



Git – Git Flow

GitHub – GitHub Flow



And SourceTree

09-2025

Nhóm thực hiện

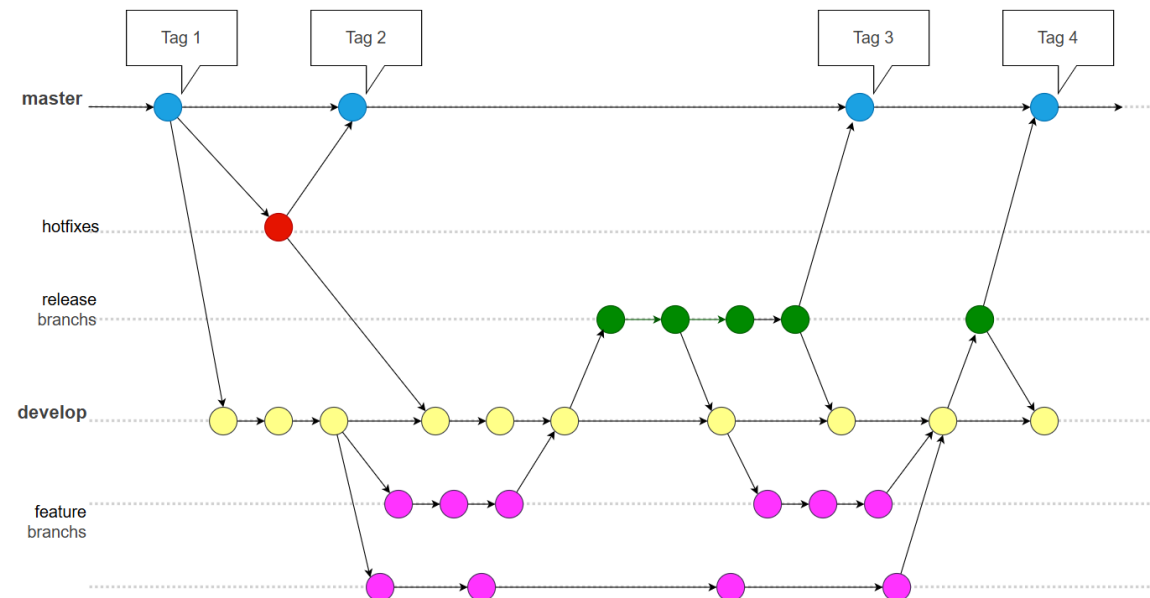
- Thân Hoàng Lộc
- Trần Quang Bình
- Trần Văn Huy
- Trần Anh Tuấn

Nội dung

1. Giới thiệu Git
 1. Các thuật ngữ
 2. Basic Git commands
 2. Giới thiệu GitHub
 1. Repository (repo)
 2. Vài thao tác cơ bản với repo
 3. GitHub Flow – Quy trình đơn giản
 4. Git Flow – Quy trình mô hình phân nhánh
 5. SourceTree – A beautiful Git GUI
1. Thực hành GitHub Flow
 2. Thực hành Git Flow
 3. Thực hành SourceTree

Mục đích

Giới thiệu tổng quan về Git và ứng dụng Git Flow trong dự án phần mềm





Giới thiệu Git

[Introduction to Git in VS Code](#)

[Git - Downloads](#)



🔍 Type / to search entire site...



About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloads



macOS



Windows



Linux/Unix

Older releases are available and the [Git source repository](#) is on GitHub.



GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)



Giới thiệu Git

Git là một hệ thống **theo dõi các thay đổi** đối với các tập tin trong quá trình phát triển dự án, **thường là mã nguồn**, và cho phép **nhiều người cùng hợp tác làm việc** trên các tập tin đó.

*Git là một **Hệ thống Kiểm soát Phiên bản Phân tán**
(**Distributed Version Control System - DVCS**),
được tạo ra bởi Linus Torvalds vào năm 2005.*



Git – Các thuật ngữ

Working tree

tập hợp các folder và file để lưu trữ nội dung

Repository (repo)

là folder lưu trữ toàn bộ lịch sử và meta data nội dung của dự án

Hash

một thông tin được mã hóa biểu diễn nội dung của một file hoặc object, được dùng để xác định sự thay đổi nội dung của file

Head

là một tham chiếu trỏ đến vài commit cuối cùng hoặc hiện tại trong nhánh hiện hành

Commit

là một cam kết về những thay đổi của nội dung

Branch

là nhánh, một nhánh là một chuỗi các Commit được liên kết với nhau. “**Main**” là nhánh chính, mặc định

Command, sub-command

Git sử dụng các command để thực hiện các thao tác

ví dụ git --version là trình bày phiên bản Git



Basic Git commands

Command

```
git --version
```

Output

```
git version 2.45.0.Windows.1
```

Command

```
git config --global user.name "<username>"
```

Command

```
git config --global user.email "<user email>"
```

Command

```
git config --list
```

Output

```
User.name=sample-name
```

```
User.email=sample-email@mycompany.com
```

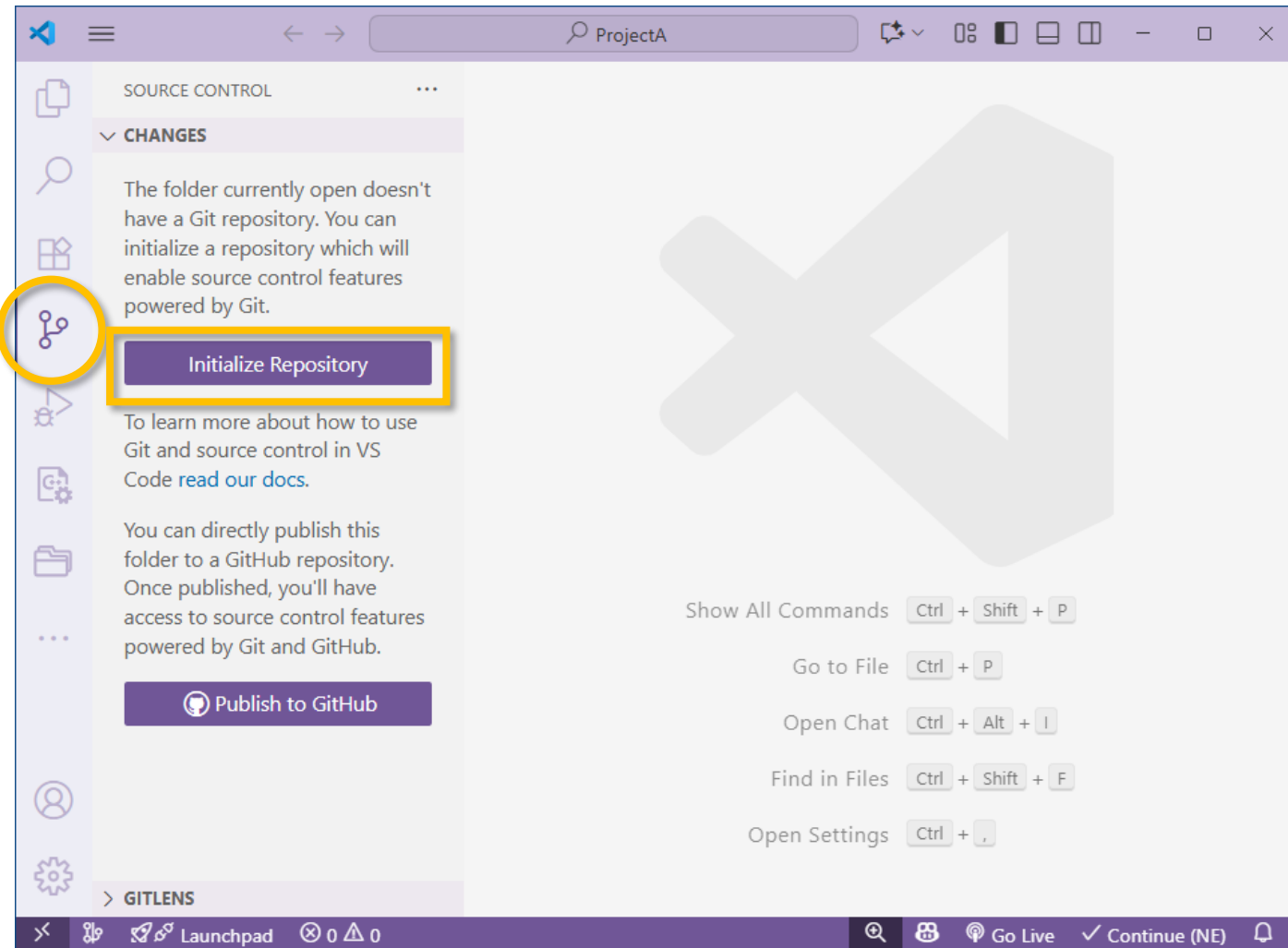


Basic Git commands

Command

```
git init
```

Khởi tạo **“.git”** folder





Basic Git commands

Command

```
git init --initial-branch=main
```

Command

```
git init -b main
```

Output

Initialized empty Git repository in E:/WorkFolder/ProjectA/.git/

Khởi tạo **repo** mới và đặt tên
mặc định cho **branch** là **main**

Một **empty repo** được tạo tại Project folder



Basic Git commands

Command

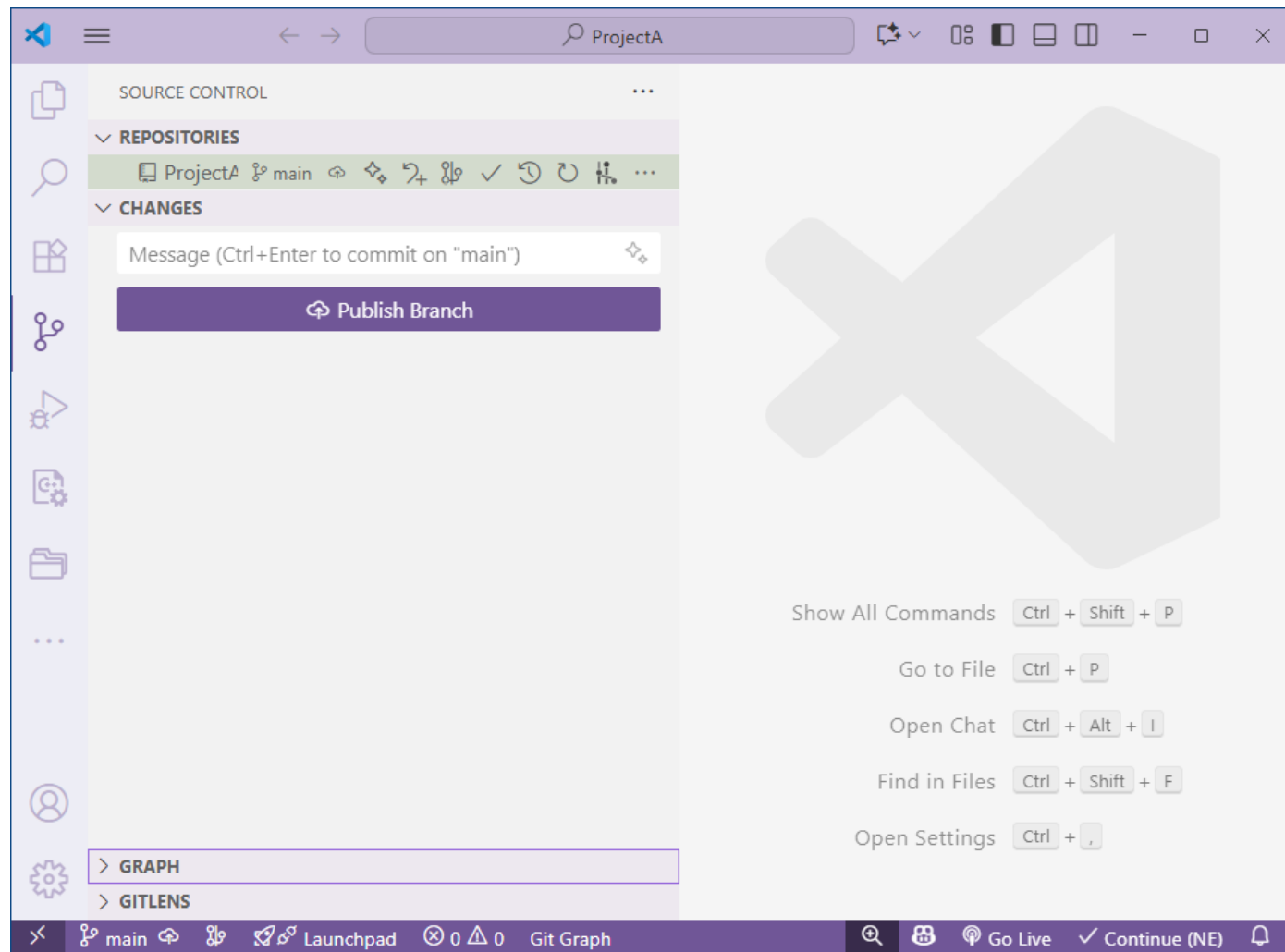
```
git status
```

Output

On branch main

No commits yet

Hiện tại ở **branch** là **main**, và
chưa có commits nào.





Basic Git commands

Command

`git help`

`git --help`

`git help <command>`

Liệt kê các git command.

Hướng dẫn sử dụng từng git
command, subcommand

```
Windows PowerShell
PS E:\WorkFolder\ProjectA> git help
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
      [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
      [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  backfill   Download missing objects in a partial clone
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```



Giới thiệu GitHub



Sign up

GitHub được xây dựng trên nền tảng Git và bổ sung thêm khả năng cộng tác với nhiều công cụ, tính năng tự động, hoạt động trên môi trường web.

 [Product](#) [Solutions](#) [Resources](#) [Open Source](#) [Enterprise](#) [Pricing](#)

 Search or jump to...

[Sign in](#)

[Sign up](#)

Build and ship software on a single, collaborative platform

Join the world's most widely adopted AI-powered developer platform.

Enter your email

[Sign up for GitHub](#)

[Try GitHub Copilot](#)

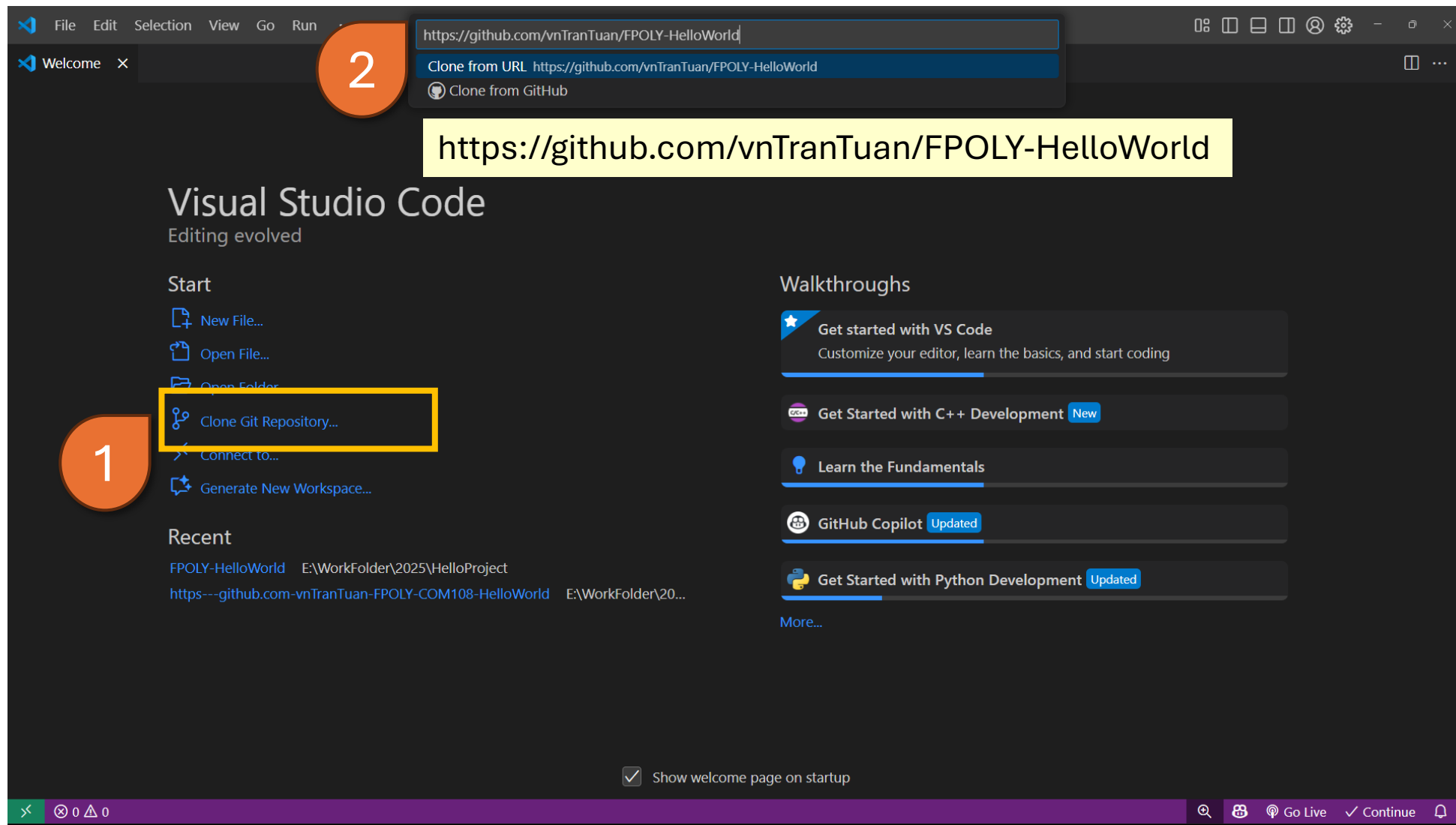


GitHub – các thao tác cơ bản

Clone repository	download một repository có sẵn
Create repository	tạo mới một repository
Initialize repository	khởi tạo repository
Commit changes	cam kết những thay đổi nội dung trong repository
Publish changes	tải lên những thay đổi nội dung trong repository



VS Code **Clone** Hello Project





VS Code **Clone** Hello Project

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'FPOLY-HELLOWORLD' with files '.vscode', 'hello.c', and 'hello.exe'. The 'hello.c' file is selected and open in the editor. The editor shows a C program with the following code:

```
1  #include<stdio.h>
2
3  int main() {
4      printf("Hello, World!");
5      return 0;
6  }
7
8
```

The status bar at the bottom indicates the current state: 'master' branch, 0 errors and 0 warnings, 'Git Graph' view, 'Spaces: 4', 'UTF-8' encoding, 'CRLF' line endings, 'C' language, 'Go Live' button, 'Win32' architecture, and 'Continue' button.



GitHub – repository (repo), **create**

The screenshot shows the GitHub 'New repository' page. The page is titled 'Create a new repository' and includes a 'Preview' button and a link to 'Switch back to classic experience'. Below the title, there is a description of repositories and a link to 'Import a repository'. The page is divided into two main sections: 'General' and 'Configuration'.

General Section:

- Owner ***: A dropdown menu showing the user 'vnTranT' with a callout '3' pointing to it.
- Repository name ***: A text input field.
- Description**: A text input field with a character count '0 / 350 characters'.

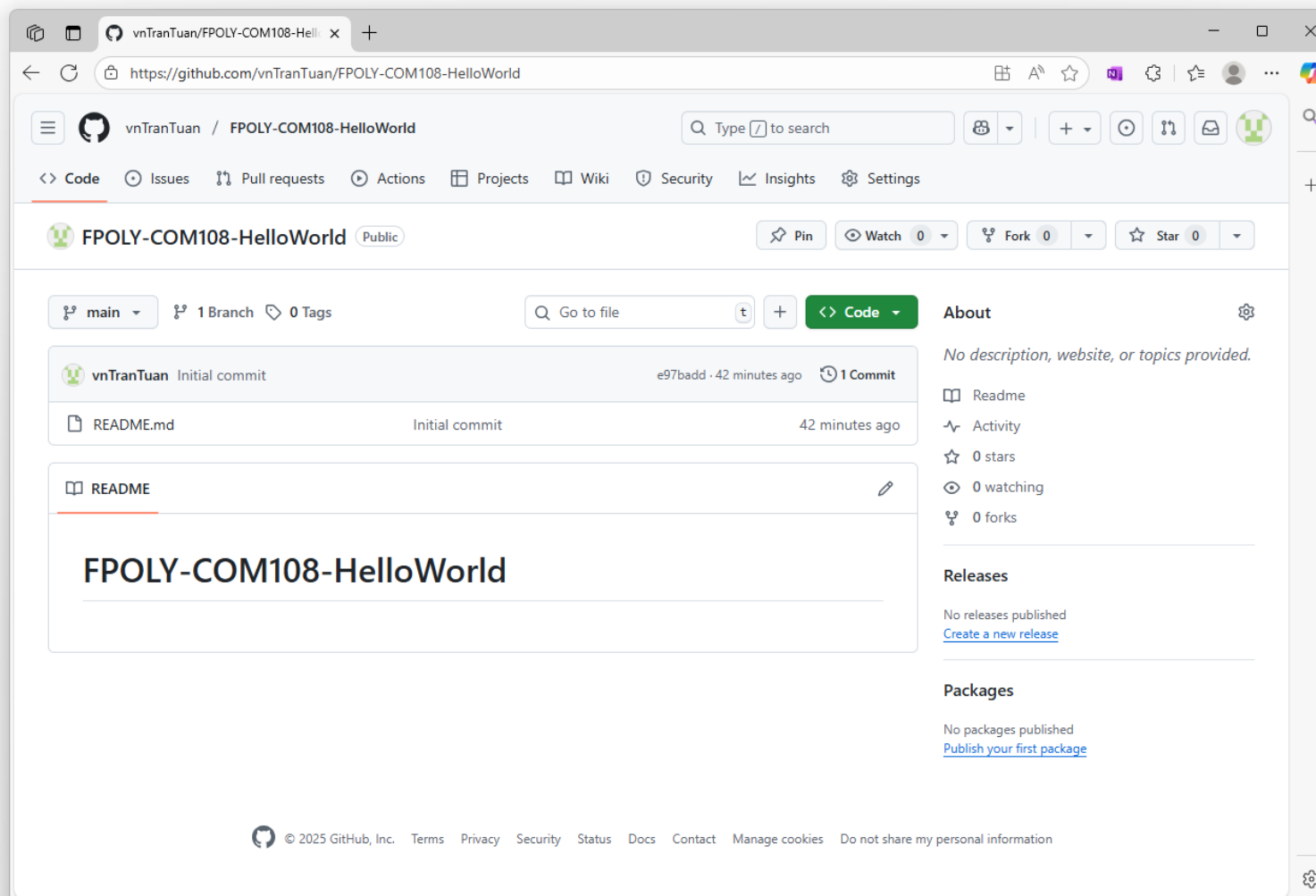
Configuration Section:

- Choose visibility ***: A dropdown menu set to 'Public'.
- Start with a template**: A dropdown menu set to 'No template'.
- Add README**: A toggle switch set to 'Off'.
- Add .gitignore**: A dropdown menu set to 'No .gitignore'.
- Add license**: A dropdown menu set to 'No license'.

At the bottom right, there is a green 'Create repository' button. A callout '1' points to the top navigation bar, and a callout '2' points to the 'New repository' option in the dropdown menu.




GitHub – repository (repo), **create**





Ví dụ - GitHub Create **Empty Repository**

Đăng nhập vào GitHub

1. Click nút 
2. Chọn menu **New repository**
3. Điền thông tin

Repository name

Ví dụ :

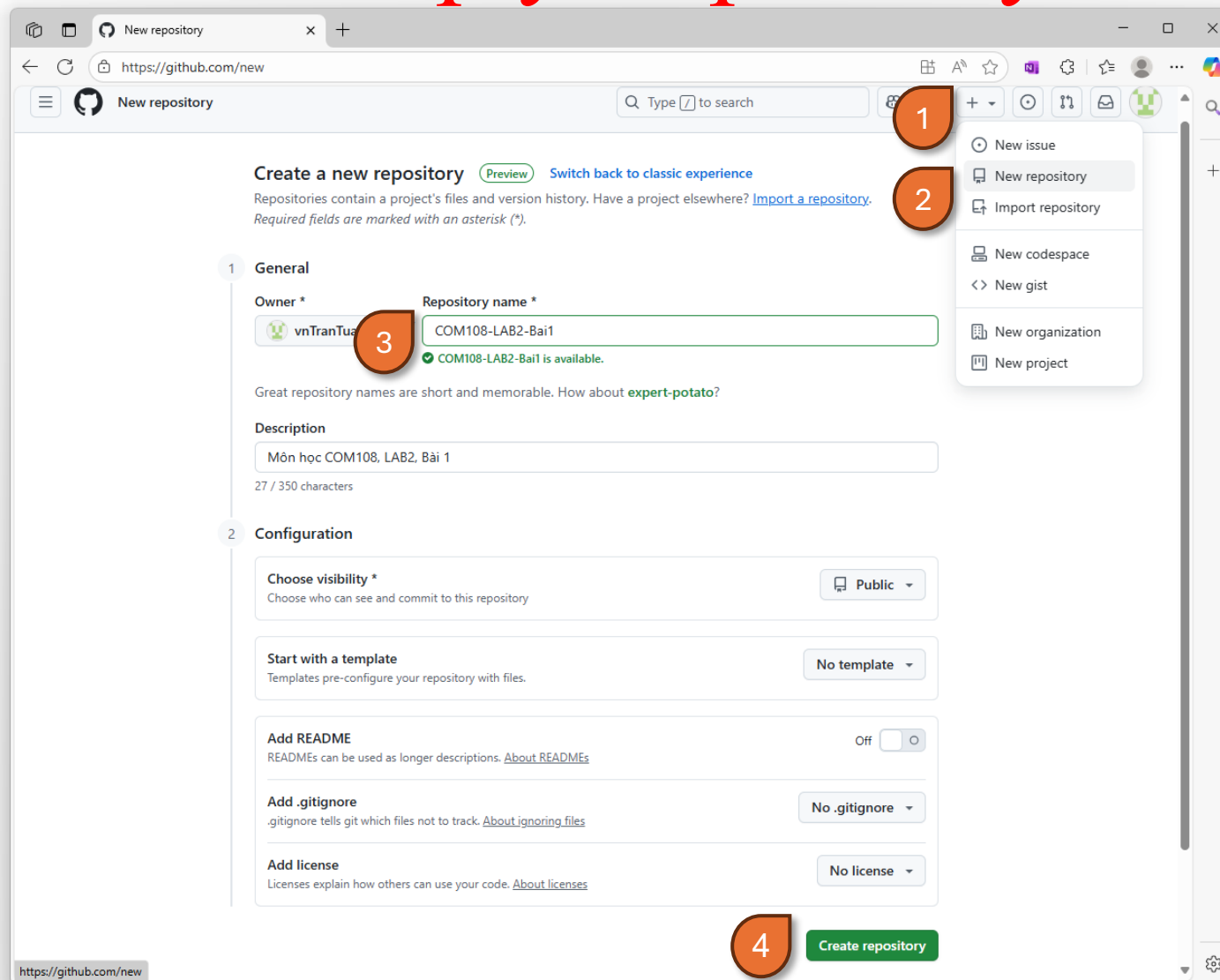
[COM108-LAB2-Bai1](#)

Description

Ví dụ :

[Môn học COM108, Repository của LAB2, Bài 1](#)

4. Click nút **Create repository**



The screenshot shows the GitHub 'New repository' page. The interface includes a search bar at the top, a navigation menu on the right, and a main form area. The form is divided into two sections: 'General' and 'Configuration'. In the 'General' section, the 'Repository name' field is filled with 'COM108-LAB2-Bai1' and a message indicates it is available. The 'Description' field is filled with 'Môn học COM108, LAB2, Bài 1'. In the 'Configuration' section, the 'Choose visibility' dropdown is set to 'Public', 'Start with a template' is set to 'No template', 'Add README' is set to 'Off', 'Add .gitignore' is set to 'No .gitignore', and 'Add license' is set to 'No license'. The 'Create repository' button is at the bottom right.

1. Click the plus icon in the top right corner.

2. Select the 'New repository' option from the dropdown menu.

3. Fill in the 'Repository name' and 'Description' fields.

4. Click the 'Create repository' button at the bottom right.

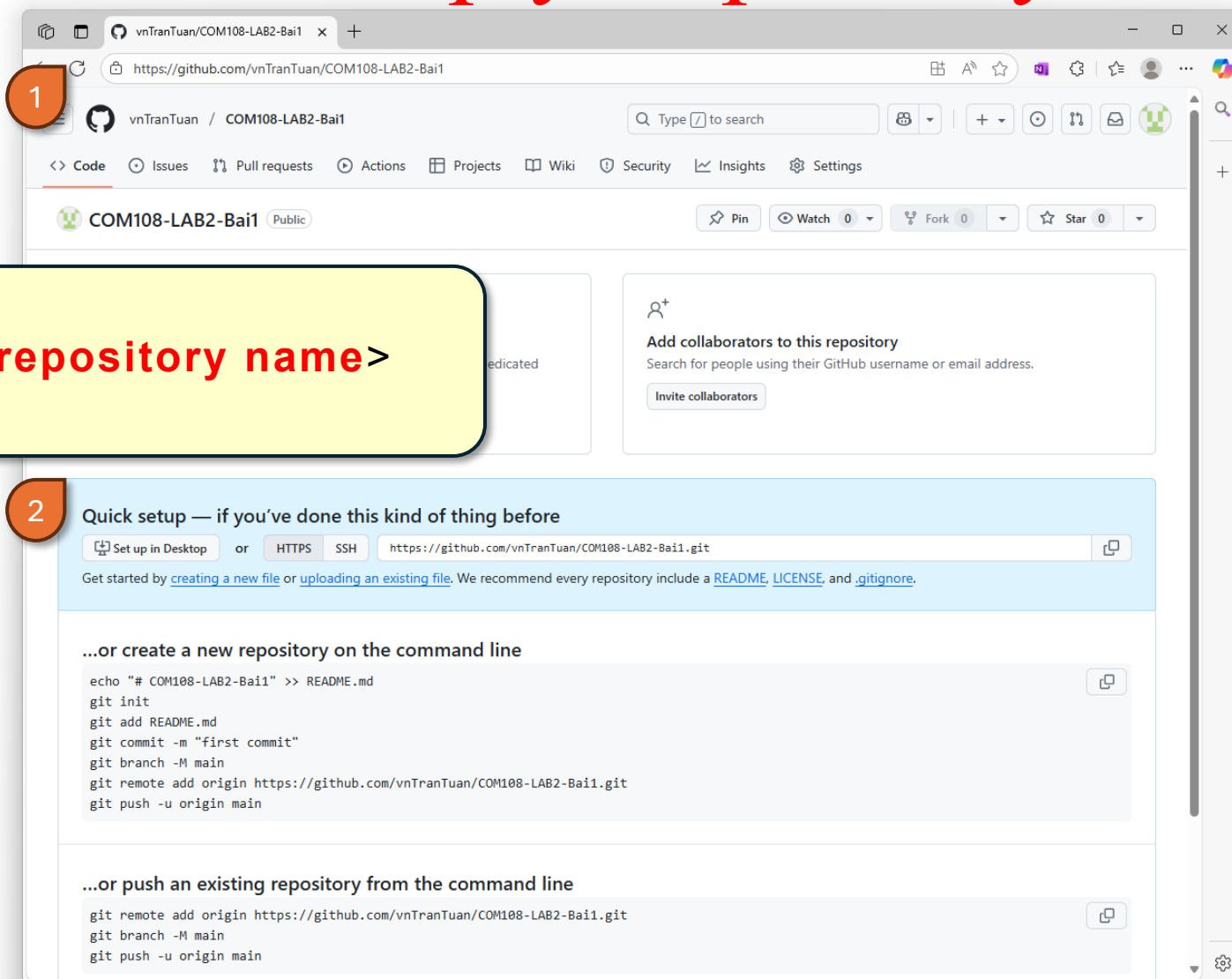


Ví dụ - GitHub Create Empty Repository

Trang web của repository đã tạo

1

`https://github.com/<user name>/<repository name>`







Ví dụ - GitHub Create Empty Repository

LƯU Ý

Quick setup của Empty repository đã tạo

2

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** 

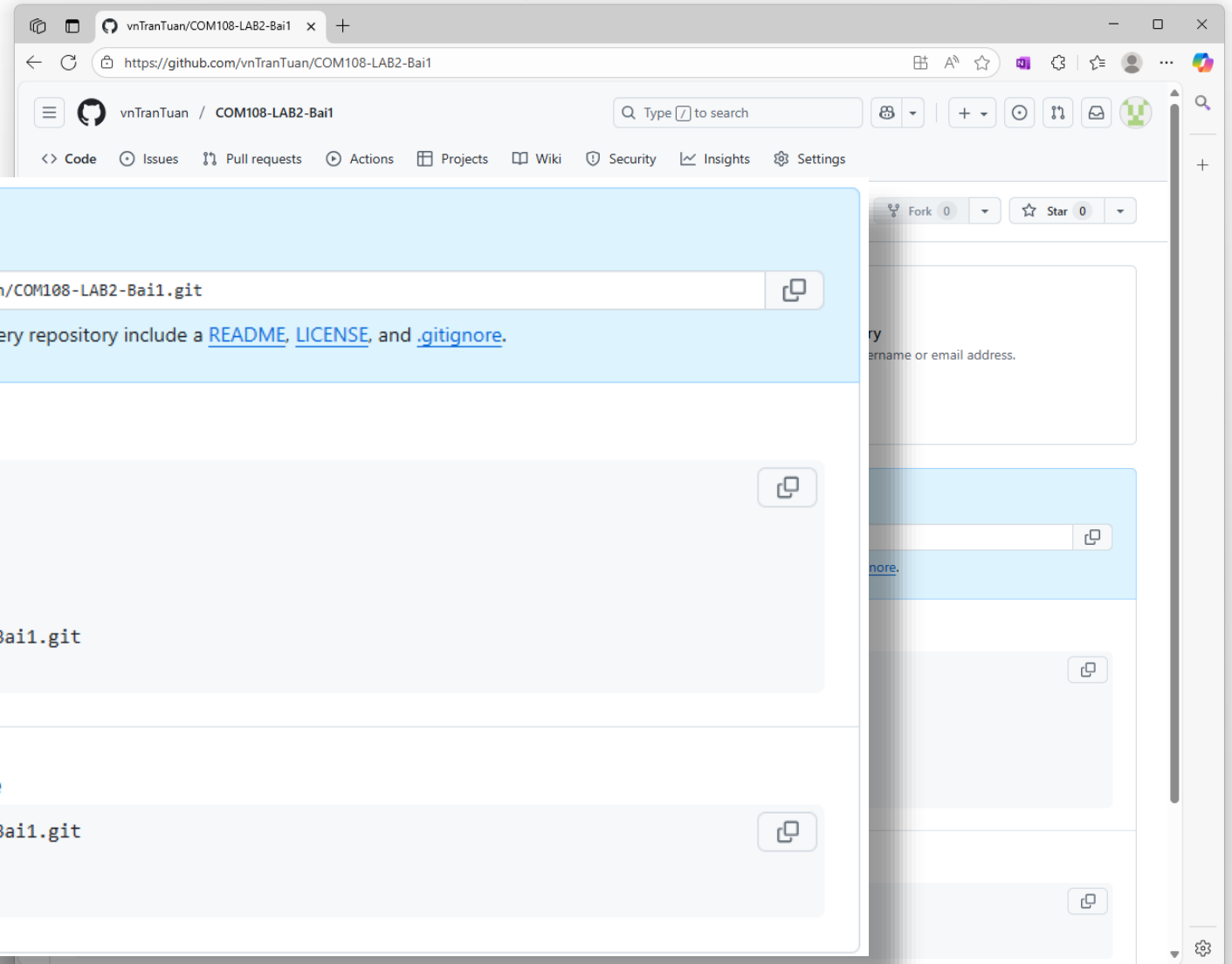
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# COM108-LAB2-Bai1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/vnTranTuan/COM108-LAB2-Bai1.git
git push -u origin main
```

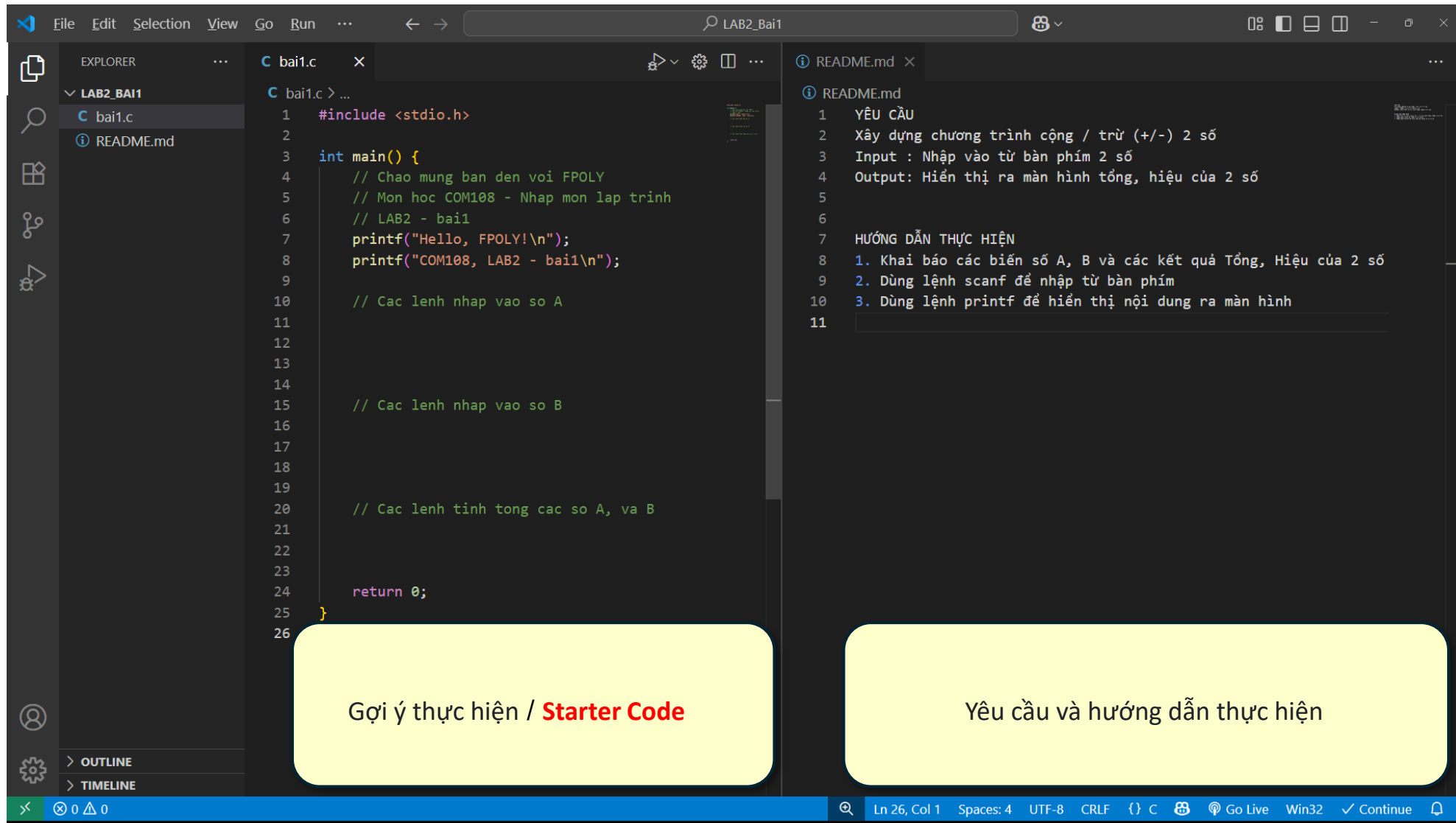
...or push an existing repository from the command line

```
git remote add origin https://github.com/vnTranTuan/COM108-LAB2-Bai1.git
git branch -M main
git push -u origin main
```



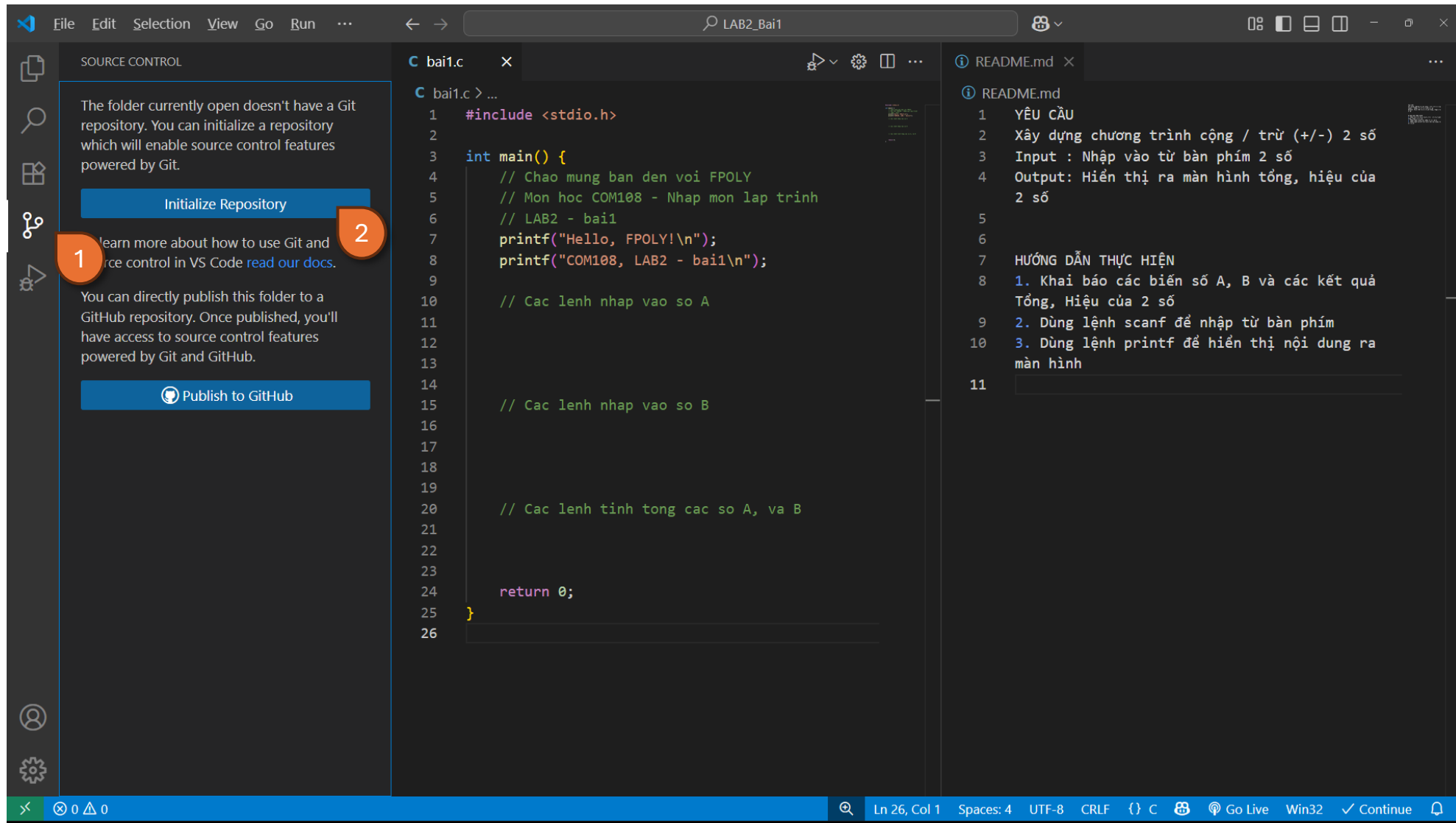


VS Code - Create Project (*ví dụ LAB2, bài 1*)



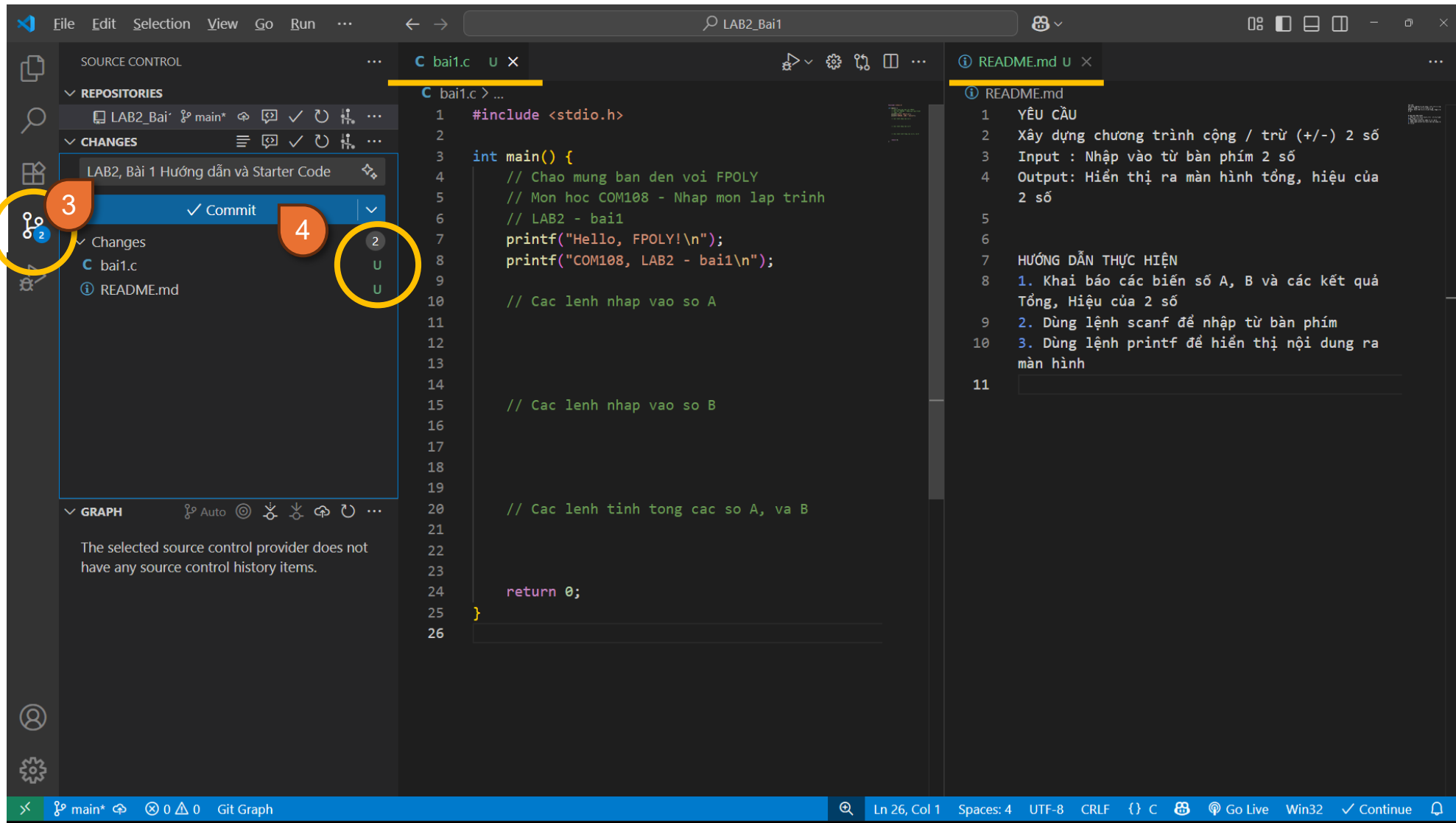


VS Code – Git Initialize Repository



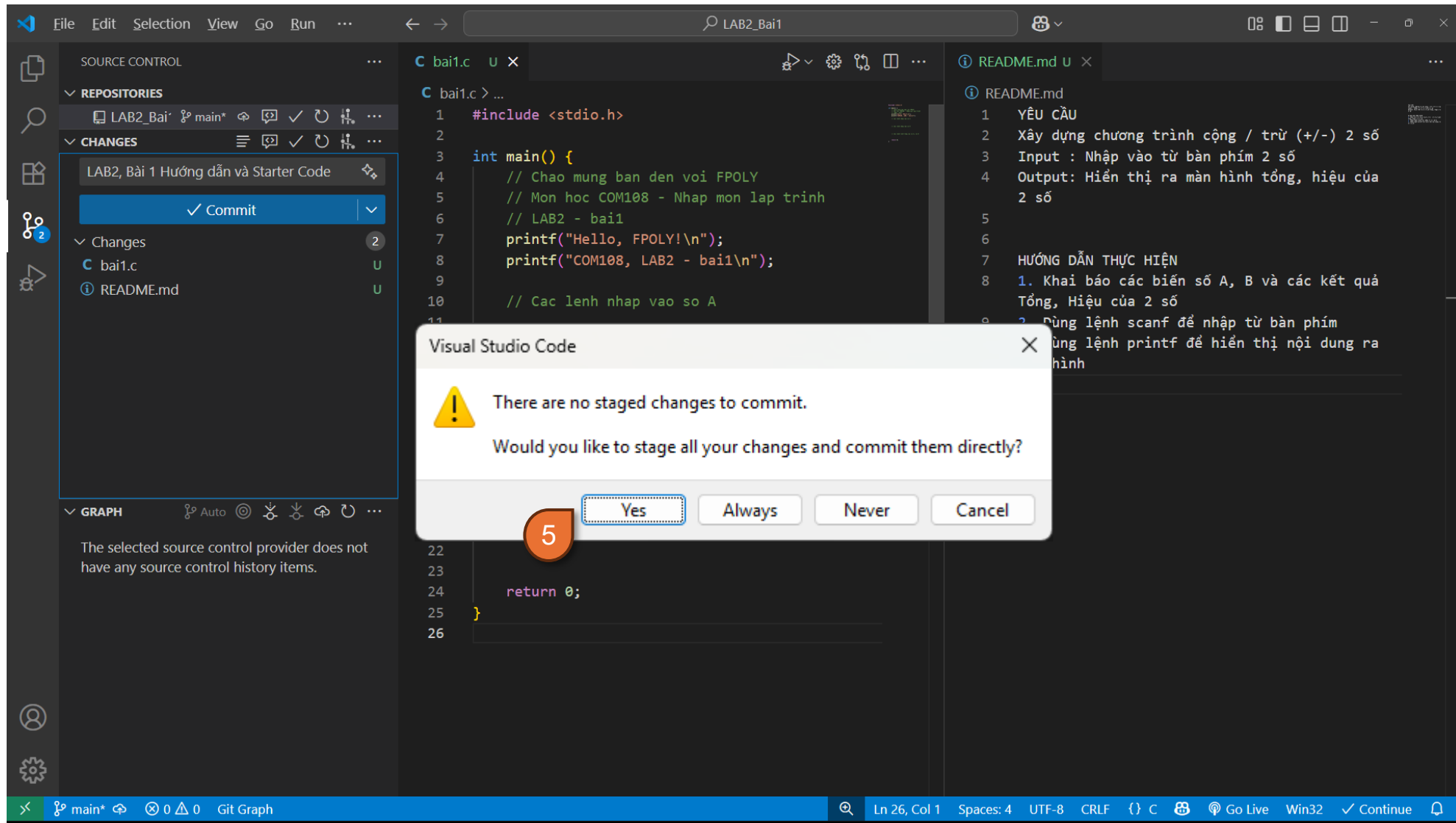


VS Code – Git Commit changes





VS Code – Git Commit changes





VS Code – Git Publish

Quick setup — if you've done this kind of thing before

Set up in Desktop or **HTTPS** **SSH**

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# COM108-LAB2-Bai1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/vnTranTuan/COM108-LAB2-Bai1.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/vnTranTuan/COM108-LAB2-Bai1.git
git branch -M main
git push -u origin main
```



VS Code – Git Publish

The screenshot shows the Visual Studio Code interface with the following components:

- SOURCE CONTROL:** Shows the repository 'LAB2_Bai1' with a 'main' branch. A 'Publish Branch' button is visible.
- Editor:** Displays a C program 'bai1.c' and a 'README.md' file. The C program includes comments in Vietnamese and uses `printf` to output a greeting and some data. The README file contains instructions in Vietnamese.
- Terminal:** The terminal window is active, showing the command `git branch -M main` being executed. A yellow circle highlights the terminal window.

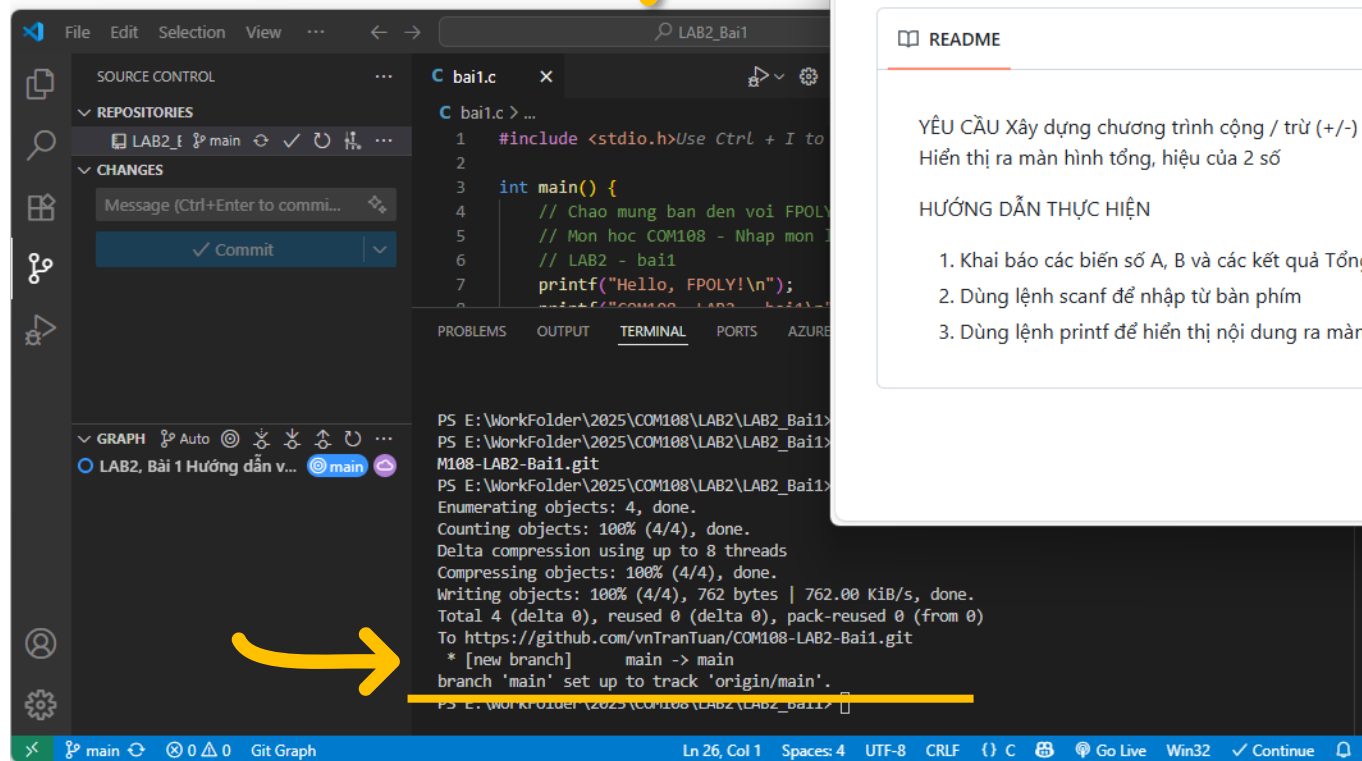
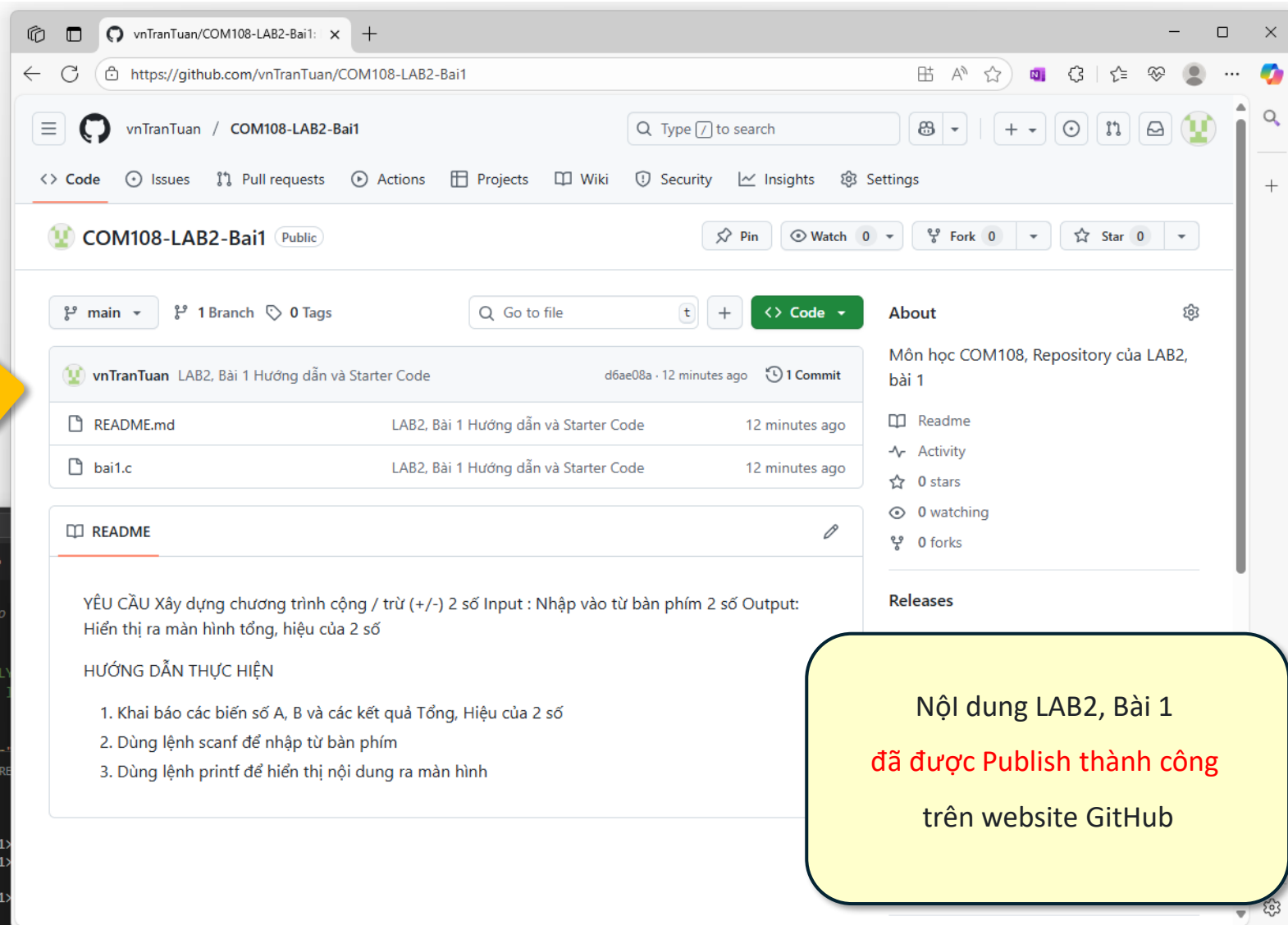
A yellow callout box contains the following text:

Chạy lần lượt các lệnh sau, với **<username>** đã đăng nhập

```
git branch -M main
git remote add origin https://github.com/<username>/COM108-LAB2-Bai1.git
git push -u origin main
```



VS Code Git Publish





GitHub Flow, **branch**

GitHub Flow được xây dựng dựa trên các khái niệm căn bản của Git

main – branch chính (mặc định)

Command

```
git init --initial-branch=main
```

Command

```
git init -b main
```

Khi mới khởi tạo main, nếu chưa có file (hay commit) nào thì main chưa thật sự tồn tại

Branch là một phần thiết yếu của trải nghiệm GitHub.

Chúng cho phép bạn thực hiện các thay đổi mà không ảnh hưởng đến main branch.

Branch là một nơi an toàn để thử nghiệm các tính năng hoặc bản sửa lỗi mới.

Nếu mắc lỗi, bạn có thể “undo” các thay đổi hoặc để sửa lỗi. Các thay đổi của bạn sẽ không được cập nhật trên main branch cho đến khi bạn hợp nhất các branch.

GitHub Flow, **commit**

Commit là một sự thay đổi của một hoặc nhiều files trên một branch.

Mỗi commit có thể được theo dõi (tracked), gồm ID, thời gian, và người thực hiện thay đổi, cho dù sự thay đổi đó được thực hiện bằng command (ở local) hay trên website GitHub.

Commit cung cấp một cơ chế có thể truy vết lịch sử sự thay đổi nội dung (file hay folder)

Command

```
git commit -m "Cập nhật file A bổ sung câu chào bạn"
```

Commits

 main ▾

Commits on Sep 1, 2023

Update 1-create-a-branch.md



Contoso committed 2 weeks ago ✓

Verified



d97b4c6



Commits on Aug 9, 2023

Update discussion link (#468) ...



Contoso committed on Aug 9 ✓

Verified



cc85b5a





GitHub Flow, **commit**

Với 1 repo, quá trình kiểm soát của Git, 1 file có các trạng thái sau : **Untracked** và **Tracked**

Untracked: trạng thái ban đầu của file, Git chưa kiểm soát

Tracked: Git đã kiểm soát file. Và có các trạng thái sau

Unmodified: nội dung chưa bị thay đổi từ lần commit gần nhất

Modified: nội dung đã thay đổi từ lần commit gần nhất, nhưng chưa Staged

Staged: nội dung đã thay đổi, đã được đưa vào chỉ mục kiểm soát, và sẵn sàng Commit

Committed: nội dung thay đổi đã kiểm soát và nằm trong repo database



GitHub Flow, pull request

Pull request là 1 cơ chế được sử dụng để báo hiệu các commit từ một branch đã sẵn sàng để được hợp nhất – merge vào nhánh khác.

Thành viên nhóm gửi yêu cầu pull request sẽ yêu cầu một hoặc nhiều người đánh giá xác minh code và phê duyệt việc hợp nhất.

Những người đánh giá này sẽ bình luận về các thay đổi, thêm nội dung của riêng họ hoặc thảo luận thêm.

Sau khi các thay đổi được phê duyệt (nếu cần), branch của yêu cầu pull request (compare branch) sẽ được hợp nhất vào base branch.



GitHub Flow, pull request

Create 0000-01-02-Contoso.md #3

Merged

Contoso merged 2 commits into `main` from `my-slide` 2 days ago


Conversation 4

Commits 2

Checks 0

Files changed 1

+6 -0



Contoso commented 2 days ago • edited


Adding a much-needed file.

Owner

😊


⋮

🔗

 Create 0000-01-02-Contoso.md

Verified

fce28d0



github-learning-lab bot requested changes 2 days ago

View changes

github-learning-lab bot left a comment

😊 ⋮


Good pull requests have a body description that tells other contributors about the change you're suggesting, so they understand the context.

Let's edit this pull request to add a body description.

Activity: Fixing your pull request

1. The first comment on your pull request will have the default text of **No description provided**. Click on the ... icon located at the top right corner of the comment box, then click on **Edit** to make an edit

Reviewers

 github-learning-lab ✓

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

Successfully merging this pull request may close these issues.

None yet

No Flow

No Flow với ý muốn nói đến không theo quy trình

KHÔNG giúp bạn thực hiện công việc một cách an toàn.

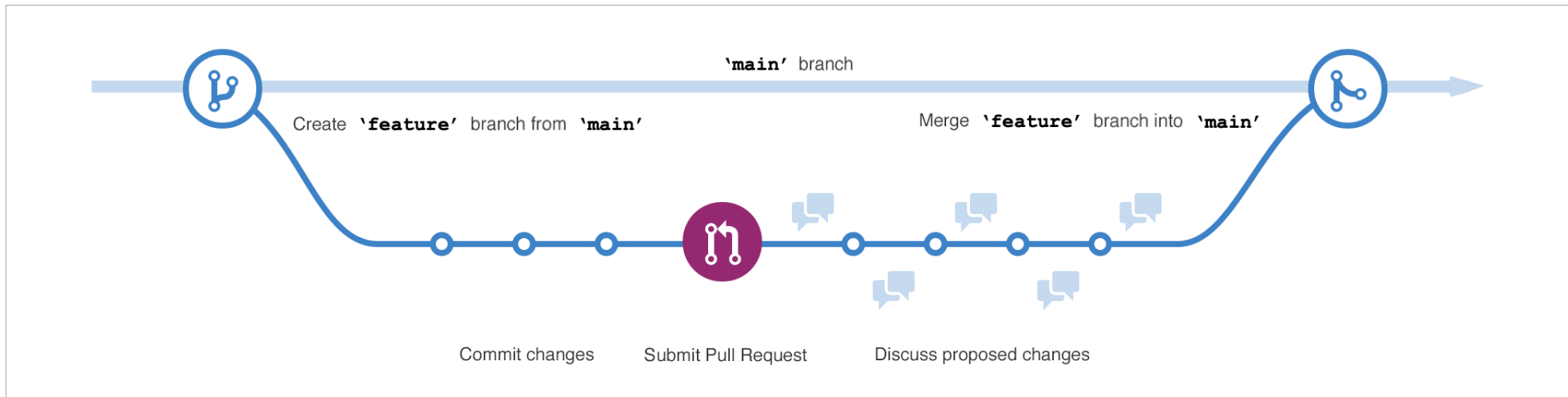
Nội dung này có nên trình bày hay không?



GitHub Flow

GitHub Flow là một quy trình làm việc đơn giản

giúp bạn thực hiện và chia sẻ các thay đổi một cách an toàn. Nó rất hữu ích để thử nghiệm các ý tưởng và cộng tác với nhóm của bạn bằng cách sử dụng các branch, pull request và merge



1. Bắt đầu bằng cách tạo một "feature" branch để các thay đổi, hoặc bản sửa lỗi của bạn không ảnh hưởng đến nhánh chính.
2. Tiếp theo, hãy thực hiện các cập nhật trong branch.
3. Bây giờ, tạo một pull request để mời phản hồi và bắt đầu đánh giá.
4. Sau đó, hãy xem xét các bình luận và thực hiện bất kỳ cập nhật cần thiết nào dựa trên phản hồi của nhóm bạn.
5. Cuối cùng, khi bạn đã tự tin vào các thay đổi của mình, hãy xin phê duyệt và merge yêu cầu pull request vào main branch.
6. Sau đó, hãy xóa branch để giữ cho repo của bạn sạch sẽ và tránh sử dụng các branch lỗi thời.



GitHub Flow, ví dụ

Giả sử bạn đang làm việc trên một trang web với nhánh chính là **main**.

1. Bắt đầu Công việc (Create a Branch)

Command

```
git init --initial-branch=main  
# Đảm bảo bạn đang ở nhánh main  
git checkout main  
# Tạo và chuyển sang nhánh tính năng mới  
git checkout -b feature/them-nut-dang-ky
```

Lưu ý

Khi mới khởi tạo main, nếu chưa có file (hay commit) nào thì main chưa thật sự tồn tại



GitHub Flow, ví dụ

Giả sử bạn đang làm việc trên một trang web với nhánh chính là **main**.

2a. Tạo file / nội dung công việc

Ví dụ tạo nút đăng ký trong trang web `"dang-ky.html"`



file `"dang-ky.html"`

Lưu ý

Khi mới khởi tạo main, nếu chưa có file (hay commit) nào thì main chưa thật sự tồn tại



GitHub Flow, ví dụ

Giả sử bạn đang làm việc trên một trang web với nhánh chính là **main**.

2b. Commit công việc (commit locally)

Command

Thêm các file đã thay đổi

```
git add .
```

Commit công việc đã thực hiện

```
git commit -m "feature: Đã thêm nút đăng ký"
```



GitHub Flow, ví dụ

Giả sử bạn đang làm việc trên một trang web với nhánh chính là **main**.

3. Sẵn sàng và Push lên GitHub

Command

```
# Khai báo remote repository  
# Ví dụ https://github.com/vnTranTuan/GitFlow-ProjectA.git  
git remote add origin <repositoryURL>
```

Command

```
# Đẩy / Push lên remote repository  
git push origin feature/them-nut-dang-ky
```



GitHub Flow, ví dụ

Giả sử bạn đang làm việc trên một trang web với nhánh chính là **main**.

3. Sẵn sàng và Push lên GitHub

Command

Khai báo remote repository

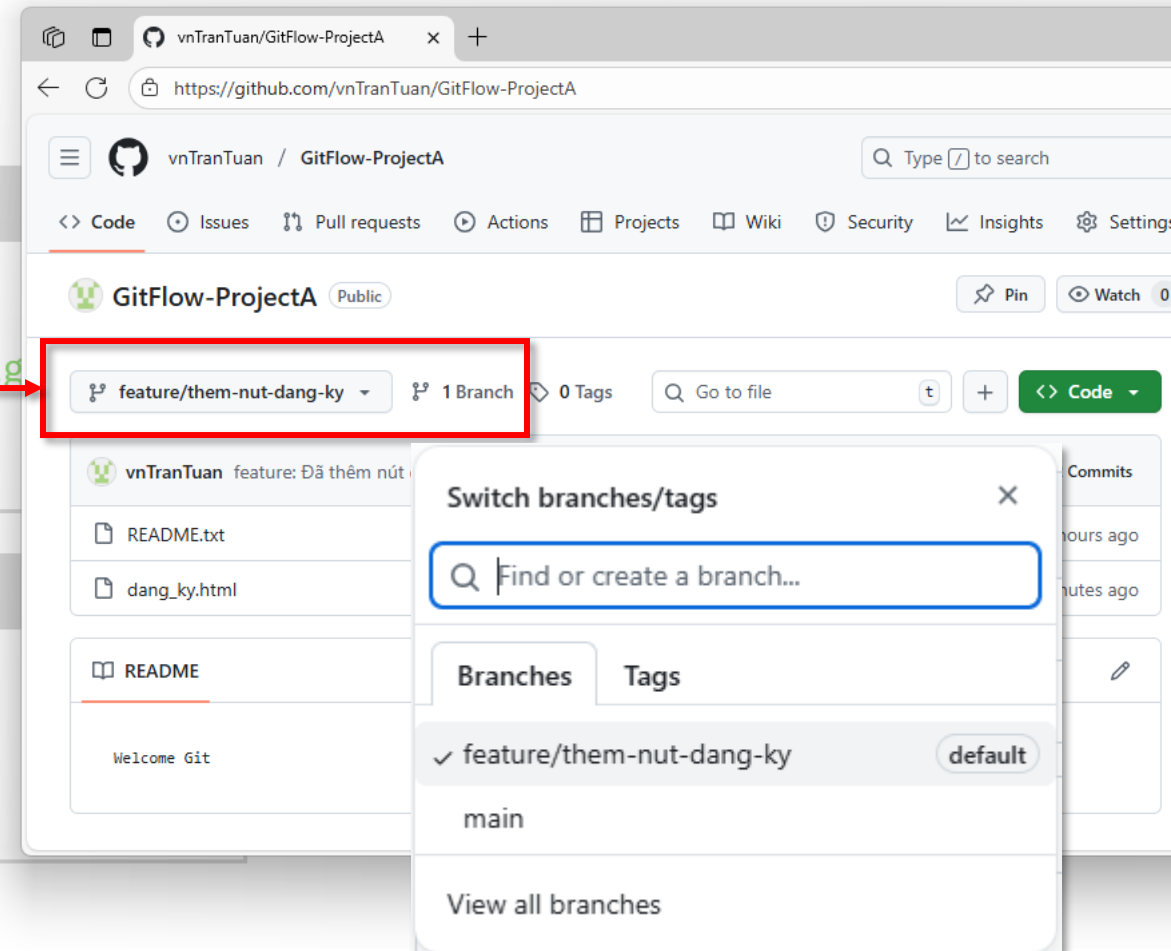
Ví dụ <https://github.com/vnTranTuan/GitFlow-ProjectA>

```
git remote add origin <repositoryURL>
```

Command

Đẩy / Push lên remote repository

```
git push origin feature/them-nut-dang-ky
```

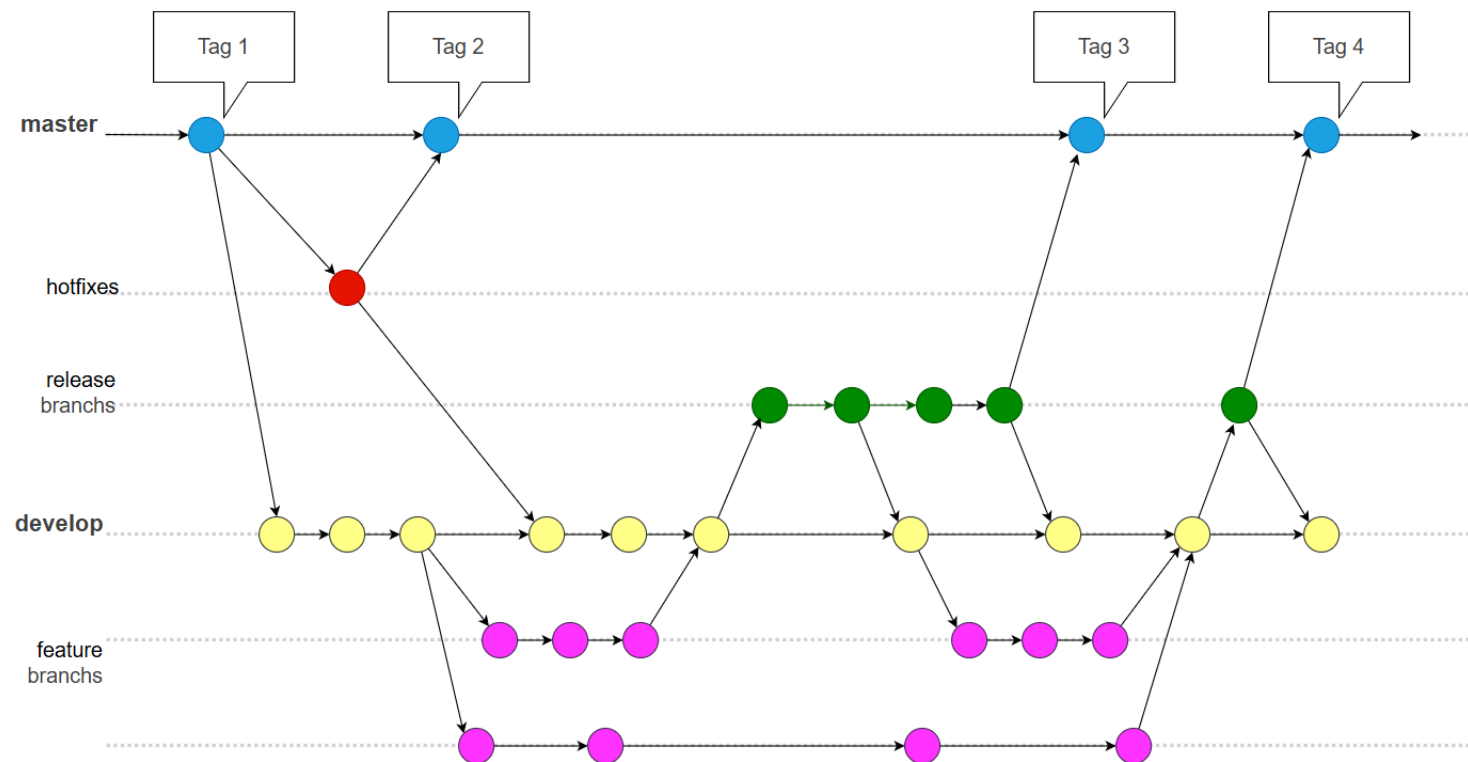




Git Flow

Trong khi GitHub Flow là một quy trình làm việc đơn giản

Git Flow là một quy trình làm việc theo mô hình phân nhánh có cấu trúc thường được sử dụng trong các môi trường làm phần mềm theo hướng phát hành (các phiên bản).
Git Flow thường dùng thuật ngữ master branch làm mặc định thay vì main branch.





Git Flow, **branch types**

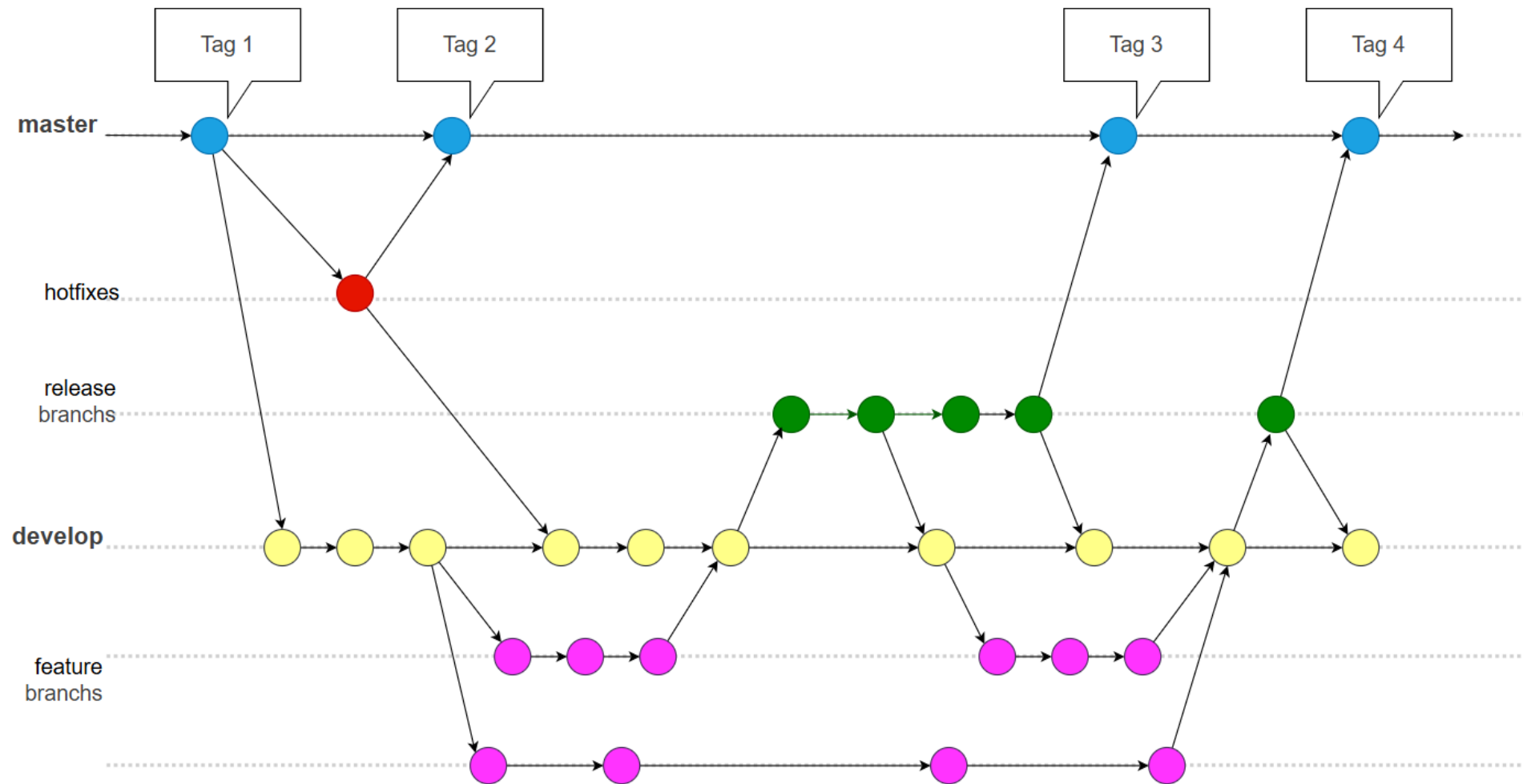
Với Git Flow, thường áp dụng các dạng branch sau

- master** branch chứa code đã sẵn sàng cho phát hành
- develop** branch chứa các công việc phát triển mới nhất cho bản phát hành tiếp theo
- feature/*** được sử dụng để tạo các tính năng mới; phân nhánh từ develop và được hợp nhất trở lại khi hoàn tất
- release/*** chuẩn bị một bản phát hành production mới từ develop; cho phép thử nghiệm cuối cùng và sửa các lỗi nhỏ
- hotfix/*** được sử dụng để vá nhanh các sự cố production; phân nhánh từ master

/* được hiểu là có thể có nhiều nhánh con



Git Flow, **how to**





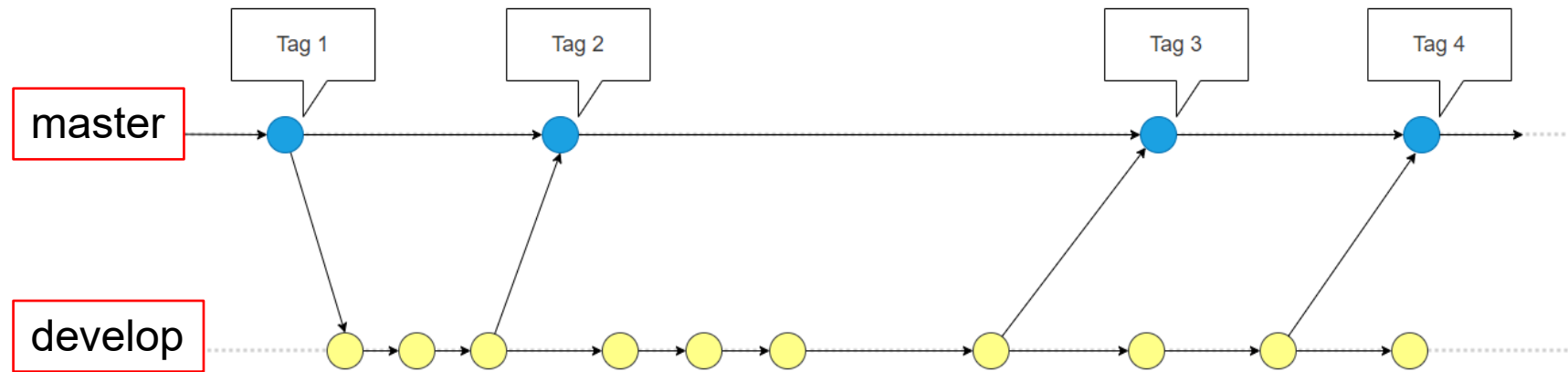
Git Flow, **how to**

Giải thích cách hoạt động

1. Developer tạo các **features branch** từ develop để xây dựng chức năng mới.
2. Khi đến thời điểm phát hành, một **release branch** sẽ được tạo từ develop.
Điều này giúp cô lập công việc chuẩn bị phát hành để quá trình phát triển có thể tiếp tục mà không bị gián đoạn.
3. Các bản sửa lỗi có thể được thêm vào release branch, nhưng các tính năng chính nên chờ bản phát hành trong tương lai.
4. Khi đã sẵn sàng, release branch sẽ được hợp nhất – merge vào master và được gắn thẻ số phiên bản.
GitHub có thể sử dụng các thẻ này để giúp bạn tạo ghi chú phát hành.
5. Cùng một release branch nên được hợp nhất – merge trở lại develop để đồng bộ.
6. Nếu phát sinh lỗi sản xuất nghiêm trọng, một **hotfix branch** sẽ được tạo từ master.
Sau khi sửa xong, nhánh này sẽ được hợp nhất – merge vào cả master và develop.



Git Flow, **main branches**



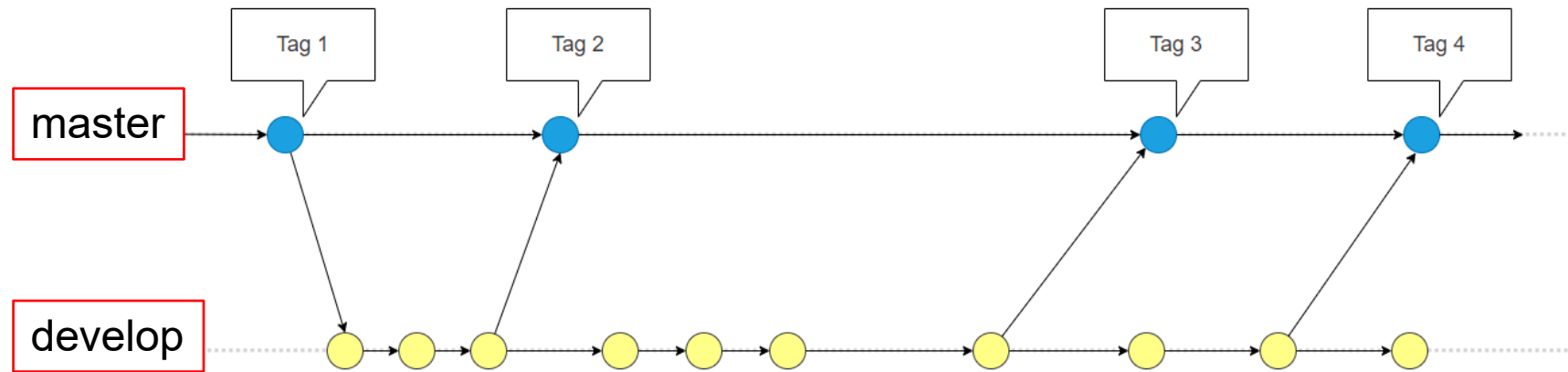
Main branches là “master” và “develop”

`origin/master` là nhánh chính (main),
`head` chứa source code **luôn là code sẵn sàng production ready state**

`origin/develop` là nhánh chính (main),
`head` chứa source code **luôn là code mới nhất – next release state**



Git Flow, **main branches**



Khi source code trong **develop branch** đạt đến điểm ổn định và sẵn sàng để phát hành, tất cả các thay đổi sẽ được tích hợp trở lại **master branch**

Vì vậy, mỗi khi các thay đổi đã được tích hợp vào **master branch**, đó chính là thời điểm có một phiên bản mới (được gắn thẻ - Tag)



Git Flow, **supporting branches**

Supporting branches là

- các branch **“hotfix”**
- các branch **“release”**
- các branch **“feature”**

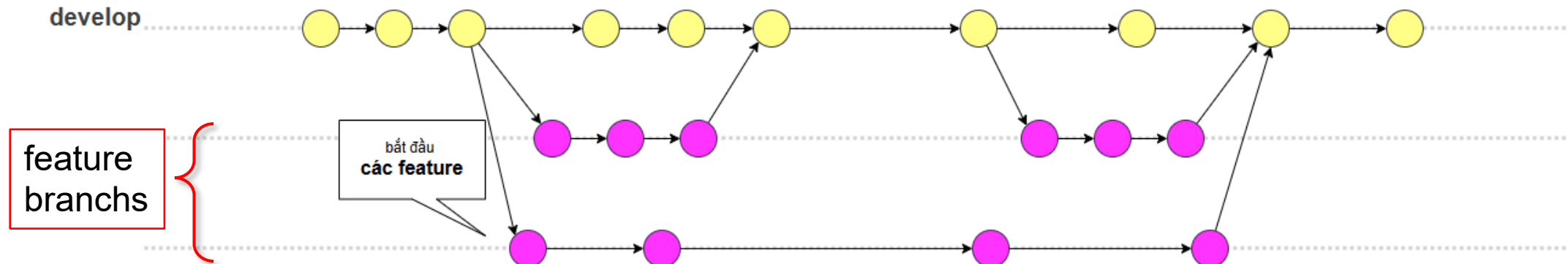


Các branch này có thời hạn tồn tại

Mỗi nhánh này có mục đích và nhiệm vụ riêng biệt, cụ thể, và bị ràng buộc bởi **các quy tắc về tích hợp – merge** giữa các nhánh.



Git Flow, **feature branches**

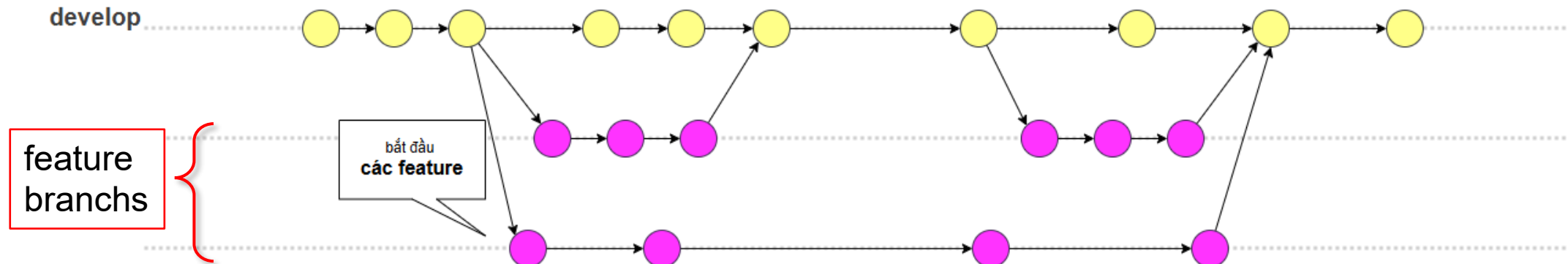


feature branches rules

- được tách ra từ develop branch
- phải được merge về develop branch
- quy ước đặt tên : tùy ý nhưng ngoại trừ các từ sau
`"master", "develop", "release-*", "hotfix-*"`
- thường chỉ tồn tại ở developer repo, không tồn tại ở **"origin"**



Git Flow, **feature branches**

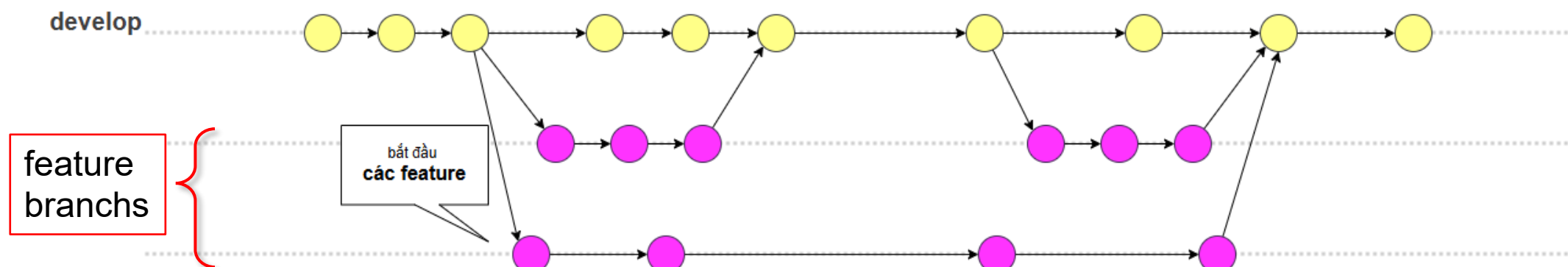


feature branches

- Dùng để phát triển các tính năng mới cho bản phát hành sắp tới.
- Khi bắt đầu phát triển một tính năng, bản phát hành mục tiêu mà tính năng này sẽ được tích hợp có thể chưa được biết tại thời điểm đó.
- Bản chất của “feature branch” là nó tồn tại trong suốt quá trình phát triển tính năng, nhưng cuối cùng sẽ được sáp nhập trở lại vào “develop branch” (để chắc chắn thêm tính năng mới vào bản phát hành sắp tới) hoặc bị loại bỏ (trong trường hợp thử nghiệm không thành công).



Git Flow, **feature branches**



feature branches

- Dùng để phát triển các tính năng mới cho bản phát hành sắp tới.
- Khi bắt đầu phát triển một tính năng, bản phát hành mục tiêu mà tính năng này sẽ được tích hợp có thể chưa được biết tại thời điểm đó.
- Bản chất của “feature branch” là nó tồn tại trong suốt quá trình phát triển tính năng, nhưng cuối cùng sẽ được sáp nhập trở lại vào “develop branch” (để chắc chắn thêm tính năng mới vào bản phát hành sắp tới) hoặc bị loại bỏ (trong trường hợp thử nghiệm không thành công).



Git Flow, **feature branches** & ví dụ

1. Tạo các feature branches, từ develop branch

Command

```
# Tạo branch và chuyển ngay đến branch vừa tạo  
# Ví dụ "feature/F" (các chức năng thuộc Front-End)  
# Ví dụ "feature/B" (các chức năng thuộc Back-End)  
git checkout -b <feature-branch-name> develop
```

Command

```
# Tạo Branch, không chuyển đến Branch mới tạo  
git branch <feature-branch-name> develop
```

Command

```
# Liệt kê các branches  
git branch
```

Output

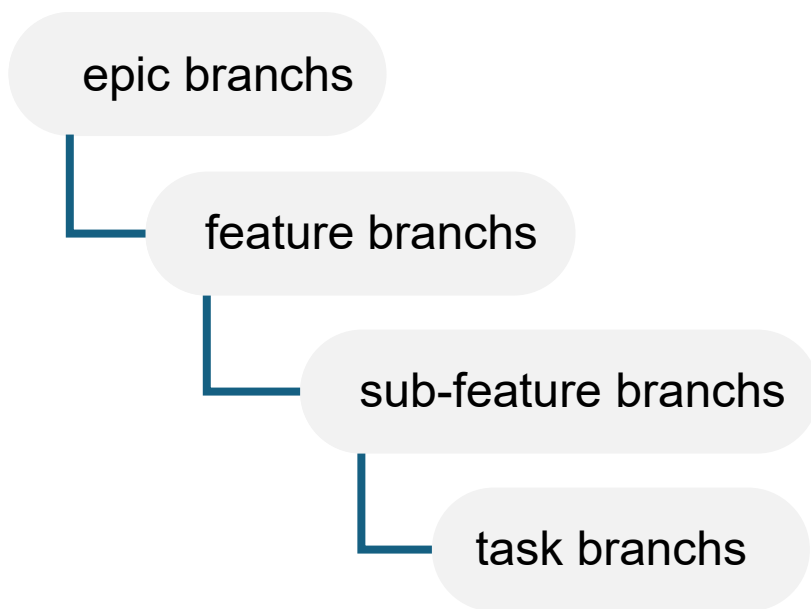
```
develop  
* feature/FE  
feature/BE  
master
```



Git Flow, **feature branches** & ví dụ

2. Lập trình xây dựng các features

Tùy theo quy mô của dự án, tất cả các chức năng có thể áp dụng cách phân cấp và quy ước đặt tên



Ví dụ

`<feature-branch>/<sub-feature-branch>`

`<feature-branch>/<task-branch>`



Git Flow, **feature branches** & ví dụ

3. Hoàn tất xây dựng các features và **merge** vào **develop**

Command

```
# Chuyển về develop branch
git checkout develop

# Merge vào develop branch
# git merge --no-ff feature/FE
git merge --no-ff <feature-branch-name>

# Xóa feature branch đã merge
git branch -d <feature-branch-name>

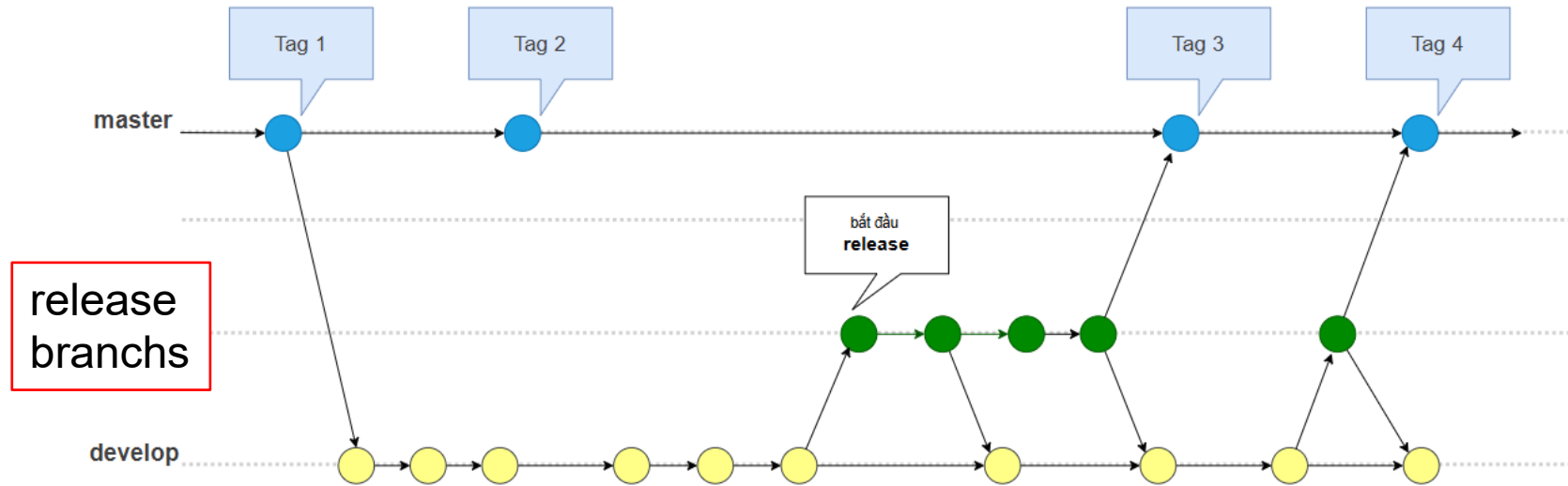
# Push develop branch đã merge các feature
git push origin develop
```

Ghi chú

Tham số *--no-ff*, được dùng để ghi lại thông tin rõ ràng về thời điểm, cách thức feature branch được merge vào develop branch



Git Flow, **release branches**

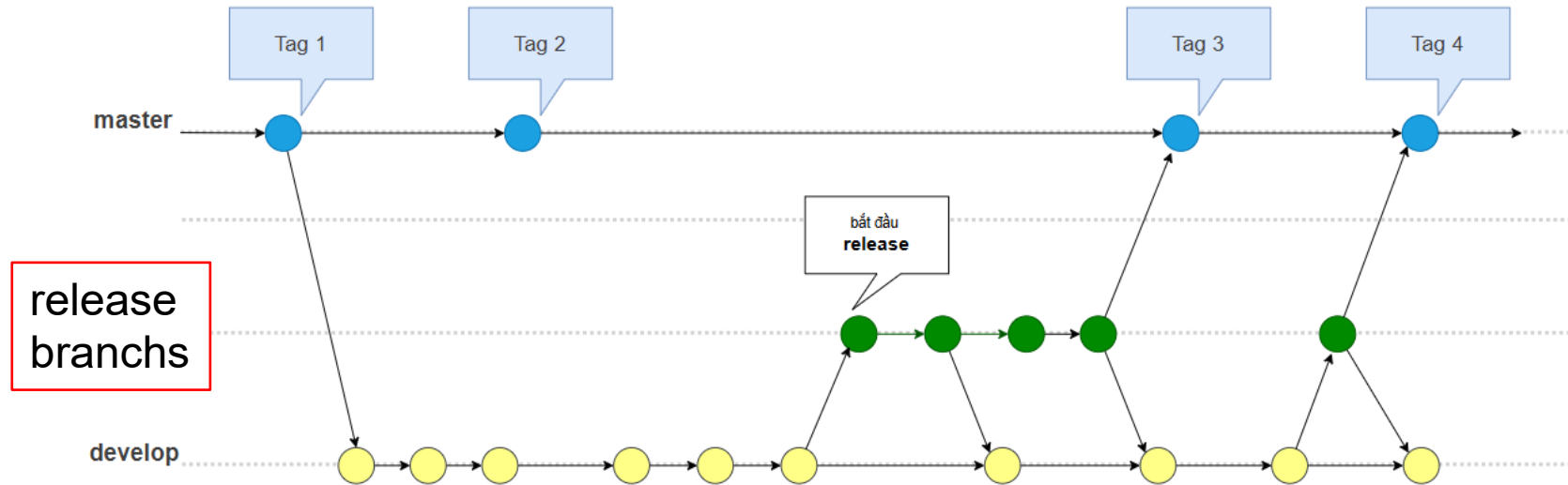


release branches rules

- được tách ra từ develop branch
- phải được merge về develop branch và master branch
- quy ước đặt tên như sau
"release-*



Git Flow, **release branches**



release branches

- Thời điểm quan trọng để tách release branch mới từ develop branch là khi develop branch (gần như) phản ánh trạng thái mong muốn của bản phát hành mới. Ít nhất tất cả các tính năng được nhắm mục tiêu cho bản phát hành sắp được xây dựng phải được hợp nhất – merge vào develop branch tại thời điểm này.
- Tất cả các tính năng được nhắm mục tiêu cho các bản phát hành trong tương lai có thể không được hợp nhất - chúng phải đợi cho đến khi nhánh phát hành được tách ra.



Git Flow, **release branches** & ví dụ

1. Tạo release branch, từ develop branch

Command

```
# Tạo branch và chuyển ngay đến branch vừa tạo
# Ví dụ "release-1.2" (bản phát hành release 1.2)
git checkout -b <release-branch-name> develop

# Commit release version
# Ví dụ
# git commit -a -m "Bản phát hành release-1.2"
git commit -a -m "release-message"
```

Ghi chú

Tham số **-a**, viết tắt của **-all** được dùng để tự động thêm (lệnh git add .) tất cả các file đã được kiểm soát vào khu vực chuẩn bị commit



Git Flow, **release branches** & ví dụ

2a. Merge release branch vào master branch

Command

```
# Chuyển đến master branch
git checkout master

# Merge release branch vào master branch
# Ví dụ git merge --no-ff release-1.2
git merge --no-ff <release-branch-name>

# Gán thẻ tag
# Ví dụ git tag -a 1.2
git tag -a <release-number>
```

Ghi chú

Tham số **-a** (*automated tag*) của lệnh `git tag` dùng để bật chế độ tự động tạo dữ liệu cho tag.



Git Flow, **release branches** & ví dụ

2b. Push master branch lên GitHub

Command

Push master branch

```
git push origin master
```

Push tag mới

```
git push origin --tags
```

Ghi chú

Sau khi merge xong release branch, cần push master branch và push tag để đồng bộ và thông báo cho các thành viên trong nhóm



Git Flow, **release branches** & ví dụ

2c. Merge release branch vào develop branch

Command

Chuyển đến develop branch

```
git checkout develop
```

Merge release branch vào develop branch

Ví dụ `git merge --no-ff release-1.2`

```
git merge --no-ff <release-branch-name>
```

Lưu ý

Cần merge release branch vào develop branch để đảm bảo tính nhất quán, những lỗi đã sửa ở release branch cũng được cập nhật vào develop branch, không bị lỗi tiếp ở những lần sau



Git Flow, **release branches** & ví dụ

2d. Delete release branch, sau khi hoàn tất hợp nhất

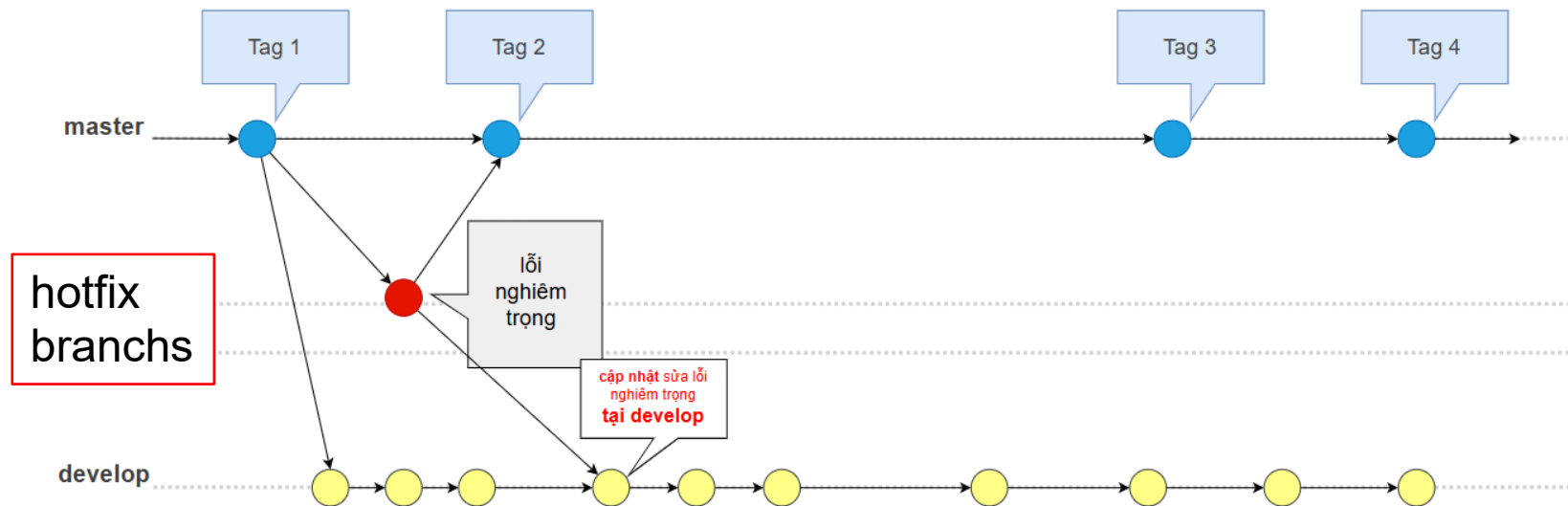
Command

```
# Xóa release branch  
# Ví dụ  
# git branch -d release-1.2  
git branch -d <release-branch-name>
```

Sau khi đã hoàn tất hợp nhất release branch vào master và develop branch, thì có thể xóa release branch để gọn gàng, sạch sẽ.



Git Flow, hotfix branches

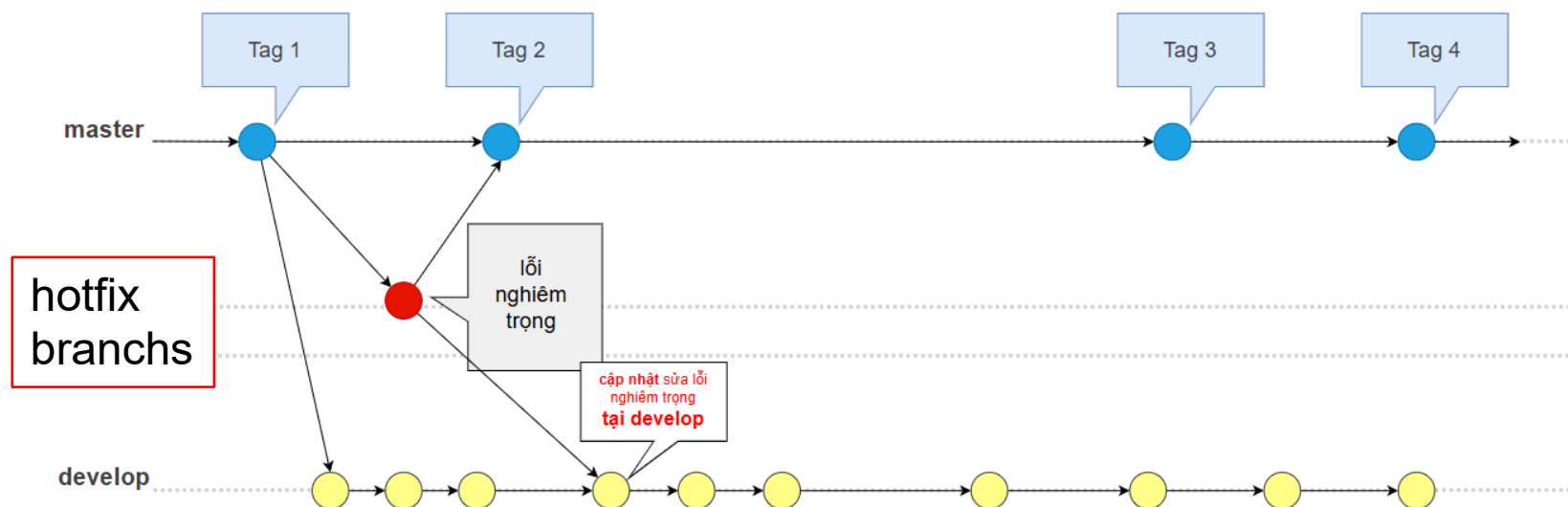


hotfix branches rules

- được tách ra từ master branch
- phải được merge về develop branch và master branch
- quy ước đặt tên như sau
"hotfix-*



Git Flow, hotfix branches



hotfix branches

- Khi một **lỗi nghiêm trọng** trong phiên bản production **cần được giải quyết ngay lập tức**, một hotfix branch có thể được tách ra khỏi thẻ tương ứng trên nhánh đánh dấu phiên bản production.
- Các hotfix branch rất giống với các release branch ở chỗ chúng cũng nhằm mục đích chuẩn bị cho một bản phát hành production mới, mặc dù không được lên kế hoạch trước.



Git Flow, hotfix branches & ví dụ

1. Tạo hotfix branch, từ master branch

Command

```
# Tạo branch và chuyển ngay đến branch vừa tạo
# Ví dụ "hotfix-1.23" (bản sửa lỗi hotfix 1.23)
git checkout -b <hotfix-branch-name> master

# Commit hotfix version
# Ví dụ
# git commit -a -m "Bản sửa lỗi hotfix-1.23"
git commit -a -m "hotfix-message"
```

Ghi chú

Tham số **-a**, viết tắt của **-all** được dùng để tự động thêm (lệnh `git add .`) tất cả các file đã được kiểm soát vào khu vực chuẩn bị commit



Git Flow, hotfix branches & ví dụ

2a. Merge hotfix branch, vào master branch

Command

Chuyển đến master branch

```
git checkout master
```

Merge hotfix branch vào master branch

Ví dụ `git merge --no-ff hotfix-1.23`

```
git merge --no-ff <hotfix-branch-name>
```

Gán thẻ tag

Ví dụ `git tag -a 1.23`

```
git tag -a <hotfix-number>
```



Git Flow, hotfix branches & ví dụ

2b. Merge hotfix branch, vào develop branch

Command

Chuyển đến develop branch

```
git checkout develop
```

Merge hotfix branch vào develop branch

Ví dụ `git merge --no-ff hotfix-1.23`

```
git merge --no-ff <hotfix-branch-name>
```




Git Flow, hotfix branches & ví dụ

3. Xóa hotfix branch

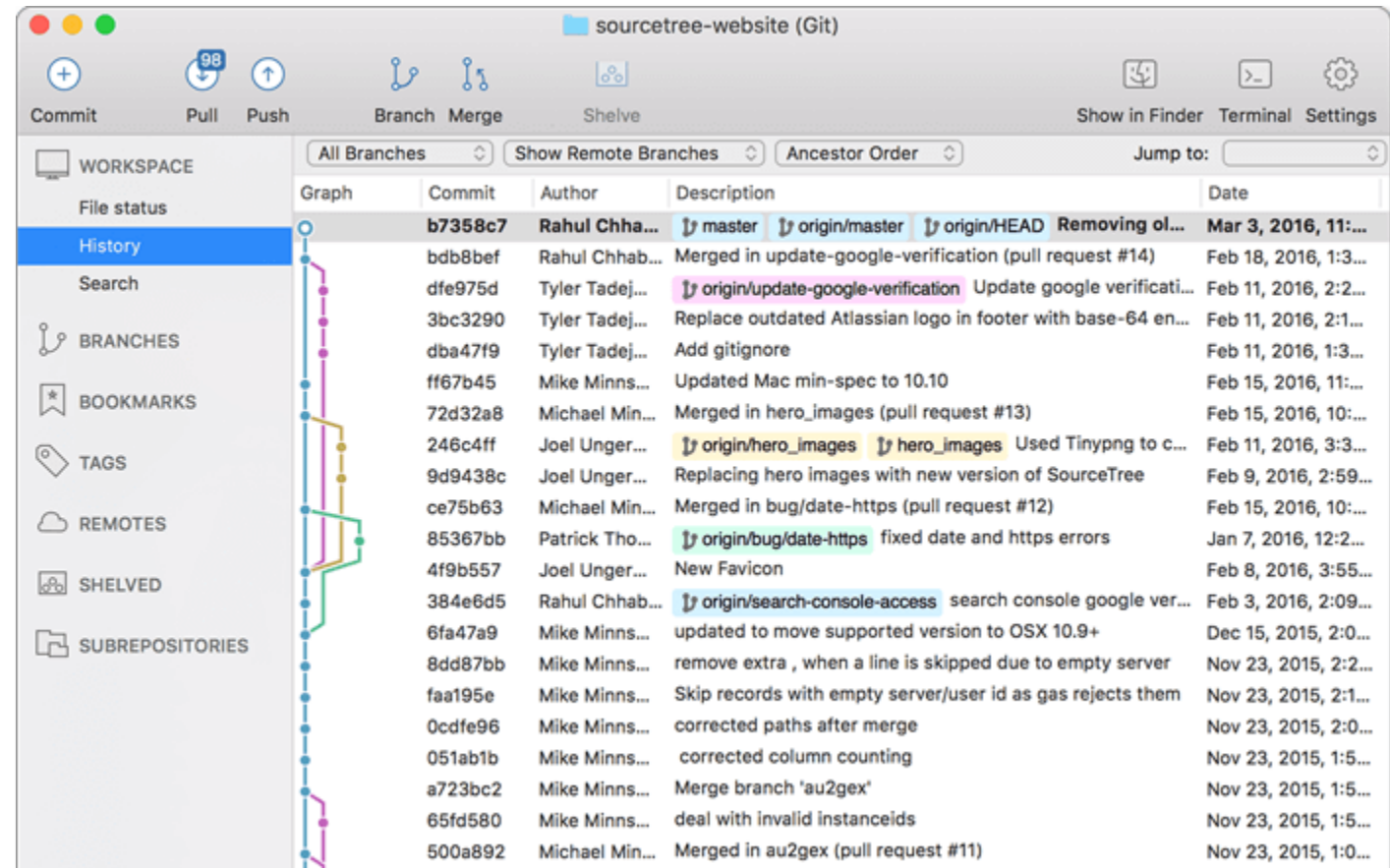
Command

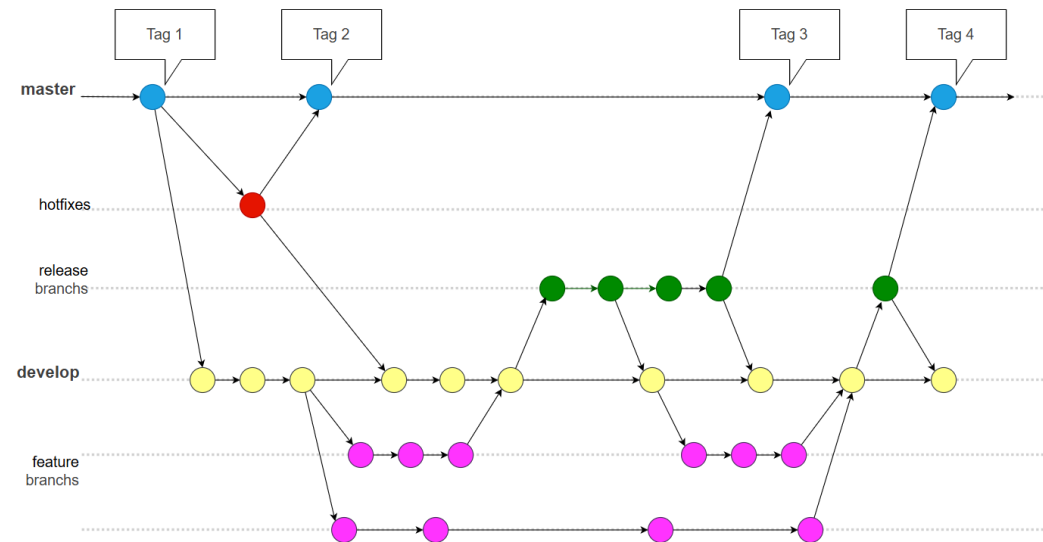
```
# Xóa hotfix branch
```

```
# Ví dụ git branch -d hotfix-1.23
```

```
git branch -d <hotfix-branch-name>
```

SourceTree – a beautiful Git GUI





XIN CẢM ƠN

09-2025
Nhóm thực hiện
Thân Hoàng Lộc
Trần Quang Bình
Trần Văn Huy
Trần Anh Tuấn