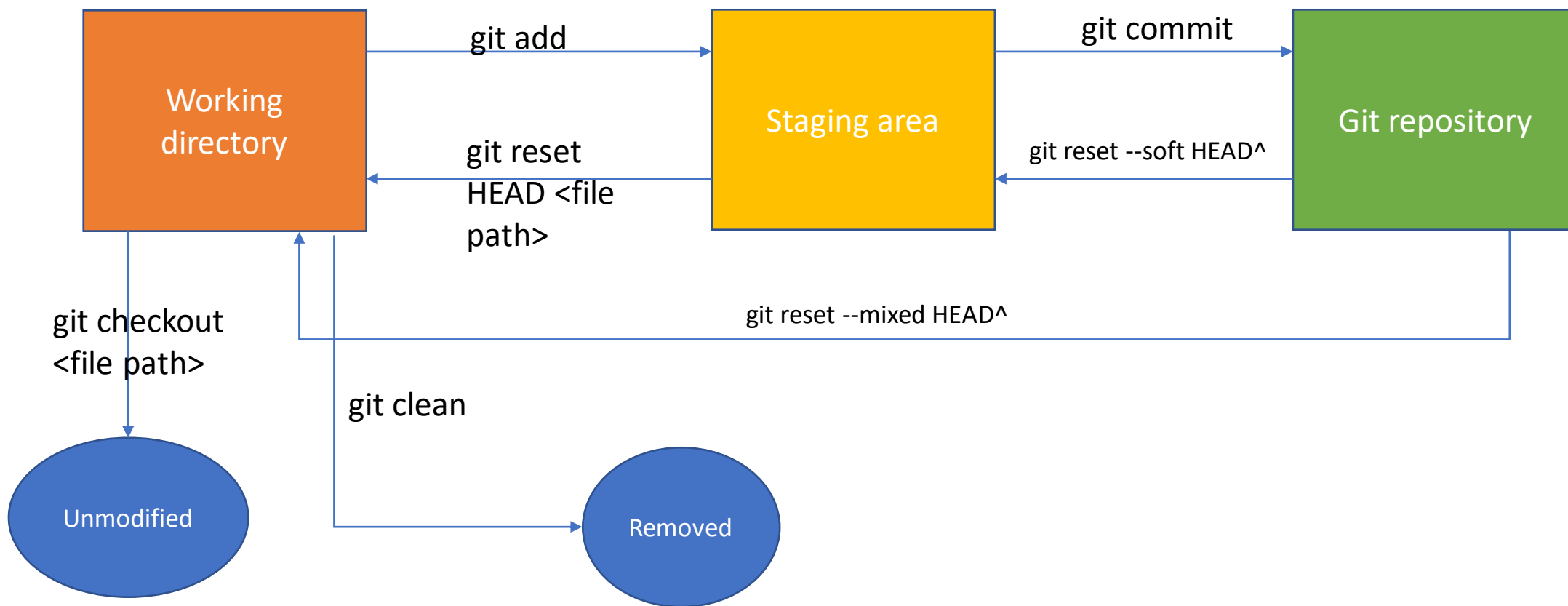
```
→ Desktop git clone https://hand@git.rikkei.org/d1/18081d1_zoo.git

Cloning into '18081d1_zoo'...
Password for 'https://hand@git.rikkei.org':
remote: Enumerating objects: 11320, done.
remote: Counting objects: 100% (11320/11320), done.
remote: Compressing objects: 100% (4170/4170), done.
Receiving objects: 52% (5922/11320), 828.00 KiB | 57.00 KiB/s
```



# Các lệnh git cơ bản

- Thao tác với repository





# Các lệnh git cơ bản

- Thao tác xem trạng thái
  - git status: Hiển thị trạng thái hiện tại của git ở cả 3 sections
  - git diff <file path>: Hiển thị những thay đổi của file được chỉ định ở Working directory
  - git diff <file path> --cached: Hiển thị những thay đổi của file được chỉ định ở Staging area
  - git show <mã commit>: Hiển thị những thay đổi của file ở commit được chỉ định

# Các lệnh git cơ bản

- Thao tác với remote
  - Git pull: Lấy dữ liệu mới nhất từ remote và tích hợp vào local repository  
git pull <remote>
  - Git push: đẩy dữ liệu mới nhất ở local lên remote  
git push <remote>
  - Git fetch: lấy dữ liệu mới nhất từ remote về, không tích hợp vào local repository

Git pull



Git fetch



Integrate



A decorative pattern in the top-left corner consisting of several overlapping triangles in red and white, some with internal geometric details.

# Các lệnh git cơ bản

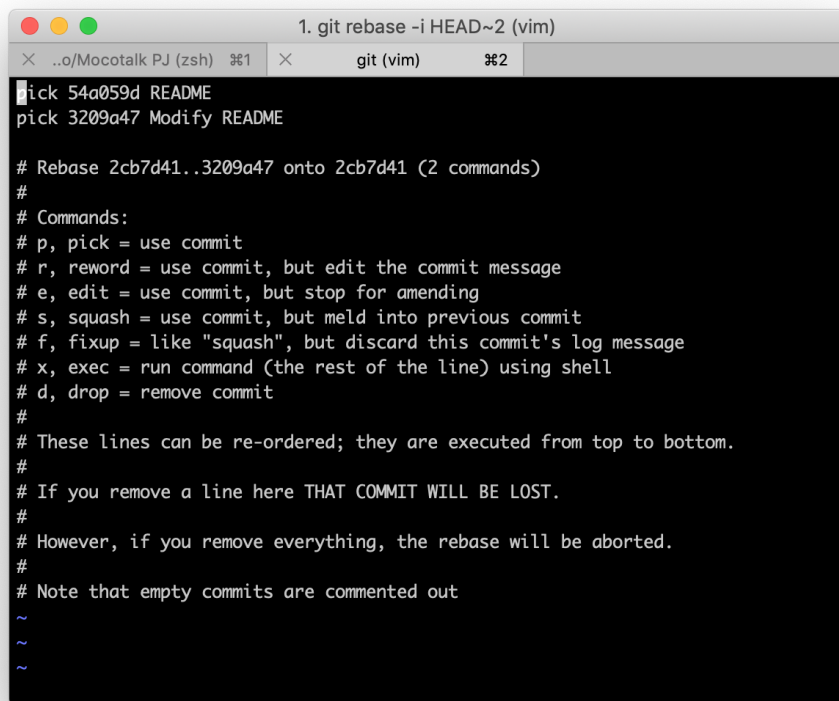
- Thao tác với branch
  - git merge
  - git rebase
  - git branch -d <branch name>
  - git checkout -b <branch name>
  - git checkout <branch name>



# Các lệnh git cơ bản

- Rebase interactive

git rebase -i HEAD~n: sửa lịch sử n commit so với HEAD



```
1. git rebase -i HEAD~2 (vim)
pick 54a059d README
pick 3209a47 Modify README

# Rebase 2cb7d41..3209a47 onto 2cb7d41 (2 commands)
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
~
~
~
```

# Một số bài toán thường gặp

- Đổi tên author, email:

git config user.name <user name>

git config user.email <user email>

```
→ git_demo git:(master) git config user.email hand@rikkeisoft.com
→ git_demo git:(master) git config user.name hand
→ git_demo git:(master) git config user.email
hand@rikkeisoft.com
→ git_demo git:(master) git config user.name
hand
→ git_demo git:(master) █
```





# Một số bài toán thường gặp

- Sửa commit mà HEAD đang trỏ đến  
git commit –amend  
git commit --amend --reset-author
- Sửa các commit cũ hơn  
Dùng git rebase interactive như bên trên đã nói





# Một số bài toán thường gặp

- Lấy commit từ nhánh này sang nhánh kia  
git cherry-pick <commit>  
git cherry-pick <before first commit>..<last commit>



# Một số bài toán thường gặp

- Lưu trữ code để chuyển branch

git stash save <message>

git stash list

git stash pop stash@{x}

git stash apply

git stash drop

→ git\_demo git:(new\_branch\_3) X git stash save "Modify README2.md"

Saved working directory and index state On new\_branch\_3: Modify README2.md

*stash@{0}: On new\_branch\_3: Modify README2.md*





# Một số bài toán thường gặp

- Phục hồi commit đã bị reset --hard
  - Dùng reflog để lấy hash của commit trước khi reset.
  - Dùng git reset --hard <commit hash vừa lấy được>

→ *git\_demo git:(new\_branch\_2) git reset --hard HEAD~1*

*HEAD is now at 88f76db Modify README*

→ *git\_demo git:(new\_branch\_2) git reflog*

*88f76db (HEAD -> new\_branch\_2) HEAD@{0}: reset: moving to HEAD~1*

*eebad78 HEAD@{1}: checkout: moving from new\_branch\_3 to new\_branch\_2*

→ *git\_demo git:(new\_branch\_2) git reset --hard eebad78*

*HEAD is now at eebad78 Modify Readme.md second time*



# Một số bài toán thường gặp

- Khôi phục branch đã xóa
  - Dùng reflog lấy hash của commit trước khi xóa branch.
  - Dùng *git branch <tên branch> <commit>* để khôi phục lại branch

→ *git\_demo git:(master) git reflog*

*2b27cee (HEAD -> master) HEAD@{0}: checkout: moving from new\_branch\_2 to master*

*eebad78 HEAD@{1}: reset: moving to eebad78*

*88f76db HEAD@{2}: reset: moving to HEAD~1*

→ *git\_demo git:(master) git branch new\_branch eebad78*





# Một số bài toán thường gặp

- Sửa lỗi line ending (thường xảy ra khi code trên Windows và deploy lên linux)
- Cách sửa: dùng git config (chỉ config cho từng repository):
  - *git config core.safecrlf true*
  - *git config core.eol lf*
  - *git config core.autocrlf input*



# Git convention

- Đặt tên branch

**Format: [prefix]/[issue\_no\_][short\_description]**

[prefix]: feature, bugfix, release, hotfix

[issue\_no\_]: số hiệu issue nếu task đang làm được quản lý bằng issue (trên git, redmine, backlog...). Có thể bỏ qua nếu task đang làm không được quản lý bằng issue hoặc branch định làm sẽ chứa nhiều issue.

Short\_description: mô tả ngắn gọn task đang làm

*Ví dụ: feature/#123\_add\_user\_authentication*



# Git convention

- Commit

- Add đầy đủ các file/đoạn code liên quan đến commit, không add những phần không liên quan
- Tránh add quá nhiều file trong 1 commit gây khó khăn cho việc review và quản lý source code.  
Số file tối đa tham khảo: 5
- Commit message phải rõ ràng, phải chứa “issue #...” nếu có issue tương ứng.  
*Ví dụ: “issue #124 validate email format, validate password max length, add term of use to register screen”*





# Git convention

- Merge request
  - Title của Merge Request: tương tự commit message. Chú ý điền issue liên quan nếu có.
  - Nếu Merge Request chưa hoàn thành, cần sửa thêm, chưa sẵn sàng để merge:  
thêm prefix “**WIP:**” (Work In Progress) vào title của Merge Request để tránh bị merge nhầm.
  - Chọn remove source branch khi tạo merge





# Git convention

- Merge and rebase
  - Hạn chế dùng merge giữa các branch.
  - Không merge ngược từ nhánh chính (develop, master) vào nhánh feature, fixbug
  - Dùng rebase nhiều nhất có thể.
  - Không dùng rebase với nhánh chính hoặc nhánh có nhiều người làm chung.





# Git work-flow

- Centralized workflow
- Feature branch workflow
- Gitflow workflow
- Forking workflow







# Git work-flow

- Feature branch workflow

- Mỗi chức năng là 1 branch riêng, được checkout từ nhánh chính.
- Dev làm việc trên nhánh feature, không ảnh hưởng tới code ở nhánh chính.
- Tạo pull request, merge request để review code, thảo luận trước khi merge.
- Xóa nhánh feature vừa được merged.



# Git work-flow

- Gitflow workflow
  - Gồm các loại branch chính: master, develop, feature, release, hot\_fix
  - Mỗi loại branch được quy định một nhiệm vụ riêng



# Git work-flow

- Gitflow workflow

- Các bước thực hiện

1. Branch master được tạo mặc định khi khởi tạo repo.
2. Tạo branch develop từ master (branch develop và master sẽ không bị xóa).
3. Tạo branch feature từ develop.
4. Hoàn thành feature -> merge vào develop.
5. Hoàn thành tất cả feature -> tạo branch release từ develop.
6. Hoàn thành release -> merge vào develop và master. Gắn tag vào master.
7. Nếu có bug ở master -> tạo branch hotfix từ master -> merge vào develop và master sau khi hoàn thành.



# Tham khảo

- <https://git-scm.com/book/en/v2>
- <https://github.com/rikkeisoft/git-training>
- <https://www.atlassian.com/git/tutorials>
- <https://gist.github.com/jedmao/5053440>





Thank you

