



Pay It Forward

GIT

After-C

Table of contents – init stuff

- Install GIT, create Github account
- Version control system
- What is GIT? Why, Where and When use GIT ?
- Working principle GIT and File lifecycle
- What is Github ?
- Github alternative
- GIT/Github FLOW
- Basic command line and github
- SSH

Table of contents – main stuff

- DEMO PROJECT
- CREATE REPO
- SYNCHRONIZE
- MERGE
- UNDO
- IGNORE
- GIT PLUS
- HOW TO GIT PROFESSIONAL
- FAQ
- APPENDIX

Install GIT

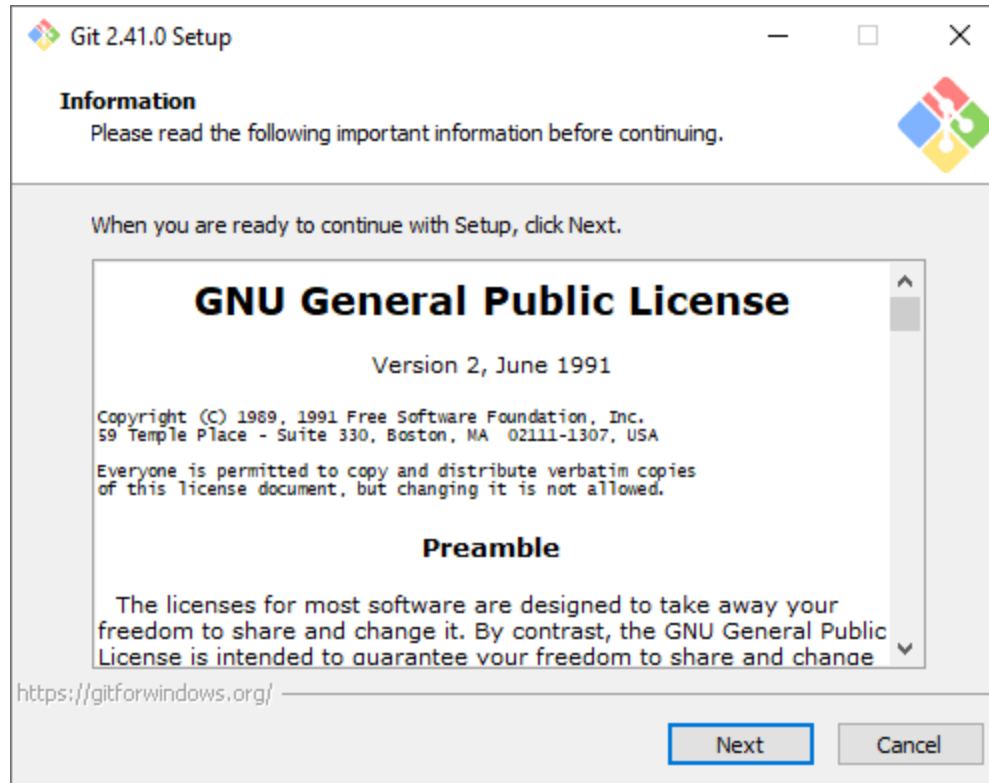
Create Github account

Install GIT

- Chỉ cần cài đối với Window, không cần với ubuntu, macOS thì không biết.
- Cài thêm GitKraken – cũng là Git nhưng sử dụng GUI, mô hình trực quan hơn, dễ sử dụng. (tự cài)
- Đối với VSCode thì có thể cài thêm extension GitLens – cũng của GitKraken luôn. (tự cài)



Install GIT



Create Github account

*Không có tutorial cho phần này đâu
Tự thân vận động đi :))))
Cho 5 phút để làm*

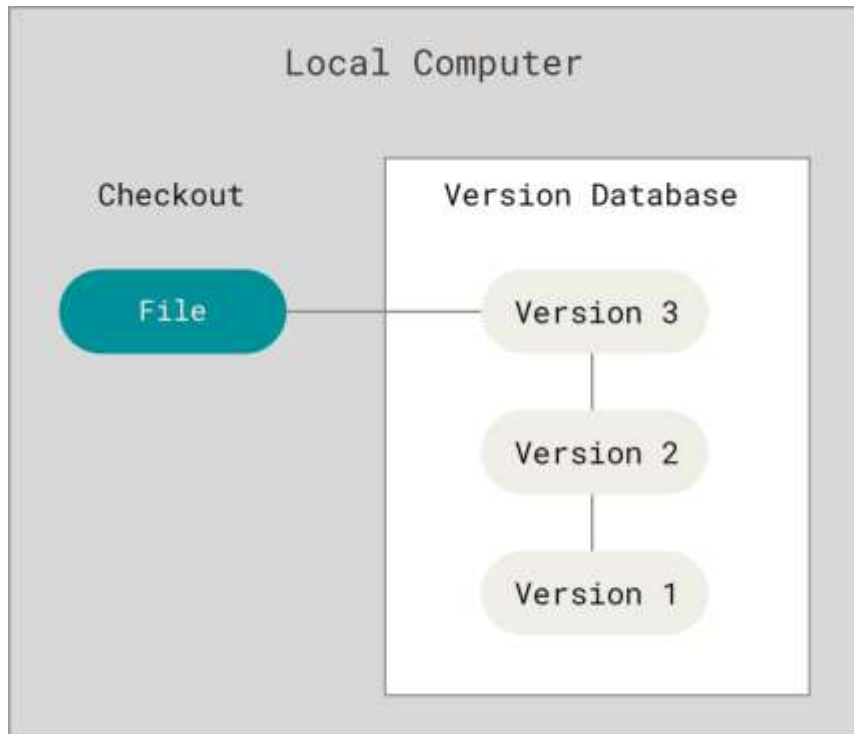
*Không xài mail trường làm mail chính
cho github*



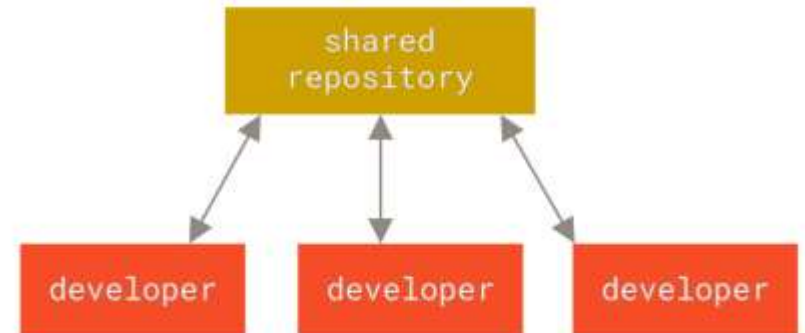
không ngờ tới phải
không

Version control system

Version control system

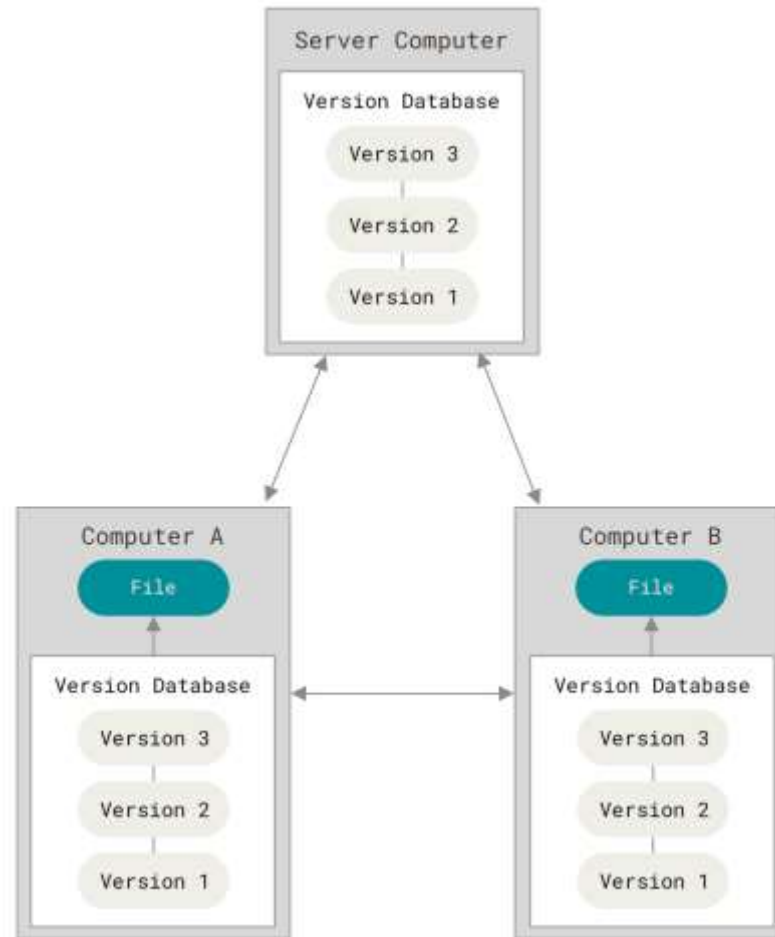


Local version control



Centralized version control

Version control system

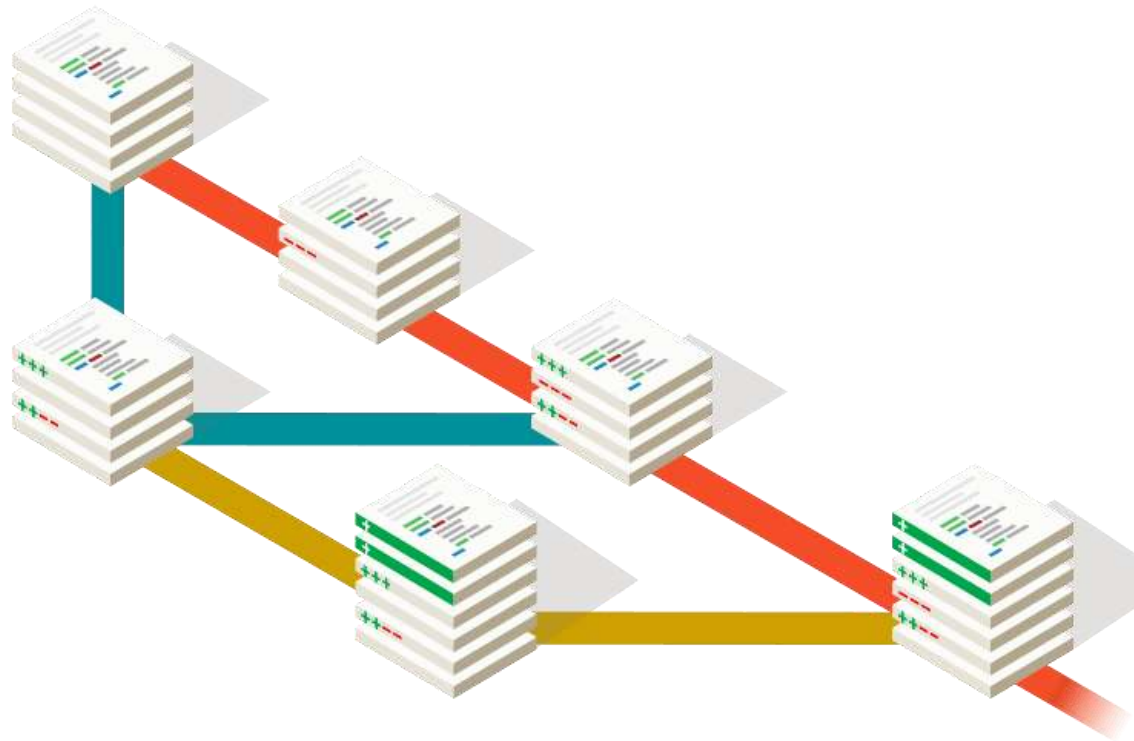


Distributed version control

What is GIT ? Why, Where and When use GIT ?

What is GIT ?

- GIT là một hệ thống quản lý phiên bản (version control system) với mã nguồn mở và miễn phí.

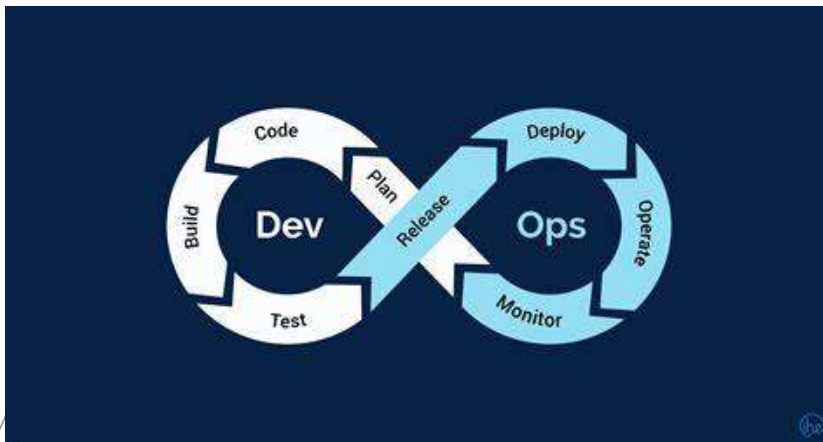


Why GIT ?

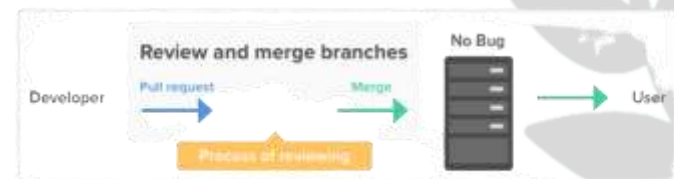
- Version control: kiểm soát các thay đổi trong file trong từng version.
- Collaboration: phối hợp với những người khác trong cùng một project.
- Branching and Merging: dùng để tạo ra những tính năng mới mà không gây ảnh hưởng nhiều tới file system đang hoạt động.
- Backup and Recovery: có thể khôi phục lại những file bị lỗi hoặc bị xóa.

Where and when use GIT ?

- Khi phát triển một dự án bất kì.
- Khi có nhiều người phát triển trong cùng một lúc.
- Cách ly phần dev và release.
- CI/CD(Continuous Integration/Continuous Deployment)



Development without pull request

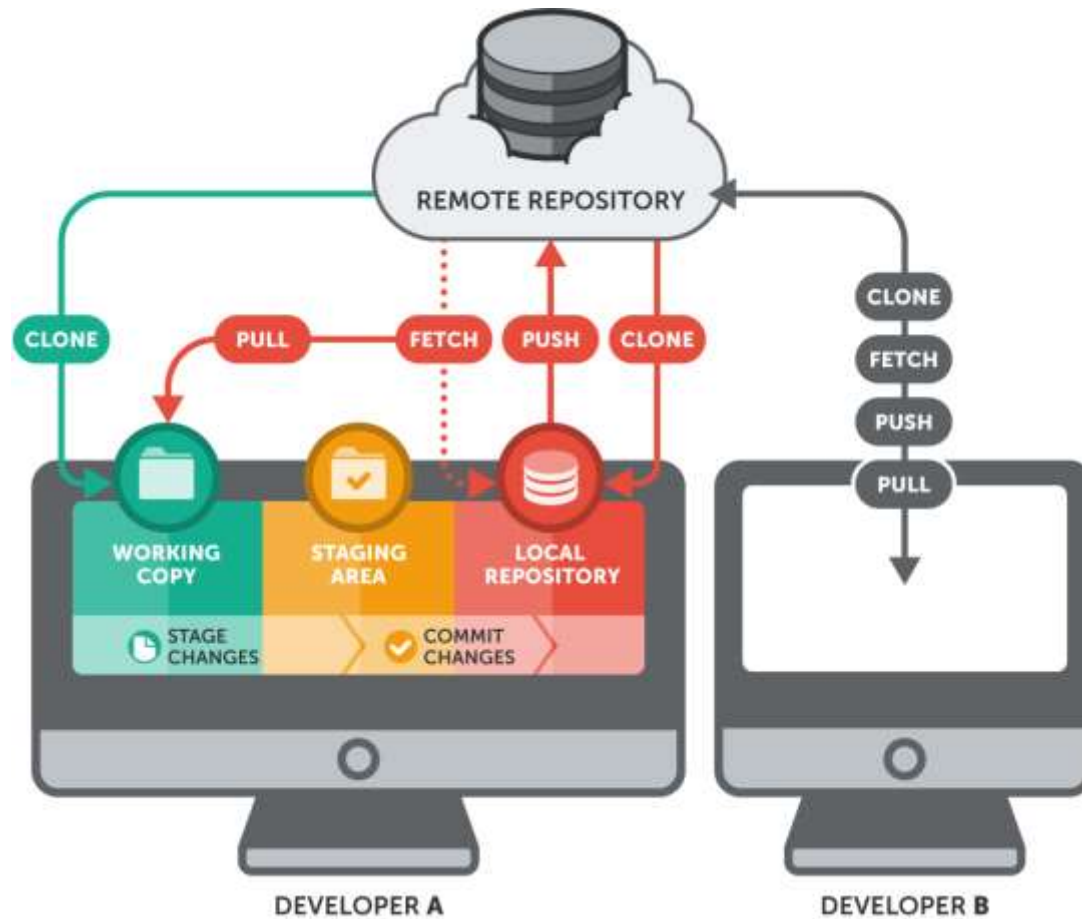


Development with pull request

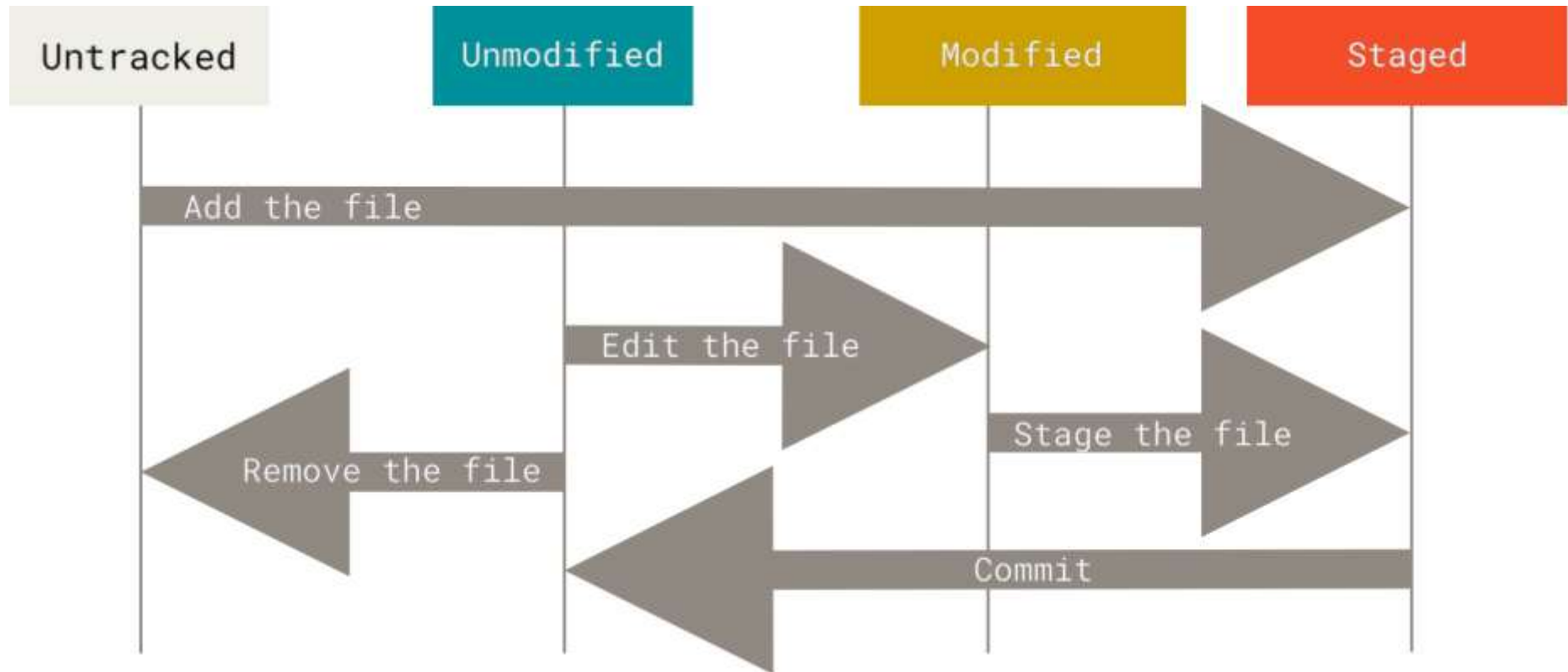
Working principle GLT and File lifecycle



Working principle GIT



File lifecycle



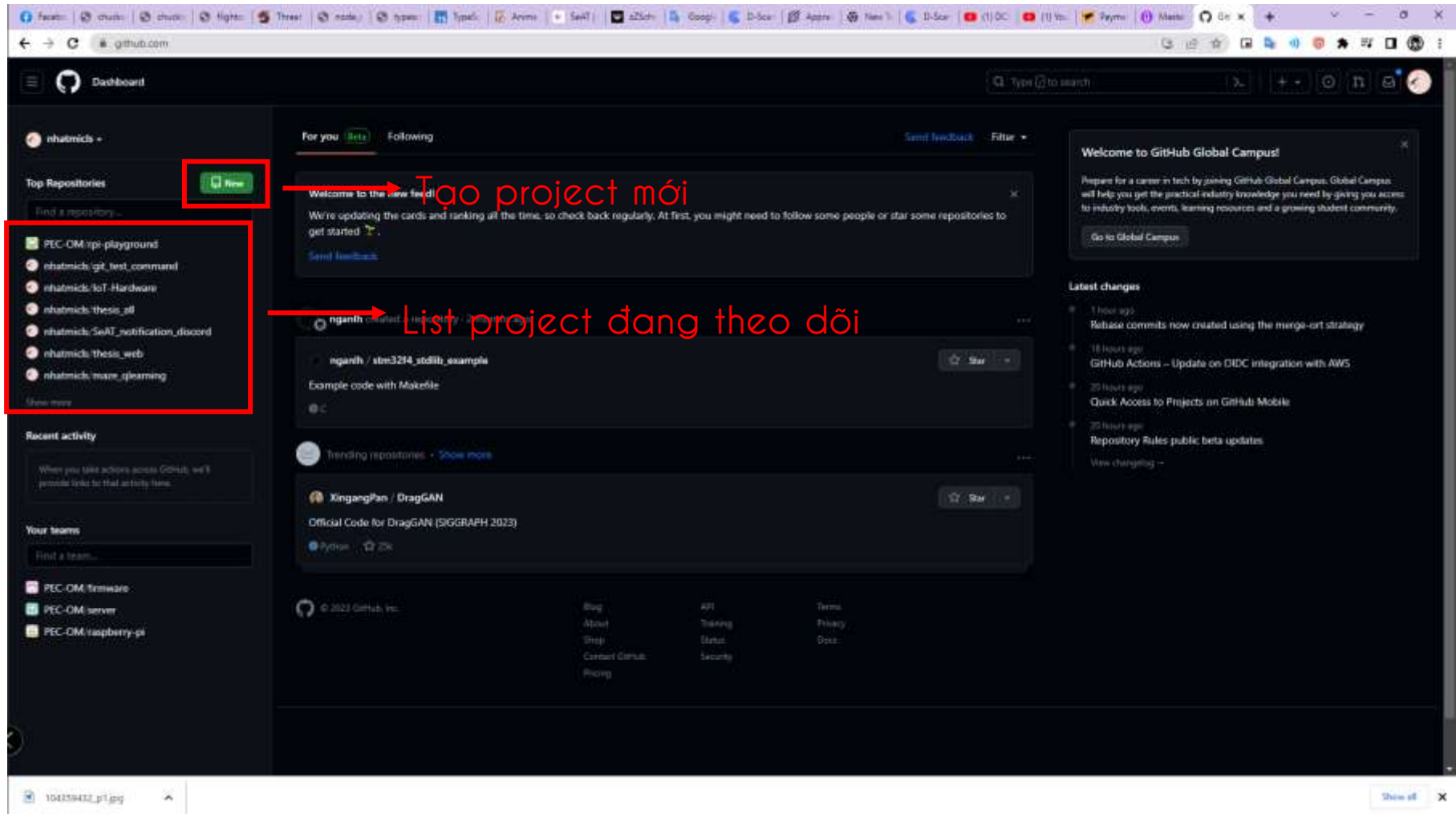
What is Github ?

Github là gì ?

- Github là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn cho các dự án phần mềm.
- Github có đầy đủ những tính năng của GIT.
- Ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.



Github front page



Github create project

The screenshot shows the GitHub 'Create a new repository' page. The page is dark-themed. At the top, there's a navigation bar with the GitHub logo and a search bar. Below the navigation bar, the main heading is 'Create a new repository'. Underneath, there's a sub-heading: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'

Below this, a note states: 'Required fields are marked with an asterisk (*)'.

The form consists of several sections:

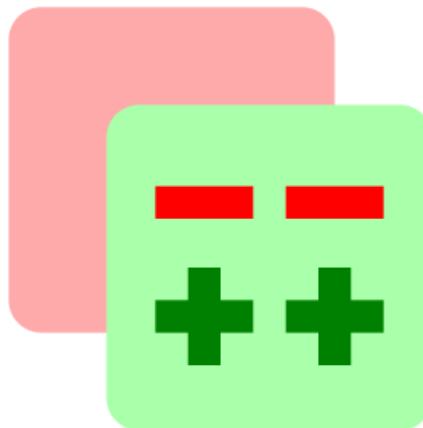
- Owner * / Repository name *:** This section is highlighted with a red box. It contains a dropdown menu for the owner (currently showing 'idhatmide') and a text input field for the repository name. A red annotation 'Tên repo' (Repo name) points to the text input field.
- Description (optional):** This section is highlighted with a yellow box. It contains a text input field for the repository description. A yellow annotation 'Mô tả repo' (Repo description) points to this field.
- Visibility:** This section is highlighted with a white box. It contains two radio buttons: 'Public' (selected) and 'Private'. A white annotation 'Chế độ hiển thị' (Display mode) points to this section.
- Initialize this repository with:** This section is highlighted with a green box. It contains a checkbox for 'Add a README file' and a text input field for 'Add .gitignore'. A green annotation 'Add readme.md' points to the 'Add a README file' checkbox.
- Choose a license:** This section is highlighted with a purple box. It contains a dropdown menu for 'License'. A purple annotation 'Add license' points to this section.

At the bottom of the form, there's a green button labeled 'Create repository'. A small note at the bottom left states: 'You are creating a public repository in your personal account.'

Github alternative

Github alternative

- Những hệ thống tương đương: GitLab, Gerrit, Bitbucket, SourceForge,

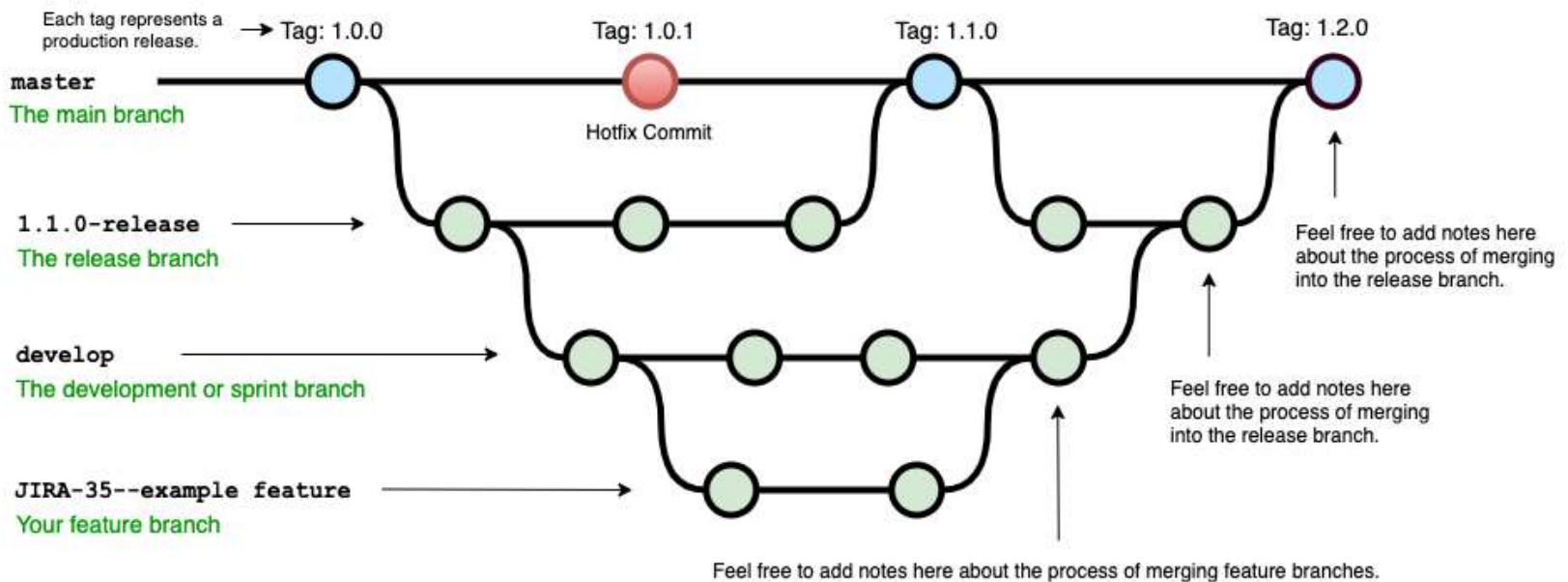


GIT/Github FLOW

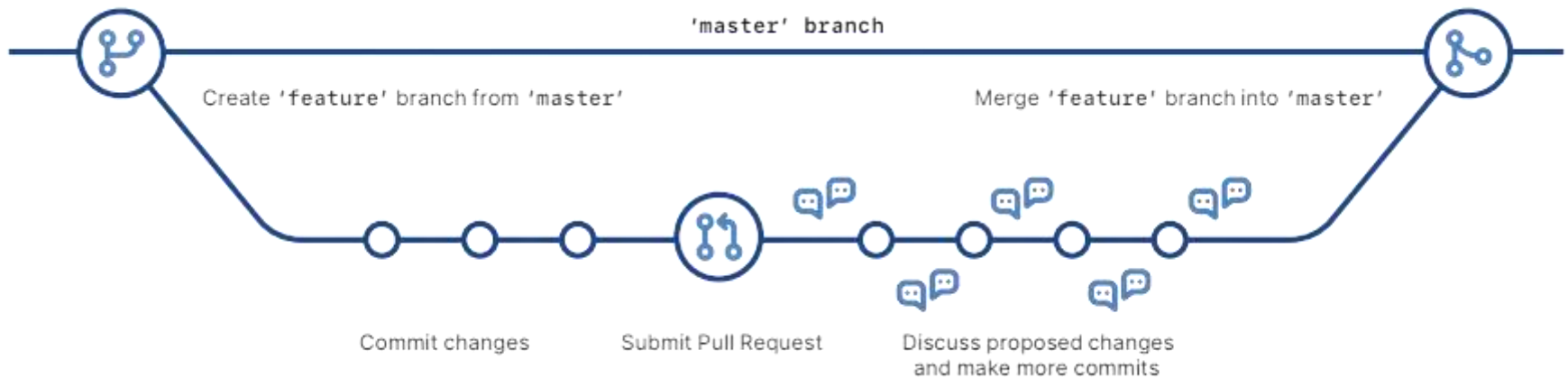
GIT FLOW

Example diagram for a workflow similar to "Git-flow" :

See: <https://nvie.com/posts/a-successful-git-branching-model/>



Github FLOW



Basic command line and github

Github TL;DR

- Up code lên git:
 - `git add .`
 - `git commit -m "something"`
 - `git push`
- Clone/download code từ git:
 - `git pull`

Command line tip

- Hầu hết mọi command trong command line đều có chức năng hỗ trợ người dùng.
- How to use:
 - Something --help.
 - Something -h.
 - Có thể có cả 2 hoặc 1 trong 2 tùy người tạo ra cái này.
- Đối với git cũng thế:
 - git **xxxx** -h (hiển thị help trên terminal).
 - git **xxxx** --help (hiển thị help trên web).

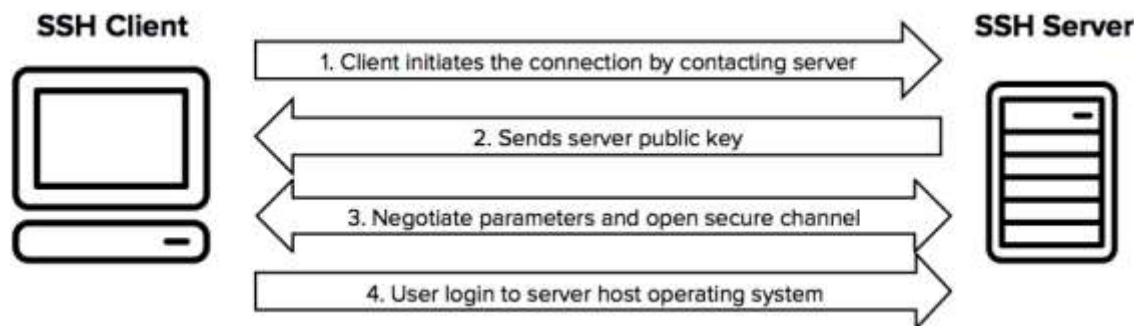
Git keyword

- Repository: một project
- Branch: một nhánh làm việc (thêm hình tượng trung)
- HEAD: là điểm làm việc hiện tại
- HEAD~: là điểm commit phía trước HEAD (HEAD~0)
- HEAD~X: là điểm commit phía trước HEAD X+1 lần
 - VD: HEAD~3 là điểm commit phía trước HEAD~2

SSH

SSH PROTOCOL

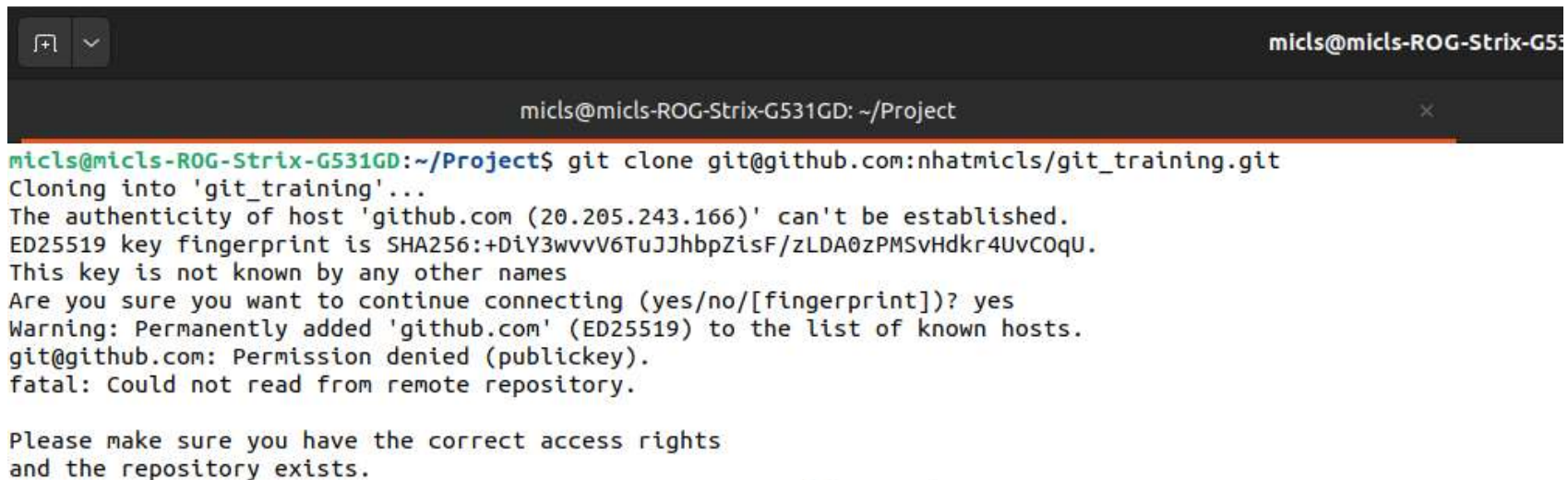
- SSH – Secure Shell là một phần mềm để nâng cao bảo mật cho hệ thống quản trị và truyền tải dữ liệu.
- SSH được sử dụng đảm bảo an toàn giao tiếp giữa client và server. Mọi xác thực, command, ... đều sẽ được mã hóa



SSH KEY PAIR

- SSH key pair là 1 cặp public key và private key:
 - Public key: là key được lưu trữ trong các server, được dùng để mã hóa dữ liệu và chỉ những người có private key mới có thể giải mã được.
 - Private key: là key **chỉ được** sở hữu bởi người dung, nó được dùng để giải mã các gói tin được mã hóa bằng public key.
- Private key phải được lưu trữ kĩ càng → lộ là tèo

SSH GENERATE KEY



A terminal window titled "mics@mics-ROG-Strix-G531GD: ~/Project" displays the output of a git clone command. The command is "git clone git@github.com:nhatmics/git_training.git". The output shows the cloning process starting, followed by a warning that the host's authenticity cannot be established. The fingerprint is displayed as "ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.". The user is prompted to confirm the connection, and they respond with "yes". A warning is shown that the host has been permanently added to the list of known hosts. The command then fails with the message "git@github.com: Permission denied (publickey). fatal: Could not read from remote repository." Below the terminal output, a message reads: "Please make sure you have the correct access rights and the repository exists."

```
mics@mics-ROG-Strix-G531GD: ~/Project$ git clone git@github.com:nhatmics/git_training.git
Cloning into 'git_training'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

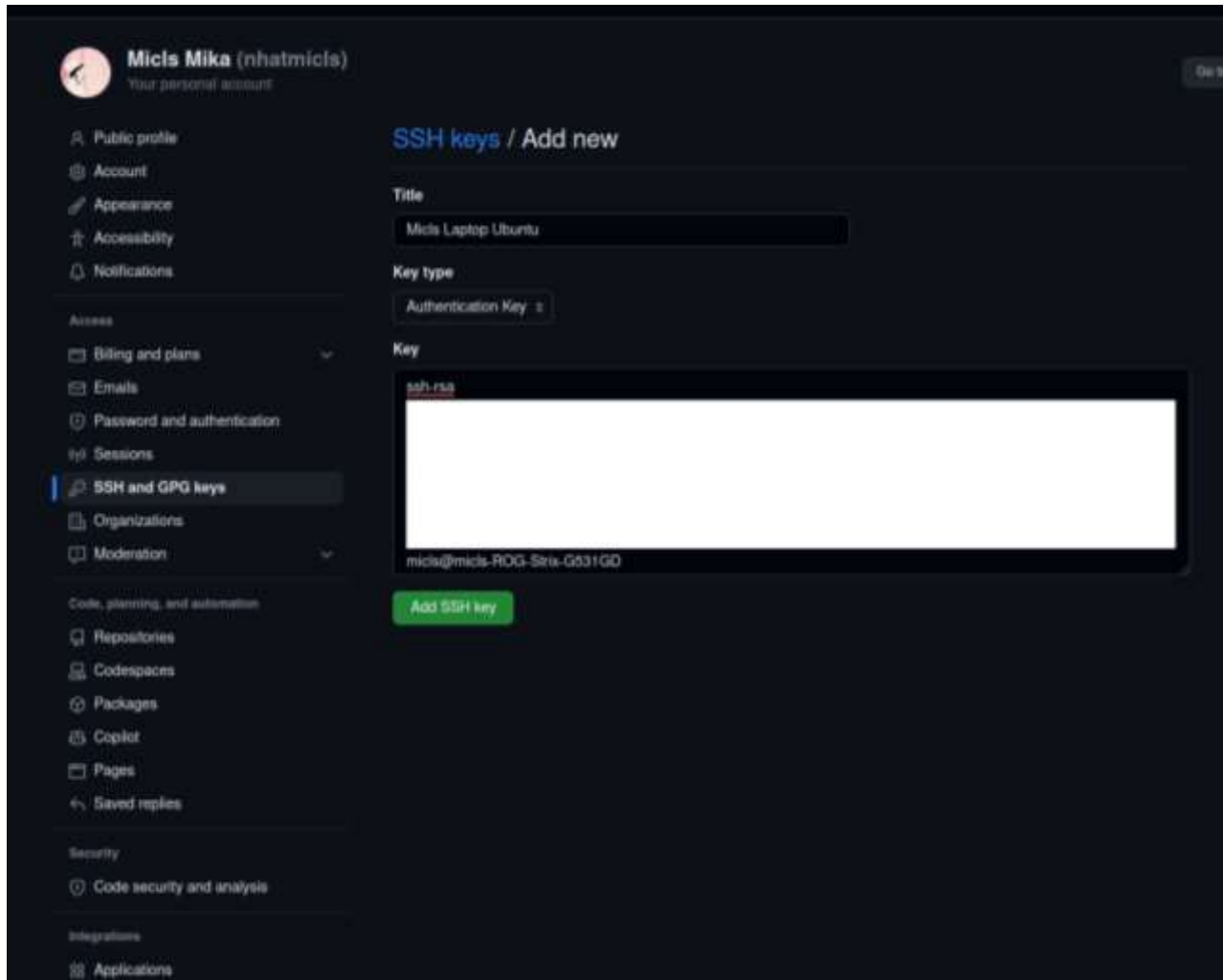
Please make sure you have the correct access rights
and the repository exists.
```

SSH GENERATE KEY

```
micsl@micsl-ROG-Strix-G531GD: ~/Project
micsl@micsl-ROG-Strix-G531GD:~/Project$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/micls/.ssh/id_rsa): /home/micls/.ssh/github
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/micls/.ssh/github
Your public key has been saved in /home/micls/.ssh/github.pub
The key fingerprint is:
[REDACTED] micsl@micsl-ROG-Strix-G531GD
The key's randomart image is:
[REDACTED]

micsl@micsl-ROG-Strix-G531GD:~/Project$ cat < ~/.ssh/
github      github.pub  id_rsa      id_rsa.pub  known_hosts
micsl@micsl-ROG-Strix-G531GD:~/Project$ cat < ~/.ssh/github.pub
ssh-rsa [REDACTED]
micsl@micsl-ROG-Strix-G531GD:~/Project$
```

SSH GENERATE KEY



The screenshot shows the GitHub user interface for managing SSH keys. On the left is a sidebar with navigation options: Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys (highlighted), Organizations, Moderation, Code, planning, and automation, Repositories, Codespaces, Packages, Copilot, Pages, Saved replies, Security, Code security and analysis, Integrations, and Applications. The main content area is titled 'SSH keys / Add new'. It contains a form with the following fields: 'Title' (filled with 'Mics Laptop Ubuntu'), 'Key type' (set to 'Authentication Key'), and 'Key' (a large text area containing 'ssh-rsa' and a redacted public key). Below the key field is the email address 'mics@mics-ROG-Strix-G531GO' and a green 'Add SSH key' button.

Mics Mika (nhatmics)
Your personal account

SSH keys / Add new

Title
Mics Laptop Ubuntu

Key type
Authentication Key

Key
ssh-rsa
[Redacted Public Key]
mics@mics-ROG-Strix-G531GO

Add SSH key

SSH GENERATE KEY

```
micls@micls-ROG-Strix-G531GD: ~/Project
micls@micls-ROG-Strix-G531GD:~/Project$ git clone git@github.com:nhatmicls/git_training.git
Cloning into 'git_training'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
micls@micls-ROG-Strix-G531GD:~/Project$ git clone git@github.com:nhatmicls/git_training.git
Cloning into 'git_training'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
micls@micls-ROG-Strix-G531GD:~/Project$
```



DEMO PROJECT INIT

Demo project init


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).


Owner *

Repository name *

 nhatnicks

 /


git_training




Great repository names are short and memorable. Need inspiration? How about [potential-winner](#)?

Description (optional)

PIF_TRAINING_DEMO


☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:


☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore




Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license



A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set [main](#) as the default branch. Change the default name in your settings.

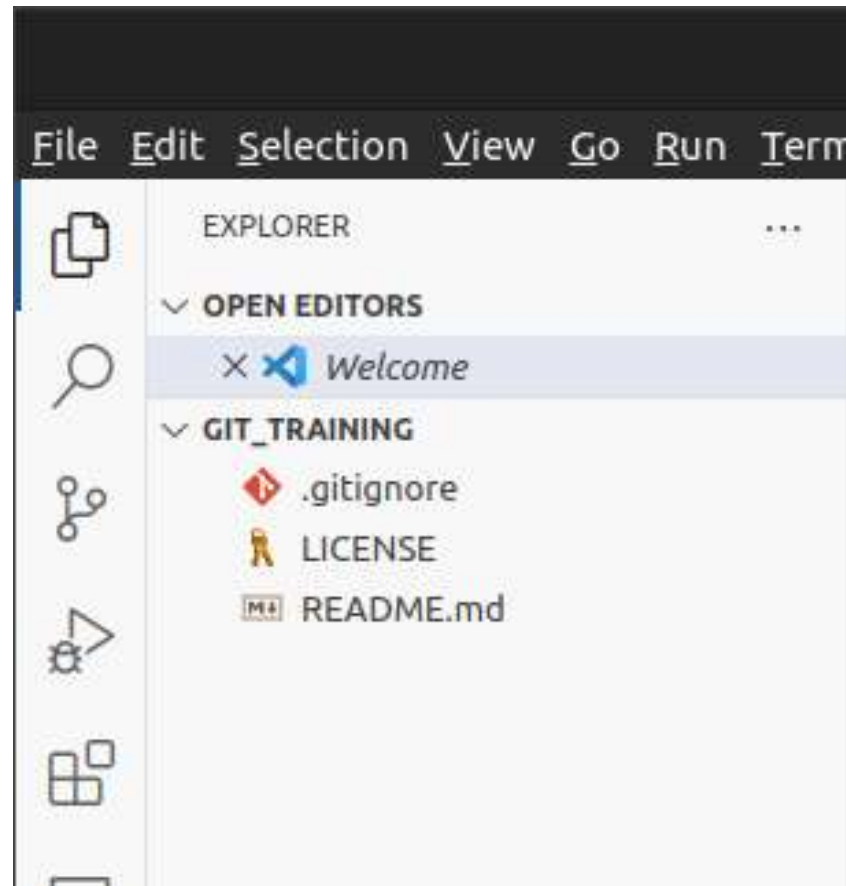
 You are creating a public repository in your personal account.

Create repository

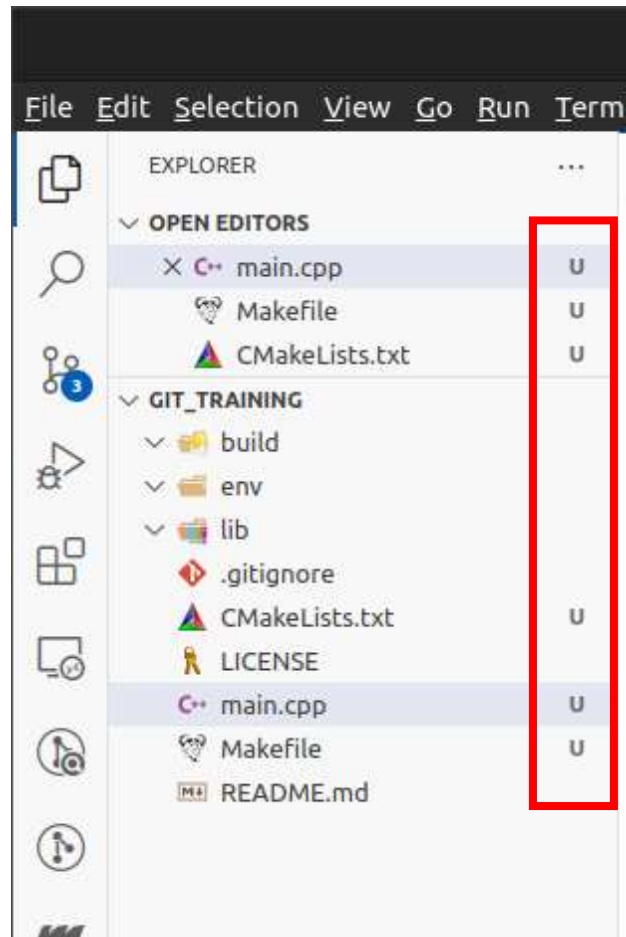
7/8/2023

40

Demo project init



Demo project init



CREATE REPO

Create repositories

- git **init**
 - Biến một thư mục local trở thành một git repository.
 - Chỉ dùng khi project đã lỡ làm :)))
- git **clone** **<url>**
 - Create + Clone một repository đã có sẵn trên Github.
 - Bao gồm:
 - Các file.
 - Các branch.
 - Các commit.

Create repositories

- Vì một lí do gì đó mà quên tạo trên repo trên git mà đã làm project thì có 2 cách:
 - **git clone way:** Tạo một repo trên Github rồi clone xuống. Xong copy and paste vào thư mục mới clone.
 - **git init way:** Tạo một repo trên Github. Xong git init cho thư mục đó rồi push lên Github.

git init

```
$ git init
```

Initialized empty Git repository in E:/Project/testgitcommand/.git/

```
$ git add .
```

```
$ git commit -m "init commit"
```

...

```
$ git remote add main git@github.com:nhatmicls/git_test_command.git
```

```
$ git push -u main
```

....

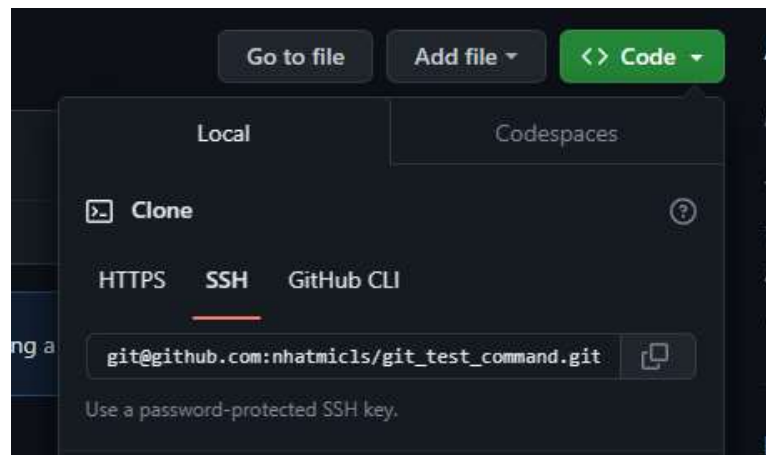
git clone <url>

- Có 2 loại url:
 - https: có thể sử dụng cho public repo thoải mái, nhưng private repo phải đăng nhập cho mỗi lần push, pull,

`https://github.com/xxxx/xxxx.git`

- ssh: có thể sử dụng cho public repo và private repo thoải mái khi đã add ssh public key

`git@github.com:xxxx/xxxx.git`



git clone <url>

```
micls@micls-ROG-Strix-G531GD: ~/Project
micls@micls-ROG-Strix-G531GD:~/Project$ git clone git@github.com:nhatmicls/git_training.git
Cloning into 'git_training'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

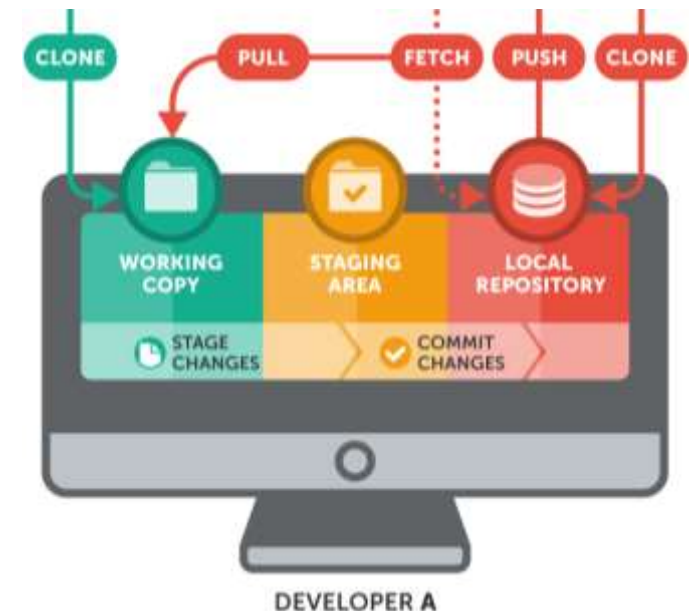
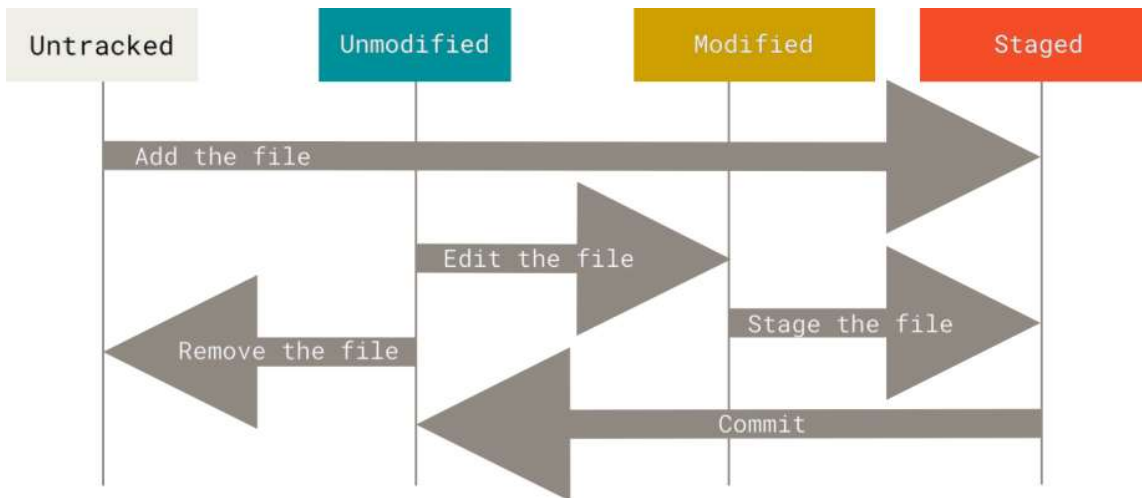
Please make sure you have the correct access rights
and the repository exists.
micls@micls-ROG-Strix-G531GD:~/Project$ git clone git@github.com:nhatmicls/git_training.git
Cloning into 'git_training'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
micls@micls-ROG-Strix-G531GD:~/Project$
```

Chưa add
ssh key

Đã add ssh
key

Something need to know before go to main part

- git **status**: kiểm tra tình trạng các file trong folder
- git **restore**: unstage các file đã trong stage



git status

```
micls@micls-ROG-Strix-G531GD: ~/Project/git_training
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CMakeLists.txt
        Makefile
        main.cpp

no changes added to commit (use "git add" and/or "git commit -a")
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```

git restore

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CMakeLists.txt
        Makefile
        main.cpp

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

git restore

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git add README.md
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch main
Your branch is up to date with 'origin/main'.
```

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
 modified: README.md

Untracked files:
(use "git add <file>..." to include in what will be committed)
 CMakeLists.txt
 Makefile
 main.cpp

git restore

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git restore --staged README.md
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
```

On branch main

Your branch is up to date with 'origin/main'.

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: README.md

Untracked files:

(use "git add <file>..." to include in what will be committed)

CMakeLists.txt

Makefile

main.cpp

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

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```



SYNCHRONIZE

Make changes and synchronize it

- Make changes:
 - git **log**: liệt kê history version cho branch hiện tại.
 - git **diff**: so sánh với các version.
 - git **show**: hiển thị các metadata và nội dung thay đổi.
 - git **add**: snapshot dữ liệu chuẩn bị cho versioning.
 - git **commit**: ghi dữ liệu đã được snapshot vào version history.
- Synchronize changes:
 - git **fetch**: download các dữ liệu history liên quan tới các branch đang theo dõi.
 - git **merge**: hợp nhất các branch với nhau.
 - git **push**: upload các commit trong các local branch lên Github.
 - git **pull**: download hết mọi thứ.

git log/show

Cấu trúc:

```
$ git log -h
```

```
usage: git log [<options>] [<revision-range>] [--] <path>...
```

```
or: git show [<options>] <object>...
```

VD:

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log
commit a6394c31fcae702fefed3f591abbfb8e9608e60e (HEAD -> main, origin/main, origin/HEAD)
Author: Micls Mika <20109101+nhatmicls@users.noreply.github.com>
Date:   Fri Jun 30 12:46:17 2023 +0700
```

```
Initial commit
```

git log – command line

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log
commit 12566955900b3c22a66e1eacba2ed3fd8e575d09 (HEAD -> main)
Author: Micls <cptprice123@outlook.com>
Date:   Fri Jun 30 15:13:55 2023 +0700
```

[Micls] Update README

```
commit a6394c31fcae702fefed3f591abbfb8e9608e60e (origin/main, origin/HEAD)
Author: Micls Mika <20109101+nhatmicls@users.noreply.github.com>
Date:   Fri Jun 30 12:46:17 2023 +0700
```

Initial commit

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```



git log – command line

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log dev/math --oneline
85c9115 (origin/dev/math, dev/math) [Micls_1] Add math func
83ced3a (HEAD -> develop, origin/main, origin/HEAD, main) [Micls] Add etc file
b827394 Update README.md
1c1bfdb Add main.cpp
9a25844 Add Makefile file
1256695 [Micls] Update README
a6394c3 Initial commit
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log dev/compare --oneline
cd0843b (origin/dev/compare, dev/compare) [Micls_2] Add compare function
83ced3a (HEAD -> develop, origin/main, origin/HEAD, main) [Micls] Add etc file
b827394 Update README.md
1c1bfdb Add main.cpp
9a25844 Add Makefile file
1256695 [Micls] Update README
a6394c3 Initial commit
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log develop --oneline
83ced3a (HEAD -> develop, origin/main, origin/HEAD, main) [Micls] Add etc file
b827394 Update README.md
1c1bfdb Add main.cpp
9a25844 Add Makefile file
1256695 [Micls] Update README
a6394c3 Initial commit
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```



git show – command line

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git show  
commit 12566955900b3c22a66e1eacba2ed3fd8e575d09 (HEAD -> main)  
Author: Micls <cptprice123@outlook.com>  
Date:   Fri Jun 30 15:13:55 2023 +0700
```

```
[Micls] Update README
```

```
diff --git a/README.md b/README.md  
index 74e5beb..c57757f 100644  
--- a/README.md  
+++ b/README.md  
@@ -1,2 +1,7 @@  
  # git_training  
+  
+   PIF_TRAINING_DEMO  
+  
+## How to use  
+  
+Don't have any tutorial  
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```



git diff

Cấu trúc:

```
$ git diff -h
```

```
usage: git diff [<options>] [<commit>] [--] [<path>...]
```

```
    or: git diff [<options>] --cached [--merge-base] [<commit>] [--] [<path>...]
```

```
    or: git diff [<options>] [--merge-base] <commit> [<commit>...] <commit> [--]
    [<path>...]
```

```
    or: git diff [<options>] <commit>...<commit> [--] [<path>...]
```

```
    or: git diff [<options>] <blob> <blob>
```

```
    or: git diff [<options>] --no-index [--] <path> <path>
```

git diff

So sánh giữa các commit

bất kì

```
$ git diff 7058f79 e69de29
```

```
diff --git a/7058f79 b/e69de29
```

```
index 7058f79..e69de29 100644
```

```
--- a/7058f79
```

```
+++ b/e69de29
```

```
@@ -1,2 +0,0 @@
```

```
-test 1 2 3 4 5 6 7 8 9
```

```
-test 9 8 7 6 5 4 3 2 1
```

```
\ No newline at end of file
```

git diff

So sánh file đang sửa với phiên bản trước đó



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows the file structure of a project named 'GIT_TRAINING'. The 'Makefile' file is highlighted, and its status is 'M' (Modified). The main editor area shows the content of the 'Makefile' file, which is a Makefile for building a C++ program. The file has been modified, as indicated by the 'M' icon in the top right corner of the editor window.

```
EXPLORER
...
OPEN EDITORS
  main.cpp U
  README.md
  Makefile M
  CMakeLists.txt U
GIT_TRAINING
  build
  env
  lib
  .gitignore
  CMakeLists.txt U
  LICENSE
  main.cpp U
  Makefile M
  README.md

main.cpp U
README.md
Makefile M
Makefile
You, 10 seconds ago | 2 authors (You and others)
1 main:
2     g++ main.cpp -o ./build/main.o
3 dev:
4     g++ main.cpp -o ./build/dev.d
```

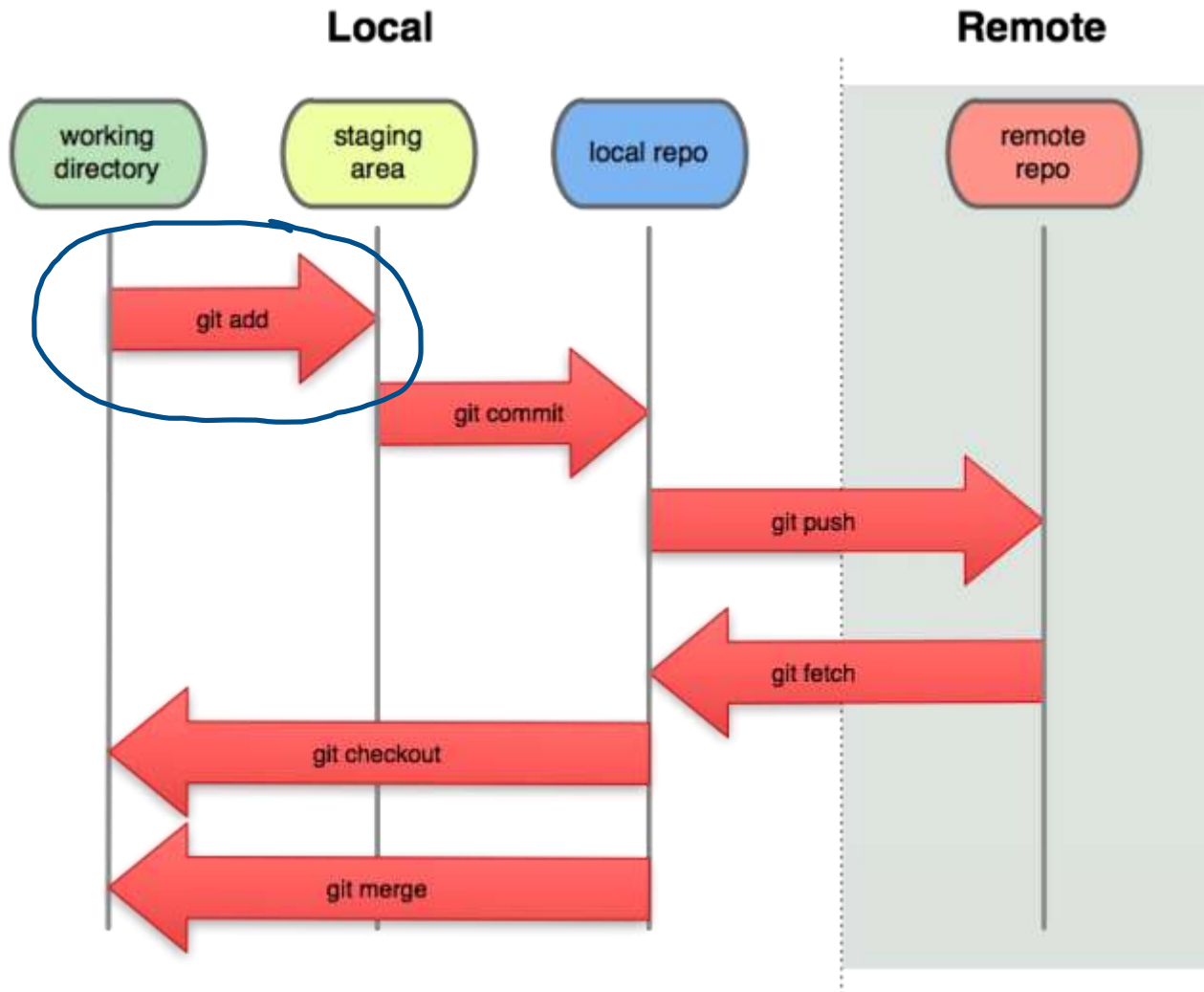
git diff

So sánh file đang sửa với phiên bản trước

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git diff
diff --git a/Makefile b/Makefile
index 5cd0af3..dd8e852 100644
--- a/Makefile
+++ b/Makefile
@@ -1,2 +1,4 @@
  main:
-      g++ main.cpp -o ./build/main.o
\ No newline at end of file
+      g++ main.cpp -o ./build/main.o
+dev:
+      g++ main.cpp -o ./build/dev.o
\ No newline at end of file
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```



git add



git add

Cấu trúc:

```
$ git add -h
```

```
usage: git add [<options>] [--] <pathspec>...
```

-n, --dry-run	dry run
-v, --verbose	be verbose

....

- Có 2 cách để git add:
 - git add . → add toàn bộ thay đổi
 - git add "file_path" → chỉ add những thứ được chỉ định

git add .

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

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:   Makefile

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CMakeLists.txt
        main.cpp

no changes added to commit (use "git add" and/or "git commit -a")
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git add .
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   CMakeLists.txt
        modified:   Makefile
        new file:   main.cpp

micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```



git add “file_path”

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

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:   Makefile

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CMakeLists.txt
        main.cpp

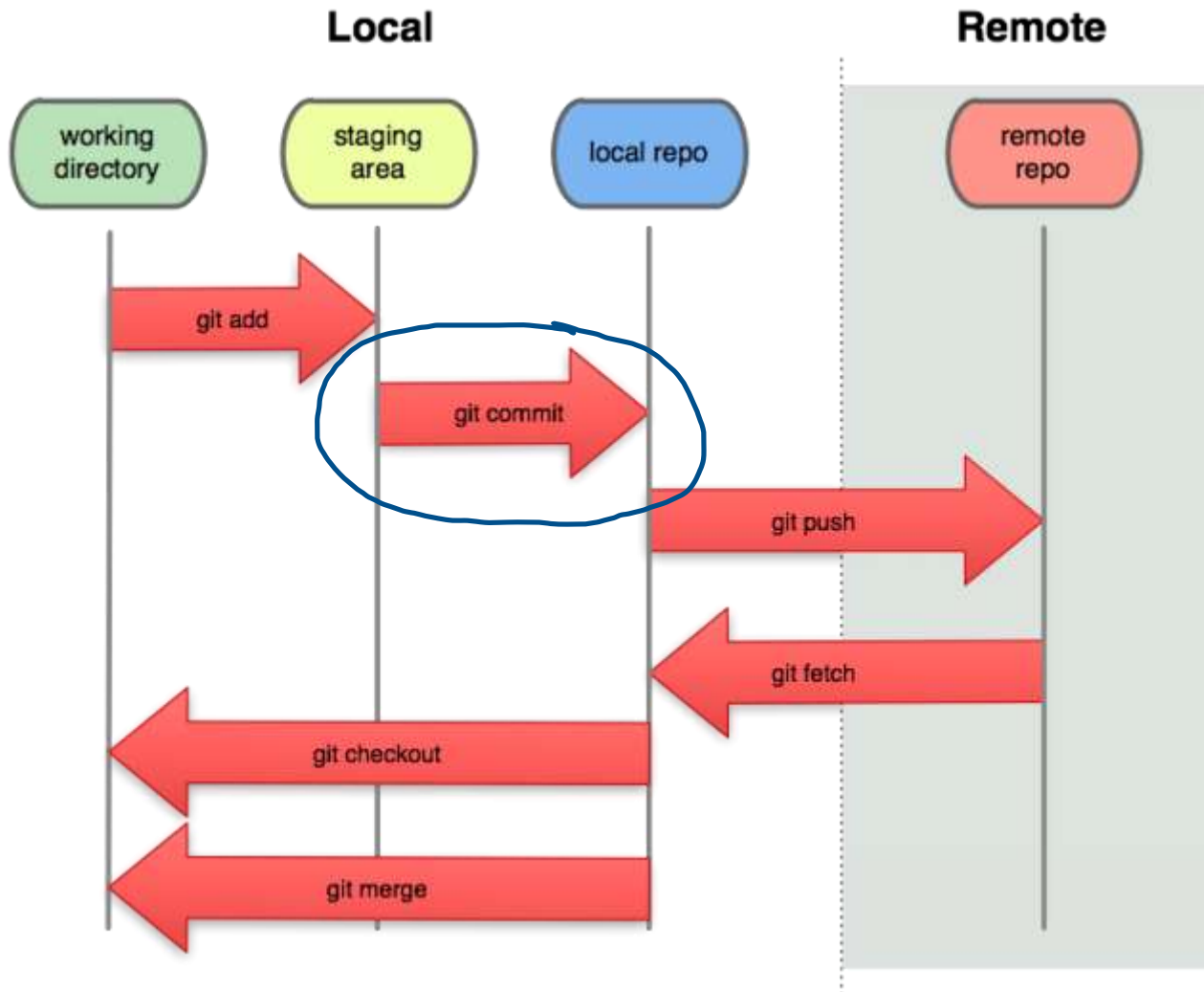
no changes added to commit (use "git add" and/or "git commit -a")
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git add main.cpp
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   main.cpp

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:   Makefile

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CMakeLists.txt
```

git commit



git commit

Cấu trúc:

```
$ git commit -h
```

```
usage: git commit [-a | --interactive | --patch] [-s] [-v] [-u<mode>] [--amend]
                [--dry-run] [(-c | -C | --squash) <commit> | --fixup
((amend|reword):)<commit>)]
                [-F <file> | -m <msg>] [--reset-author] [--allow-empty]
                [--allow-empty-message] [--no-verify] [-e] [--
author=<author>]
                [--date=<date>] [--cleanup=<mode>] [--[no-]status]
                [-i | -o] [--pathspec-from-file=<file> [--pathspec-file-nul]]
                [(--trailer <token>[(=|:)<value>])...] [-S[<keyid>]]
                [--] [<pathspec>...]
```

git commit -m "something"

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git commit -m "Add main.cpp"
[main 1c1bfdb] Add main.cpp
1 file changed, 7 insertions(+)
create mode 100644 main.cpp
```

TIP:

Something = "[Your name] something u have done"

VD:

"[Micls] Fix unstage system"

git commit – config

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log
commit a6394c31fcae702fefed3f591abbfb8e9608e60e (HEAD -> main, origin/main, origin/HEAD)
Author: Micls Mika <20109101+nhatmicls@users.noreply.github.com>
Date:   Fri Jun 30 12:46:17 2023 +0700
```

Initial commit

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git add README.md
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git commit -m "[Micls] Update README"
Author identity unknown
```

*** Please tell me who you are.

Run

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

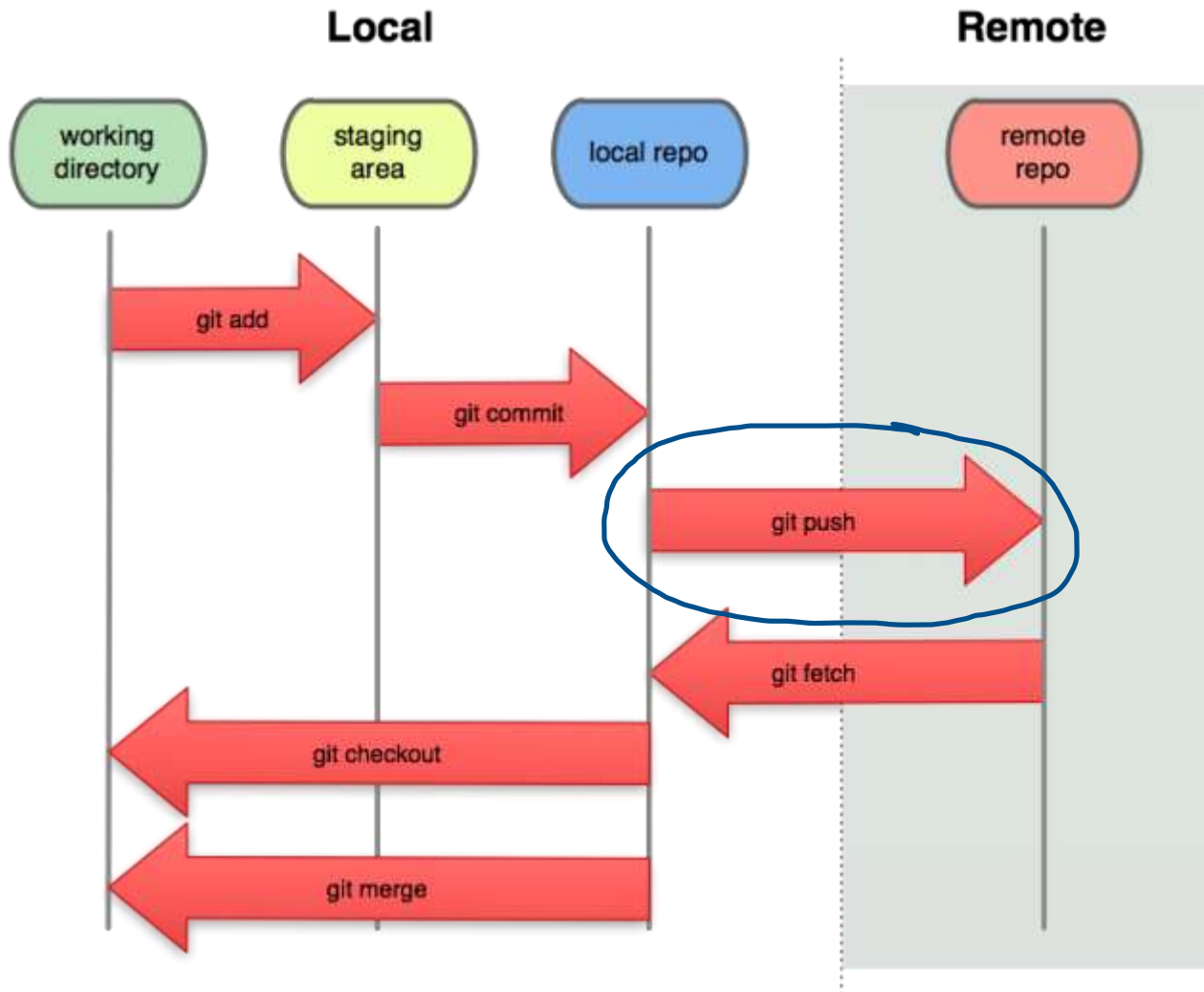
to set your account's default identity.

Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'micls@micls-ROG-Strix-G531GD.(none)')

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git config --global user.email "cptprice123@outlook.com"
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git config --global user.name "Micls"
```

git push



git push

Cấu trúc:

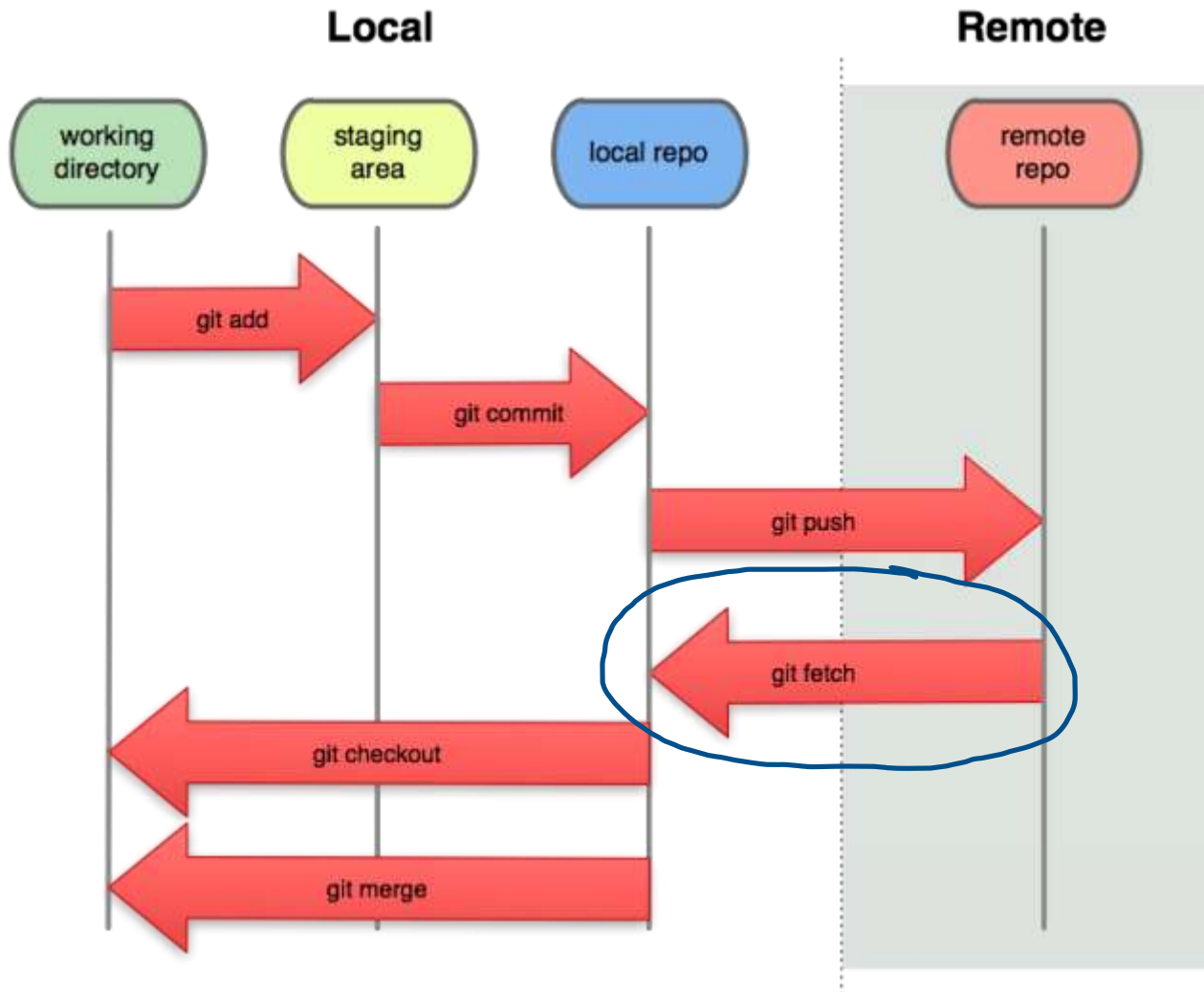
```
git push -h
```

```
usage: git push [<options>] [<repository> [<refspec>...]]
```

VD

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 924 bytes | 308.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To github.com:nhatmicls/git_training.git
   a6394c3..1c1bfdb  main -> main
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```


git pull/fetch



git pull/fetch

Cấu trúc git pull

```
$ git pull -h
```

```
usage: git pull [<options>] [<repository> [<refspec>...]]
```

Cấu trúc git fetch

```
$ git fetch -h
```

```
usage: git fetch [<options>] [<repository> [<refspec>...]]
```

```
or: git fetch [<options>] <group>
```

```
or: git fetch --multiple [<options>] [(<repository> | <group>)...]
```

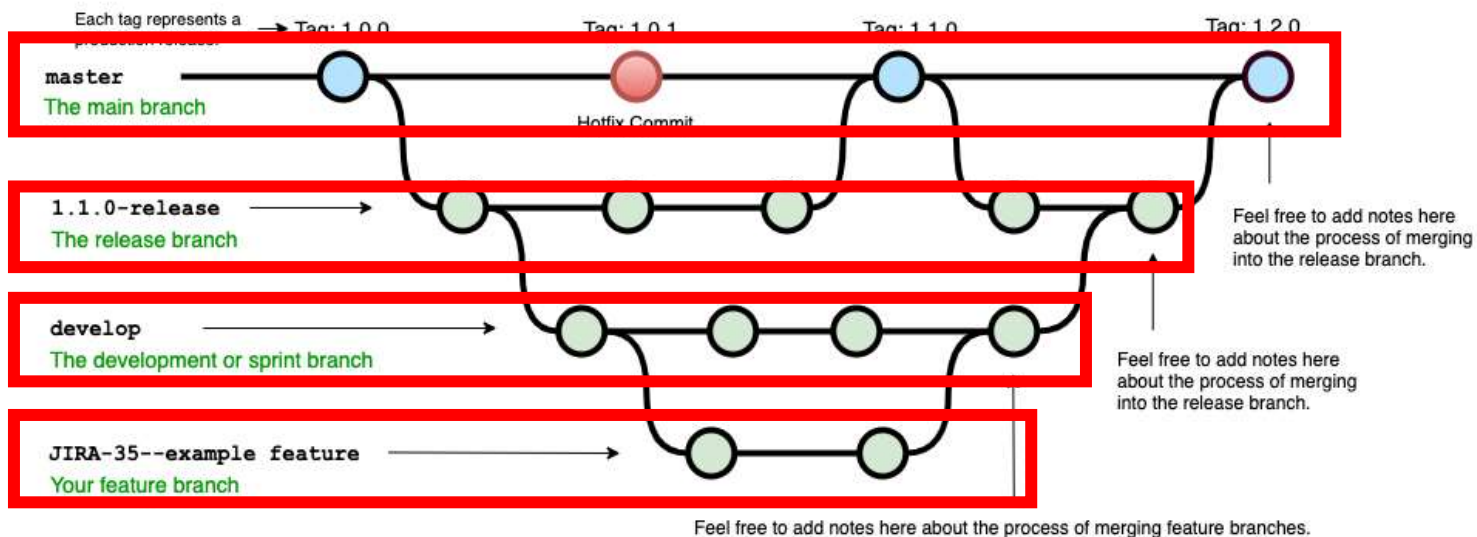
```
or: git fetch --all [<options>]
```

Branches

- git **branch**: tập hợp các lệnh tạo, xóa, ... tới branch
- git **checkout**: dùng để di chuyển giữa các branch với nhau.

Example diagram for a workflow similar to "Git-flow" :

See: <https://nvie.com/posts/a-successful-git-branching-model/>



git branch

Cấu trúc:

```
$ git branch -h
```

```
usage: git branch [<options>] [-r | -a] [--merged] [--no-merged]
```

```
    or: git branch [<options>] [-f] [--recurse-submodules] <branch-name>  
    [<start-point>]
```

```
    or: git branch [<options>] [-l] [<pattern>...]
```

```
    or: git branch [<options>] [-r] (-d | -D) <branch-name>...
```

```
    or: git branch [<options>] (-m | -M) [<old-branch>] <new-branch>
```

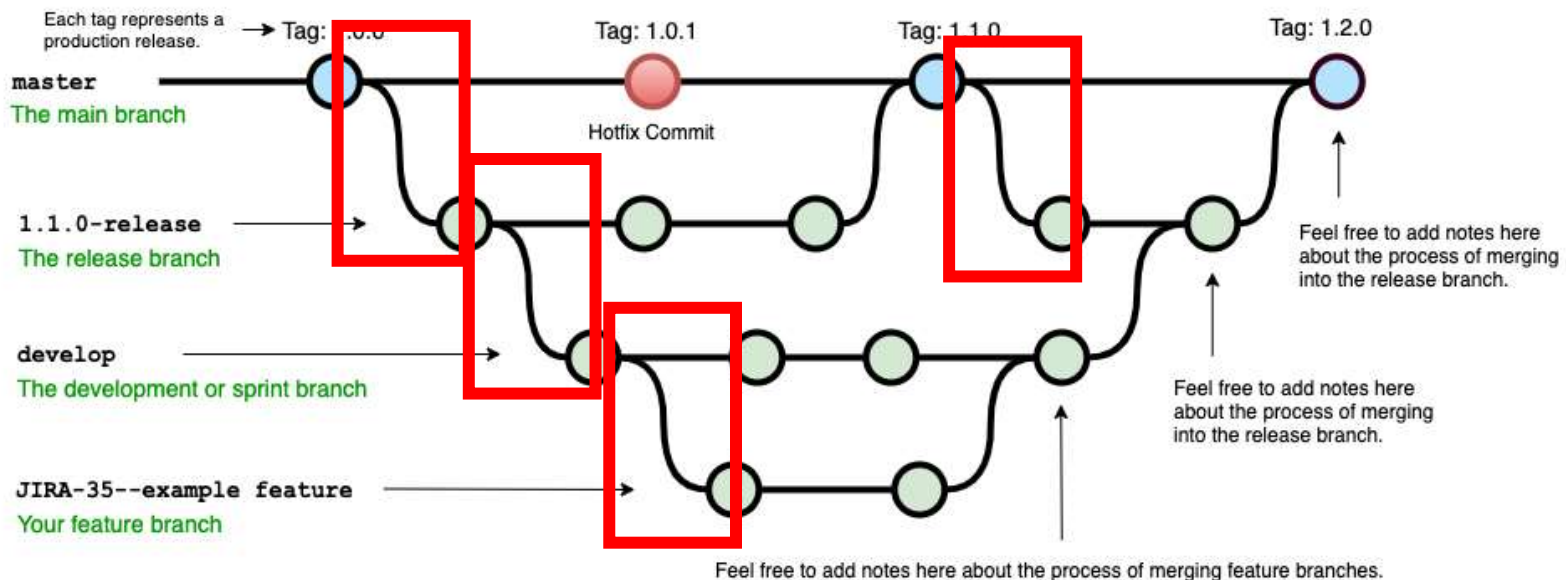
```
    or: git branch [<options>] (-c | -C) [<old-branch>] <new-branch>
```

```
    or: git branch [<options>] [-r | -a] [--points-at]
```

```
    or: git branch [<options>] [-r | -a] [--format]
```

git branch

- `git branch <branch_name>`: tạo ra branch mới
- `git branch -d <branch_name>`: xóa branch có merge



git branch

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git branch -l
* main
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git branch develop main
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git branch -l
  develop
* main
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git checkout develop
Switched to branch 'develop'
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git branch -l
* develop
  main
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git checkout -b dev/math
Switched to a new branch 'dev/math'
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git branch -l
* dev/math
  develop
  main
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git checkout develop
Switched to branch 'develop'
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git checkout -b dev/compare
Switched to a new branch 'dev/compare'
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git branch -l
* dev/compare
  dev/math
  develop
  main
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```

git **branch** related

- Xóa branch remote

```
git push <repo> --delete <branch_name>
```

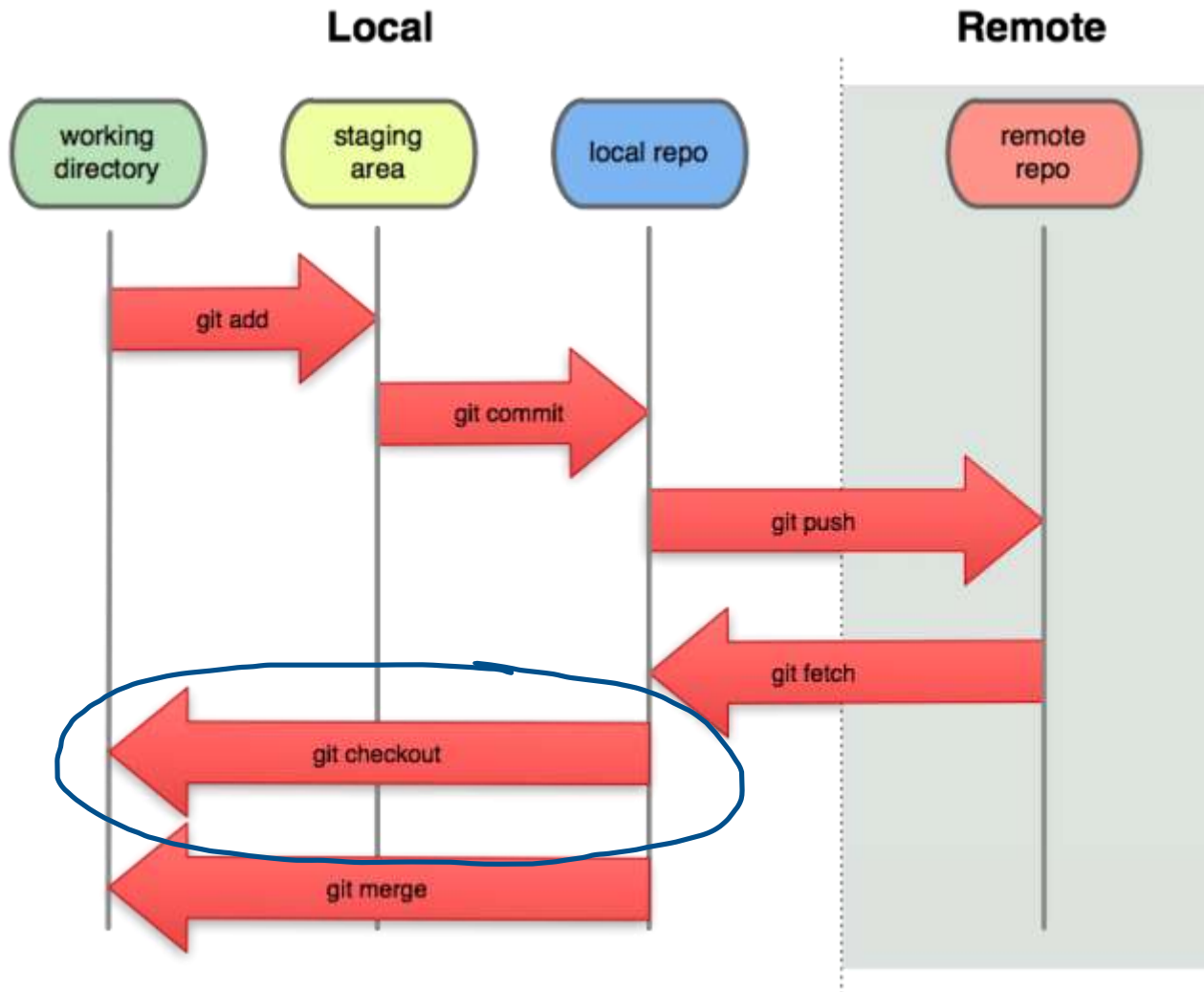
VD:

```
$ git push origin --delete dev
```

```
To github.com:nhatmicls/git_test_command.git
```

```
- [deleted]          dev
```

git checkout



git checkout

Cấu trúc:

- git checkout <branch_name>: chuyển branch
- git checkout -b <branch_name>: tạo branch
- git checkout -b <branch_name> <commit_name>: tạo branch tại 1 commit bất

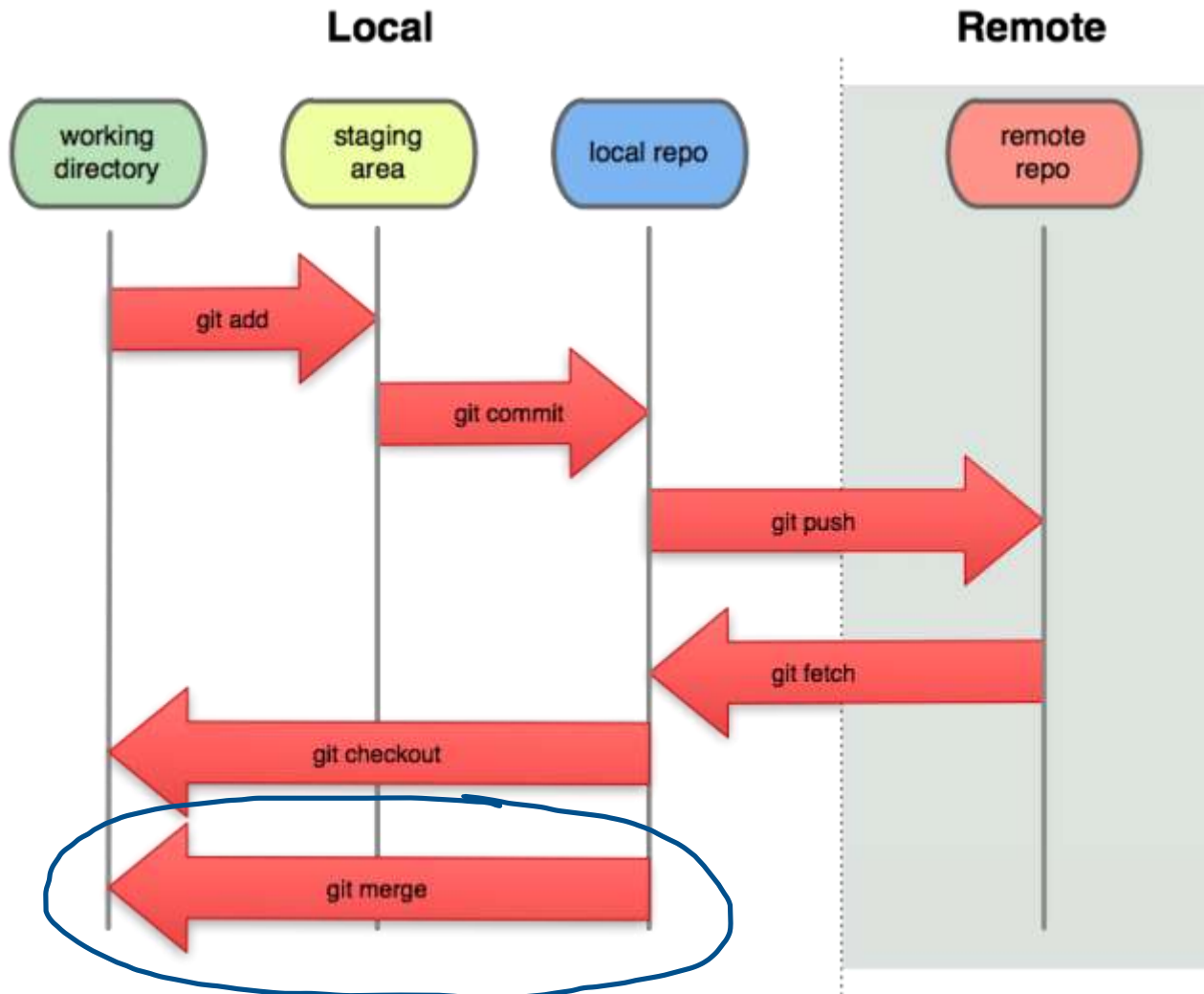
```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git checkout -b dev/hotfix 4263cb5
Switched to a new branch 'dev/hotfix'
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```



BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR	CHANGES	COMMIT DATE / TIME	SHA
develop		[Micls] Add HCMUT	You	1	28 minutes ago	d32b1a5
		[Micls] Add PIF	You	1	29 minutes ago	65f1954
✓ dev/hotfix		[Micls] Add hello	You	2	29 minutes ago	4263cb5
main		[Micls] Edit gitignore	You	1	16 hours ago	acfa69b
		[Micls] Merge develop branch	You		16 hours ago	90b12e9
		[Micls] Add random line code	You	1	17 hours ago	533fe47

MERGE

Merge

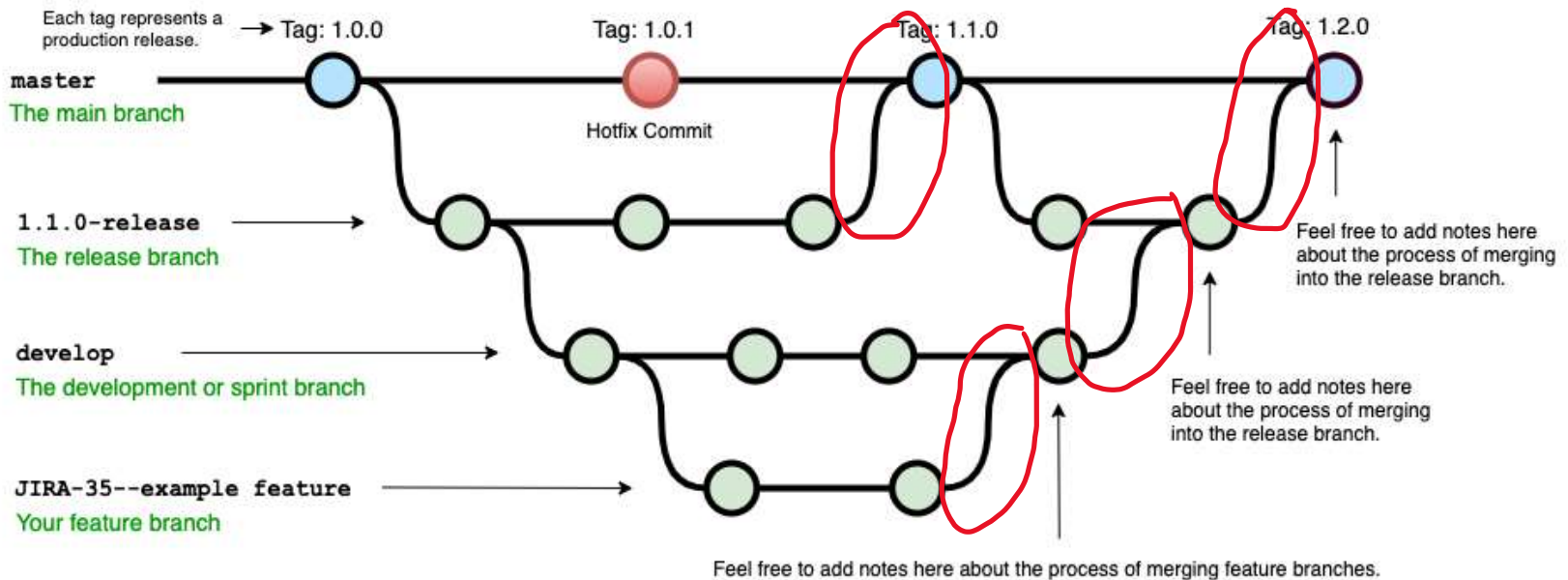


Merge

- Dùng để kết hợp các branch khác nhau lại với nhau

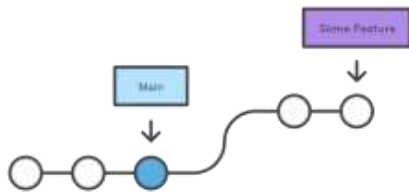
Example diagram for a workflow similar to "Git-flow" :

See: <https://nvie.com/posts/a-successful-git-branching-model/>

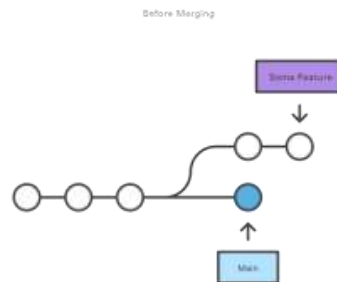
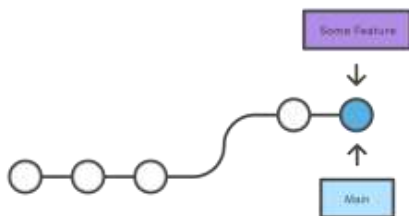


Merge

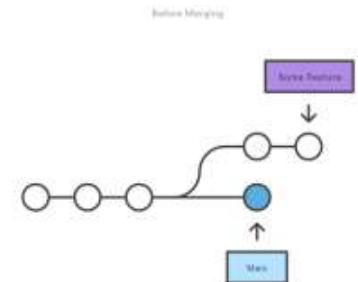
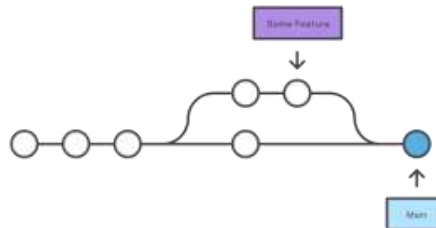
- Có 3 loại merge:
 - Fast forward merge.
 - 3-way merge.
 - Squash merge.



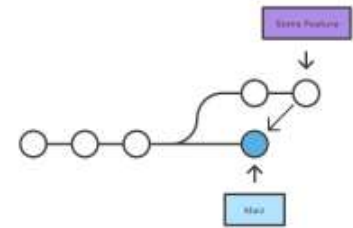
After a Fast-Forward Merge



After a 3-way Merge

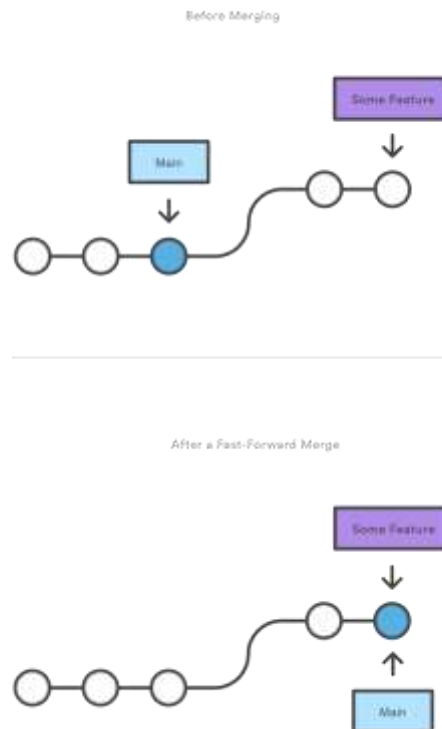


After a Squash Merge



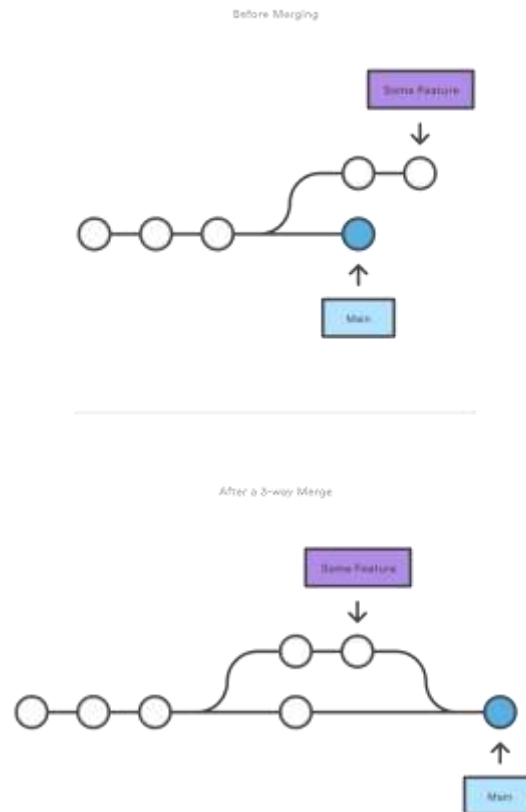
Fast forward merge

- Được tự động sử dụng khi branch được merge không có thay đổi gì.



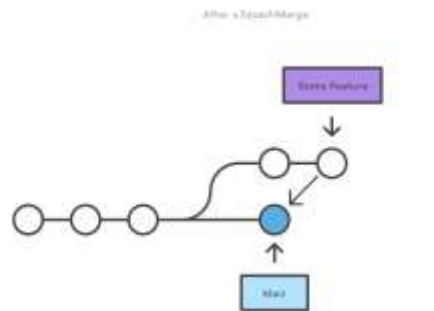
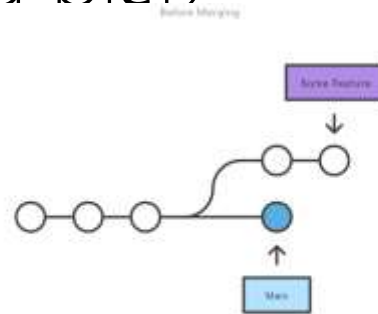
3-way merge

- Được sử dụng khi branch được merge có sự thay đổi nào đó.



Squash merge

- Một loại merge đặc biệt được sử dụng trong những trường hợp đặc biệt. (trường hợp nào đặc biệt thì chưa biết)



Merge

- Cấu trúc của lệnh merge

```
$ git merge -h
```

```
usage: git merge [<options>] [<commit>...]
```

```
or: git merge --abort
```

```
or: git merge --continue
```

- Flow của merge

- Push data của branch A
- Chuyển sang branch B để merge với branch A
- Cập nhật dữ liệu branch B nếu có
- Merge

Merge – command line

The screenshot displays the Visual Studio Code interface during a merge operation. The Explorer sidebar on the left shows the project structure, with 'GIT_TRAINING' selected. The main editor shows 'README.md' with a merge conflict. The bottom panel shows the 'COMMIT GRAPH' for 'GIT_TRAINING'.

Explorer Sidebar:

- OPEN EDITORS
 - main.cpp
 - README.md
 - Makefile
 - CMakeLists.txt
- GIT_TRAINING
 - build
 - env
 - lib
 - .gitignore
 - CMakeLists.txt
 - LICENSE
 - main.cpp
 - Makefile
 - README.md

Main Editor (README.md):

```
1 # git_training
2
3 PIF_TRAINING_DEMO
4
5 # How to use
6
7 Don't have any tutorial
8
9 # How to pull
10
11 Don't have any tutorial
12
```

Bottom Panel (COMMIT GRAPH: GIT_TRAINING):

BRANCH / TAG	COMMIT	COMMIT MESSAGE	AUTHOR
dev/compare	[Micks]	Add etc file	You
main	[Micks]	Update README.md	Micks Mika
develop	[Micks]	Add main.cpp	You
dev/math	[Micks]	Add Makefile file	You
	[Micks]	Update README	You

Merge – command line

The screenshot displays the Visual Studio Code interface for a C++ project. The Explorer panel on the left shows the project structure, including files like `main.cpp`, `README.md`, `compare.cpp`, `compare.h`, `Makefile`, and `CMakeLists.txt`. The main editor shows the `compare.cpp` file with the following code:

```
1 #include "compare.h"
2
3 #include <iostream>
4
5 void compare_number(int input_1, int input_2)
6 {
7     if (input_1 > input_2)
8     {
9         std::cout << "Number 1 is bigger then number 2";
10     }
11     else if (input_1 < input_2)
12     {
13         std::cout << "Number 1 is smaller then number 2";
14     }
15     else
16     {
17         std::cout << "Number 1 is equal number 2";
18     }
19
20     return;
21 }
```

The bottom panel shows the Git interface with a commit graph for the `dev/compare` branch. The graph shows a merge of the `main` branch into `dev/compare`. The commit history is as follows:

Branch	Commit	Message	Author
dev/compare	[Mick_2]	Add compare function	You
main	[Mick]	Add etc file	You
		Update README.md	Mick Mike
		Add main.cpp	You
		Add Makefile file	You

Merge – command line

The screenshot displays the Visual Studio Code interface. The Explorer panel on the left shows the project structure, including files like `main.cpp`, `README.md`, `math_func.cpp`, and `math_func.h`. The main editor window shows the content of `math_func.h`, which includes headers for `math_func.h` and `<iostream>`, and defines two functions: `plus` and `minus`. The Git Explorer panel at the bottom shows the commit history for the `dev/math` branch, with a merge of `dev/main` into `dev/math` in progress. The status bar at the bottom indicates the current branch is `dev/math`.

```
1  #include "math_func.h"
2
3  #include <iostream>
4
5  int plus(int input_1, int input_2)
6  {
7      return input_1 + input_2;
8  }
9
10 int minus(int input_1, int input_2)
11 {
12     return input_1 - input_2;
13 }
14
```

COMMIT GRAPH: GIT_TRAINING

git_training > dev/math ~ > Fetch (1 hour ago)

Search commits (↑ for history), e.g. "Updates dependencies" author:esmodis

BRANCH / FILE	GRAPH	COMMIT MESSAGE	AUTHOR
		Work in progress + 2	
dev/compare		[Mick] Add compare function	You
dev/math ~ +2		[Mick] Add etc File	You
		Update README.md	Mys Mika
		Add main.cpp	You
		Add README File	

Merge – command line

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git branch -l
  dev/compare
  dev/math
* develop
  main
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log dev/math --oneline
85c9115 (origin/dev/math, dev/math) [Micls_1] Add math func
83ced3a (HEAD -> develop, origin/main, origin/HEAD, main) [Micls] Add etc file
b827394 Update README.md
1c1bfdb Add main.cpp
9a25844 Add Makefile file
1256695 [Micls] Update README
a6394c3 Initial commit
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log dev/compare --oneline
cd0843b (origin/dev/compare, dev/compare) [Micls_2] Add compare function
83ced3a (HEAD -> develop, origin/main, origin/HEAD, main) [Micls] Add etc file
b827394 Update README.md
1c1bfdb Add main.cpp
9a25844 Add Makefile file
1256695 [Micls] Update README
a6394c3 Initial commit
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git log develop --oneline
83ced3a (HEAD -> develop, origin/main, origin/HEAD, main) [Micls] Add etc file
b827394 Update README.md
1c1bfdb Add main.cpp
9a25844 Add Makefile file
1256695 [Micls] Update README
a6394c3 Initial commit
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```

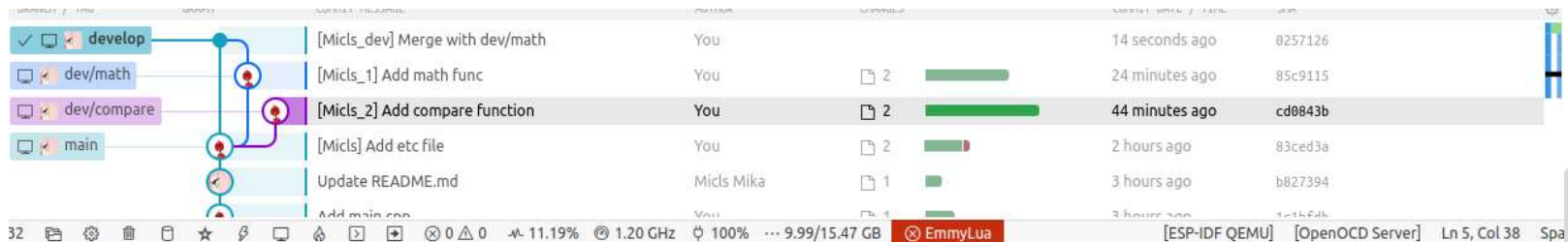
Merge – command line

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git branch -l
```

```
dev/compare
dev/math
* develop
main
```

Chọn 3-way merge và không tự com

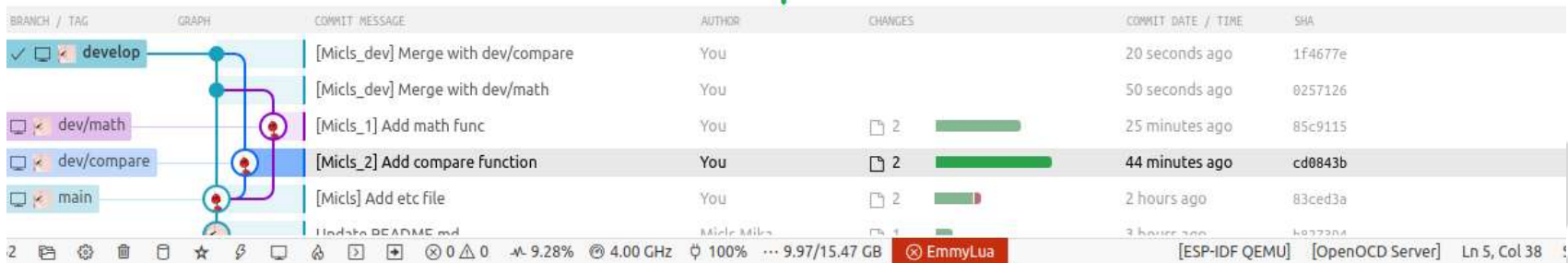
```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git merge dev/math --no-commit --no-ff
Automatic merge went well; stopped before committing as requested
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git commit -m "[Micls_dev] Merge with dev/math"
[develop 0257126] [Micls_dev] Merge with dev/math
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git push
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 226 bytes | 226.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:nhatmicls/git_training.git
83ced3a..0257126 develop -> develop
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```



Merge – command line

```
05ce03d..0257126 develop -> develop
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git merge dev/compare --no-commit --no-ff
Automatic merge went well; stopped before committing as requested
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git commit -m "[Micls_dev] Merge with dev/compare"
[develop 1f4677e] [Micls_dev] Merge with dev/compare
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 366 bytes | 366.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:nhatmicls/git_training.git
   0257126..1f4677e develop -> develop
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```

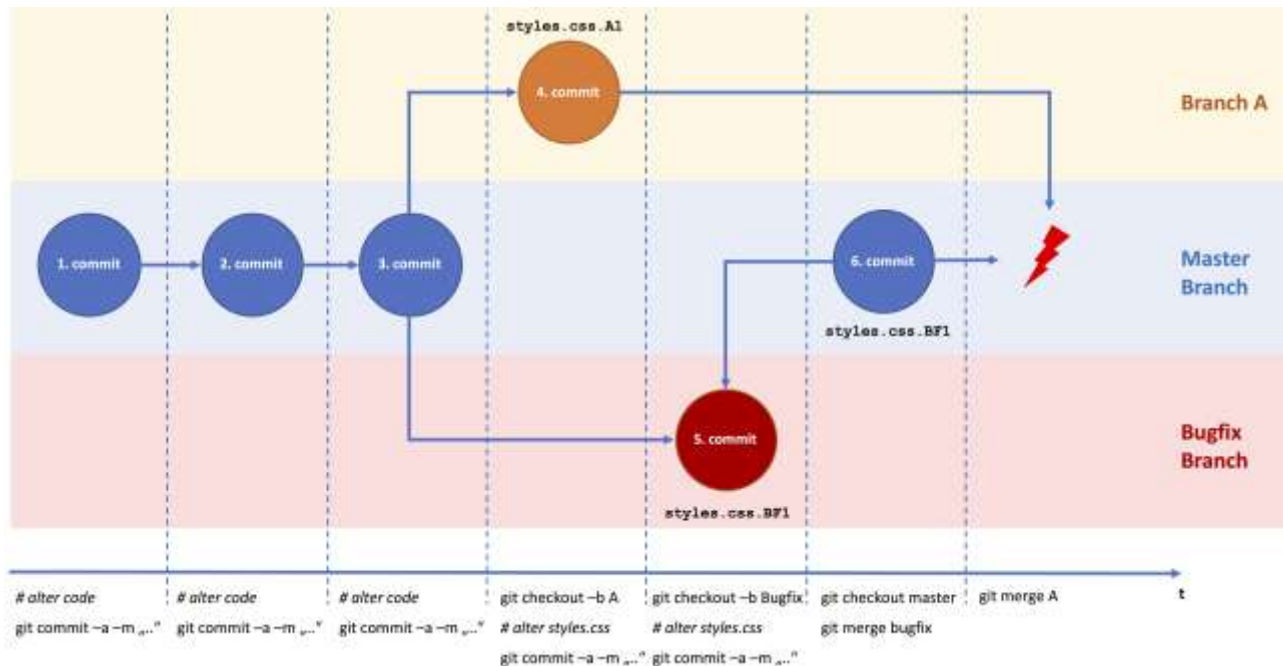
Chọn 3-way merge và không tự com



BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR	CHANGES	COMMIT DATE / TIME	SHA
✓ develop		[Micls_dev] Merge with dev/compare	You		20 seconds ago	1f4677e
		[Micls_dev] Merge with dev/math	You		50 seconds ago	0257126
dev/math		[Micls_1] Add math func	You	2	25 minutes ago	85c9115
dev/compare		[Micls_2] Add compare function	You	2	44 minutes ago	cd0843b
main		[Micls] Add etc file	You	2	2 hours ago	83ced3a
		Update README.md	hhatle Mike	1	3 hours ago	4e37204

Merge – Resolve conflict

- Conflict là hiện tượng xảy ra khi hai file đang sửa trên hai branch cùng một lúc, khiến cho git không biết dòng nào cần thay đổi.



Merge – Resolve conflict



main



develop

Merge – Resolve conflict

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git merge --no-commit --no-ff develop
Auto-merging main.cpp
CONFLICT (content): Merge conflict in main.cpp
Automatic merge failed; fix conflicts and then commit the result.
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```

```
406 → → → → → snippet: i
407 → → → → → });
408 → → → → → }));
409 → → → → → });

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
410 <<<<<< HEAD (Current Change)
411 → → → → → this.updateSizeClasses();
412 → → → → → this.multiCursorModifier();
413 → → → → → this.contentDisposables.push(this.configurationService.onDic
414 =====
415 → → → → → this.toggleSizeClasses();
416 >>>>>> Test (Incoming Change)
417 → → → → → if (input.onReady) {
418 → → → → →     input.onReady(innerContent);
419 → → → → → }
420 → → → → → this.scrollbar.scanDomNode();
421 → → → → → this.loadTextEditorViewState(input.getResource());
422 → → → → → this.updatedScrollPosition();
423 → → → → → });
424 → → → → → }
```

Merge – Resolve conflict

Accept current change

```
int main()
{
    std::cout << "hi\n";
    std::cout << "hello\n";
    std::cout << "bo\n";
    return 0;
}
```

Accept incoming

```
int main()
{
    std::cout << "hi\n";

    int new_1 = plus(1, 2);
    std::cout << "ANS: " << new_1 << "\n";

    return 0;
}
```

Accept both

```
int main()
{
    std::cout << "hi\n";
    std::cout << "hello\n";
    std::cout << "bo\n";

    int new_1 = plus(1, 2);
    std::cout << "ANS: " << new_1 << "\n";

    return 0;
}
```

Merge – Resolve conflict

- Có thể xài những trình code khác như VSCode, Vim (chưa thử nên không biết nó có không), ... để giải quyết các vấn đề conflict.

UNDO

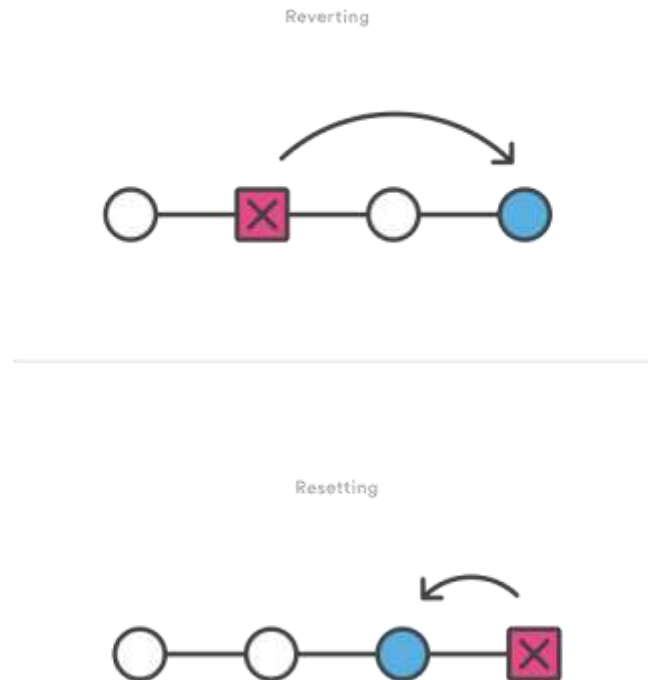


Redo – rewind – undo

- Để undo một hành động nào đó trong git thì bao gồm những trường hợp sau đây (chỉ gợi ý một số lỗi cho biết thế giới đã có cái đó :V):
 - Lỗi stage lộn file
 - Lỗi commit lộn
- Không những thế git đã được trang bị những tool để phục vụ cho những công việc như quay lại quá khứ mà không ảnh hưởng nhiều
 - GIT revert
 - GIT stash

Redo – rewind – undo

- Để redo một lỗi lầm nào đó thì chúng ta có những lệnh sau đây
 - git **reset**
 - git **revert**



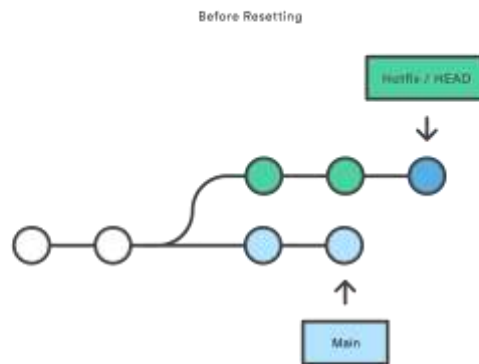
Redo – rewind – undo

- So sánh giữa revert – reset:
 - Commit – level:
 - Reset: xóa các commit ở các branch hoặc xóa các thay đổi chưa được commit.
 - Revert: undo các commit.
 - File – level:
 - Reset: unstage file.
 - Revert: N/A

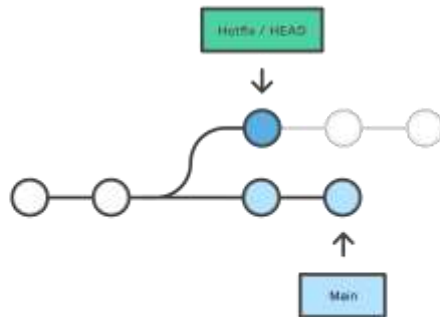
Redo – rewind – undo

- Commit – level:

Resetting the hotfix branch to HEAD-2

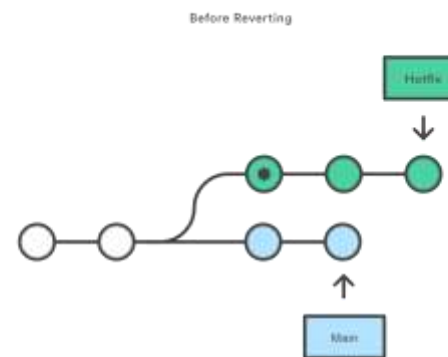


After Resetting



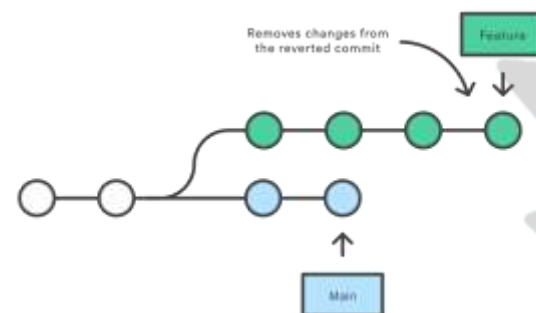
○ - Orphaned Commits

Reverting the 2nd to last commit



★ Commit to be reverted

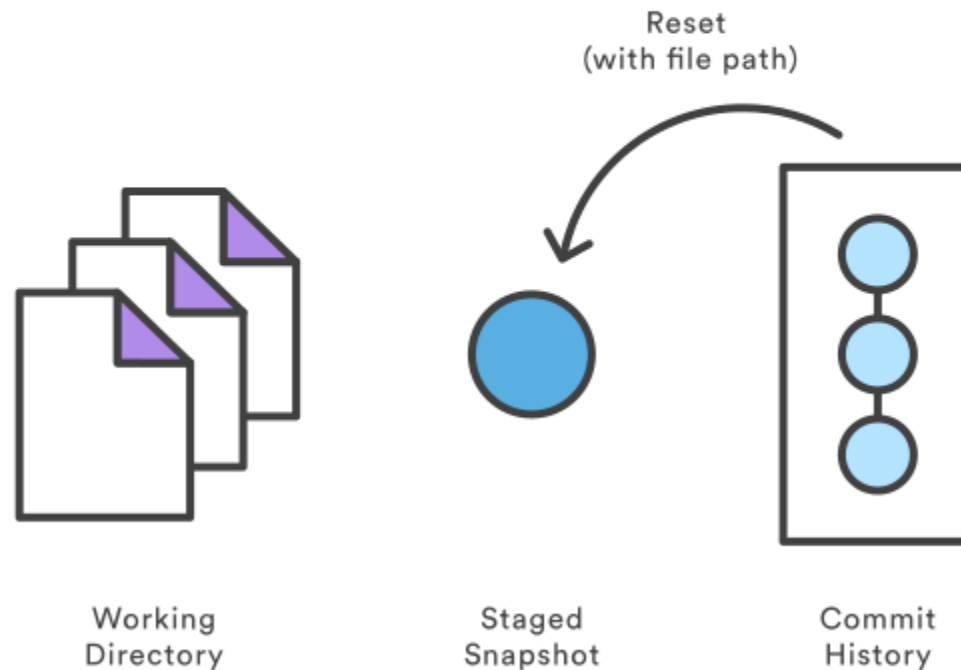
After Reverting



Redo – rewind – undo

- File – level:

Moving a file from the commit history into the staged snapshot



Redo – rewind – undo

- Để undo một commit nào đó thì người ta hay sử dụng git revert.
- GIT revert không thay đổi lịch sử commit → có thể redo tới 1 commit bất kì mà không có gì khó khăn.
- GIT reset sẽ xóa các commit cho tới khi tới commit mình mong muốn → không thể redo vì nó đã bị xóa vĩnh viễn.

git **revert**

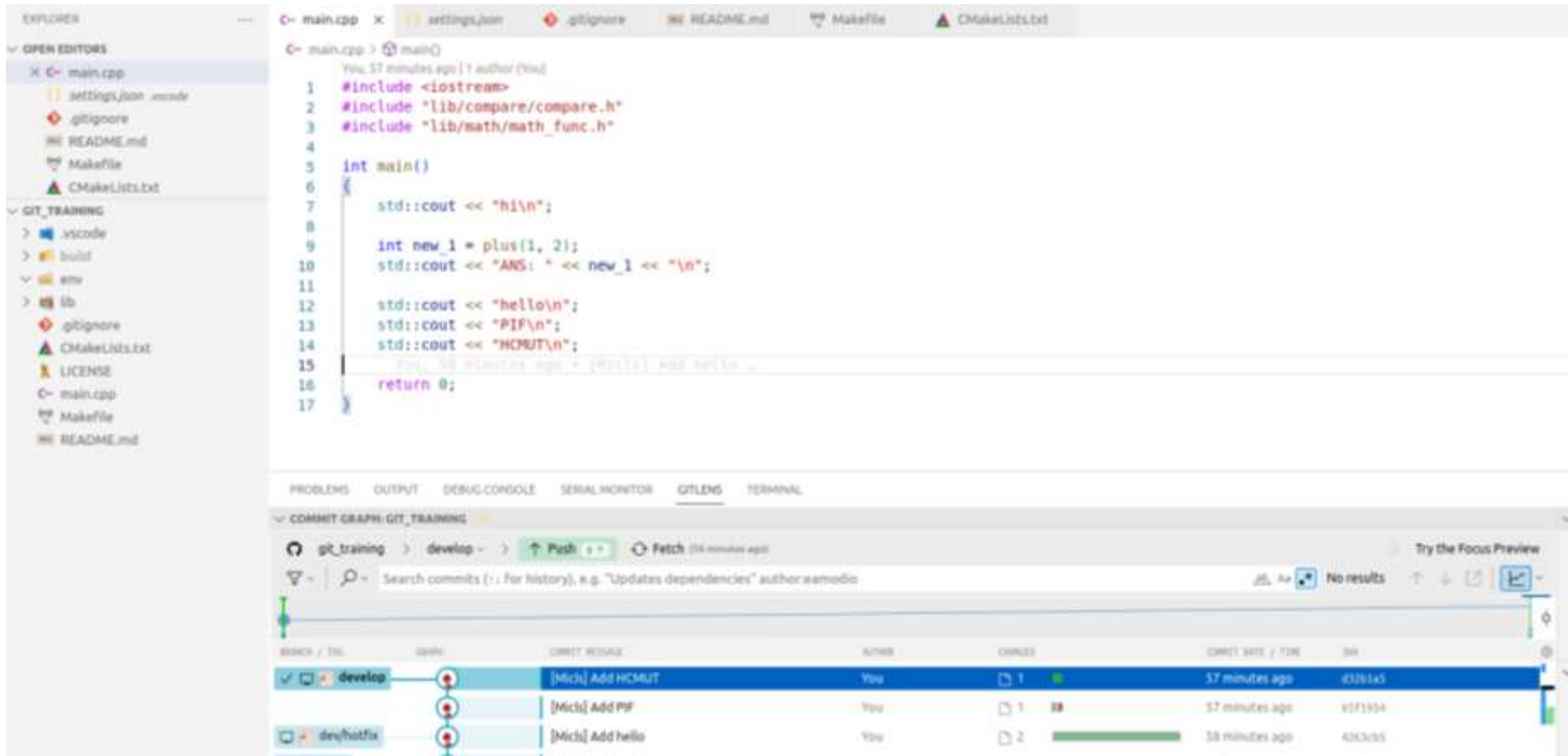
Cấu trúc:

```
$ git revert -h
```

```
usage: git revert [--[no-]edit] [-n] [-m <parent-number>] [-s] [-S[<keyid>]]  
<commit>...
```

```
or: git revert (--continue | --skip | --abort | --quit)
```

git revert



The screenshot displays the Visual Studio Code interface. The Explorer sidebar on the left shows the project structure, including files like `main.cpp`, `settings.json`, `.gitignore`, `README.md`, `Makefile`, and `CMakeLists.txt`. The main editor window shows the `main.cpp` file with the following content:

```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11
12    std::cout << "hello\n";
13    std::cout << "PIF\n";
14    std::cout << "HCMUT\n";
15
16    return 0;
17 }
```

Below the editor, the GitLens extension shows the commit graph for the `develop` branch. The graph includes a search bar and a table of commits:

BRANCH / FILE	GRAPH	COMMIT MESSAGE	AUTHOR	CHANGES	COMMIT DATE / TIME	SHA
✓ <code>develop</code>		[Mick] Add HCMUT	You	1	57 minutes ago	d3b1a5
		[Mick] Add PIF	You	1	57 minutes ago	95f1954
<code>dev/hotfix</code>		[Mick] Add hello	You	2	58 minutes ago	42c3c55

git revert

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git revert HEAD
[develop 5ab5835] Revert "[Micls] Add HCMUT"
 1 file changed, 1 deletion(-)
micls@micls-ROG-Strix-G531GD:~/Project/git_training$
```

The screenshot shows the Visual Studio Code interface. The top editor displays the file `main.cpp` with the following content:

```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11
12    std::cout << "hello\n";
13    std::cout << "PIF\n";
14
15    return 0;
16 }
```

The bottom panel shows the 'COMMIT GRAPH: GIT_TRAINING' view. It displays a list of commits on the `develop` branch:

Branch / SHA	Comm	Commit Message	Author	Changes	Commit Date / Time	Age
develop	5ab5835	Revert "[Micls] Add HCMUT" [This reverts commit ...]	You	1	9 seconds ago	5ab5835
develop	413b1a6	[Micls] Add HCMUT	You	1	37 minutes ago	413b1a6
dev/hello	65f1934	[Micls] Add PIF	You	1	38 minutes ago	65f1934
dev/hello	4263c33	[Micls] Add hello	You	2	38 minutes ago	4263c33
main	461f8b6	[Micls] Edit gitignore	You	1	17 hours ago	461f8b6
main	88613d6	[Micls] Merge develop branch	You	1	17 hours ago	88613d6

7/8/2023

git **reset**

Cấu trúc:

```
$ git reset -h
```

```
usage: git reset [--mixed | --soft | --hard | --merge | --keep] [-q] [<commit>]
```

```
or: git reset [-q] [<tree-ish>] [--] <pathspec>...
```

```
or: git reset [-q] [--pathspec-from-file [--pathspec-file-nul]] [<tree-ish>]
```

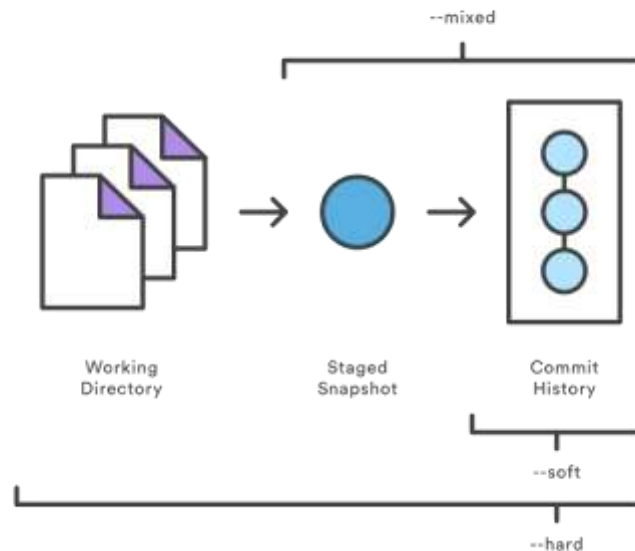
```
or: git reset --patch [<tree-ish>] [--] [<pathspec>...]
```

```
or: DEPRECATED: git reset [-q] [--stdin [-z]] [<tree-ish>]
```

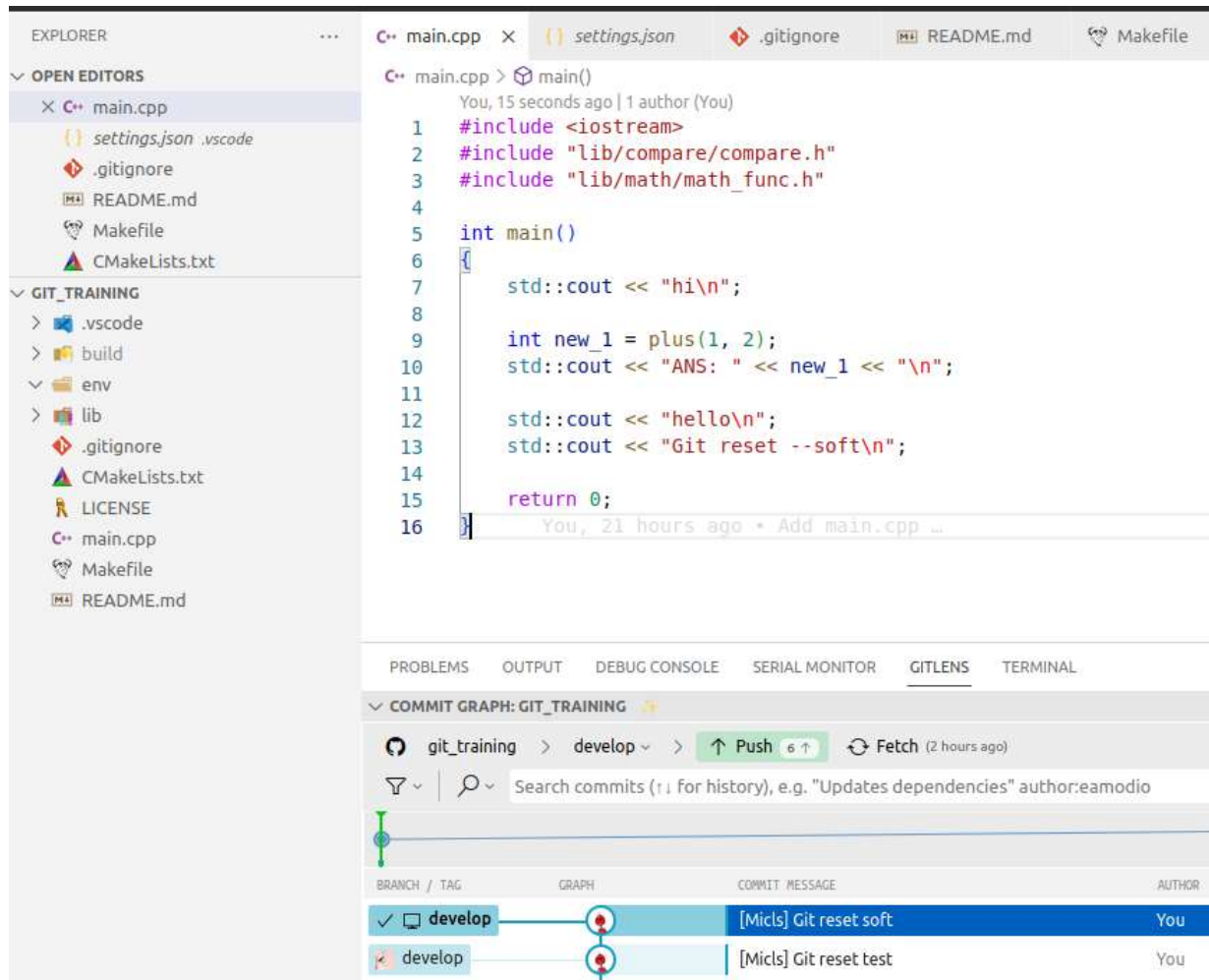
git reset

- Git reset gồm 3 mức độ
 - Soft: xóa commit
 - Mixed: xóa commit và unstage
 - Hard: mọi file được đưa về lại HEAD trên git

The scope of git reset's modes



git reset - soft



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays the project structure for 'GIT_TRAINING', including files like .vscode, build, env, lib, .gitignore, CMakeLists.txt, LICENSE, main.cpp, Makefile, and README.md. The main editor shows the content of 'main.cpp', which includes headers for <iostream>, lib/compare/compare.h, and lib/math/math_func.h. The main function prints 'hi\n', the result of plus(1, 2), 'hello\n', and 'Git reset --soft\n'. The bottom panel shows the 'GITLENS' view with a commit graph for the 'develop' branch. The graph shows two commits: a recent one by 'You' with the message '[Mics] Git reset soft', and a previous one by 'You' with the message '[Mics] Git reset test'.

```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11
12    std::cout << "hello\n";
13    std::cout << "Git reset --soft\n";
14
15    return 0;
16 }
```

BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR
✓ develop	(commit icon)	[Mics] Git reset soft	You
develop	(commit icon)	[Mics] Git reset test	You

git reset - soft

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch develop
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

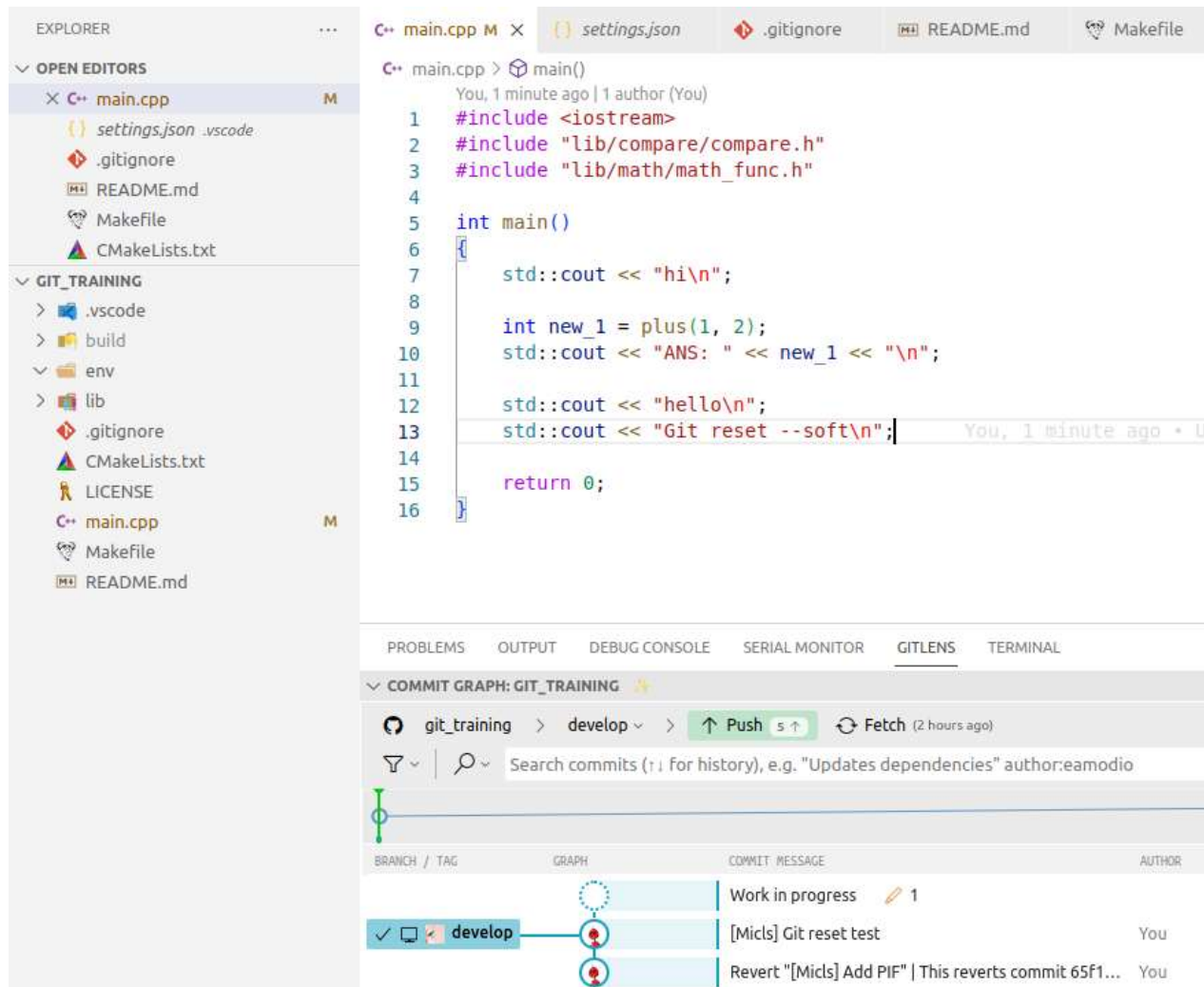
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.cpp

no changes added to commit (use "git add" and/or "git commit -a")
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git add main.cpp
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git commit -m "[Micls] Git reset soft"
[develop c932999] [Micls] Git reset soft
 1 file changed, 1 insertion(+), 1 deletion(-)
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git reset --soft HEAD~
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git status
On branch develop
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.cpp

micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```

git reset - soft



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the file structure of a project named 'GIT_TRAINING'. The file 'main.cpp' is open in the editor. The code in 'main.cpp' includes headers for `<iostream>`, `lib/compare/compare.h`, and `lib/math/math_func.h`. It defines a `main()` function that prints 'hi\n', calculates the sum of 1 and 2, prints 'ANS: ' followed by the result, prints 'hello\n', and prints 'Git reset --soft\n'. The bottom panel shows the 'GIT TRAINING' commit graph. The 'develop' branch is highlighted, showing a sequence of commits: 'Work in progress', '[Mics] Git reset test', and 'Revert "[Mics] Add PIF" | This reverts commit 65f1...'. The 'Git reset test' commit is the current state of the 'develop' branch.

EXPLORER

OPEN EDITORS

- main.cpp
- settings.json
- .gitignore
- README.md
- Makefile
- CMakeLists.txt

GIT_TRAINING

- .vscode
- build
- env
- lib
- .gitignore
- CMakeLists.txt
- LICENSE
- main.cpp
- Makefile
- README.md

main.cpp

```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11
12    std::cout << "hello\n";
13    std::cout << "Git reset --soft\n";
14
15    return 0;
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE SERIAL MONITOR GITLENS TERMINAL

COMMIT GRAPH: GIT_TRAINING

git_training > develop > Push 5 ↑ Fetch (2 hours ago)

Search commits (↑ for history), e.g. "Updates dependencies" author:eamodio

BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR
develop		Work in progress	You
		[Mics] Git reset test	You
		Revert "[Mics] Add PIF" This reverts commit 65f1...	You

git **reset** - mixed

The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows the file structure of a project named 'GIT_TRAINING'. The main editor displays the file 'main.cpp' with the following code:

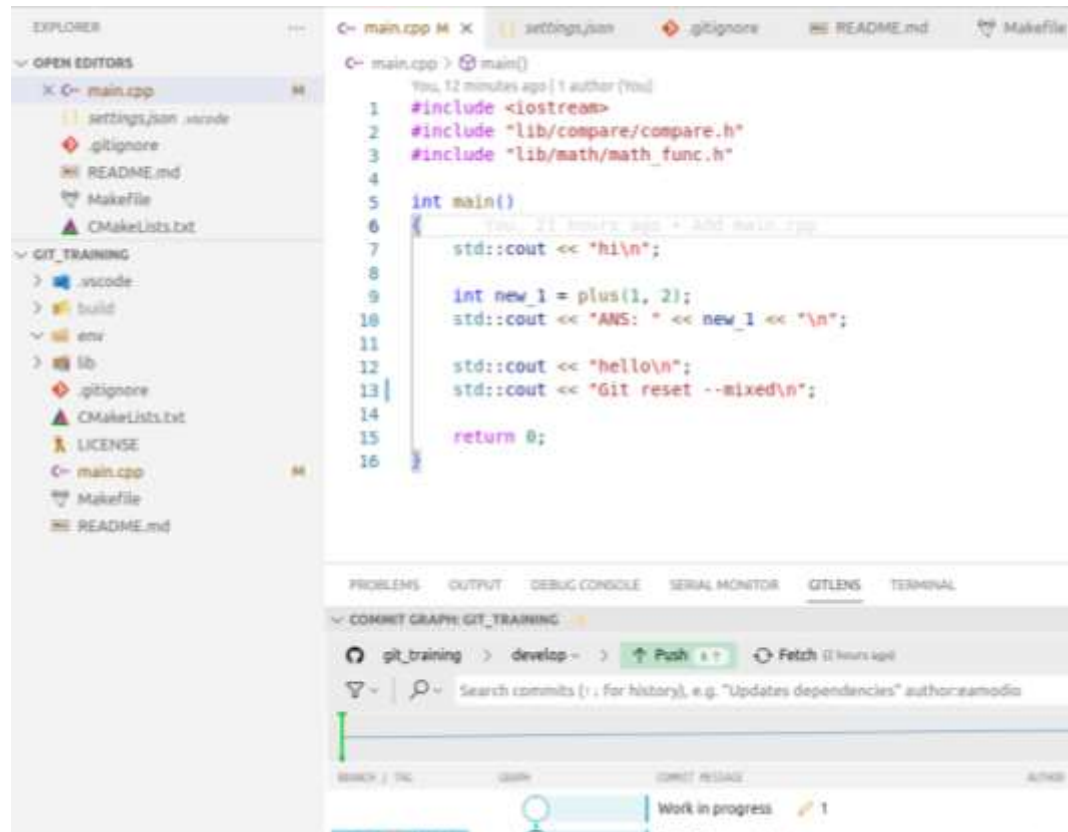
```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11    You, 18 hours ago * [Micl's dev] Implemen new func ...
12    std::cout << "hello\n";
13    std::cout << "Git reset --mixed\n";
14
15    return 0;
16 }
```

Below the editor, the Git pane shows the 'COMMIT GRAPH: GIT_TRAINING'. The current branch is 'develop'. The commit history is as follows:

BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR
✓ develop	(commit)	[Micl's] Git reset mixed	You
develop	(commit)	[Micl's] Git reset test	You
	(commit)	Revert "[Micl's] Add PIF" This reverts commit 65f1...	You

git **reset** - mixed

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git add .
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git commit -m "[Micls] Git reset mixed"
[develop 8918fb1] [Micls] Git reset mixed
1 file changed, 1 insertion(+), 1 deletion(-)
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git reset --mixed HEAD~
Unstaged changes after reset:
M   main.cpp
```



git **reset** - hard

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays the file structure of a project named 'GIT_TRAINING'. The Open Editors panel shows 'main.cpp' as the active file. The main editor area displays the content of 'main.cpp', which includes headers for `<iostream>`, `lib/compare/compare.h`, and `lib/math/math_func.h`. The `main` function prints 'hi', the result of `plus(1, 2)`, 'hello', and messages about a 'Git reset --hard' and 'Git test'. The bottom panel shows the 'GITLENS' view, specifically the 'COMMIT GRAPH: GIT_TRAINING'. It indicates the current branch is 'develop' and shows a series of commits. The most recent commit, '[Micl] Git reset test' by 'You', is highlighted in blue. Below it are two other commits: 'Revert "[Micl] Add PIF" | This reverts commit 65f1...' and '[Micl] Add HOMIT'.

```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11
12    std::cout << "hello\n";
13    std::cout << "Git reset --hard\n";
14    std::cout << "Git test\n";
15
16    return 0;
17 }
```

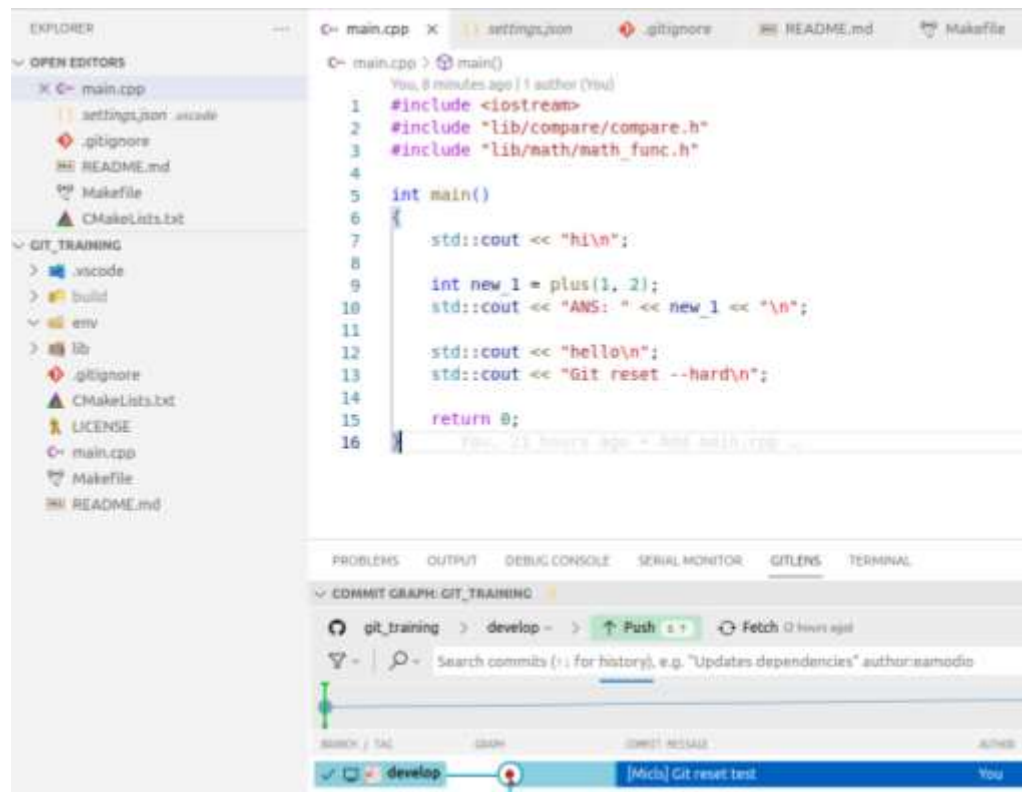
git_training > develop > Push 5 ↑ Fetch (2 hours ago)

Search commits (↑ for history), e.g. "Updates dependencies" author:eamodio

BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR
		Work in progress 1	
✓ develop		[Micl] Git reset test	You
		Revert "[Micl] Add PIF" This reverts commit 65f1...	You
		[Micl] Add HOMIT	You

git ~~reset~~ - hard

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git add .
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git commit -m "[Micls] git reset hard"
[develop e458af1] [Micls] git reset hard
1 file changed, 1 insertion(+)
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git reset --hard HEAD
HEAD is now at e458af1 [Micls] git reset hard
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git reset --hard HEAD~
HEAD is now at 6eba769 [Micls] Git reset test
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```



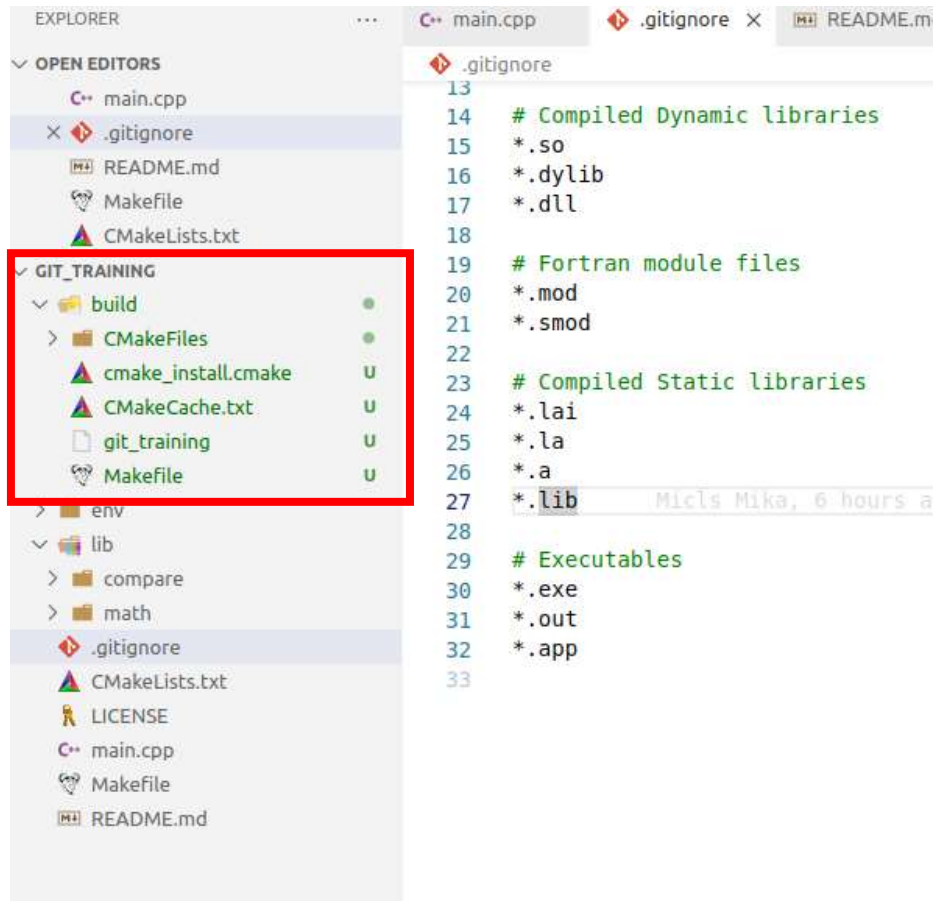
IGNORE

Ignore

- Hay còn gọi là “.gitignore”
- File này giúp cho git biết những file, folder nào git sẽ không cần quan tâm trong quá trình track file, stage,
- .gitignore có hai loại:
 - Local gitignore: là file .gitignore nằm trong sub folder
 - Global gitignore: là file .gitignore nằm trong main folder

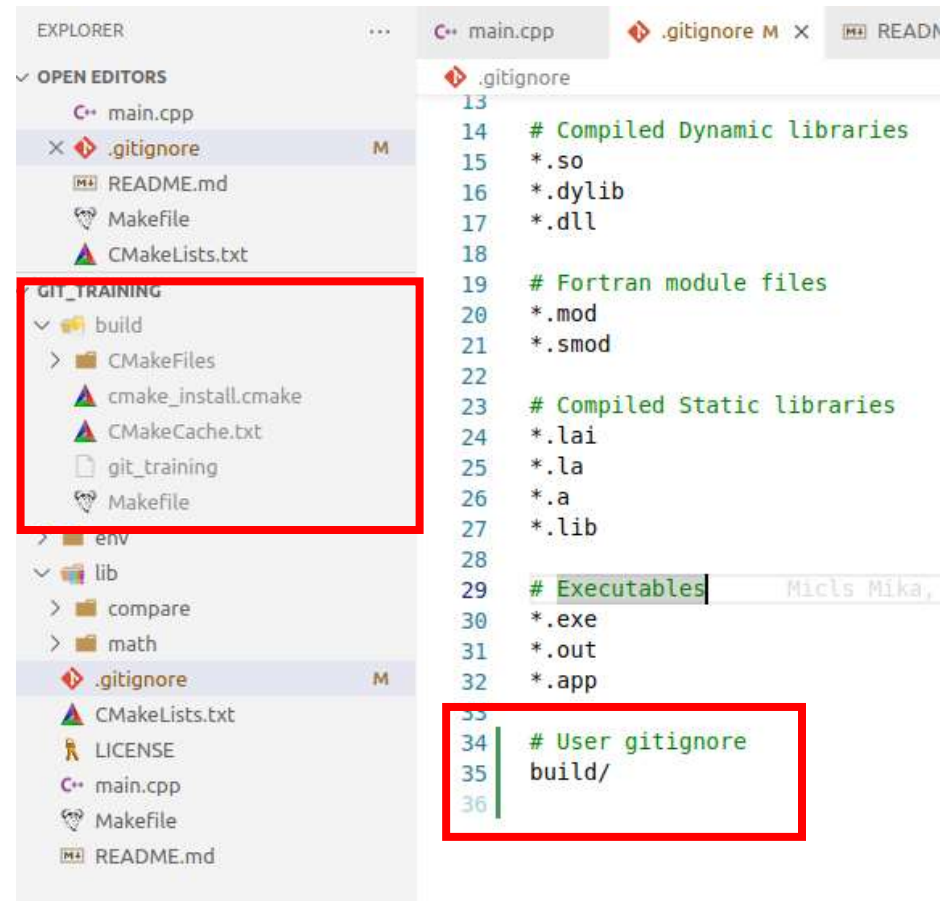
Ignore

VD



This screenshot shows the VS Code Explorer on the left with a red box highlighting the `build` directory under `GIT_TRAINING`. The `.gitignore` file is open in the center editor, showing the following content:

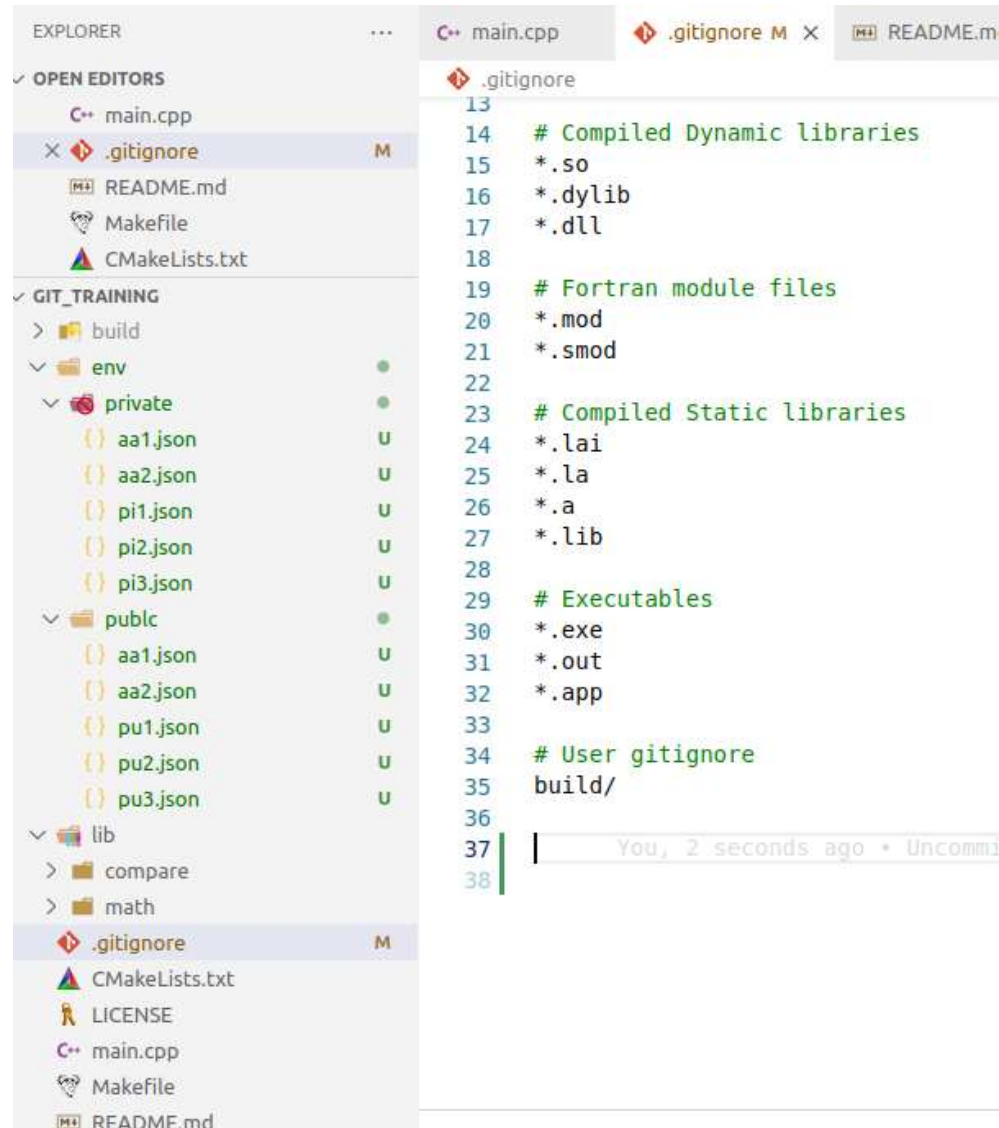
```
13
14 # Compiled Dynamic libraries
15 *.so
16 *.dylib
17 *.dll
18
19 # Fortran module files
20 *.mod
21 *.smod
22
23 # Compiled Static libraries
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
```



This screenshot shows the VS Code Explorer on the left with a red box highlighting the `build` directory under `GIT_TRAINING`. The `.gitignore` file is open in the center editor, showing the following content:

```
13
14 # Compiled Dynamic libraries
15 *.so
16 *.dylib
17 *.dll
18
19 # Fortran module files
20 *.mod
21 *.smod
22
23 # Compiled Static libraries
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
34 # User gitignore
35 build/
36
```

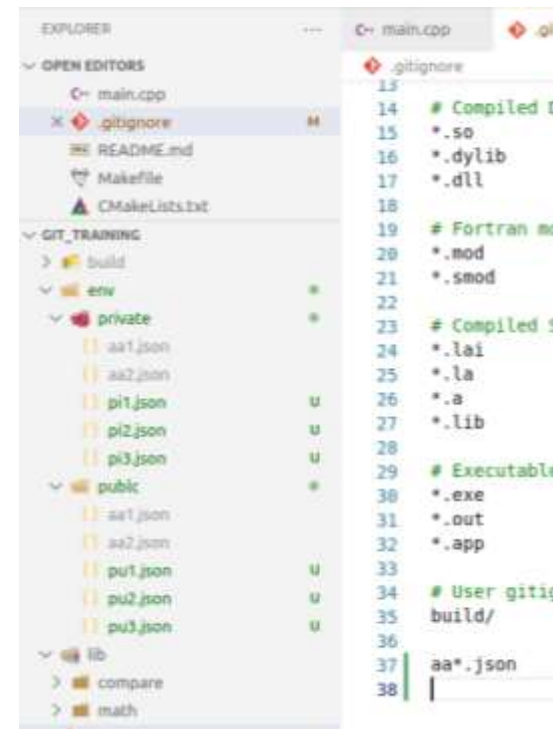
Ignore



```
13
14 # Compiled Dynamic libraries
15 *.so
16 *.dylib
17 *.dll
18
19 # Fortran module files
20 *.mod
21 *.smod
22
23 # Compiled Static libraries
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
34 # User gitignore
35 build/
36
37 | You, 2 seconds ago • Uncommi
38
```

Ignore

- Global(a.k.a Glob) pattern
 - Dấu sao(*): một hoặc nhiều giá trị bất kì nào cũng được chọn.
 - Dấu hỏi(?): một giá trị bất kì nào cũng được chọn.
 - [abc]: nếu giá trị trùng với giá trị nào trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a, b và c).
 - [a-c]: nếu giá trị trùng với giá trị nào trong khoảng kí tự trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a tới c).
- Hai dấu sao(**): mọi thư mục.



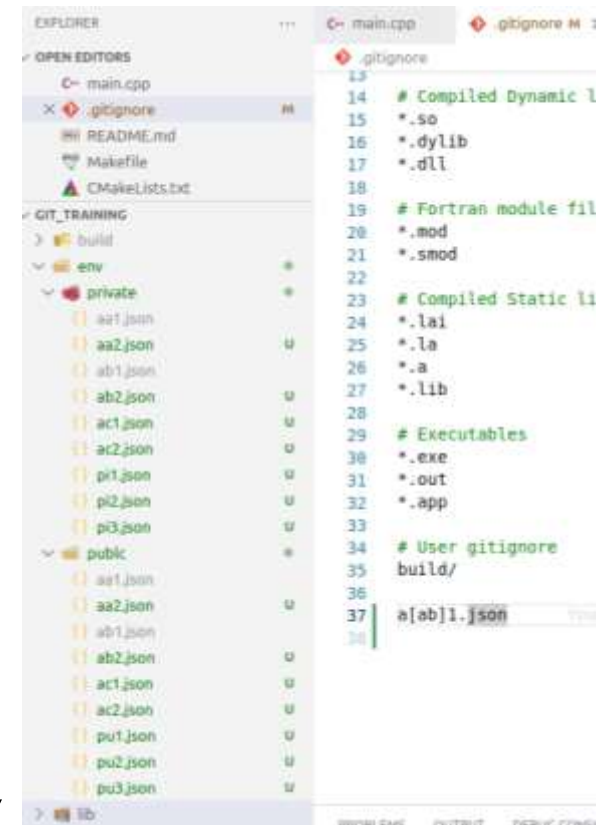
Ignore

- Global(a.k.a Glob) pattern
 - Dấu sao(*): một hoặc nhiều giá trị bất kì nào cũng được chọn.
 - Dấu hỏi(?): một giá trị bất kì nào cũng được chọn.
 - [abc]: nếu giá trị trùng với giá trị nào trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a, b và c).
 - [a-c]: nếu giá trị trùng với giá trị nào trong khoảng kí tự trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a tới c).
 - Hai dấu sao(**): mọi thư mục.



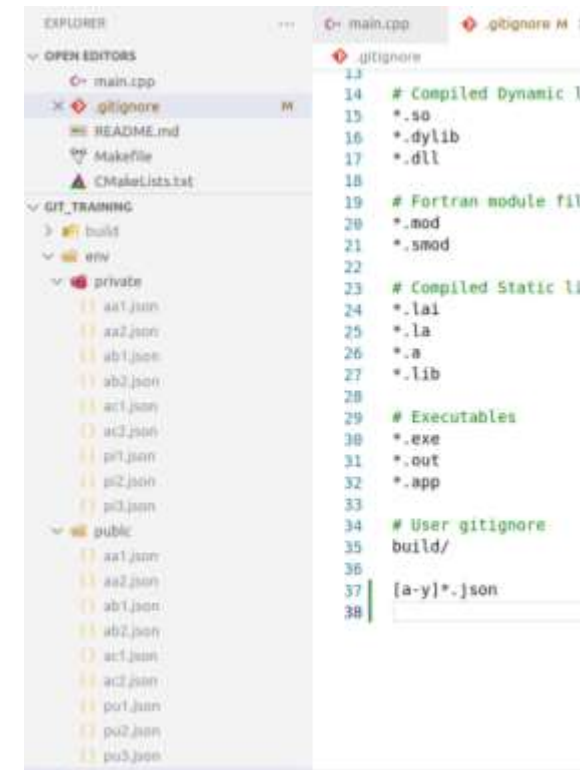
Ignore

- Global(a.k.a Glob) pattern
 - Dấu sao(*): một hoặc nhiều giá trị bất kì nào cũng được chọn.
 - Dấu hỏi(?): một giá trị bất kì nào cũng được chọn.
 - [abc]: nếu giá trị trùng với giá trị nào trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a, b và c).
 - [a-c]: nếu giá trị trùng với giá trị nào trong khoảng kí tự trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a tới c).
 - Hai dấu sao(**): mọi thư mục.



Ignore

- Global(a.k.a Glob) pattern
 - Dấu sao(*): một hoặc nhiều giá trị bất kì nào cũng được chọn.
 - Dấu hỏi(?): một giá trị bất kì nào cũng được chọn.
 - [abc]: nếu giá trị trùng với giá trị nào trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a, b và c).
 - [a-c]: nếu giá trị trùng với giá trị nào trong khoảng kí tự trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a tới c).
- Hai dấu sao(**): mọi thư mục.



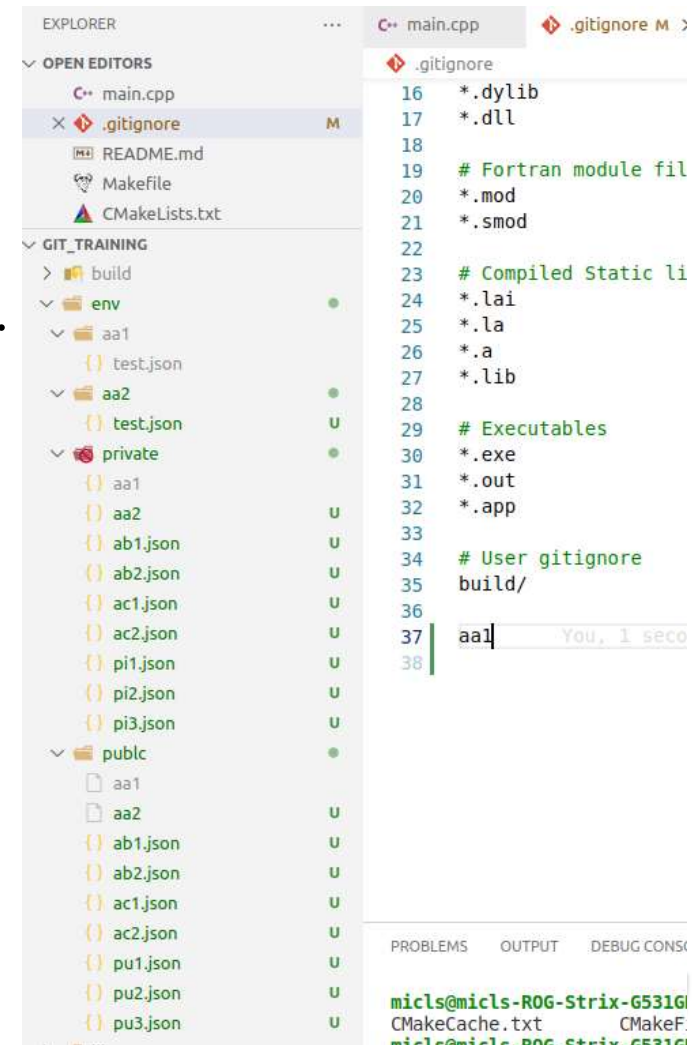
Ignore

- Global(a.k.a Glob) pattern
 - Dấu sao(*): một hoặc nhiều giá trị bất kì nào cũng được chọn.
 - Dấu hỏi(?): một giá trị bất kì nào cũng được chọn.
 - [abc]: nếu giá trị trùng với giá trị nào trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a, b và c).
 - [a-c]: nếu giá trị trùng với giá trị nào trong khoảng kí tự trong ngoặc vuông thì sẽ được chọn(trong trường hợp này là a tới c).
 - Hai dấu sao(**): mọi thư mục.



Ignore

- Gitignore patterns
 - Dòng trống hoặc các dòng bắt đầu bằng dấu # sẽ được bỏ qua.
 - .gitignore sẽ được áp dụng cho toàn bộ thư mục làm việc
 - Dấu slash(/) phía trước để không loại trừ các file trùng tên folder.
 - Dấu slash(/) phía sau để loại trừ các folder.
 - Dấu chấm than(!) để đảo các giá trị phía trước.



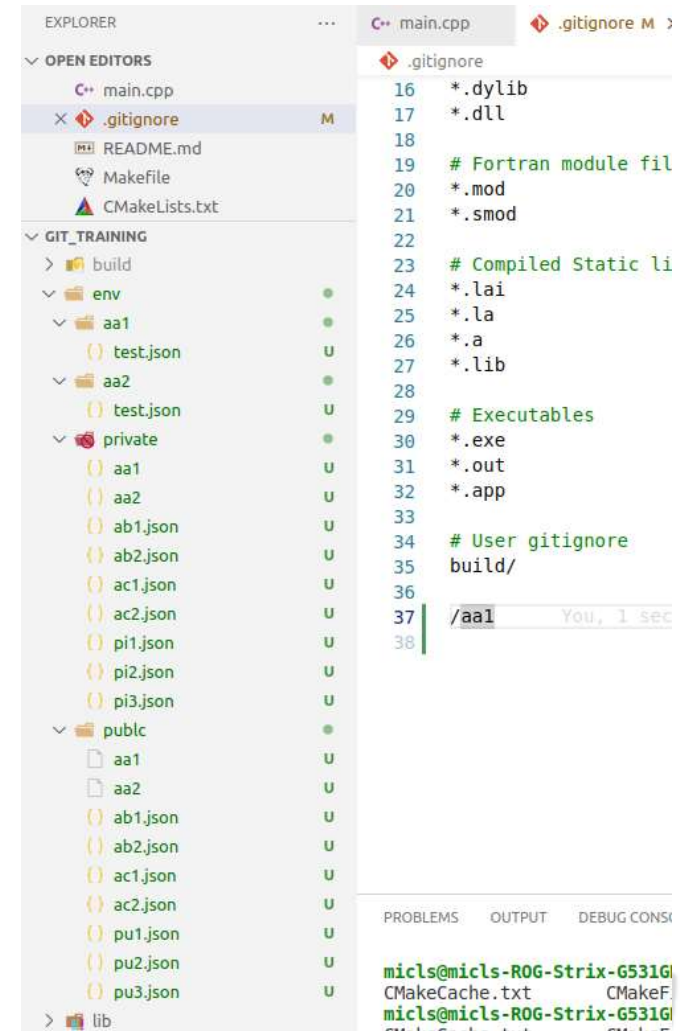
The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane displays a project structure under 'GIT_TRAINING'. It includes folders like 'build', 'env', 'aa1', 'aa2', 'private', and 'public'. The 'private' folder is highlighted. On the right, the Editor pane shows the content of the '.gitignore' file. The file contains patterns for ignoring various file types and folders. The patterns include: *.dylib, *.dll, # Fortran module file, *.mod, *.smod, # Compiled Static li, *.lai, *.la, *.a, *.lib, # Executables, *.exe, *.out, *.app, # User gitignore, build/, and aa1. The line 'aa1' is currently being edited.

```
16 *.dylib
17 *.dll
18
19 # Fortran module file
20 *.mod
21 *.smod
22
23 # Compiled Static li
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
34 # User gitignore
35 build/
36
37 aa1
38
```

Ignore

- Gitignore patterns

- Dòng trống hoặc các dòng bắt đầu bằng dấu # sẽ được bỏ qua.
- .gitignore sẽ được áp dụng cho toàn bộ thư mục làm việc
- Dấu slash(/) phía trước để không loại trừ các file trùng tên folder.
- Dấu slash(/) phía sau để loại trừ các folder.
- Dấu chấm than(!) để đảo các giá trị phía trước.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a project structure for 'GIT_TRAINING'. It includes folders like 'build', 'env', 'aa1', 'aa2', 'private', and 'public', each containing various files like 'test.json', 'aa1', 'aa2', 'ab1.json', 'ab2.json', 'ac1.json', 'ac2.json', 'pu1.json', 'pu2.json', and 'pu3.json'. The 'private' folder is highlighted. On the right, the Editor pane shows the '.gitignore' file with the following content:

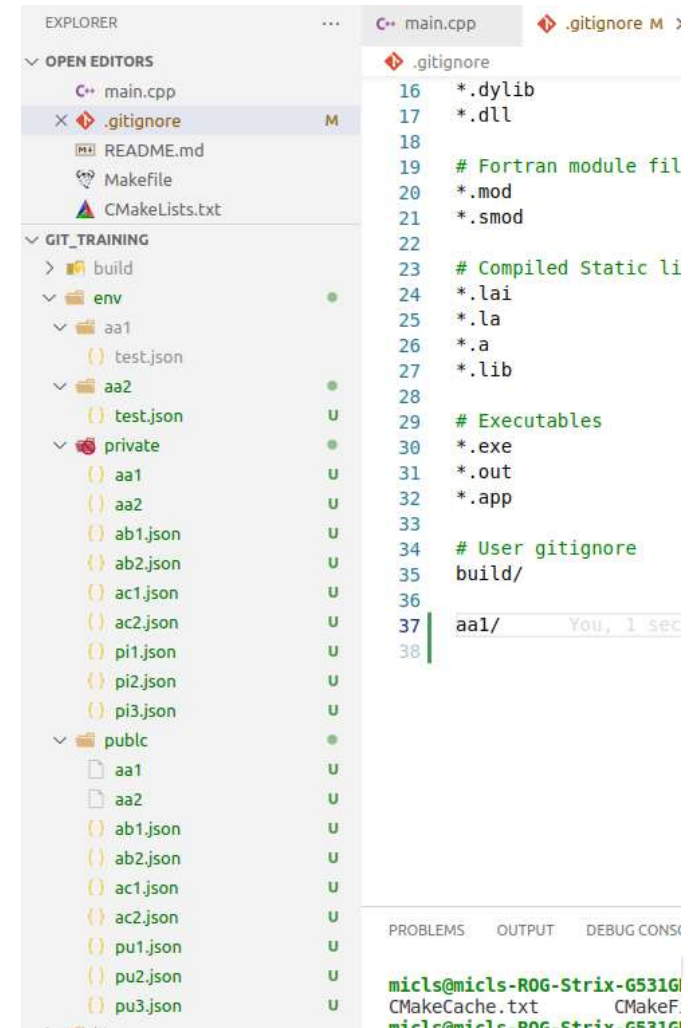
```
16 *.dylib
17 *.dll
18
19 # Fortran module fil
20 *.mod
21 *.smod
22
23 # Compiled Static li
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
34 # User gitignore
35 build/
36
37 /aa1 You, 1 sec
38
```

At the bottom, the PROBLEMS pane shows error messages related to CMakeCache.txt files.

Ignore

- Gitignore patterns

- Dòng trống hoặc các dòng bắt đầu bằng dấu # sẽ được bỏ qua.
- .gitignore sẽ được áp dụng cho toàn bộ thư mục làm việc
- Dấu slash(/) phía trước để không loại trừ các file trùng tên folder.
- Dấu slash(/) phía sau để loại trừ các folder.
- Dấu chấm than(!) để đảo các giá trị phía trước.




The screenshot shows a code editor with two panels. The left panel, titled 'EXPLORER', displays a project structure for 'GIT_TRAINING'. It includes a 'build' directory, an 'env' directory, and a 'public' directory. The 'env' directory contains subdirectories 'aa1' and 'aa2', each with a 'test.json' file. The 'public' directory contains subdirectories 'aa1' and 'aa2', each with files 'ab1.json', 'ab2.json', 'ac1.json', 'ac2.json', 'pu1.json', 'pu2.json', and 'pu3.json'. The right panel shows the content of the '.gitignore' file, which lists various file types to be ignored, including '*.dylib', '*.dll', Fortran module files, compiled static libraries, executables, and user-specific gitignore files. The file also includes a comment about the 'build/' directory and a pattern 'aa1/'.

```
.gitignore
16 *.dylib
17 *.dll
18
19 # Fortran module fil
20 *.mod
21 *.smod
22
23 # Compiled Static li
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
34 # User gitignore
35 build/
36
37 aa1/ You, 1 sec
38
```

Ignore

- Gitignore patterns
 - Dòng trống hoặc các dòng bắt đầu bằng dấu # sẽ được bỏ qua.
 - .gitignore sẽ được áp dụng cho toàn bộ thư mục làm việc
 - Dấu slash(/) phía trước để không loại trừ các file trùng tên folder.
 - Dấu slash(/) phía sau để loại trừ các folder.
 - Dấu chấm than(!) để đảo các giá trị phía trước.



The screenshot shows a code editor with two panels. The left panel, titled 'EXPLORER', displays a project structure under 'GIT_TRAINING'. It includes folders like 'build', 'env', 'aa1', 'aa2', 'private', and 'public', each containing various files such as 'test.json', 'ab1.json', 'ab2.json', 'ac1.json', 'ac2.json', 'pi1.json', 'pi2.json', 'pi3.json', 'aa1', 'aa2', 'ab1.json', 'ab2.json', 'ac1.json', 'ac2.json', 'pu1.json', 'pu2.json', and 'pu3.json'. The right panel shows the content of the '.gitignore' file, which lists patterns to ignore: '*.dylib', '*.dll', Fortran module files (*.mod, *.smod), Compiled Static Libraries (*.lai, *.la, *.a, *.lib), Executables (*.exe, *.out, *.app), and a user-defined pattern 'build/'. It also includes a comment '# User gitignore' and a pattern '/env/**/aa*' with an exception '!aa1/'.

```
.gitignore
16 *.dylib
17 *.dll
18
19 # Fortran module file
20 *.mod
21 *.smod
22
23 # Compiled Static li
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
34 # User gitignore
35 build/
36
37 /env/**/aa*
38 !aa1/
39 |
```

GIT/GITHUB ADVANCED

Git/Github advanced

- Git stash
- README.md
- Pull request
- Issue
- `Git add -p "file_name"`

README.md

README.md

- README.md là một dạng file text được nâng cấp 1 xíu về đồ họa dựa trên 1 ngôn ngữ tên Markdown
- README.md như một bộ mặt của bất kì một project nào trên git – người ta luôn đọc README trước khi tìm hiểu code đó làm gì.
 - ➔ Không có README.md thì một project có xin cách mấy thì cũng không được người khác chú ý.
- README.md bao gồm những giới thiệu chung về project, hướng dẫn, reference,

README.md – basic syntax

Element	Markdown Syntax
Heading	<code># H1</code> <code>## H2</code> <code>### H3</code>
Bold	<code>**bold text**</code>
Italic	<code>*italicized text*</code>
Blockquote	<code>> blockquote</code>
Ordered List	<code>1. First item</code> <code>2. Second item</code> <code>3. Third item</code>
Unordered List	<code>- First item</code> <code>- Second item</code> <code>- Third item</code>
Code	<code>`code`</code>
Horizontal Rule	<code>---</code>
Link	<code>[title](https://www.example.com)</code>
Image	<code>![alt text](image.jpg)</code>



README.md – extended syntax

Element	Markdown Syntax
Table	<pre> Syntax Description ----- ----- Header Title Paragraph Text </pre>
Fenced Code Block	<pre>``` { "firstName": "John", "lastName": "Smith", "age": 25 } ```</pre>
Footnote	<pre>Here's a sentence with a footnote. [^1] [^1]: This is the footnote.</pre>
Heading ID	<pre>### My Great Heading {#custom-id}</pre>

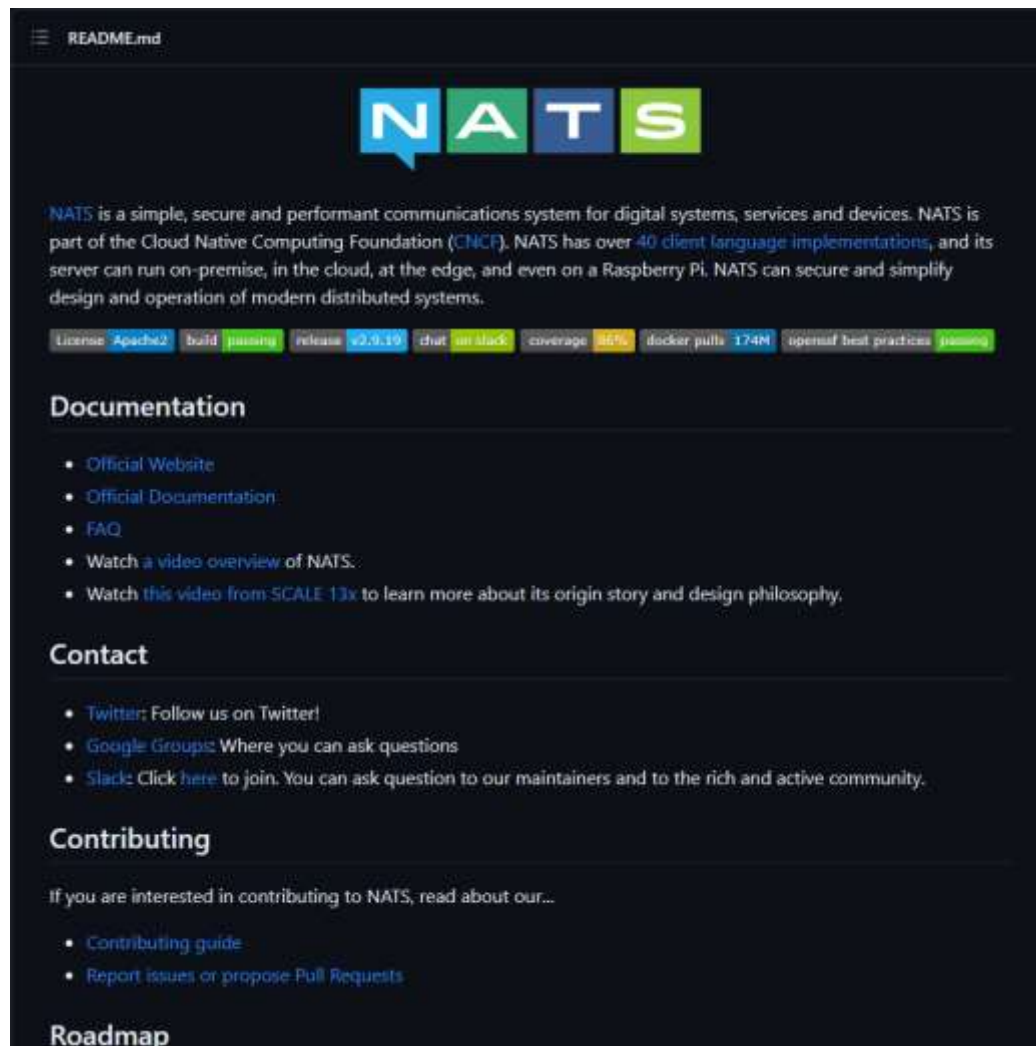
README.md – extended syntax

Definition List	<code>term</code> <code>: definition</code>
Strikethrough	<code>~~The world is flat.~~</code>
Task List	<ul style="list-style-type: none">- <code>[x]</code> Write the press release- <code>[]</code> Update the website- <code>[]</code> Contact the media
Emoji (see also Copying and Pasting Emoji)	That is so funny! <code>:joy:</code>
Highlight	I need to highlight these <code>==very important words==</code> .
Subscript	<code>H~2~0</code>
Superscript	<code>X^2^</code>

7/8/2023

141

README.md – E.g – preview



The screenshot shows the README.md file for NATS. At the top, the NATS logo is displayed. Below it, a paragraph describes NATS as a simple, secure, and performant communications system for digital systems, services, and devices. It mentions that NATS is part of the Cloud Native Computing Foundation (CNCF) and has over 40 client language implementations. A row of badges follows, showing license (Apache2), build status (passing), release version (v2.9.19), chat on Slack, coverage (95%), docker pulls (174M), and openSSF best practices (passing). The 'Documentation' section lists links to the official website, documentation, FAQ, a video overview, and a video from SCALE 13x. The 'Contact' section lists links to Twitter, Google Groups, and Slack. The 'Contributing' section includes a link to the contributing guide and a link to report issues or propose pull requests. The 'Roadmap' section is partially visible at the bottom.

README.md

NATS

NATS is a simple, secure and performant communications system for digital systems, services and devices. NATS is part of the Cloud Native Computing Foundation (CNCF). NATS has over 40 client language implementations, and its server can run on-premise, in the cloud, at the edge, and even on a Raspberry Pi. NATS can secure and simplify design and operation of modern distributed systems.

License: [Apache2](#) build: [passing](#) release: [v2.9.19](#) chat: [on slack](#) coverage: [95%](#) docker pulls: [174M](#) openSSF best practices: [passing](#)

Documentation

- [Official Website](#)
- [Official Documentation](#)
- [FAQ](#)
- Watch a [video overview](#) of NATS.
- Watch [this video from SCALE 13x](#) to learn more about its origin story and design philosophy.

Contact

- [Twitter](#): Follow us on Twitter!
- [Google Groups](#): Where you can ask questions
- [Slack](#): Click [here](#) to join. You can ask question to our maintainers and to the rich and active community.

Contributing

If you are interested in contributing to NATS, read about our...

- [Contributing guide](#)
- [Report issues or propose Pull Requests](#)

Roadmap

GIT STASH

Git stash

- Git stash là một trong những công cụ hữu ích trong việc hot fix một bug nào đó trong khi chúng ta đang dev một tính năng gì đó trên cùng một branch

git stash

Cấu trúc:

```
$ git stash -h

usage: git stash list [<log-options>]

    or: git stash show [-u | --include-untracked | --only-untracked] [<diff-
options>] [<stash>]

    or: git stash drop [-q | --quiet] [<stash>]

    or: git stash pop [--index] [-q | --quiet] [<stash>]

    or: git stash apply [--index] [-q | --quiet] [<stash>]

    or: git stash branch <branchname> [<stash>]

    or: git stash [push [-p | --patch] [-S | --staged] [-k | --[no-]keep-index]
[-q | --quiet]

    [-u | --include-untracked] [-a | --all] [(-m | --message)
<message>]

    [--pathspec-from-file=<file> [--pathspec-file-nul]]
    [--] [<pathspec>...]

    or: git stash save [-p | --patch] [-S | --staged] [-k | --[no-]keep-index]
[-q | --quiet]

    [-u | --include-untracked] [-a | --all] [<message>]

    or: git stash clear

    or: git stash create [<message>]

    or: git stash store [(-m | --message) <message>] [-q | --quiet] <commit>
```



git stash

The screenshot shows the Visual Studio Code interface. On the left, the Explorer and Open Editors panels are visible. The main editor displays a C++ file named `main.cpp` with the following code:

```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11
12    std::cout << "hello\n";
13    std::cout << "Git stash\n";
14    std::cout << "Git stash\n";
15
16    return 0;
17 }
```

The commit graph at the bottom shows the following commits:

BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR
develop	(commit)	Work in progress	1
develop	(commit)	[Micl] git stash	You
develop	(commit)	[Micl] Git reset test	You

git stash

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git stash
Saved working directory and index state WIP on develop: 19b3996 [Micls] git stash
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ █
```

git stash

The screenshot shows the Visual Studio Code interface. On the left, the Explorer and Open Editors panels are visible. The main editor displays `main.cpp` with the following code:

```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11
12    std::cout << "hello\n";
13    std::cout << "Git stash\n";
14
15    return 0;
16 }
```

Below the editor, the GitLens panel shows the commit graph for the `develop` branch. The graph shows a sequence of commits, with the most recent commit being a stash operation.

BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR
develop		WIP on develop: 19b3996 [Micls] git stash	You
develop		[Micls] git stash	You
develop		[Micls] Git reset test	You
develop		Revert "[Micls] Add PIF" This reverts commit 65f1...	You

git stash

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git stash list
```

```
stash@{0}: WIP on develop: 19b3996 [Micls] git stash
```

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ git stash pop
```

```
\On branch develop
```

```
Your branch is ahead of 'origin/main' by 6 commits.
```

```
(use "git push" to publish your local commits)
```

```
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:   main.cpp
```

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

```
Dropped refs/stash@{0} (41890356de95798930510173f7e49a69d1dba951)
```

```
micls@micls-ROG-Strix-G531GD:~/Project/git_training$ \
```



git stash

The screenshot shows the Visual Studio Code interface with a C++ project. The Explorer sidebar on the left shows the file structure, including `main.cpp`, `settings.json`, `.gitignore`, `README.md`, `Makefile`, and `CMakeLists.txt`. The main editor displays the `main.cpp` file with the following code:

```
1 #include <iostream>
2 #include "lib/compare/compare.h"
3 #include "lib/math/math_func.h"
4
5 int main()
6 {
7     std::cout << "hi\n";
8
9     int new_1 = plus(1, 2);
10    std::cout << "ANS: " << new_1 << "\n";
11
12    std::cout << "hello\n";
13    std::cout << "Git stash\n";
14    std::cout << "Git stash\n";
15
16    return 0;
17 }
```

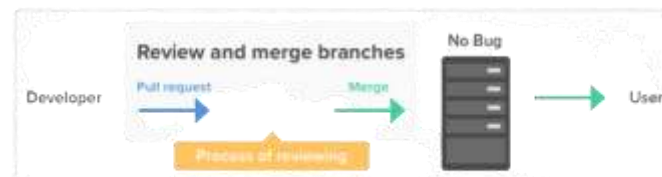
Below the editor, the GitLens panel shows the commit graph for the `develop` branch. The graph includes the following commits:

BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR
develop	(commit icon)	Work in progress	1
develop	(commit icon)	[Micl] git stash	You
develop	(commit icon)	[Micl] Git reset test	You
develop	(commit icon)	Revert "[Micl] Add PIF" This reverts commit 65f1...	You

Pull/merge request



Development without pull request



Development with pull request

Issue



HOW TO GIT PROFESSIONAL

How to git professional

- [GIT for Professionals Tutorial - Tools & Concepts for Mastering Version Control with GIT - YouTube](#)
- [Gist – robertpainsi](#)
- [Git training with interactive](#)

Other keyword maybe you need

- Squash merge
- Cherrypick
- (to be continue)

FAQ



FAQ

Không có đâu,
hỏi google cho



Docs

- How to master git: [GIT - Book \(git-scm.com\)](https://git-scm.com/book)
- Github cheat sheet: [Github Training Kit - Github Cheatsheets](#), [Github Education Kit - Github Cheatsheets](#)
- Other materials: [GIT - Documentation \(git-scm.com\)](https://git-scm.com/docs)
- GIT but professional: [GIT for Professionals Tutorial - Tools & Concepts for Mastering Version Control with GIT - YouTube](#)
- If you have bugs or need something ? Ask: [Google](#), [Bing](#), [Stack Overflow](#), [ChatGPT \(openai.com\)](#), everyone near you(just in case they know)

 payitforward.edu.vn