

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN - BỘ MÔN CÔNG NGHỆ PHẦN MỀM



Topic 4:

Source Code Management – Git&GitHub

Giảng viên: Phạm Thị Thương – Bộ môn CNPM – Khoa CNTT
Email: ptthuong@ictu.edu.vn

Mục tiêu

- ▶ Giúp SV làm quen với Git, các khái niệm và các câu lệnh của Git
- ▶ Biết tạo dự án Jenkins với mã nguồn lấy từ kho trên GitHub
 - ▶ Tích hợp, dịch và chạy code tự động, liên tục

Main Content

- I. Giới thiệu Git & Github
- II. Hệ thống quản lý phiên bản phân tán Git
- III. Dịch vụ repository (free, tốt nhất)- GitHub
- IV. Mô hình phân nhánh (tham khảo).

Git



I. Giới thiệu Git & Github

► Git là gì? Tại sao nên dùng?

► **Git - Hệ thống quản lý phiên bản phân tán** *(Distributed Version Control System – DVCS)*

- Là một trong những DVCS phổ biến nhất hiện nay.
- Nhiều tổ chức, dự án lớn đang sử dụng Git:
 - Google, Facebook, Microsoft, twitter, LinkedIn, Android, Unix, Eclipse,

I. Giới thiệu Git & Github

► Git là gì? Tại sao nên dùng?

- Sử dụng Git - một kỹ năng bắt buộc đối với lập trình viên khi tham gia vào các dự án có nhiều thành viên.
- Git đang trở nên thông dụng và dần thay thế các SVN truyền thống
- Có nhiều hệ thống dựa trên nền tảng Git như github, gitlab, ... ra đời nhằm giải quyết vấn đề lưu trữ và quản lý mã nguồn **trực tuyến** (~ ta không cần phải xây dựng một remote repository mà đã có các hệ thống đó trợ giúp)

I. Giới thiệu Git & Github

- ▶ Git ≠ Github:

- ▶ Git là hệ thống quản lý mã nguồn phân tán, nó cung cấp các lệnh giúp quản lý mã nguồn dễ dàng hơn.

- ▶ Github:

- ▶ ~ Dịch vụ kho chứa, cho phép tạo các remote repository để quản lý mã nguồn online,

- ▶ Các chức năng được phát triển dựa trên thư viện của Git.

I. Giới thiệu Git & Github

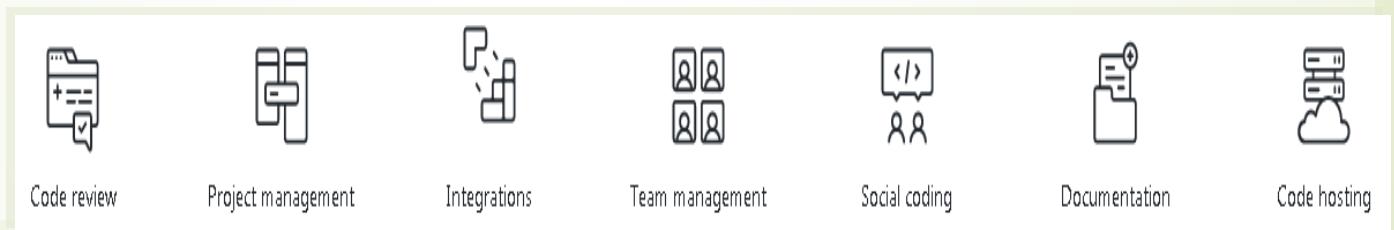
- ▶ Github là gì? tại sao nên dùng
 - ▶ Cho phép tạo remote repository mà không cần mua server
 - Mua server & xây dựng remote trên đó => tốn chi phí và có thể không bảo mật, thay vào đó sử dụng Github để tạo remote repo => Không tốn kinh phí + cơ chế bảo mật cao

I. Giới thiệu Git & Github

- ▶ Github là gì? tại sao nên dùng
 - ▶ **All your code in one place**
 - ▶ GitHub is one of the largest code hosts in the world with over 100 million* projects. Private, public, or open source, all repositories are equipped with tools to help you host, version, and release code.

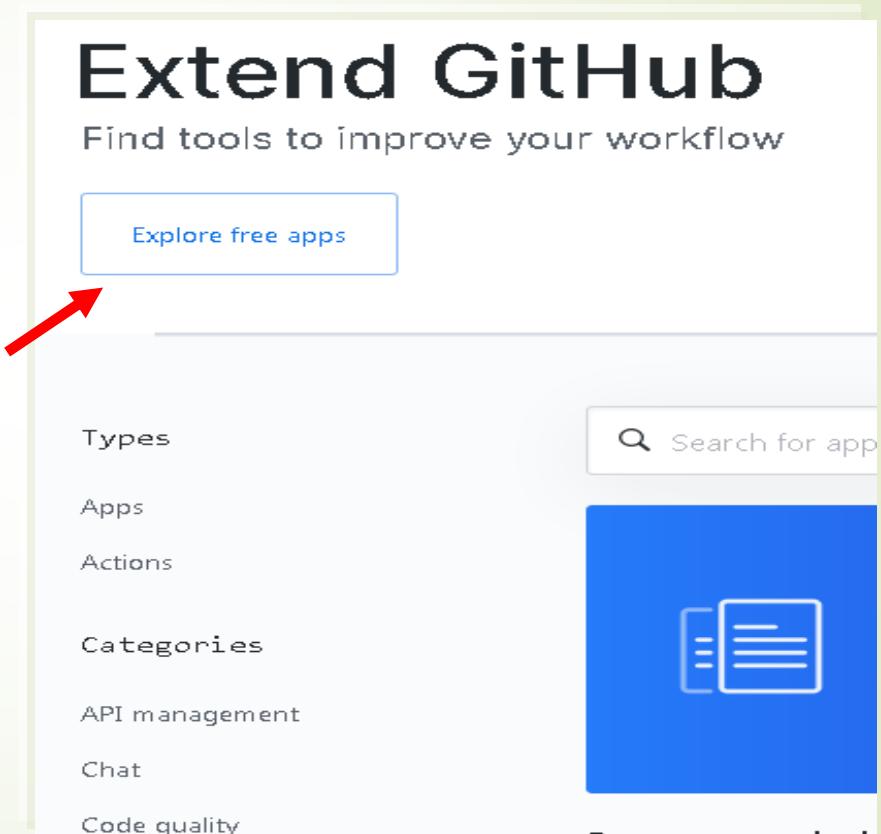
I. Giới thiệu Git & Github

- ▶ GitHub – Features:
 - ▶ Developers: works



I. Giới thiệu Git & Github

- ▶ GitHub – Marketplace:



Types

Apps

Actions

Categories

API management

Chat

Code quality

Code review

Continuous integration

Dependency management

Deployment

IDEs

Search for apps and actions

Free

Tools that provide a free tier.

143 results filtered by Free x



Buddy ✓

One-click delivery automation for Web Developers



Dependabot Preview ✓

Automated dependency updates for Ruby, JavaScript, Python, Go, PHP, Elixir, Rust, Java and .NET



AccessLint ✓

Find accessibility issues in your pull requests



Rollbar ✓

Real-time, full-stack error monitoring and debugging for developers



Codacy ✓

Automated code reviews to help developers ship better software, faster



Better Code Hub ✓

A Benchmarked Definition of Done for Code Quality



Airbrake ✓



ZenHub ✓

Main Content

- I. Giới thiệu Git & Github
- II. Hệ quản lý phiên bản phân tán Git**
- III. Dịch vụ repository (free, best) - GitHub
- IV. Mô hình phân nhánh (tham khảo)



II. Hệ quản lý phiên bản phân tán Git

1. Cài đặt Git và thiết lập ban đầu
2. Cách tạo một repository
3. Hiểu thêm về Commit và Staging Area
4. Git Log và Undo Commit
5. Đánh dấu commit với Tag
6. Remote Repository & Origin
7. Branch – Kỹ thuật phân nhánh

1. Cài đặt Git & thiết lập ban đầu

- ▶ Toàn bộ quy trình làm việc với Git đều diễn ra các dòng lệnh (tuy nhiên, với Windows có Git với GUI).
 - ▶ Để có thể sử dụng Git, phải cài đặt trên MT
 - ▶ Cài trên Linux, Mac OS and **Window**

1. Cài đặt Git & thiết lập ban đầu

- ▶ Cài Git trên Windows
 - ▶ Tải file .exe cài đặt Git tại địa chỉ <http://git-scm.com/download/win>.
 - ▶ Giữ nguyên tùy chọn mặc định khi cài đặt.
 - ▶ Cài đặt thành công
=> Mở **Git Bash** để bắt đầu sử dụng các lệnh của Git.

1. Cài đặt Git & thiết lập ban đầu

- ▶ Thiết lập chứng thực cá nhân sau cài đặt:
 - ▶ Khai báo tên và địa chỉ email vào trong file cấu hình của Git trên máy:
 - ▶ `$ git config --global user.name "Pham Thuong"`
 - ▶ `$ git config --global user.email ptthuong@ictu.edu.vn`
 - ▶ Kiểm tra thông tin chứng thực bằng cách:
 - ▶ (1) xem tập tin `~/.gitconfig` (dấu `~` nghĩa là thư mục gốc của user).

```
$ cat ~/.gitconfig
[user]
    name = Pham Thuong
    email = ptthuong@ictu.edu.vn
```
 - ▶ (2) dùng lệnh `git config --list` để liệt kê ds các thiết lập hiện tại



II. Hệ thống quản lý phiên bản phân tán Git

1. Cài đặt Git và thiết lập ban đầu
2. **Cách tạo repository**
3. Hiểu thêm về Commit và Staging Area
4. Git Log và Undo Commit
5. Đánh dấu commit với Tag
6. Remote Repository & Origin
7. Branch – Kỹ thuật phân nhánh

2. Cách tạo một repository

- ▶ **Repository (kho chứa)**

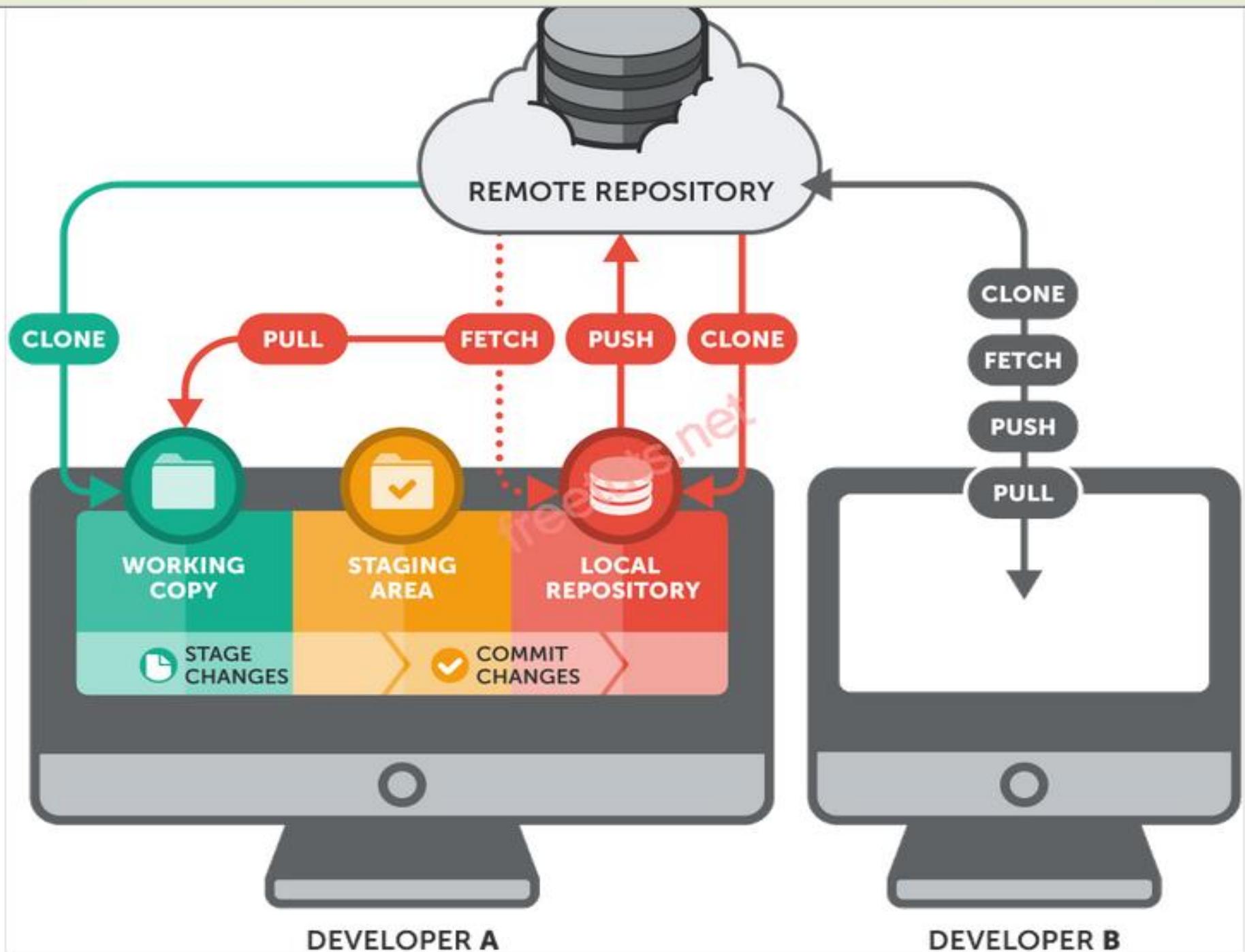
- ▶ 2 loại repository:

- ▶ ***Local Repository :***

- ▶ Kho chứa trên máy cá nhân, repo này sẽ đồng bộ hóa với remote repo bằng các lệnh của git

- ▶ ***Remote Repository***

- ▶ Kho chứa trên một máy chủ từ xa (ví dụ: Github).



2. Cách tạo một repository

a. Tạo local repository

- ▶ Cách 1: Clone kho remote
- ▶ Cách 2: Share code dự án trong IDE
- ▶ Cách 3: Khởi tạo kho rỗng chưa có dữ liệu

2. Cách tạo một repository

a. Tạo local repository

- ▶ Cách 1: Tạo kho rỗng

B1. cd đến thư mục mã nguồn của dự án

- ▶ Ví dụ

C:\Users\Administrator\Desktop\TestBuild1\HelloWorld

B2. gõ: git init để khởi tạo repository trong thư mục đó

```
Administrator@HaiThangBeo MINGW64 ~
$ cd C:/Users/Administrator/Desktop/TestBuild1/HelloWorld

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld
$ git init
Initialized empty Git repository in C:/Users/Administrator/Desktop/TestBuild1/HelloWorld/.git/

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ |
```

2. Cách tạo một repository

a. Tạo local repository

- ▶ Cách 1: Tạo kho rỗng

- ▶ Kết quả:

- ▶ Khởi tạo một kho Git rỗng tại thư mục hiện thời.

- ▶ Lưu ý:

- ▶ Thư mục ẩn **.git/** : chứa các thiết lập về Git & toàn bộ thông tin về kho chứa (chúng ta không cần can thiệp vào thư mục **.git/** này).

2. Cách tạo một repository

a. Tạo local repository

► Lưu ý:

- Nếu kho đã có sẵn mã nguồn => cần **đưa các tập tin về trạng thái Tracked** để có thể làm việc được với Git
- ⇒ Sử dụng lệnh **git add tên_file (*)** - gom toàn bộ file)
 - ⇒ **git status**: kiểm tra d.sách các file được tracked.

► Ví dụ:

git add *.* ↵

git status ↵

=> Kết quả: màn hình (dưới)

```
MINGW64:/c/Users/Administrator/Desktop/TestBuild1/HelloWorld
Administrator@HaiThangBeo MINGW64 ~
$ E:
bash: E:: command not found

Administrator@HaiThangBeo MINGW64 ~
$ cd C:/Users/Administrator/Desktop/TestBuild1/HelloWorld

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld
$ git init
Initialized empty Git repository in C:/Users/Administrator/Desktop/TestBuild1/HelloWorld/.git/
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git add *.*

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git status
On branch master
No commits yet

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

    new file:   ANT_script.xml
    new file:   HelloWorld.jar
    new file:   build.xml
    new file:   manifest.mf

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

    .classpath
    .project
    build/
    dist/
    nbproject/
    src/

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ |
```

Các
file đã

2. Cách tạo một repository

a. Tạo local repository

- ▶ Commit/ủy thác mã nguồn lên kho:

git commit -m "Lời nhắn"

```
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git commit -m "this is the first commit"
[master (root-commit) daf1fad] this is the first commit
 5 files changed, 95 insertions(+)
 create mode 100644 ANT_script.xml
 create mode 100644 HelloWorld.jar
 create mode 100644 build.xml
 create mode 100644 file1.txt
 create mode 100644 manifest.mf

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ |
```

2. Cách tạo một repository

a. Tạo local repository

- ▶ Cách 2: Clone dữ liệu từ kho remote về
- ▶ Cách 3: Share code dự án để làm việc với team
=> Hoàn thành việc tạo kho cục bộ Git.

2. Cách tạo một repository

b. Tạo remote repository trên Github

- ▶ **Bước 1:** Đăng ký tài khoản, đăng nhập vào [Github](#)
- ▶ **Bước 2:** Click: New repository.
- ▶ **Bước 3:** Đặt tên kho: ví dụ: hoc_git_basic; Chọn loại kho: public (ai cũng có thể clone) hoặc private (chỉ có những người được cấp quyền mới có thể clone).
 - ▶ Click: Create Repository
 - ⇒ Màn hình (dưới)

Create a new repository

A repository contains all project files, including the revision history.

Owner

Repository name *



/ hoc_git_basic



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

Description (optional)

Public

Anyone can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

2. Cách tạo một repository

b. Tạo remote repository trên Github

- ▶ Khi tạo xong kho chứa sẽ có địa chỉ URL:
[https://github.com/\\$user-name/\\$repository](https://github.com/$user-name/$repository),
- ▶ ví dụ https://github.com/PhamThuongBlog/hoc_git_basic.

 Code

 Issues 0

 Pull requests 0

 Projects 0

 Wiki

 Insights

 Settings

No description, website, or topics provided.

 Edit

Manage topics

 1 commit

 1 branch

 0 releases

 1 contributor

Branch: master ▾

New pull request

Create new file

Upload files

Find File

Clone or download ▾

 PhamThuongBlog Initial commit

Latest commit 5d3020b a minute ago

 README.md

Initial commit

a minute ago

 README.md



hoc_git_basic

2. Cách tạo một repository

c. Làm việc với các repository

- ▶ Push mã nguồn từ kho local lên kho remote:
 - ▶ **Git push <URL of the remote repository> <branch>**
 - ▶ Hoặc, clone kho Github (nếu kho đã có sẵn mã nguồn) về máy tính: **git clone địa chỉ**.

```
$ git clone https://github.com/PhamThuongBlog/hoc_git_basic
Cloning into 'hoc_git_basic'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

2. Cách tạo một repository

c. Làm việc với các repository

- ▶ Truy cập vào working tree (thư mục vừa clone kho về)
- ▶ Make change: ví dụ:
 - ▶ Tạo 1 file *README_more.md*, sau đó dùng **git add** để đưa file vào Staging Area và commit nó.

```
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ echo "Huong dan khai thac Git co ban" > Readme_more.md

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git add Readme_more.md
warning: LF will be replaced by CRLF in Readme_more.md.
The file will have its original line endings in your working directory

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git commit -m "The next commit on GitHub"
[master 10e2f06] The next commit on GitHub
 1 file changed, 1 insertion(+)
 create mode 100644 Readme_more.md

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ |
```

2. Cách tạo một repository

c. Làm việc với các repository

- ▶ Sau khi commit, tập tin vẫn không xuất hiện trong kho Github => Cần phải đẩy nó lên Github, dùng lệnh:

git push <ten remote rep>/URL <tên nhanh>

- ▶ Ví dụ: **\$ git push origin master**

- ▶ Lưu ý: Cần nhập tài khoản và mật khẩu Github khi push
- ▶ Ví dụ: xem hình (dưới)

2. Cách tạo một repository

```
$ git push origin master
Username for 'https://github.com': PhamThuongBlog
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/PhamThuongBlog/hoc_git_basic
  5d3020b..d860e37  master -> master
```

- Kết quả:
 - Push thành công ~ đồng bộ file lên kho Github
 - Xem hình (dưới)

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

No description, website, or topics provided.

Edit

Manage topics



2 commits

1 branch

0 releases

1 contributor

Branch: master ▾

New pull request

Create new file

Upload files

Find File

Clone or download ▾

ptthuongECJ First commit on Github ...

Latest commit d860e37 9 minutes ago

README.md

Initial commit

30 minutes ago

README_more.md

First commit on Github

9 minutes ago

README.md



hoc_git_basic



II. Hệ thống quản lý phiên bản phân tán Git

1. Cài đặt Git và thiết lập ban đầu
2. Cách tạo repository
3. **Hiểu thêm về Staging Area**
4. Git Log và Undo Commit
5. Đánh dấu commit với Tag
6. Sơ lược Remote Respository và Origin
7. Branch – Kỹ thuật phân nhánh.

3. Hiểu thêm về Staging Area

- ▶ Staging Area:

- ▶ ~ Khu vực nắm giữ ∀ thay đổi code (snapshot) chuẩn bị để commit lên kho
- ▶ Git sử dụng thêm khu vực trung gian này để giúp theo dõi, điều khiển, đối chiếu các thay đổi so với tệp gốc dễ dàng hơn.
 - ▶ Helps in reviewing changes: by using: **git diff**
 - ▶ Helps when a merge has conflicts: show the merge conflicts
 - ▶ ...



working directory

git add

staging area

git commit

repository

Mô hình giải thích cách hoạt động của Staging Area.

3. Hiểu thêm về Staging Area

- ▶ Để commit một tập tin đã được Tracked mà không đưa nó vào Staging Area => sử dụng tham số -a trong lệnh *git commit*.

- ▶ Ví dụ:

git commit -a -m "Skipped Staging Are to commit".

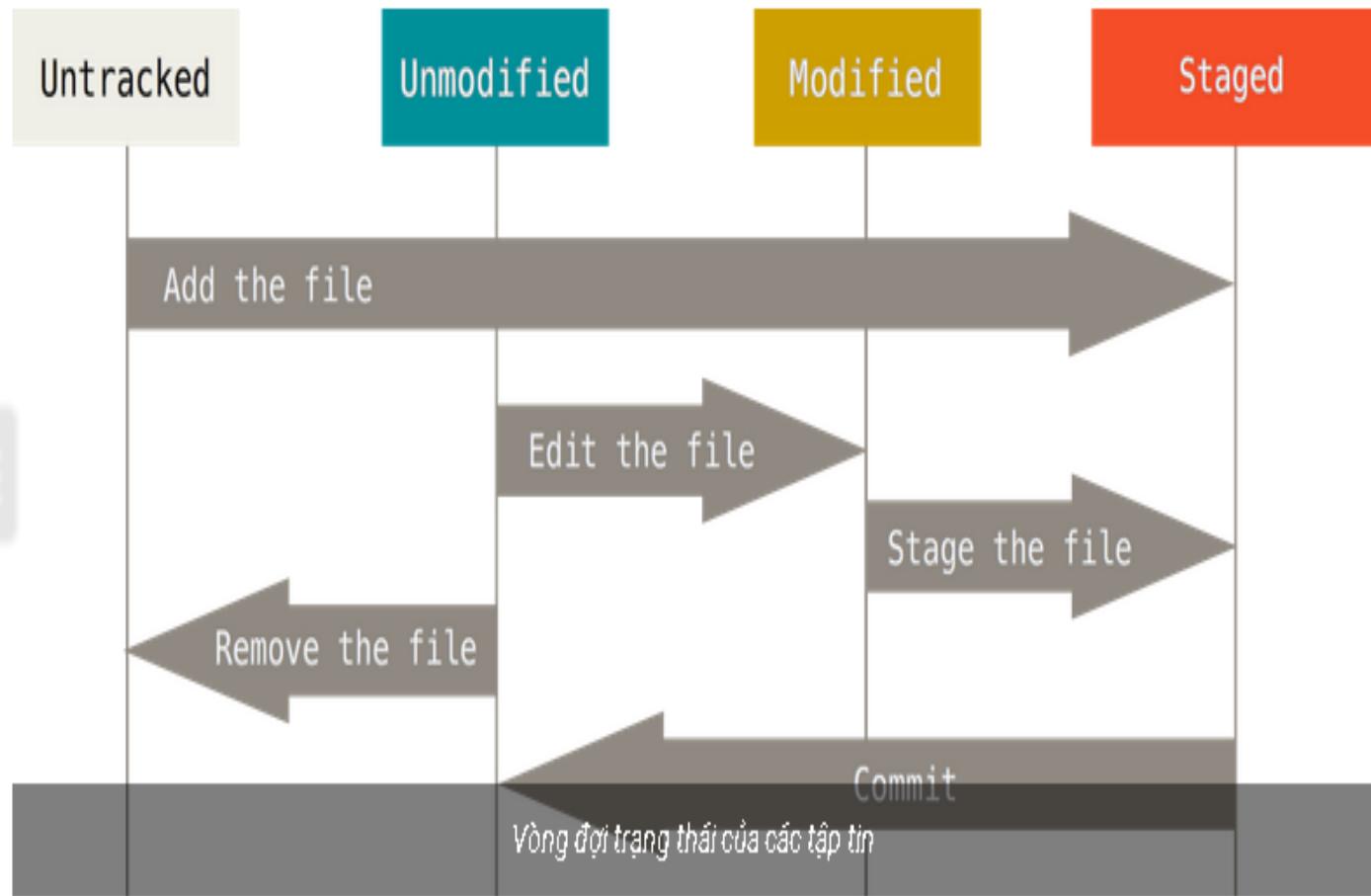
⇒ Các trạng thái của tệp tin?

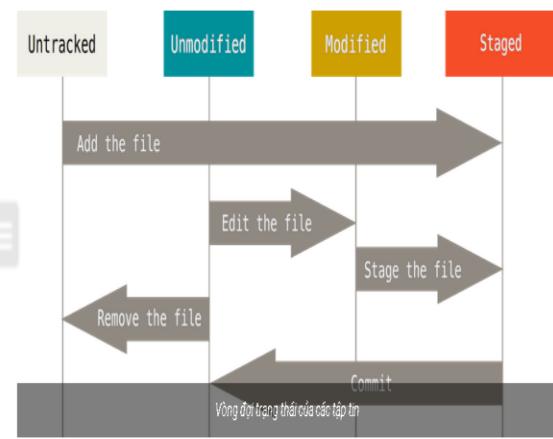
3. Hiểu thêm về Staging Area

► Các trạng thái của tệp tin:

- ▶ => Cần nắm vững để chủ động khi sd các lệnh git và tránh các lỗi có thể mắc phải
- ▶ 2 trạng thái chính của file: **Tracked** và **Untracked**:
 - ▶ **Tracked** – Tập tin đánh dấu được theo dõi trong Git. Ở trạng thái này nó có thêm các trạng thái phụ khác: **Unmodified** (chưa chỉnh sửa gì), **Modified** (đã chỉnh sửa) và **Staged** (sẵn sàng để commit).
 - ▶ **Untracked** – Là tập tin còn lại mà ta không muốn làm việc với nó trong Git.

Tìm hiểu thêm về trạng thái





Các trạng thái tệp tin?

► Untracked

- Khi tạo/thêm mới 1 tập tin vào working space nó sẽ ở trạng thái Untracked.
- Ví dụ: Tạo file `test.html`, sau đó gõ **git status** để xem trạng thái của nó:

Lệnh `touch` tạo ra một tập tin rỗng

Tập tin ở trạng thái untracked

```
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ touch test.html

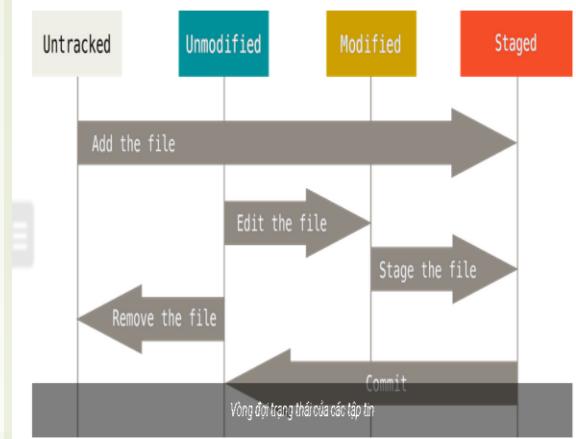
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    .classpath
    .project
    build/
    dist/
    nbproject/
    src/
    test.html

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

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ |
```

Các trạng thái tệp tin?



► Tracked:

- Để đưa tệp tin về Tracked, sử dụng lệnh **git add test.html** và xem lại trạng thái của nó.

- Ví dụ:

```
$ git add test.html
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/Helloworld (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
    new file:   test.html
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .classpath
    .project
    build/
    dist/
    nbproject/
    src/
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/Helloworld (master)$
```

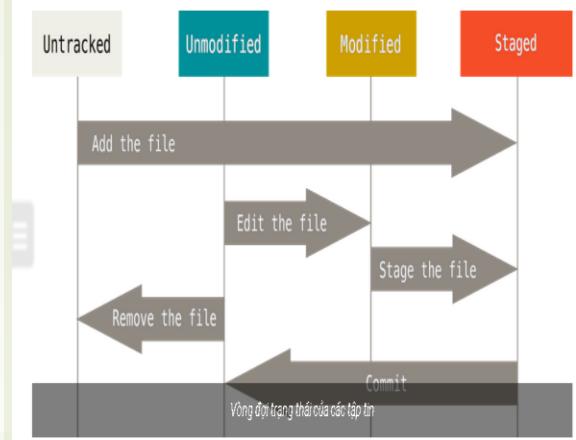
A red arrow points to the line 'new file: test.html'. A green callout bubble contains the text: 'Test.html giờ là sẵn sàng để commit'.

Các trạng thái tập tin?

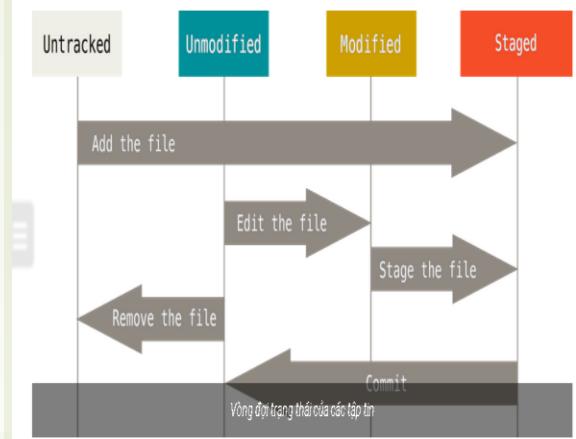
► Tracked

► Lưu ý:

- Sau khi commit test.html sẽ chuyển về trạng thái Unmodified.
- Nếu ta thay đổi nội dung tập tin này thì nó sẽ đưa về trạng thái Modified và nó không thể commit, để commit được cần gõ lệnh ***git add <tệp tin> /git add .***



Các trạng thái tệp tin?



► Tracked

► Lưu ý:

- Khi một tập tin đã được đưa về Tracked thì nó có thể thay đổi giữa 3 trạng thái khác nhau: **Modified**, **Unmodified** và **Staged**.

► Chuyển tập tin từ Untracked về Tracked

- Sử dụng lệnh ***rm tên_file***

=> Đưa tập tin về trạng thái Untracked nhưng không xóa hẳn trong ổ cứng.

- Ví dụ: Xem hình (dưới)

```
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ rm test.html

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   test.html

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

    deleted:   test.html

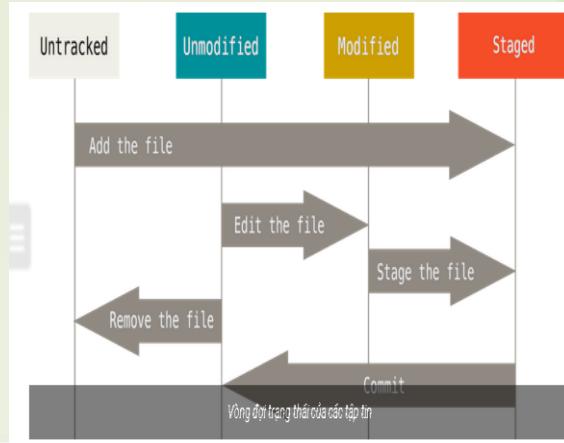
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    .classpath
    .project
    build/
    dist/
    nbproject/
    src/
```



```
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ |
```

Các trạng thái tệp tin?



- ➡ Lệnh: **git rm -f tên_file**
 - ➡ Xóa hẳn file trong bộ nhớ
 - ➡ **Cẩn thận** khi dùng lệnh này.

Các trạng thái tệp tin?

► Lưu ý:

- Lệnh **git add .** => Add mọi thay đổi của kho
- Associate commit with issue:
 - Tạo issue, gán cho ai, fix the issue, commit, push ...
 - Lệnh commit đính kèm thông báo đã fixed issues được giao và close issue
 - **git commit -m "I fixed the pro. close #2"**
 - **git push (để push update lên remote repository)**
 - Nếu gặp lỗi thì gõ: git pull(để tránh xung đột), gõ lại lệnh commit ở trên, và gõ git push



II. Hệ thống quản lý phiên bản phân tán Git

1. Cài đặt Git và thiết lập ban đầu
2. Cách tạo repository
3. Hiểu thêm về Staging Area
- 4. Git Log và Undo Commit**
5. Đánh dấu commit với Tag
6. Sơ lược Remote Respository và Origin
7. Branch – Kỹ thuật phân nhánh

4. Git Log và Undo Commit

- ▶ Khi làm việc theo nhóm, ta thường xuyên phải kiểm tra xem những ai đã commit vào dự án (git log), và undo lần commit trước đó khi cần thiết
 - ▶ Mục này sẽ tìm hiểu về git log & undo commit



4. Git Log và Undo Commit

- a. Xem git log
- b. Lọc log với --pretty
- c. Undo Commit
- d. Bỏ tập tin ra khỏi Staging Area

a. Xem git log

- Xem lịch sử commit, dùng lệnh **git log**.
 - Ví dụ:

```
$ git log
commit d860e37ba53a3effef3dc358d368da1715524973 (HEAD -> master, origin/master,
origin/HEAD)
Author: Pham Thuong <ptthuong@ictu.edu.vn>
Date:   Sun Mar 31 14:17:31 2019 +0700

    First commit on Github
    git status

commit 5d3020be67aad777ae3a9a5db95f3effcc08f6aa
Author: PhamThuongBlog <49114029+PhamThuongBlog@users.noreply.github.com>
Date:   Sun Mar 31 13:55:48 2019 +0700

    Initial commit
```

Mỗi lần commit
sẽ có một
checksum riêng
, ghi rõ ai
commit , ngày
giờ commit

a. Xem git log

- ▶ Xem chi tiết commit hơn:
 - ▶ Bổ sung tham số **-p** vào lệnh
 - ▶ Ví dụ: **git log -p ↵**
 - ▶ Note: Press p to exit review results
 - ▶ Chỉ xem 1 lần commit gần nhất => thêm tham số **-1** vào lệnh.
 - ▶ Ví dụ: **git log -p -1 ↵**
 - ~ Xem chi tiết lần commit gần đây nhất
 - ⇒ Xem hình (dưới)



```
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/Helloworld (master)
$ git log -p -1
commit d04b74f244ab8e43f3f8f1707acef4bc34a01e0a (HEAD -> master)
Author: Pham Thuong <ptthuong@ictu.edu.vn>
Date:   Sun Sep 8 06:03:46 2019 +0700
```

The third commit

```
diff --git a/test.html b/test.html
index e69de29..111c1fb 100644
--- a/test.html
+++ b/test.html
@@ -0,0 +1,5 @@
+<body>
+
+
+
+</body>
\ No newline at end of file
```

```
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/Helloworld (master)
$ |
```

a. Xem git log

► Lưu ý:

- Sử dụng thêm các tùy chọn để xem log.
 - since, --after: Xem các lần commit kể từ ngày nhất định.
 - until: Xem các lần commit trước từ ngày nhất định.
 - author: Xem các lần commit của một người nào đó.
 - grep: Lọc các chuỗi trong log và in ra.

► Ví dụ:

```
$ git log --author=ptthuong@ictu.edu.vn --pretty="%s"  
First commit on Github git status
```

b. Lọc log với --pretty

- ▶ Tham số **--pretty** rất có ích nếu muốn lọc để xem một đối tượng nào đó trong lịch sử commit
 - ▶ Ví dụ: Chỉ xem lời nhắn commit hoặc chỉ xem email của người commit.
 - ⇒ Tham số **--pretty** phải sử dụng kèm với các tag của nó:

git log --pretty = "% tag"

b. Lọc log với --pretty

- ▶ Danh sách %tag:
 - ▶ Các **%tag** phải được đặt trong cặp dấu ngoặc kép; có thể sử dụng nhiều **%tag** khác nhau.
 - ▶ Ví dụ:

```
$ git log --pretty="%an - %s"
Pham Thuong - First commit on Github git status
PhamThuongBlog - Initial commit
```

a. Xem git log

► Danh sách các %tag:

- %H – Commit hash
- %h – Abbreviated commit hash
- %T – Tree hash
- %t – Abbreviated tree hash
- %P – Parent hashes
- %p – Abbreviated parent hashes
- %an – Author name
- %ae – Author e-mail
- %ad – Author date (format respects the –date=option)
- %ar – Author date, relative
- %cn – Committer name
- %ce – Committer email
- %cd – Committer date
- %cr – Committer date, relative
- %s – Subject

c. Undo Commit

- ▶ Undo để xóa message of commit trước & commit lại:
 - ▶ Sử dụng tham số **--amend** trong lệnh **git commit**.
 - ▶ Ví dụ: Undo để sửa commit message

```
$ git log --pretty="%s"
First commit on Github git status
Initial commit
$ git commit --amend -m "FIRST COMMIT ON GITHUB !!!"
[master 3951428] FIRST COMMIT ON GITHUB git log --
pretty=%s!
Date: Sun Mar 31 14:17:31 2019 +0700
2 files changed, 1 insertion(+)
create mode 100644 README_more.md
create mode 100644 fag.html
$ git log --pretty="%s"
FIRST COMMIT ON GITHUB git log --pretty=%s!
Initial commit
```

d. Bỏ tập tin ra khỏi Staging Area

- ▶ Nếu một tập tin đã được đưa vào Staging Area, nhưng khi commit ta lại không muốn commit tệp tin này => cần loại bỏ nó ra khỏi staging area
=> lệnh loại bỏ: *git reset HEAD <tên_file>*.
 - ▶ HEAD ~ Con trỏ trả về nhánh đang làm việc.

4. Git Log và Undo Commit

► Tóm lại:

- Xem lại lịch sử commit, phục hồi liên quan đến Commit là quan trọng trong giám sát những thay đổi mã nguồn.
 - SV cần thực hành thành thạo các lệnh này.
- Phần tiếp theo sẽ trình bày về quản lý phiên bản trong Git. Nhưng trước hết cần làm quen với tính năng Tagging (đánh dấu) của Git và đây là cách để chúng ta quản lý từng phiên bản trong mã nguồn (mỗi version có 1 tag đính kèm).



II. Hệ thống quản lý phiên bản phân tán Git

1. Cài đặt Git và thiết lập ban đầu
2. Cách tạo repository
3. Hiểu thêm về Commit và Staging Area
4. Git Log và Undo Commit
5. **Đánh dấu commit với Tag**
6. Sơ lược Remote Respository và Origin
7. Branch – Kỹ thuật phân nhánh

5. Đánh dấu commit với Tag

- ▶ Mỗi lần chỉnh sửa code là một lần commit
 - ▶ ∀ thay đổi stored trong history log & có thể xem lại
=> commit quá nhiều => gây khó khăn khi xem lại thông tin commit.
 - ▶ Gắn thẻ đánh dấu (tag) cho mỗi commit: thuận lợi:
 - ▶ Xem lại commit dễ dàng qua tag: **git show tên_tag**
 - ▶ Dễ dàng **diff (đối chiếu)** các thông tin commit mà không cần nhớ checksum của commit đó
 - ▶ Thêm branch từ tag.
 - ▶ Quản lý phiên bản mã nguồn (**mỗi version có 1 tag đính kèm**).



5. Đánh dấu commit với Tag

- a. **Lightweight Tag và Annotated Tag**
- b. **Push Tag**
- c. **Nhập tag vào branch**

a. Lightweight Tag & Annotated Tag

- ▶ Gắn tag cho commit ~ gắn nhãn phiên bản
- ▶ Trong Git có hai kiểu tag chính:
 - ▶ **Lightweight Tag:**
 - ▶ Đánh dấu snapshot của commit.
 - ▶ **Annotated Tag:**
 - ▶ Cho phép gắn nhãn tag & khi log sẽ hiện thị thêm thông tin về người tag, ngày tag,....

Không quản lý
các thông tin: ai
gắn tag, ngày gắn
tag, lời nhắc, ...

a. Lightweight Tag & Annotated Tag

► Cách tạo Lightweight Tag

► Để gắn tag cho lần commit sau cùng, sử dụng lệnh *git tag tên_tag* .

⇒ Gõ *git tag* để xem danh sách các tag có trong dự án

Ví dụ: Tạo tag “*v1.0*” & xem thông tin về tag:

```
$ git tag v1.0  
$ git tag  
v1.0
```

a. Lightweight Tag & Annotated Tag

► Cách tạo Lightweight Tag

► Xem thông tin của lần commit được gắn tag này, sử dụng lệnh *git show tên_tag*.

► Ví dụ:

```
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git tag
v1.0

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ git show v1.0
commit 3951e5c79acb3f82de2c79a814a5d0ed1feead2e (HEAD -> master, tag: v1.0)
Author: Pham Thuong <cptthuong@ictu.edu.vn>
Date:   Sun Sep 8 06:03:46 2019 +0700

    Commit on GitHub

diff --git a/test.html b/test.html
index e69de29..111c1fb 100644
--- a/test.html
+++ b/test.html
@@ -0,0 +1,5 @@
+<body>
+
+
+
+</body>
\ No newline at end of file

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld (master)
$ |
```

a. Lightweight Tag & Annotated Tag

► Cách tạo Annotated Tag

- Sử dụng lệnh ***git tag*** + tham số ***-a*** và tham số ***-m*** để thiết lập lời nhắn cho tag này.
- Ví dụ:
 - ***git tag -a v.1.0-an -m "Ra mat version dau tien"***
 - Xem hình dưới

```
$ git tag -a v1.0-an -m "Ra mat phien ban 1.0"

Administrator@HaiThangBep MINGW64 ~/Desktop/TestBuild1/HelloWorld/src/helloworld
/hoc_git_basic (master)
$ git show v1.0-an
tag v1.0-an
Tagger: Pham Thuong <ptthuong@ictu.edu.vn>
Date:   Sun Mar 31 15:26:13 2019 +0700

Ra mat phien ban 1.0

commit 3951428d494ac87d77556f5c91533c04ca7e274a (HEAD -> master, tag: v1.0-an,
ag: v1.0)
Author: Pham Thuong <ptthuong@ictu.edu.vn>
Date:   Sun Mar 31 14:17:31 2019 +0700

    FIRST COMMIT ON GITHUB git log --pretty=%s!

diff --git a/README_more.md b/README_more.md
new file mode 100644
index 000000..6662107
--- /dev/null
+++ b/README_more.md
@@ -0,0 +1 @@
+">#Huong dan Git co ban
diff --git a/fag.html b/fag.html
new file mode 100644
index 000000..e69de29
```

Annotated tag cung cấp thêm các thông tin này so với lightweight tag. Đây là kiểu tag ta nên sử dụng để có nhiều thông tin hơn



a. Lightweight Tag & Annotated Tag

- ▶ **Gắn tag cho các commit cũ**
 - ▶ Các lệnh trên chỉ tạo ra tag cho commit cuối cùng.
 - ⇒ ? Nếu có rất nhiều commit trước đó mà cần gắn tag thì làm ntn?

a. Lightweight Tag & Annotated Tag

► Gắn tag cho các commit cũ

► Cách làm:

B1. Xác định mã checksum/ một đoạn mã checksum của commit cần tạo tag

► Xem mã checksum của các lần commit trước đó => sử dụng **git log** với tham số **--pretty** nhận giá trị **oneline** để lọc log.

► Ví dụ:

Mã checksum

```
$ git log --pretty=oneline  
3951428d494ac87d77556f5c91533c04ca7e274a (HEAD -> master, tag: v1.0-an, tag: v1.  
0) FIRST COMMIT ON GITHUB git log --pretty=%s!  
5d3020be67aad777ae3a9a5db95f3effcc08f6aa Initial commit
```

a. Lightweight Tag & Annotated Tag

► Gắn tag cho các commit cũ

► **B2:** Tạo tag cho commit sử dụng mã checksum hoặc 1 phần mã checksum

► Ví dụ:

► Tạo tag cho **Initial commit** sử dụng lệnh:

git tag -a <ten tag> <mã checksum> -m "comment"

► Xem các tag đã tạo:

```
$ git tag -a v0.0 5d30 -m "Tag for initial commit"  
$ git tag  
v0.0  
v1.0  
v1.0-an
```

b. Push Tag

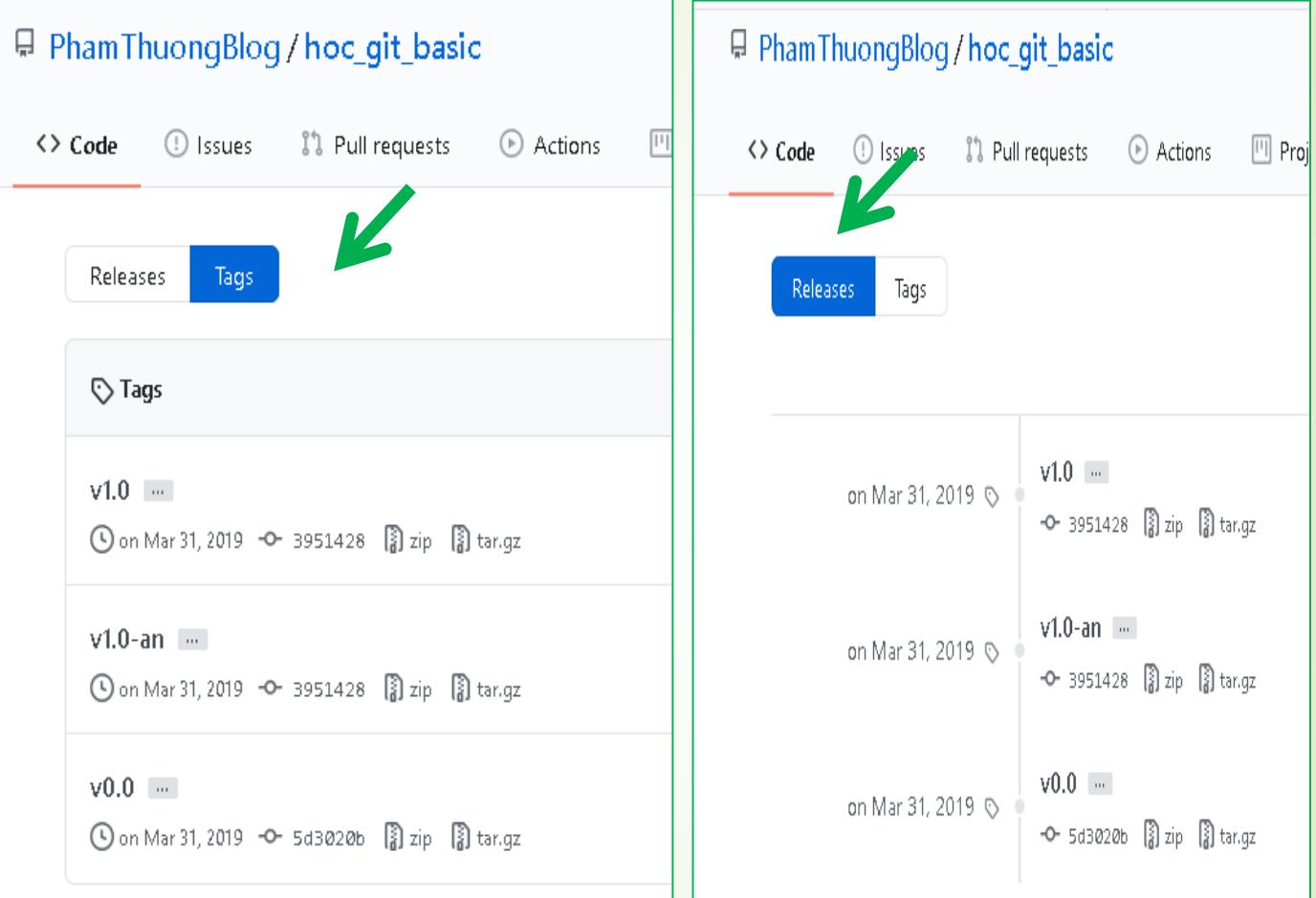
- ▶ Mặc định lệnh **git push** sẽ không push các tag đã tạo lên repository
 - ▶ Đẩy toàn bộ tag lên kho, dùng lệnh: **\$ git push --tags**, hoặc: **\$ git push <URL of remote Repository> --tags**

```
$ git push --tags
Username for 'https://github.com': PhamThuongBlog
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 659 bytes | 219.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/PhamThuongBlog/hoc_git_basic
 * [new tag]          v0.0 -> v0.0
 * [new tag]          v1.0 -> v1.0
 * [new tag]          v1.0-an -> v1.0-an
```



b. Push Tag

- ▶ Kết quả: tags xuất hiện ở remote repository.
 - ▶ Xem hình (dưới)



PhamThuongBlog / hoc_git_basic

Code Issues Pull requests Actions

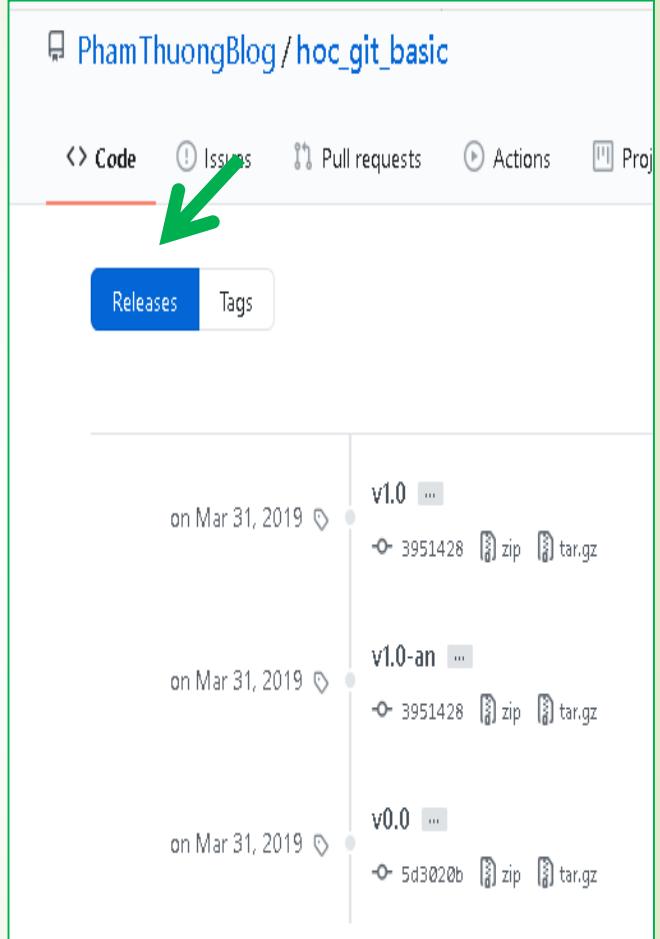
Releases Tags

Tags

v1.0 ...
on Mar 31, 2019 · 3951428 · zip · tar.gz

v1.0-an ...
on Mar 31, 2019 · 3951428 · zip · tar.gz

v0.0 ...
on Mar 31, 2019 · 5d3020b · zip · tar.gz



PhamThuongBlog / hoc_git_basic

Code Issues Pull requests Actions Proj

Releases Tags

v1.0 ...
on Mar 31, 2019 · 3951428 · zip · tar.gz

v1.0-an ...
on Mar 31, 2019 · 3951428 · zip · tar.gz

v0.0 ...
on Mar 31, 2019 · 5d3020b · zip · tar.gz

c. Nhập tag vào branch

► Branch

► ~ **một phân nhánh trong cây dự án**, nơi 1 developer có thể sửa mã nguồn mà không ảnh hưởng đến phân nhánh gốc (master) và các phân nhánh khác (developers khác).

=> Nhập tag vào branch để làm việc tại branch tương ứng.

c. Nhập tag vào branch

- ▶ Nhập tag vào branch
 - ▶ Truy cập vào dữ liệu đã commit thông qua tag kèm theo tạo một branch mới với lệnh:
git checkout -b <tên_branch> <tên_tag>.
 - ▶ Ví dụ:
 - ▶ Đưa **tag v1.0-an** vào 1 branch mới có tên **version1** hoặc **develop**:

```
$ git checkout -b version1 v1.0-an
Switched to a new branch 'version1'
D      fag.html
```

c. Nhập tag vào branch

- ▶ Kết quả:
 - ▶ Branch mới **version1** được tạo và tự động chuyển qua branch **này** thay vì **master** ban đầu, kèm theo đó là dữ liệu của lần commit được gắn tag **v1.0-an**.
=> Bây giờ, ta có thể làm việc với branch này như khi làm việc tại branch master.
 - ▶ => Để chuyển về lại master, gõ lệnh **git checkout master**.

c. Nhập tag vào branch

- ▶ Để push branch *version1* này lên repository, dùng lệnh

git push <remote URL> <branch name>

Ví dụ: ***git push origin/URL version1***

- ▶ ***Origin:*** tên địa chỉ remote repository mặc định mà Git tự đặt khi ta clone.
- ▶ Kết quả:
 - ▶ Xuất hiện thêm branch có tên ***version1*** trên kho Github.
 - ▶ màn hình (dưới)

The screenshot shows a GitHub repository page for 'PhamThuongBlog / hoc_git_basic'. The repository has no description, website, or topics provided. It contains 2 commits, 2 branches (master, version1), 3 releases (v1.0, v1.0-an, v0.0), and 2 contributors. A red arrow points to the 'version1' branch in the recently pushed branches list. Another red arrow points to the 'version1' entry in the dropdown menu under 'Switch branches/tags'. A green thought bubble in the top right corner states: '3 releases ~ 3 version (v1.0, v1.0-an, v0.0): với các tệp .zip, tar.gz đính kèm' (3 releases ~ 3 versions (v1.0, v1.0-an, v0.0): with zip, tar.gz files attached). A red box highlights the release section, showing three releases: v1.0 (commit d860e37, 2 hours ago), v1.0-an (commit 3951428, 2 hours ago), and v0.0 (commit 5d3020b, 2 hours ago). A red arrow points to the 'Compare & pull request' button.

No description, website, or topics provided.

Manage topics

2 commits 2 branches 3 releases 2 contributors

Your recently pushed branches:

version1 (1 minute ago)

Branch: master New pull request

Switch branches/tags

Find or create a branch...

Branches Tags

✓ master version1

v1.0 ...
3951428 zip tar.gz

v1.0-an ...
3951428 zip tar.gz

v0.0 ...
5d3020b zip tar.gz

Compare & pull request

Initial commit 2 hours ago

First commit on Github 2 hours ago

Latest commit d860e37 2 hours ago

Find File Clone or download ▾

hoc_git_basic

PhamThuongBlog/hoc_git_basic/tree/master

5. Đánh dấu commit với Tag

- ➡ Tóm lại:
 - ➡ Git + branch mã nguồn tạo sức mạnh cho việc quản lý mã nguồn và sửa đổi (bảo trì) sản phẩm khi làm việc nhóm/lập trình cộng tác.



II. Hệ thống quản lý phiên bản phân tán Git

1. Cài đặt Git và thiết lập ban đầu
2. Cách tạo repository
3. Hiểu thêm về Commit và Staging Area
4. Git Log và Undo Commit
5. Đánh dấu commit với Tag
6. **Remote Repository và Origin**
7. Branch – Kỹ thuật phân nhánh

6. Remote Respository & Origin

a. Kiểm tra tên remote repository:

\$ git remote -v

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

=> Mọi repository được clone đều có tên là **origin**,
và mỗi repository đều có 2 hành động là **fetch** (lấy
dữ liệu về từ server) và **push** (gửi dữ liệu lên
server)

6. Remote Respository & Origin

b. Đổi tên remote

- ▶ Nếu có nhiều remote rep trong dự án, để dễ quản lý, ta cần đổi tên *origin* bằng tên khác = lệnh:

git remote rename <tên cũ> <tên mới>.

- ▶ Ví dụ: đổi từ *origin* sang *Thuong* :

```
$ git remote rename origin Thuong  
  
Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/Helloworld/src/helloworld  
/hoc_git_basic (version1)  
$ git remote -v  
Thuong  https://github.com/PhamThuongBlog/hoc_git_basic (fetch)  
Thuong  https://github.com/PhamThuongBlog/hoc_git_basic (push)
```

- ▶ Khi commit hay push dữ liệu ta gõ ***git push Thuong master***

6. Remote Respository & Origin

c. Thêm một remote để lấy dữ liệu khi cần

- ▶ Sử dụng lệnh *git remote add <ten new remote rep> <tên_remote URL>*
- ▶ Ví dụ:
 - ▶ Thêm remote repository có tên là **inuit**
 - ▶ Liệt kê các remote của dự án

```
$ git remote add unuit https://github.com/csswizardry/inuit.css-web-template
$ git remote -v
Thuong https://github.com/PhamThuongBlog/hoc_git_basic (fetch)
Thuong https://github.com/PhamThuongBlog/hoc_git_basic (push)
inuit https://github.com/csswizardry/inuit.css-web-template (fetch)
inuit https://github.com/csswizardry/inuit.css-web-template (push)
```

6. Remote Respository & Origin

c. Thêm một remote

► Lấy dữ liệu **inuit** về, sử dụng lệnh **git fetch inuit**

```
$ git fetch inuit
warning: no common commits
remote: Counting objects: 94, done.
remote: Total 94 (delta 0), reused 0 (delta 0), pack
Unpacking objects: 100% (94/94), done.
From https://github.com/csswizardry/inuit.css-web-template
 * [new branch] master -> inuit/master
```

6. Remote Respository & Origin

c. Thêm một remote

► Lưu ý:

► Lệnh **git fetch** chỉ lấy dl về và lưu vào database của Git trên máy chứ không được gộp vào repository.

=> Để gộp dùng thêm lệnh **git merge inuit**

► Lệnh lấy dl và gộp vào branch hiện tại: **git pull
tên_remote**

Lưu ý:

Cẩn thận khi gộp, nên tạo thư mục mới chứa code lấy về.

6. Remote Respository & Origin

d. Sự khác nhau giữa clone, fetch và pull

- ▶ Là 3 lệnh dùng để lấy dữ liệu về từ repository
 - ⇒ Sự khác nhau giữa chúng là gì?

(1) git clone

- ▶ Sao chép toàn bộ dữ liệu trên repository và các thiết lập về repository (các nhánh đã thiết lập).
=> Chỉ nên sử dụng lệnh này khi ta cần tạo mới một kho Git trên máy tính với toàn bộ dữ liệu và thiết lập của remote repository.

6. Remote Respository & Origin

d. Sự khác nhau giữa clone, fetch và pull

(2) git pull

- ▶ Lấy code từ remote repository gộp vào branch hiện tại đang làm việc.

(3) git fetch

- ▶ Lấy code từ remote repository về local rep nhưng không merge vào code hiện tại. Cần merge thủ công.

6. Remote Respository & Origin

e. Các loại giao thức của Remote Repository

► HTTP Repository

- Giao thức được sử dụng thông dụng nhất khi truy cập các dịch vụ remote repository như Github hay Assembla
- Các định dạng gồm:
 - **http://**domain.com/repository.git hoặc
 - **https://**domain.com/repository.git.

6. Remote Respository & Origin

e. Các loại giao thức của Remote Repository

► Local Repository

- Dùng khi kết nối tới một repository trên máy tính => URL của giao thức sẽ có dạng **/path/repository/**.

► SSH Repository

- Connect tới một server repository (server riêng) sử dụng giao thức SSH.

=> Đường dẫn của giao thức này sẽ có dạng:

user@server:/path/repository.git



II. Hệ thống quản lý phiên bản phân tán Git

1. Cài đặt Git và thiết lập ban đầu
2. Cách tạo repository
3. Hiểu thêm về Commit và Staging Area
4. Git Log và Undo Commit
5. Đánh dấu commit với Tag
6. Sơ lược Remote Respository và Origin
7. **Branch – Kỹ thuật phân nhánh**



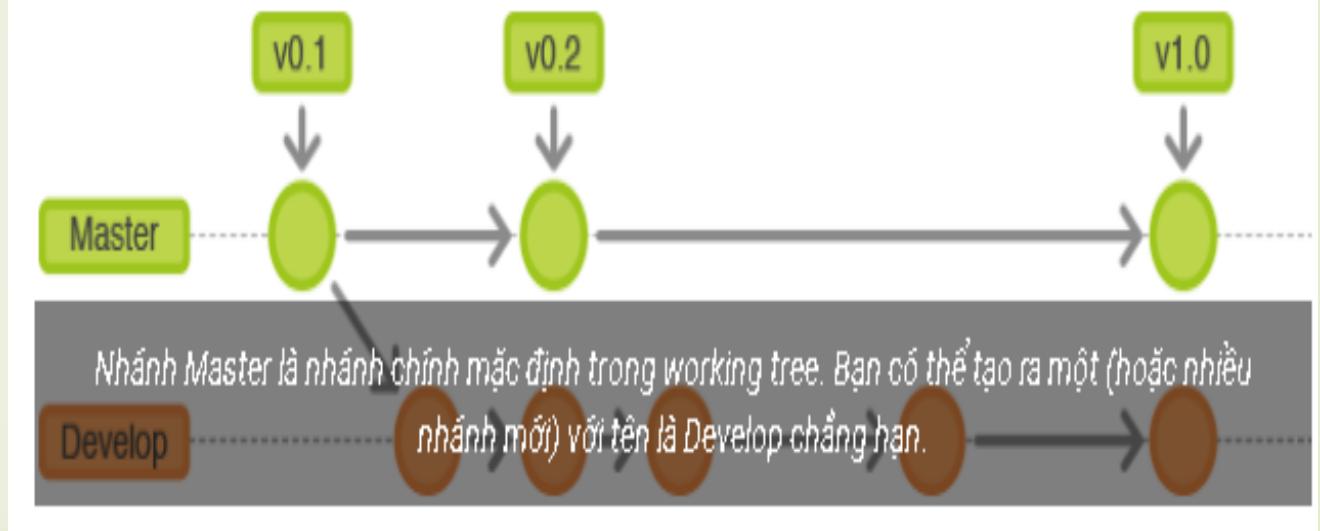
7. Branch – Kỹ thuật phân nhánh

- a. **Branch trong Git là gì?**
- b. **HEAD – con trỏ vị trí**
- c. **Cách tạo một branch**
- d. **Checkout một branch**
- e. **Gộp dữ liệu từ một branch**
- f. **Xóa branch**
- g. **Làm việc với remote branch**

a. Branch trong Git là gì?

- ▶ Khi bắt đầu khởi tạo một repository hoặc clone một repository, mặc định sẽ có một nhánh chính tên là *master*
 - ▶ master ~ thân cây: chứa toàn bộ mã nguồn chính (baselines/codebase) trong repository.
 - ▶ Ngoài branch master, ta có thể tạo ra một hoặc nhiều nhánh mới để làm việc với các variant

a. Branch trong Git là gì?



a. Branch trong Git là gì?

- ▶ Các nhánh làm việc độc lập với nhau
 - ▶ Ví dụ:
 - ▶ Bên branch master:
 - ▶ Tạo một tệp **master.html** rồi commit lên kho
=> master.html sẽ xuất hiện trên nhánh master của kho
 - ▶ Bên branch develop
 - ▶ Tệp **master.html** không có ở branch này vì master, không liên quan đến develop.
=> tương tự, thực hiện việc sửa đổi bên develop cũng không liên quan đến bên master.

a. Branch trong Git là gì?

- ▶ Các nhánh làm việc độc lập với nhau
 - ▶ Lưu ý: Mỗi khi checkout vào branch thì toàn bộ mã nguồn trên working tree sẽ được đổi sang môi trường dành cho branch đó
 - ▶ Ví dụ: thực hành trên: **DemoGit_GitHub_2020 project**
 - ▶ Bật sang các nhánh tương ứng, quan sát code in working place & run code => code khác nhau, kết quả chạy khác nhau!!!

b. HEAD – con trỏ vị trí

- ▶ Từ khóa HEAD:

- ▶ ~ con trỏ cho biết đang ở nhánh nào.
- ▶ Ví dụ: gõ lệnh

git log =>HEAD trỏ tới branch hiện thời trong working tree.

```
$ git log
commit 3951428d494ac87d77556f5c91533c04ca7e274a (HEAD -> version1, tag: v1.0-an,
tag: v1.0, Thuong/version1, master)
Author: Pham Thuong <pptthuong@ictu.edu.vn>
Date:   Sun Mar 31 14:17:31 2019 +0700
        ↑

        FIRST COMMIT ON GITHUB git log --pretty=%s!

commit 5d3020be67aad777ae3a9a5db95f3effcc08f6aa (tag: v0.0)
Author: PhamThuongBlog <49114029+PhamThuongBlog@users.noreply.github.com>
Date:   Sun Mar 31 13:55:48 2019 +0700

        Initial commit
```

b. HEAD – con trỏ vị trí

- ▶ Kiểm tra branch hiện tại (qua Head) bằng cách đọc tệp `.git/HEAD`
 - ▶ Ví dụ:

```
$ cat .git/HEAD  
ref: refs/heads/version1
```

c. Cách tạo branch

- ▶ Tạo branch: ***git branch <tên_branch>***

- ▶ Ví dụ: tạo branch ***develop***

\$ git branch develop

=> ∀ code of current branch is copied to new branch

- ▶ Xem toàn bộ các branch trong working tree:

git branch.

```
/hoc_git_basic (version1)
$ git branch develop

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld/src/helloworld
/hoc_git_basic (version1)
$ git branch
  develop
  master
* version1
```

d. Checkout một branch

- ▶ Checkout: bật branch và làm việc với branch đó:

git checkout <tên_branch>

```
$ git checkout develop  
Switched to branch 'develop'
```

=> Mọi update code chỉ diễn ra ở branch hiện tại

- ▶ Để chuyển lại về branch chính, gõ ***git checkout master.***

e. Gộp dữ liệu từ một branch

- ▶ Mỗi branch hoạt động một cách độc lập => đồng bộ các thay đổi giữa các branch liên quan ntn?
 - ⇒ Dùng lệnh *git merge* để chuyển dữ liệu từ branch cũ về branch hiện thời.
- Lưu ý: dl ở branch cần chuyển về phải đã được commit.

e. Gộp dữ liệu từ một branch

- Ví dụ: các bước:
 - ▶ Bật branch *master*:
 - ▶ \$ *git checkout master*
 - ▶ Gõ lệnh ***git merge develop*** để chuyển dữ liệu từ branch ***develop*** về branch ***master***;
 - ▶ Gõ ***ls*** để xem kq

Tệp
develop.html
của branch
develop đã
xuất hiện trong
master

```
$ git merge develop
Updating 3951428..3775114
Fast-forward
  develop.html | 0
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 develop.html

Administrator@HaiThangBeo MINGW64 ~/Desktop/TestBuild1/HelloWorld/src/helloworld
/hoc_git_basic (master)
$ ls
develop.html  README.md  README_more.md
```

f. Xóa branch

- ▶ Sử dụng lệnh ***git branch -d <tên_branch>***.
- ▶ Lưu ý: branch muốn xóa phải được gộp dữ liệu (merge) về nhánh chính trước khi xóa để tránh mất code.

```
$ git branch -d develop  
Deleted branch develop (was 8c68896).
```

Sau khi xóa xong, sẽ có thông báo branch develop đã được mốc vào commit với mã checksum 8c68896

g. Làm việc với remote branch

- ▶ Tại working tree, ta tạo thêm một remote có tên **inuit** từ địa chỉ <https://github.com/csswizardry/inuit.css-web-template>

```
$ git remote add unuit https://github.com/csswizardry/inuit.css-web-template
$ git remote -v
Thuong https://github.com/PhamThuongBlog/hoc_git_basic (fetch)
Thuong https://github.com/PhamThuongBlog/hoc_git_basic (push)
inuit https://github.com/csswizardry/inuit.css-web-template (fetch)
inuit https://github.com/csswizardry/inuit.css-web-template (push)
```

g. Làm việc với remote branch

- ▶ Lệnh xem các branch của remote ***inuit***:

git remote show inuit

- ▶ Lệnh xem các file từ branch:

ls

Main Content

- I. Giới thiệu Git & Github
- II. Hệ thống quản lý phiên bản phân tán Git
- III. Dịch vụ remote repository - GitHub**
- IV. Mô hình phân nhánh (tham khảo)



III. Dịch vụ repository - GitHub

- a. **Fetch, pull,**
- b. Fork and Branch with fork
- c. GitHub graph
- d. Add contributor
- e. Labels and milestones
- f. Create issue



a. Fetch, Pull

- ▶ Make change in remote repo → commit → pull or fetch this update code to local repo:
 - ▶ Type: `git pull/fetch <origin/remote repo URL> <branch>`
 - ▶ Note:
 - ▶ Fetch: copy code but not merge into previous code
 - ▶ Pull = Fetch + Merge



III. Dịch vụ repository - GitHub

- a. Fetch, pull
- b. Fork and Branch with Fork**
- c. GitHub graph
- d. Add contributor
- e. Labels and milestones
- f. Create issue

b. Fork and Branch with Fork

- ▶ Fork: copy a remote parent repo to our remote repo
 - ▶ Make a new remote repo
 - ▶ Explore → choose remote parent rep → Fork
- ▶ Branch with fork

git clone <URL of remote parent Repository>

git branch // to check branches of project

- ▶ Tạo thêm các branch và làm việc độc lập với remote parent repository



III. Dịch vụ repository - GitHub

- a. Fetch, pull
- b. Fork and Branch with Fork
- c. **GitHub graph**
- d. Add contributor
- e. Labels and milestones
- f. Create issue



c. GitHub graph

- ▶ Use to check the details and what happened in the remote repository
 - ▶ Click: Insights → type of graph need to check
 - ▶ See figure (below)



PhamThuongBlog/TestGit_GitHub_Year2020



Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

master ▾

3 branches

3 tags

Go to file

Add file ▾

Code ▾



ptthuongECJ the nine commit from master

616b39c 24 days ago 8 commits



TestGit_GitHub_Demo

the nine commit from master

24 days ago

Help people interested in this repository understand your project by adding a README.

Add a README

 ⏵

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

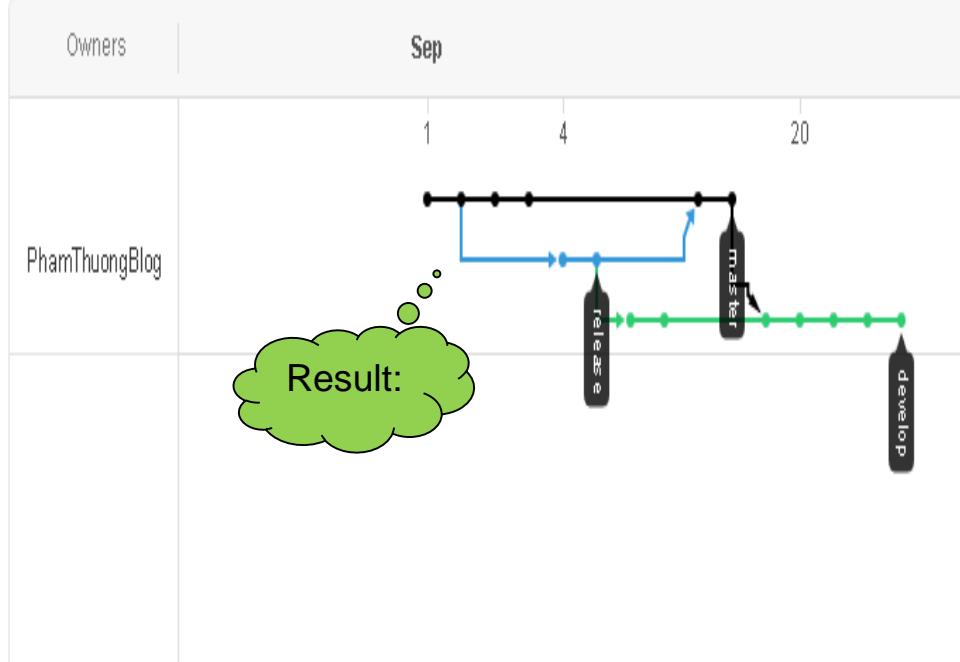
Pulse Check here to see results Contributors Community Traffic Commits Code frequency Dependency graph Network Forks

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

Owners Sep

PhamThuongBlog



Result:

Pulse

Contributors

Traffic

Commits

Code frequency

Dependency graph

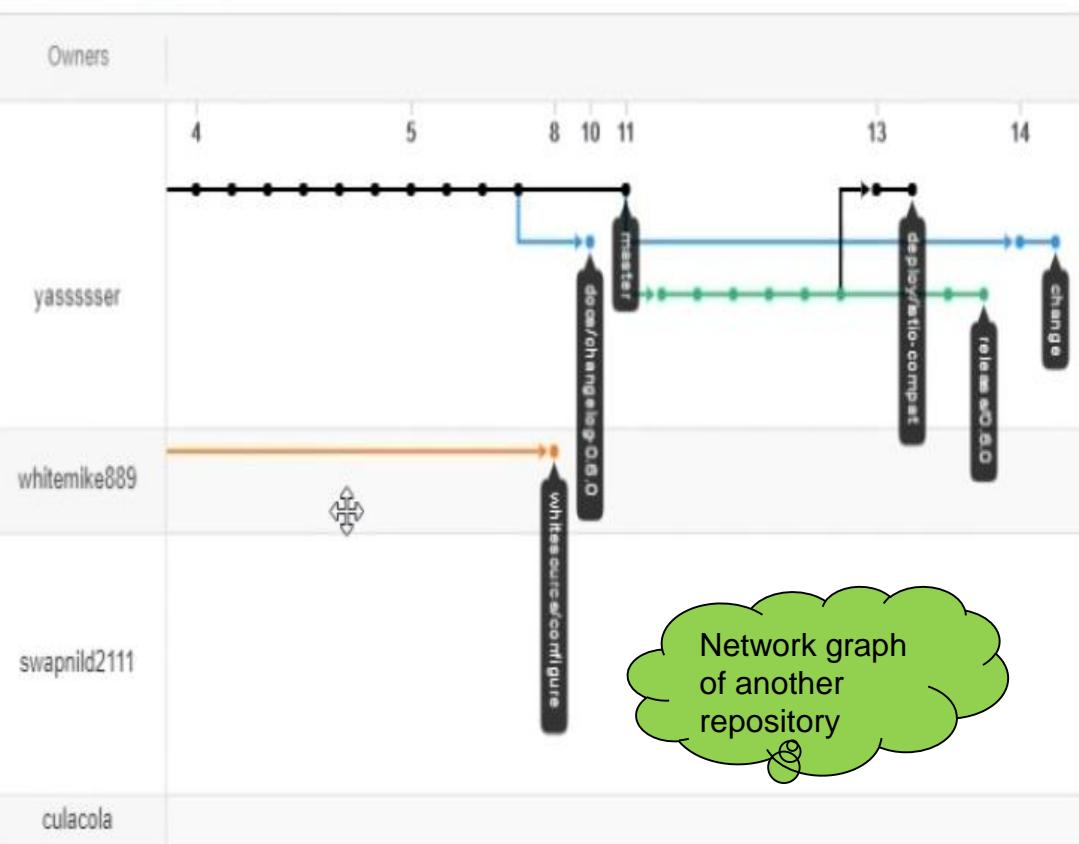
Network

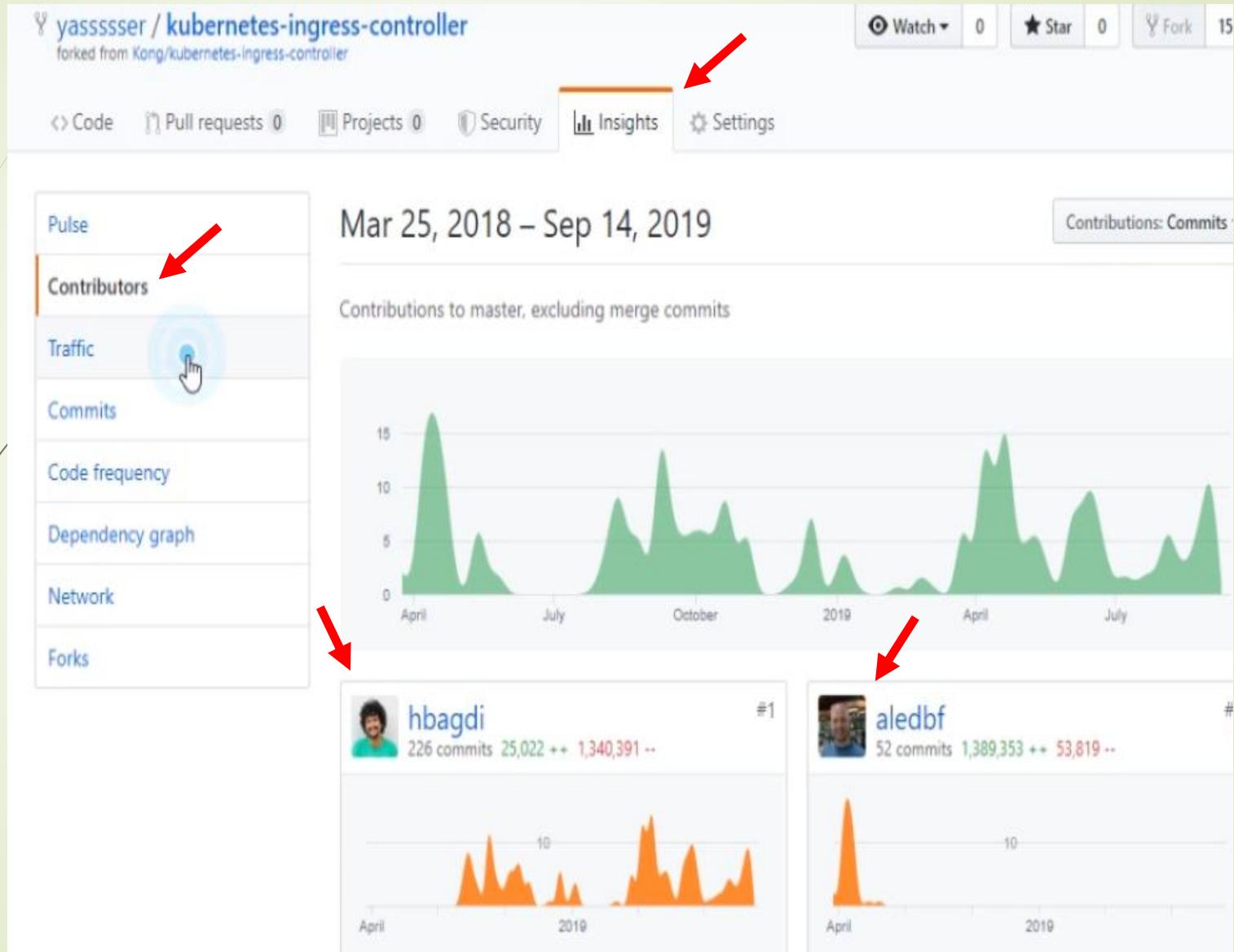
Forks

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

The repository network shows the 100 most recently pushed forks. Do you need to see more forks? Please [give us feedback](#) on your usage of this feature.





yassssser / kubernetes-ingress-controller

forked from Kong/kubernetes-ingress-controller

Watch ▾ 0

Star 0

Fork 153

Code

Pull requests 0

Projects 0

Security

Insights

Settings

Pulse

Contributors

Traffic

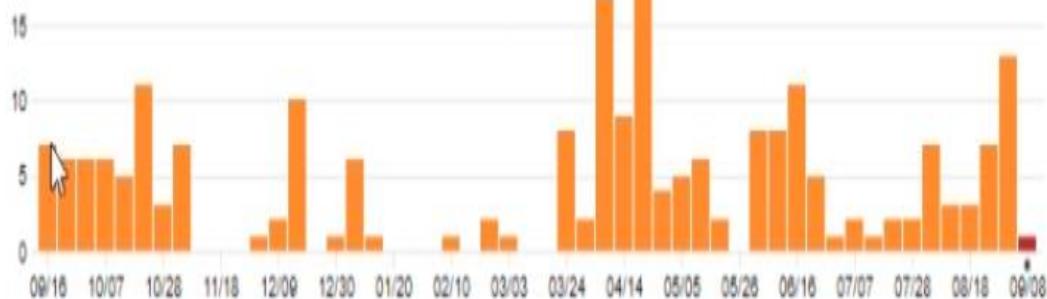
Commits

Code frequency

Dependency graph

Network

...





III. Dịch vụ repository - GitHub

- a. Fetch, pull
- b. Fork and Branch with Fork
- c. GitHub graph
- d. Add contributor**
- e. Labels and milestones
- f. Create issue



d. Add contributor

- ▶ Add contributor/developer to help code in our project
 - ▶ **Setting → contributors → input username, password → type username of contributor who we want to invite → click add collaborator**
=> GitHub will send him/her email to invite him/her to be a collaborator with us.
 - ▶ After that, contributor can accept to join or not this invitation!



III. Dịch vụ repository - GitHub

- a. Fetch, pull
- b. Fork and Branch with Fork
- c. GitHub graph
- d. Add contributor
- e. **Labels and milestones**
- f. Create issue

e. Labels and milestones

- ▶ Labels ~ type of issues:
 - ▶ **Setting** → **Issues** → **Labels** (show all type of project problems: see figure 1) → **New label** → **input: label name** (Ex., “update”), **description; and choose color** (see figure 2) → **create label** (result: figure 3)

**Fig 1:
Labels**

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Labels Milestones Search all labels New label

9 labels Sort ▾

Label	Description	Edit	Delete
bug	Something isn't working	Edit	Delete
documentation	Improvements or additions to documentation	Edit	Delete
duplicate	This issue or pull request already exists	Edit	Delete
enhancement	New feature or request	Edit	Delete
good first issue	Good for newcomers	Edit	Delete
help wanted	Extra attention is needed	Edit	Delete
invalid	This doesn't seem right	Edit	Delete
question	Further information is requested	Edit	Delete

The screenshot shows the GitHub Labels creation interface. At the top, there are navigation links: Code, Issues (0), Pull requests (0), Projects (0), Wiki, Security, Insights, and Settings. Below these are buttons for Labels, Milestones, and a search bar labeled "Search all labels". A large green callout bubble in the center says "Fig 2: Create a new label". Red arrows point from this callout to the "Label name" field ("update"), the "Description" field ("need an update"), the "Color" section (hex code "#9de57b"), and the "Create label" button. To the right of the form is a list of existing labels: bug, documentation, duplicate, enhancement, and good first issue, each with an edit and delete link. A red arrow also points from the top right towards the "New label" button.

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Labels Milestones Search all labels New label

update Label name Description Color Create label

① 0 ② 0 ③ 0 ④ 0 ⑤ 0 ⑥ 0 ⑦ 0 ⑧ 0 ⑨ 0

bug documentation duplicate enhancement good first issue

Something isn't working Improvements or additions to documentation This issue or pull request already exists New feature or request Good for newcomers

Sort ▾

Fig 2:
Create a
new label

 yassssser / my-website

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Labels Milestones Search all labels New label

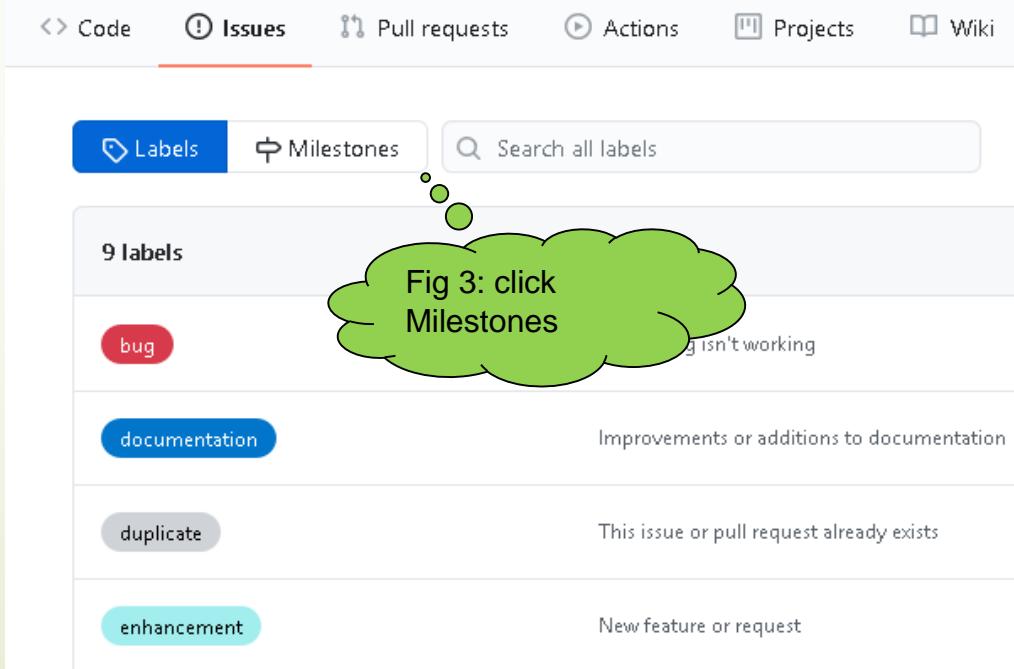
Fig 3: Result of creating a new label

Label	Description	Actions
update	need an update	Edit Delete
bug	Something isn't working	Edit Delete
documentation	Improvements or additions to documentation	Edit Delete
duplicate	This issue or pull request already exists	Edit Delete
enhancement	New feature or request	Edit Delete
good first issue	Good for newcomers	Edit Delete
help wanted	Extra attention is needed	Edit Delete

e. Labels and milestones

► Milestones

- Click **Milestones** (see Fig3) → **New Milestones** → Input information (see Fig4) → **Create Milestones** (result: see Fig5)



The screenshot shows the GitHub interface with the 'Issues' tab selected. A large red arrow points from the left margin towards the 'Issues' tab. On the right side, a green thought bubble contains the text: 'Fig4: New Milestones → Input information'. Red arrows point from the text in the thought bubble to the 'Title', 'Due date (optional)', and 'Description' fields in the milestone creation form. Another red arrow points to the 'Create milestone' button at the bottom right.

New milestone

Create a new milestone to help organize your issues and pull requests. Learn more about [milestones and issues](#).

Title

v1.0

Due date (optional)

09/28/2020

Description

Release version 1.0

Create milestone

PhamThuongBlog/TestGit_GitHub_Year2020

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Labels

Milestones

1 Open ✓ 0 Closed

v1.0

Due by September 28, 2020 Last updated less than a minute ago

Release version 1.0

Fig5:
Result

0% complete 0 open 0 closed

Edit Close Delete



III. Dịch vụ repository - GitHub

- a. Fetch, pull
- b. Fork and Branch with Fork
- c. GitHub graph
- d. Add contributor
- e. Labels and milestones
- f. **Create issue**



f. Create a issue

- ▶ Create a new issue belong a special label
 - ▶ Click Issues → New issue (see Fig6) → Input information & choose label for this issue (ví dụ: enhancement) & set milestone (ví dụ: v1.0) & assignees (gán cho ai, ví dụ: PhamThuongBlog) (see Fig7) → click: Submit new issue (result: see Fig8)

PhamThuongBlog/TestGit_GitHub_Year2020

Unwatch 1 Star 0 Fork

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

Filters isissue isopen Labels 9 Milestones 1 New issue

Fig6:
Issues
→ NEW
issue

Welcome to issues!

Issues are used to track todos, bugs, feature requests, and more. As issues are created, they'll appear here in a searchable and filterable list. To get started you should [create an issue](#).

PhamThuongBlog/TestGit_GitHub_Year2020

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Add a new feature: add user

Write Preview

We need add a new feature to our website: Add a new user

Attach files by dragging & dropping, selecting or pasting them.

Styling with Markdown is supported

Submit new issue

Fig7:
Input
information

Assignees
No one—assign yourself

Labels
enhancement

Projects
None yet

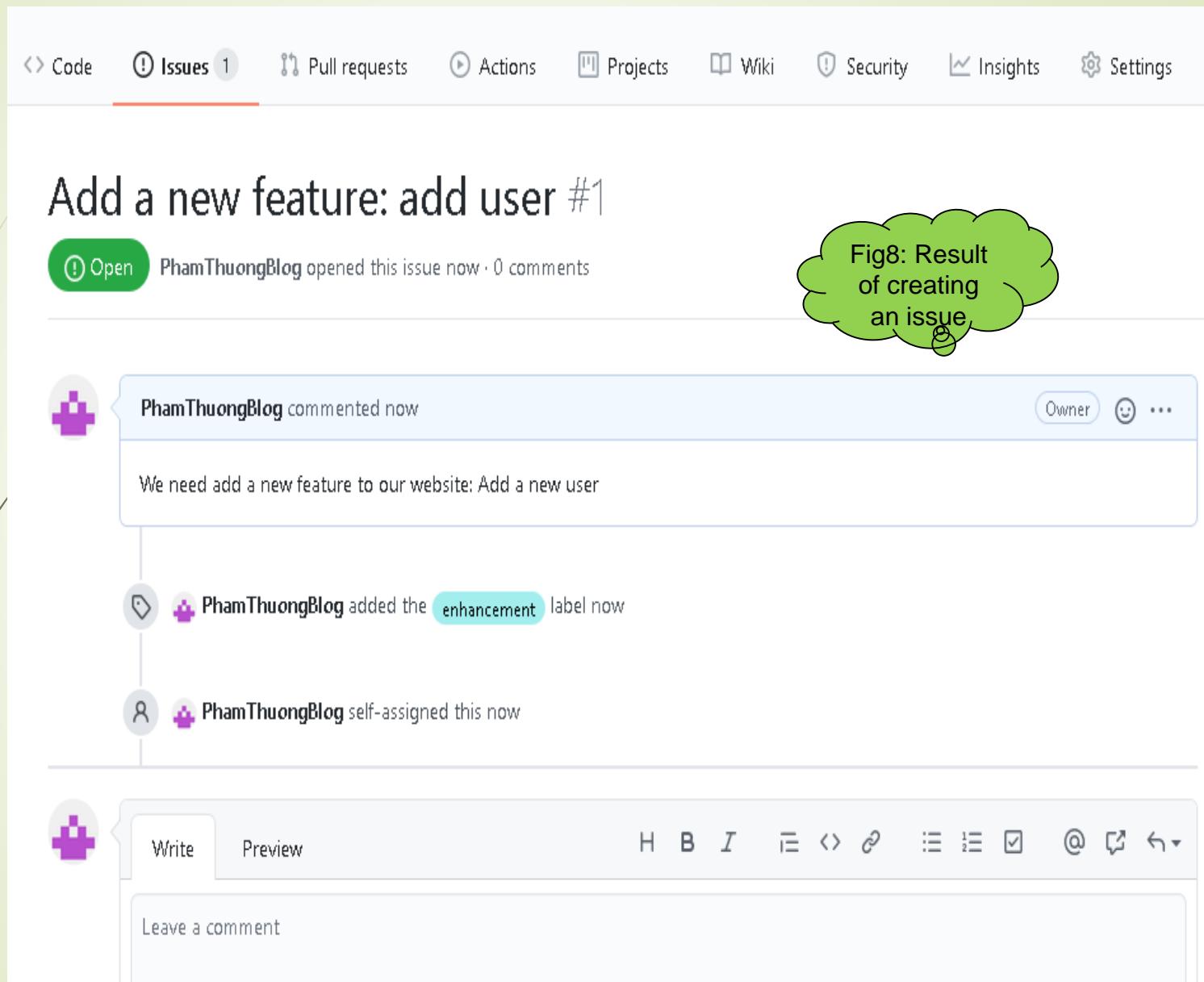
Milestone
Set milestone

Filter milestones

Open Closed

v1.0
Due by September 28, 2020

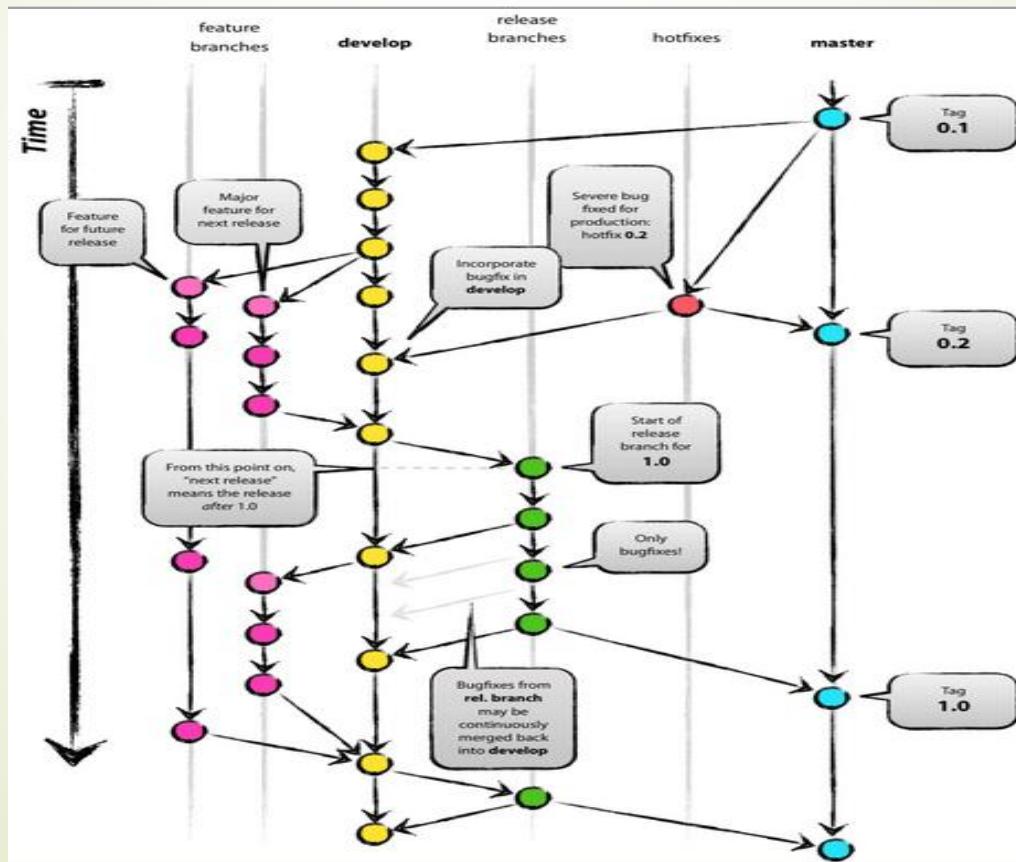
Helpful resources



Main Content

- I. Giới thiệu Git & Github
- II. Hệ thống quản lý phiên bản phân tán Git
- III. Dịch vụ repository (free) tốt nhất - GitHub
- IV. Mô hình phân nhánh (tham khảo)**

IV. A successful Git branching model



Vincent Driessens
(a software
engineer, at
Netherlands)



Demo dự án kết hợp: Git, GitHub & Jenkins

Tiến trình demo – kịch bản 1

1. Tạo dự án
2. Khởi tạo kho Git cục bộ cho dự án vừa tạo
2. Đưa mã nguồn lên GitHub
4. Tạo nhánh theo mô hình phân nhánh
5. Phân công công việc cho các thành viên/nhánh
6. Bảo trì sửa đổi mã/nhánh
7. Commit changes
8. Push lên kho remote
8. Theo dõi, xem lịch sử commit
9. Gắn nhãn version
10. Tạo dự án Jenkins gắn với code
11. Build code and check results
12. Lặp lại từ bước 6 để bắt đầu cho update code tiếp theo.

Demo: kịch bản 1

► Bước 1-9:

- Tạo dự án Java với tên: **TestGit_GitHub_Demo** tại path:
E:\Namhoc_2020\Hockyl_2020_2021\2_Baotri_PM__4.Demo_Projects\1_TestGit_GiiHub__Project.Repositories
- Khởi tạo kho Git cho dự án tại path trên:
 - Right click on project → Team → share project → chọn path of repository → click Create Repository → Finish
 - Right click on project → Team → Add to index to work with repository
- ⇒ Kết quả: địa chỉ kho:
E:\Namhoc_2020\Hockyl_2020_2021\2_Baotri_PM__4.Demo_Projects\1_TestGit_GiiHub__Project.Repositories\.git
- Tạo các nhánh trên kho gồm: **develop, master, release, ...**
=> Xem hình (dưới)

Project Explorer X

- > 1_TestGit_GitHub
- > java-project
 - java-web-project
- > Servers
- > StringCheckPlugin
- > TestGit_GitHub_Demo [_Project_Repository]
 - > src/main/java
 - ictu.testGit_GitHub.TestGit_GitH
 - > App.java
 - > src/test/java
 - > JRE System Library [J2SE-1.5]
 - > Maven Dependencies
 - > src
 - > target
 - pom.xml
- > TestGUI [TestGUI NO-HEAD]
 - > src/main/java
 - > src/test/java (1)
 - > JRE System Library [J2SE-1.5]
 - > Maven Dependencies
 - > JUnit 4
 - > package_variousFormats_Thuong
 - > src
 - > target
 - nom vml

App.java X

```

3 public class App {
4     private int x;
5     private int y;
6     public int z;
7
8
9     public static void main(String[] args) {

```

Outline X

- ictu.testGit_GitH
- App
- x: int
- y: int
- z: int

Markers Properties Servers Data Source Explorer Snippets Console Coverage JUnit History X

Project: TestGit_GitHub_Demo [_Project.Repositories]

ID	Message	Author	Authored Date	Committer
3d0c2e0	[develop] HEAD this is the next commit: make change in the App class	Pham Thuong	47 minutes ago	Pham Thuong
3dc980c	This is the next commit: edit the Hello class	Pham Thuong	55 minutes ago	Pham Thuong
f386da2	[origin_2020_Git...] the eleventh commit	Pham Thuong	2 weeks ago	Pham Thuong
616b39c	[master] the nine commit from master	Pham Thuong	2 weeks ago	Pham Thuong
bd0867c	this is the eight commit from master branch	Pham Thuong	2 weeks ago	Pham Thuong
c3f1757	This is the seven commit with the change: Add the more nhan(x,y,z) method	Pham Thuong	2 weeks ago	Pham Thuong
3ed638d	this is the six commit with change: Add more the tru(x,y,z) method	Pham Thuong	2 weeks ago	Pham Thuong
2bb5e65	[v1.2-an] [release] [origin_2020_Git...] Modify the cong(x,y,z) method	Pham Thuong	2 weeks ago	Pham Thuong
3fb8bde	Add more two attributes of the class and getter, setter methods corresponding!	Pham Thuong	2 weeks ago	Pham Thuong
4ff986b	[v1.0-an] The third commit from the feature branch.	Pham Thuong	3 weeks ago	Pham Thuong
12aea27	The forth commit from feature branch	Pham Thuong	3 weeks ago	Pham Thuong
d207ea7	The second commit	Pham Thuong	3 weeks ago	Pham Thuong
ddb5bc1	The first commit	Pham Thuong	3 weeks ago	Pham Thuong

Demo: kịch bản 1

► **Bước 1-9 (t.t):**

► Push kho Git của dự án lên kho GitHub

► Tạo kho GitHub (rỗng, public) với tên:
TestGit_GitHub_Year2020

► Push kho Git của dự án lên kho từ xa này

► Kết quả: kho GitHub với địa chỉ URL:

https://github.com/PhamThuongBlog/TestGit_GitHub_Year2020.git

=> Xem hình (dưới)

PhamThuongBlog/TestGit_GitHub_Year2020

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

develop had recent pushes less than a minute ago

Compare & pull request

develop ▾

3 branches

3 tags

Go to file

Add file ▾

Code ▾

This branch is 5 commits ahead of master.



ptthuongECJ this is the next commit: make change in the App class

TestGit_GitHub_Demo

this is the next commit: make change in

Help people interested in this repository understand your project by adding a README.

Clone

HTTPS SSH GitHub CLI

https://github.com/PhamThuongBlog/TestGit_GitHub_Year2020

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

Demo: kịch bản 1

- ▶ **Bước 10-11:** Tạo dự án Jenkins gắn liền với kho code trên GitHub (đã tạo ở trên)
 - ▶ Tạo một dự án (new item) tại trang chủ của Jenkins
 - ▶ Ví dụ: **DemoGit_GitHub_Year2020**, chọn kiểu dữ án **free style** -> OK.
 - ▶ Xem hình (dưới)



Jenkins

search

Jenkins > All

Enter an item name

DemoGit_GitHub_Year2020

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with anything else you want, for something other than software build.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration required.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines and/or organizing complex activities that do not easily fit in free-style job type.

External Job

OK

of job allows you to record the execution of a process run outside Jenkins, even on a remote host. You can use Jenkins as a dashboard of your existing automation system.

Demo: kịch bản 1

► Bước 10-11 (t.t)

- Cấu hình cho dự án **DemoGit_GitHub_Year2020** như sau:
 - Nhập thông tin tại các tab tương ứng
 - *Tại tab: General → Nhập thông tin mô tả dự án*
 - Xem hình (dưới)



Jenkins

search ? 📡

Jenkins › DemoGit_GitHub_Year2020 ›

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Description This is the first project that combine Git&Jenkins

[Plain text] [Preview](#)

Demo: kịch bản 1

► **Bước 10-11 (t.t)**

- Cấu hình cho dự án **DemoGit_GitHub_Year2020** như sau:
 - Nhập thông tin tại các tab tương ứng
 - Tại tab Source Code Management: Chọn tool: Git; Nhập vào URL của kho Git dự án & chọn branch to build.
 - Xem hình (dưới)

DemoGit_GitHub_Year2020

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Source Code Management

None Git

Repositories

Repository URL: https://github.com/PhamThuongBlog/TestGit_GitHub_Year2020.git

Credentials: - none - [Add](#)

Advanced... [Add Repository](#)

Branches to build

Branch Specifier (blank for 'any'): */master

[Add Branch](#)

The screenshot shows the Jenkins configuration interface for a job named "DemoGit_GitHub_Year2020". The "Source Code Management" tab is selected, highlighted by a large red arrow. Within this tab, the "Git" option is chosen, indicated by another red arrow. The "Repository URL" field contains the value "https://github.com/PhamThuongBlog/TestGit_GitHub_Year2020.git", with a red arrow pointing to it. Below the repository URL, the "Branch Specifier" field is set to "*master", also with a red arrow pointing to it. Other tabs like "General", "Build Triggers", "Build Environment", "Build", and "Post-build Actions" are visible at the top. On the right side, there are "Advanced..." and "Add Repository" buttons, and at the bottom, there is an "Add Branch" button.

Demo: kịch bản 1

► **Bước 10-11 (t.t)**

- Cấu hình cho dự án **DemoGit_GitHub_Year2020** như sau:
 - Nhập thông tin tại các tab tương ứng
 - Tại tab Build triggers: chọn chế độ build (thủ công, hoặc tự động theo lịch biểu), ví dụ: build mỗi phút:
 - Xem hình dưới

DemoGit_GitHub_Year2020

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- Gerrit event
- GitHub hook trigger for GITScm polling
- Poll SCM

Schedule

⚠ Do you really mean "every minute" when you say "*****"? Perhaps you meant "H *****" to poll once per hour

Would last have run at Sunday, September 20, 2020 5:08:18 PM ICT; would next run at Sunday, September 20, 2020 5:08:18 PM ICT.

Build triggers

► Examples (POM SCM):

- Start build daily at 08:30 in the morning, Monday - Friday:

30 08 * * 1-5

- Weekday daily build twice a day, at lunchtime 12:00 and midnight 00:00, Sunday to Thursday:

00 0,12 * * 0-4

- Start build daily in the late afternoon between 4:00 p.m. - 4:59 p.m. or 16:00 - 16:59 depending on the projects hash:

H 16 * * 1-5

- Start build at midnight: @midnight or start build at midnight, every Saturday:

59 23 * * 6

- Every first of every month between 2:00 a.m. - 02:30 a.m.: **H(0,30) 02 01 * ***

=> Build mỗi khi push code changes to remote repository: see link:

<https://www.edureka.co/community/49753/auto-build-job-jenkins-when-change-code-github-repository>

Demo: kịch bản 1

► **Bước 10-11 (t.t)**

- Cấu hình cho dự án **DemoGit_GitHub_Year2020** như sau:
 - Nhập thông tin tại các tab tương ứng
 - Tại tab Build Environment: Check và tùy chọn: Delete Workspace before build:

=> Mục đích: đảm bảo thư mục workspace luôn rỗng (không chứa file, thư mục cũ) nhằm tránh xung đột với dữ liệu Git clone từ Git Repository về

- Xem hình (dưới)

DemoGit_GitHub_Year2020

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Build Environment

Delete workspace before build starts

Advanced...

Use secret text(s) or file(s)

Provide Configuration files

Abort the build if it's stuck

Add timestamps to the Console Output

Define Upstream Maven Repository

Inspect build log for published Gradle build scans

With Ant

Build

Add build step ▾

Save

Apply

Demo: kịch bản 1

► **Bước 10-11 (t.t)**

- Cấu hình cho dự án **DemoGit_GitHub_Year2020** như sau:
 - Nhập thông tin tại các tab tương ứng
 - Tại tab Build: chọn Execute Window Batch Command
 - Ví dụ: viết kịch bản build như hình (dưới)

DemoGit_GitHub_Year2020

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Build

Execute Windows batch command

Command

```
git tag  
cd TestGit_GitHub_Demo  
java -jar TestGit_GitHub_Demo-1.0-SNAPSHOT.jar
```

X

See [the list of available environment variables](#)

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save Apply



Click save để lưu lại mọi thiết lập cấu hình

Demo: kịch bản 1

► Lưu ý:

- SV cần nắm vững các câu lệnh (ví dụ: biên dịch, kiểm thử, đóng gói, ,deploy, or run project) của Git, JDK, Maven, Nexus, ... nhằm thiết lập cấu hình cho dự án Jenkins sau khi đã plugin các tool này vào Jenkins.

Demo: kịch bản 1

- ▶ Finish thiết lập cấu hình cho dự án
 - ▶ Tiếp theo, ta có thể làm việc trực tiếp với dự án trên Jenkins
 - ▶ Ví dụ:
 - ▶ Click **Build now** để build dự án sử dụng kịch bản đã thiết lập tại tab **Build** trong cấu hình
 - ▶ Xem hình (dưới)

Jenkins > DemoGit_GitHub_Year2020

Back to Dashboard

Status

Changes

Workspace

Build Now

Configure

Delete Project

Git Polling Log

Rename

Build History trend ^

find X

#3 Sep 20, 2020 9:37 PM

Project DemoGit_GitHub_Year2020

This is the first project that combine Git & Jenkins.

Workspace

Recent Changes

Permalinks

- Last build (#1), 4 min 57 sec ago
- Last stable build (#1), 4 min 57 sec ago
- Last successful build (#1), 4 min 57 sec ago
- Last completed build (#1), 4 min 57 sec ago

Demo: kịch bản 1

- ▶ Kết quả:
 - ▶ Tại Console Output:
 - ▶ Xem hình (dưới)

| > #3

```
Fetching upstream changes from https://github.com/PhamThuongBlog/TestGit_GitHub_Year2020.git
> git.exe fetch --tags --force --progress -- https://github.com/PhamThuongBlog/TestGit_GitHub_Year2020.git +refs/heads
/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/develop^(commit)" # timeout=10
> git.exe rev-parse "refs/remotes/origin/origin/develop^(commit)" # timeout=10
Checking out Revision 62c4625f672084a16ec8f23afda673277ddaca50 (refs/remotes/origin/develop)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 62c4625f672084a16ec8f23afda673277ddaca50 # timeout=10
Commit message: "Add files via upload"
> git.exe rev-list --no-walk 62c4625f672084a16ec8f23afda673277ddaca50 # timeout=10
[DemoGit_GitHub_Year2020] $ cmd /c call C:\Users\ADMINI~1\AppData\Local\Temp\jenkins5001884174396935693.bat
C:\Users\Administrator\.jenkins\workspace\DemoGit_GitHub_Year2020>git tag
release-1.1.0
v1.0-an
v1.2-an
C:\Users\Administrator\.jenkins\workspace\DemoGit_GitHub_Year2020>cd TestGit_GitHub_Demo
C:\Users\Administrator\.jenkins\workspace\DemoGit_GitHub_Year2020\TestGit_GitHub_Demo>java -jar TestGit_GitHub-Demo-1.0-
SNAPSHOT.jar
Welcome to this class!!!
C:\Users\Administrator\.jenkins\workspace\DemoGit_GitHub_Year2020\TestGit_GitHub_Demo>exit 0
Finished: SUCCESS
```

Demo: kịch bản 1

- ▶ Thực hiện các update code trên kho Git và GitHub và lặp lại click **Build now** để xem kết quả chạy
 - ▶ SV cần hiểu rõ mối quan hệ giữa:
 - ▶ Kho local Git + Kho remote GitHub & kết nối kho này với dự án Jenkins và làm việc với dự án jenkins.

Demo: Lịch bản 2

- ▶ SV thực hành trên dự án Maven với mã nguồn mở có sẵn:
 - ▶ Link: <https://mkyong.com/maven/how-to-create-a-java-project-with-maven/>
 - ▶ Download Source Code:
 - ▶ \$ git clone <https://github.com/mkyong/maven-examples.git>
 - ▶ \$ cd java-project
 - ▶ \$ mvn package
 - ▶ \$ java -jar target/java-project-1.0-SNAPSHOT.jar 123456

Demo: Lịch bản 3

- ▶ SV tự tạo dự án Maven sử dụng lệnh của maven theo hướng dẫn trong [this file](#)
 - ▶ Tên dự án: **Java-project**
 - ▶ Path:
E:\Namhoc_2020\Hockyl_2020_2021\2_Baotri_PM__4.Demo_Projects\5.TestMaven\java-project
 - ▶ Thực hành như kịch bản demo 1 sử dụng các lệnh maven khi thiết lập cấu hình tại tab Build

Discussion



Tài liệu tham khảo

- ▶ https://docs.google.com/presentation/d/1IQCR_PHEIX-qKo7QFxsD3V62yhyGA9_5YsYXFOiBpgkk/present?ueb=true#slide=id.g4d6b1121f4_2_60
- ▶ <https://www.vogella.com/tutorials/EclipseGit/article.html> (need read carefully!!!)
- ▶ <https://www.edureka.co/blog/git-tutorial/>(need read carefully!!!)
- ▶ <https://mkyong.com/maven/how-to-create-a-java-project-with-maven/>

Next Topic

► **Topic 5: Build management - Maven**