



# GIT & GITHUB

ThS. Dương Hữu Thành  
Khoa CNTT, Đại học Mở, Tp.HCM  
thanh.dh@ou.edu.vn



# Nội dung chính

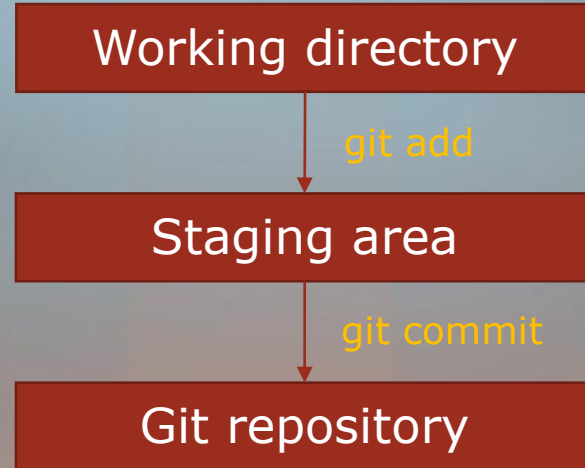
- Hệ thống quản lý phiên bản
- Giới thiệu Git & GitHub
- Các lệnh căn bản trong Git
- Tổng kết: Case Study

# Hệ thống quản lý phiên bản

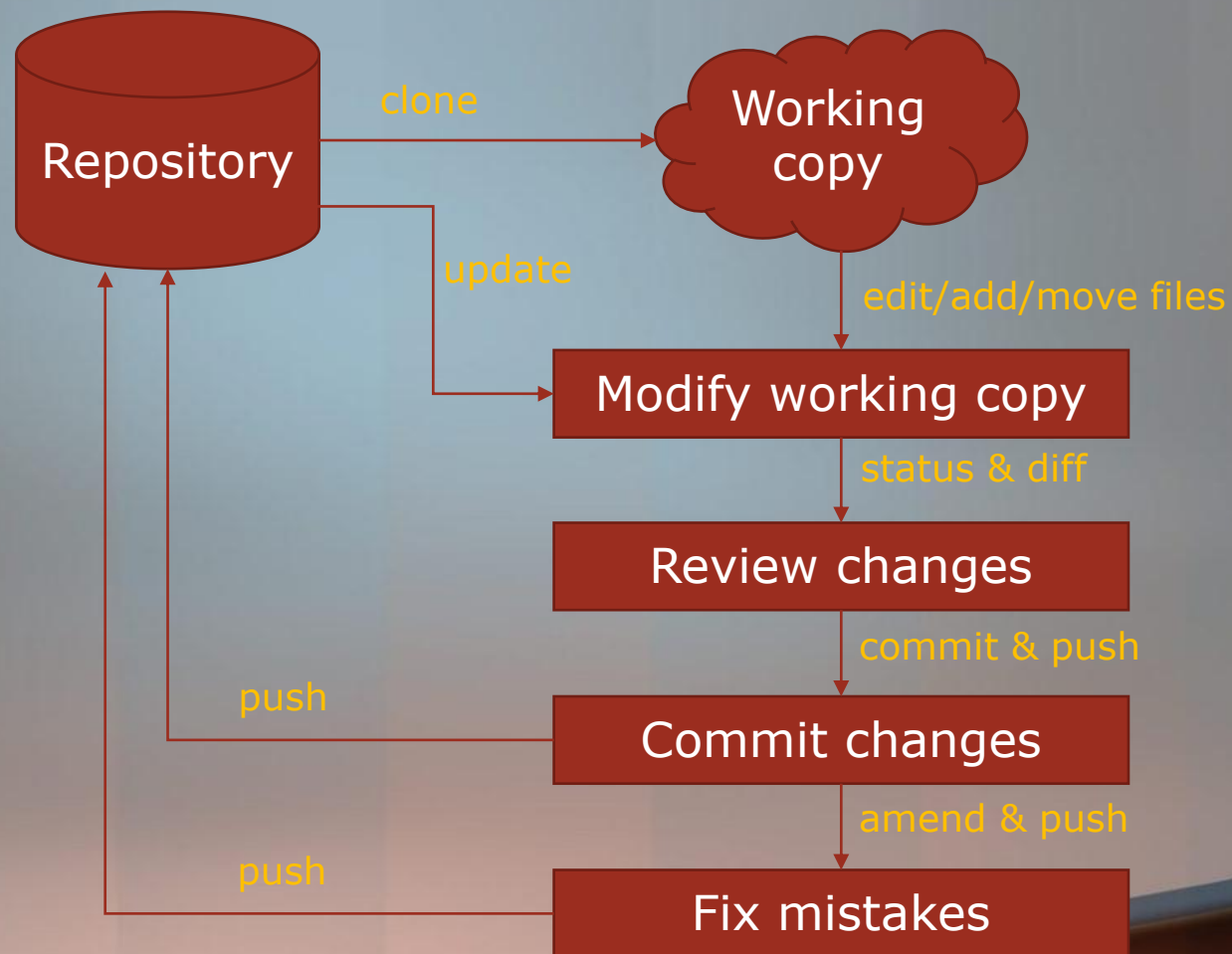
- Hệ thống quản lý phiên bản (VCS: Version Control System) là phần mềm cho phép các lập trình viên có thể làm việc đồng thời với nhau và lưu lại lịch sử của mỗi phiên bản.
- Có hai loại VCS:
  - Hệ thống quản lý phiên bản tập trung (CVCS: Centered VCS)
  - Hệ thống quản lý phiên bản phân tán (DVCS: Distributed VCS)

# Giới thiệu Git

- Một trong những công cụ VCS phổ biến nhất hiện nay là Git – một hệ thống quản lý phiên bản phân tán.
- Quy trình cơ bản làm việc git như sau:

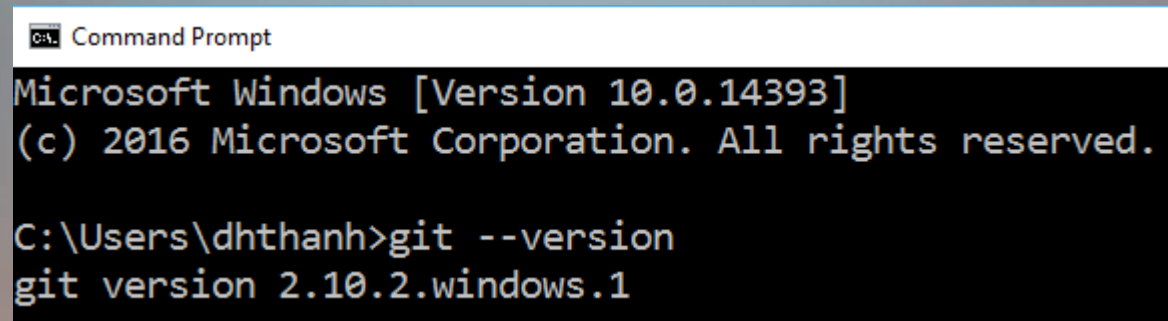


# Vòng đời làm việc với Git



# Cài đặt Git trên Window

- Tải Git tại <https://git-for-windows.github.io/>
- Sau khi tải, double-click vào file cài đặt và thực hiện theo hướng dẫn.
- Để kiểm tra cài đặt thành công, mở command line gõ lệnh "git --version" kết quả sẽ hiển thị giống như sau:



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\dhthanh>git --version
git version 2.10.2.windows.1
```

# Cài đặt Git trên Window

- Mở command line và gõ các lệnh sau để cấu hình username và email cho Git

```
git config --global user.name '<name>'
git config --global user.email '<email>'
```

```
C:\Users\dhthanh>git config --global user.name "thanh duong"
C:\Users\dhthanh>git config --global user.email "thanh.dh@ou.edu.vn"
```



# Github

- Github là một nền tảng cung cấp kho lưu trữ mã nguồn git. Nó cho phép nhiều người làm việc trên một project ở bất cứ nơi đâu.



# Đăng ký tài khoản GitHub

- Truy cập GitHub tại <https://GitHub.com/>
- Click button "Sign up" ở góc trên bên phải.
- Đến trang "Join GitHub" điền đầy đủ thông tin như username, email và password rồi click "Create an account" để tạo tài khoản.
- Sau khi tạo tài khoản, GitHub sẽ gửi một email để xác nhận đến email đã đăng ký với github
- Mở email và click vào link "Verify email address" để xác nhận.

# Join GitHub

The best way to design, build, and ship software.

1



Step 1:  
Set up a personal account



Step 2:  
Choose your plan



Step 3:  
Tailor your experience

## Create your personal account

Username

thanhduong-ou



This will be your username — you can enter your organization's username next.

Email Address

thanh.dh@ou.edu.vn



You will occasionally receive account related emails. We promise not to share your email with anyone.

Password

.....

Use at least one lowercase letter, one numeral, ar

By clicking on "Create an account" below, of [Service](#) and the [Privacy Policy](#).

Create an account

## You'll love GitHub

Unlimited collaborators

Unlimited public repositories

- ✓ Great communication
- ✓ Frictionless development
- ✓ Open source community

# Welcome to GitHub

You've taken your first step into a larger world, @thanhduong-ou.

2



Completed  
Set up a personal account



Step 2:  
Choose your plan



Step 3:  
Tailor your experience

## Choose your personal plan

- ☒ Unlimited public repositories for free.
- ☐ Unlimited private repositories for \$7/month. ([view in VND](#))

Don't worry, you can cancel or upgrade at any time.

## Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

[GitHub] Please verify your email address.

Hộp thư đến x



GitHub <noreply@github.com>

tới tôi

22:44 (2 phút trước)



Tiếng Anh



Tiếng Việt

Dịch thư

Tắt đổi với: Tiếng Anh x

3

Hi @thanhduong-ou!

Help us secure your GitHub account by verifying your email address ([thanh.dh@ou.edu.vn](mailto:thanh.dh@ou.edu.vn)). This lets you access all of GitHub's features.

[Verify email address](#)

Button not working? Paste the following link into your browser:

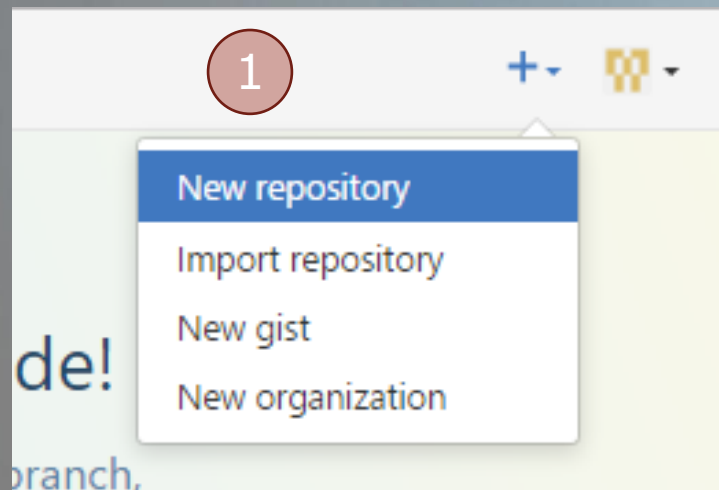
[https://github.com/users/thanhduong-ou/emails/26729753/confirm\\_verification/f7cb5fc46f92bbd1cb7c95599dd7dbc2ab8bd3c5](https://github.com/users/thanhduong-ou/emails/26729753/confirm_verification/f7cb5fc46f92bbd1cb7c95599dd7dbc2ab8bd3c5)

You're receiving this email because you recently created a new GitHub account or added a new email address. If this wasn't you, please ignore this email.

# Tạo thùng chứa (Repository)

- Thùng chứa là nơi tổ chức project, nó có thể chứa thư mục, tập tin, ảnh, video, ...
- Một project trên GitHub thông thường bao gồm tập tin README chứa mô tả về project
- Ví dụ: tạo thùng chứa tên là tutorial
  - Để tạo thùng chứa, ở góc trên bên phải click biểu tượng "+" chọn "New Repository"
  - Sau đó điền tên thùng chứa, check chọn "Initilize this repository with a README" và click "Create Repository"

# Tạo thùng chứa (Repository)




## Create a new repository


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

Owner

Repository name


 thanhduong-ou

 / 


tutorial 

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

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





# Tạo thùng chứa (Repository)

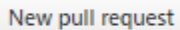
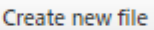
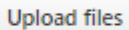
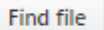
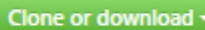
 thanhduong-ou / tutorial


 Watch 0  Star 0  Fork 0


 Code  Issues 0  Pull requests 0  Projects 0  Wiki  Pulse  Graphs  Settings


No description or website provided. — Edit

 1 commit  1 branch  0 releases  1 contributor

Branch: master     

 thanhduong-ou Initial commit Latest commit 5f25b9f 17 seconds ago

 README.md Initial commit 17 seconds ago

 README.md

## tutorial

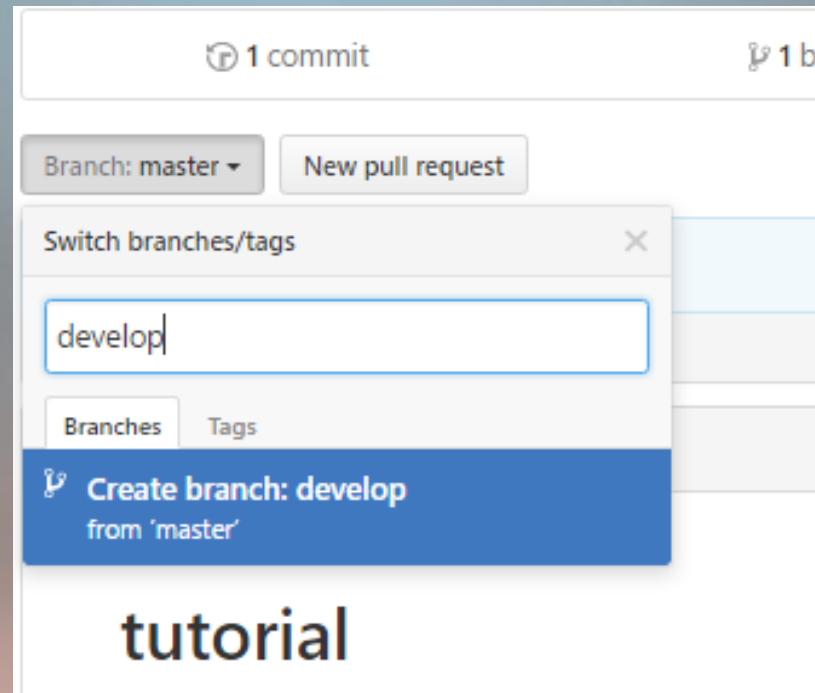
# Tạo nhánh (branch)

- Nhánh là cách thức để làm việc trên một phiên bản khác của thùng chứa tại cùng một thời điểm.
- Mặc định khi tạo thùng chứa, GitHub sẽ tạo sẵn nhánh master
- Khi tạo nhánh từ master, ta sẽ có một bản sao của master tại thời điểm đó trên nhánh mới.
- Ví dụ: tạo nhánh develop từ nhánh master



# Tạo nhánh (branch)

- Vào thùng chứa tutorial, click dropdown “Branch:master”
- Gõ vào develop ở textbox và click “Create branch: develop”



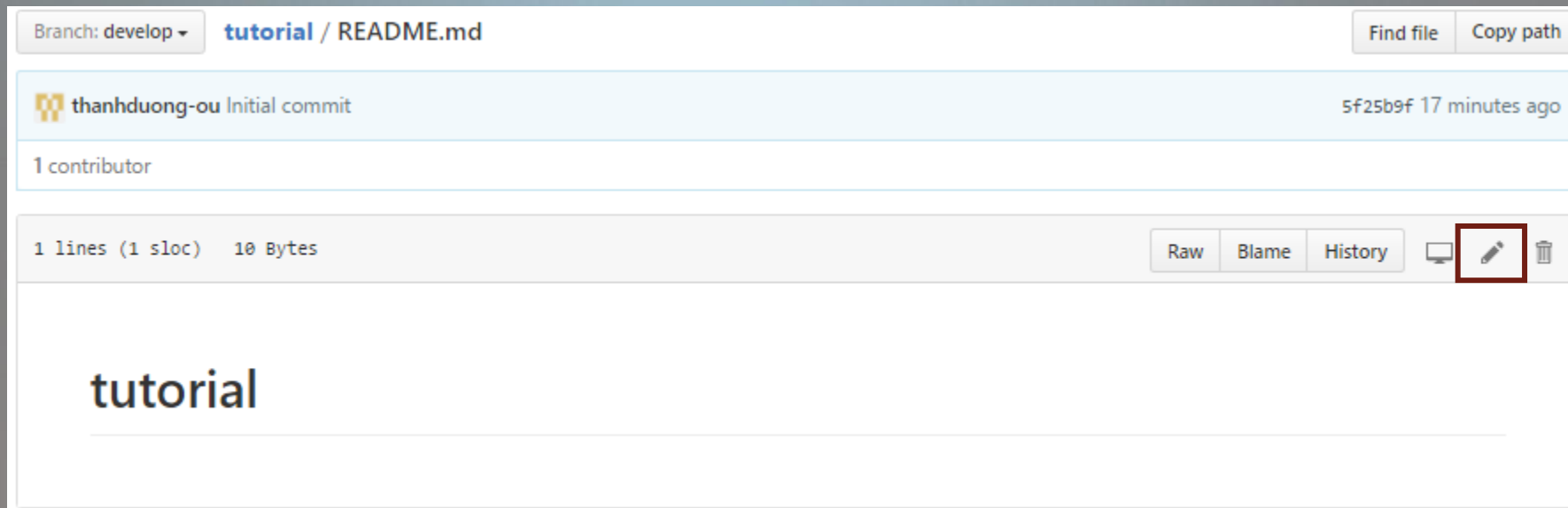


# Commit

- Trong GitHub, những thay đổi được lưu lại gọi là commit, mỗi commit có một “ghi chú” để mô tả những thay đổi trong commit đó.
- Ví dụ: chỉnh sửa file README.

# Commit

- Click vào tập tin README.md và click vào hình cây bút ở góc trên bên phải của tập tin để chỉnh sửa



# Commit

- Tiến hành chỉnh sửa tập tin, viết ghi chú (comment) mô tả thay đổi và click “commit changes”

The screenshot shows the GitHub web interface for committing changes to a file named `README.md` in a repository named `tutorial`. At the top, there's a breadcrumb `tutorial / README.md` and a button to `or cancel`. Below this is a tabbed interface with `<> Edit file` and `Preview changes`. The `Edit file` tab is active, showing the content of `README.md` with line numbers 1 through 7. The content is: `# tutorial`, `Welcome to GitHub!`, `This is the first project`, and `You will love it.`. To the right of the editor are settings for `Spaces` (set to 2) and `Soft wrap`. Below the editor is a `Commit changes` dialog box. It has a title `Commit changes` and a small GitHub logo. Inside the dialog, there's a text input field with the value `Update README.md`. Below that is a larger text area containing `Update description for the first project`. At the bottom of the dialog, there are two radio button options: `Commit directly to the develop branch.` (which is selected) and `Create a new branch for this commit and start a pull request. Learn more about pull requests.`. At the very bottom of the dialog are two buttons: `Commit changes` (in green) and `Cancel` (in grey).

tutorial / README.md or cancel

<> Edit file Preview changes Spaces 2 Soft wrap

```
1 # tutorial
2
3 Welcome to GitHub!
4
5 This is the first project
6
7 You will love it.
```

**Commit changes**

Update README.md

Update description for the first project

☒ Commit directly to the `develop` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

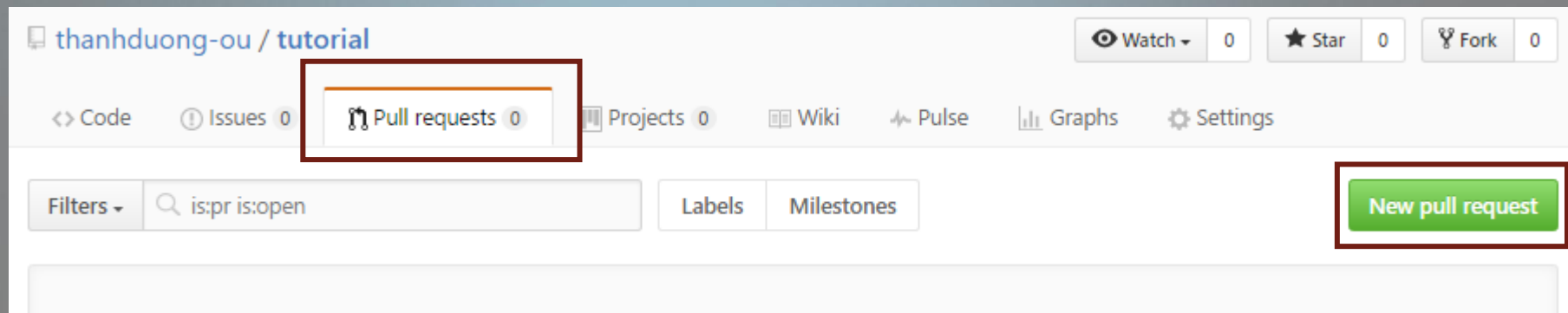
Commit changes Cancel

# Tạo Pull Request

- Pull Request (PR) dùng để đề nghị (proposing) những thay đổi và yêu cầu ai đó xét duyệt (review) để merge từ một nhánh vào nhánh nào đó.
- Pull Request sẽ thể hiện sự khác nhau nội dung giữa hai nhánh
- Ví dụ: Tạo PR để yêu cầu merge từ nhánh develop vào nhánh master.

# Tạo Pull Request

- Tại thùng chứa tutorial, click chọn tab “Pull Requests” và click vào button “New pull request”



# Tạo Pull Request

- Click dropdown “compare:” chọn nhánh develop. Sau khi chọn, kéo xuống dưới sẽ thấy sự so sánh nội dung hai nhánh.
- Sau đó, click “Create pull request” hai lần để tạo PR

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



base: master ▾

...

compare: develop ▾

✓ Able to merge. These branches can be automatically merged.



Create pull request

Discuss and review the changes in this comparison with others.



Showing 1 changed file with 7 additions and 1 deletion.

Unified Split

8 README.md



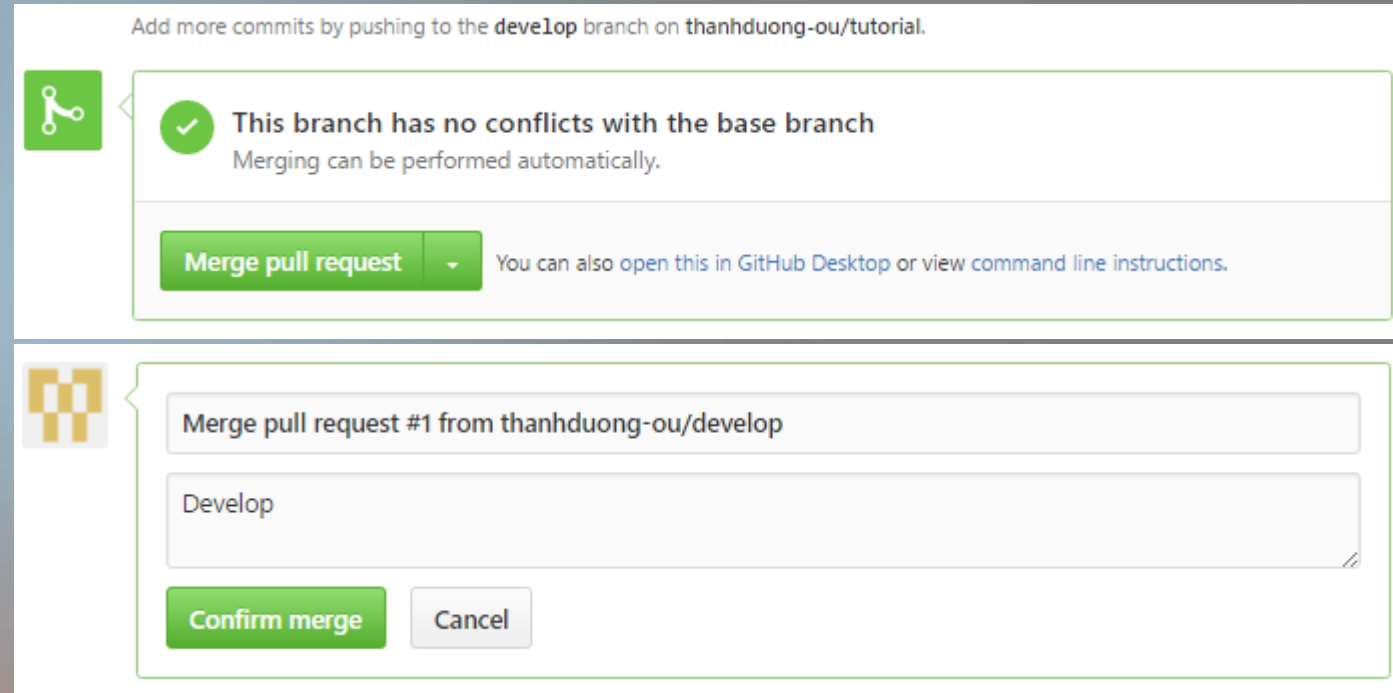
View



```
...  ...  @@ -1 +1,7 @@
1      -# tutorial
      1      +# tutorial
      2      +
      3      +Welcome to GitHub!
      4      +
      5      +This is the first project
      6      +
      7      +You will love it.
```

# Merge Pull Request

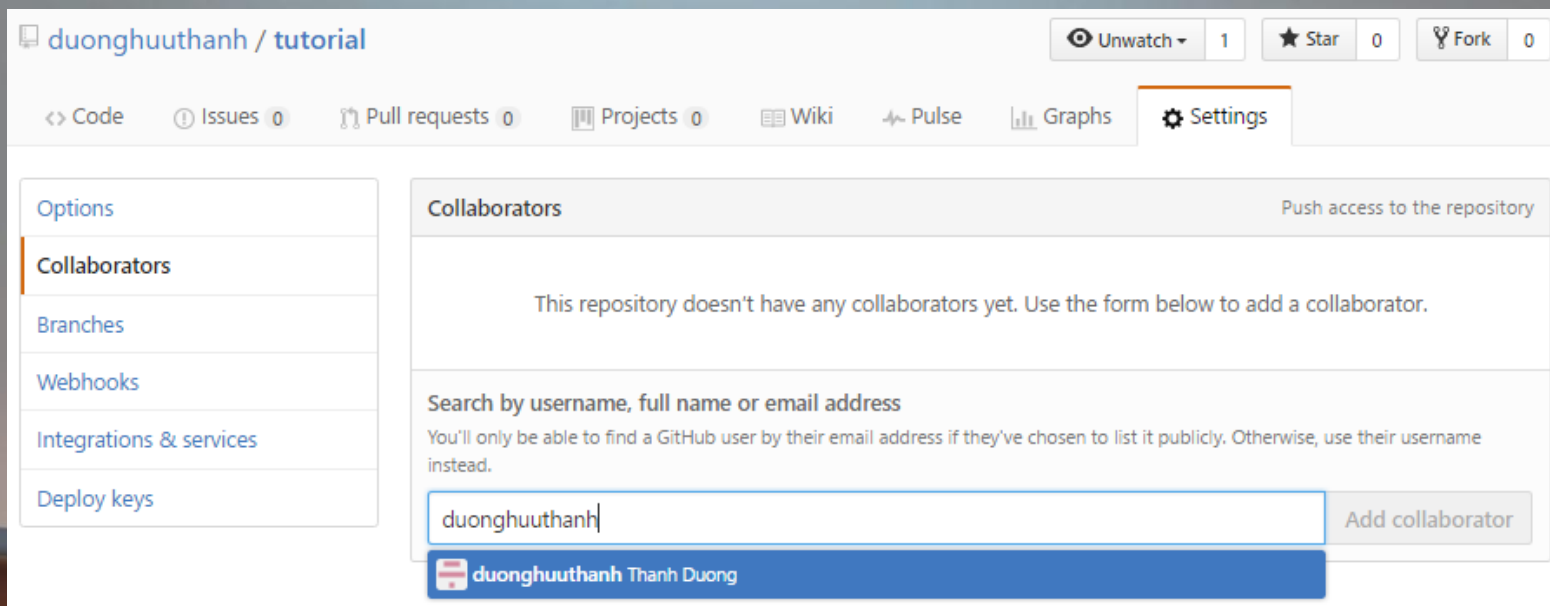
- Sau khi tạo PR, các thay đổi được tập trung trong PR để có thể xét duyệt và quyết định merge hay không.
- Để merge PR, tại PR vừa tạo click "Merge pull request" và "Confirm merge"
- Sau đó quay lại branch master, ta sẽ thấy những thay đổi trên develop đã xuất hiện trên master





# Thêm người cộng tác vào thùng chứa

- Tại thùng chứa, click chọn tab Settings, và click vào mục Collaborators ở mục menu bên trái.
- Nhập tên người muốn cộng tác vào textbox để tìm và click “Add Collaborator”, thì lời mời sẽ được gửi vào mail người vừa chọn và chờ họ chấp nhận/từ chối



# Các lệnh căn bản trong git

- Git init
- Git status
- Git log
- Git add
- Git commit
- Git branch
- Git checkout
- Git merge
- Git remote
- Git clone
- Git fetch
- Git pull
- Git push

# git init

- Lệnh này để tạo thùng chứa mới. Sau khi lệnh này thực thi thì thư mục .git (thư mục ẩn) chứa tất cả metadata của thùng chứa.

- Tạo thùng chứa tại thư mục hiện hành

```
git init
```

- Tạo thùng chứa tại thư mục chỉ định

```
git init <thư-mục>
```

- Tạo thùng chứa tại thư mục chỉ định nhưng không có thư mục .git – không thể chỉnh sửa và commit thay đổi vào thùng chứa

```
git init -bare <thư-mục>
```

# git init

- Ví dụ: tạo thùng chứa rỗng thư mục example ở Destop

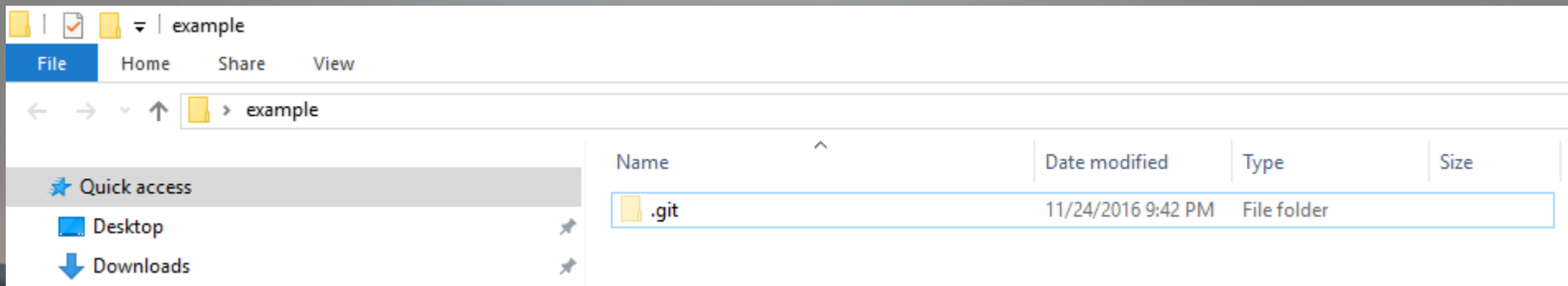
```
Command Prompt

C:\Users\dhthanh>cd Desktop

C:\Users\dhthanh\Desktop>mkdir example

C:\Users\dhthanh\Desktop>cd example

C:\Users\dhthanh\Desktop\example>git init
Initialized empty Git repository in C:/Users/dhthanh/Desktop/example/.git/
```



# git add

- Lệnh này dùng để thêm những thay đổi trong working directory vào staging area.

```
git add <tập-tin> // thêm thay đổi tập tin chỉ định  
git add <thư-mục> // thêm thay đổi thư mục thư mục  
Git add -p // thêm tất cả thay đổi
```

- Staging area giống như vùng đệm giữa working directory và project history.
- Nó dùng nhóm tất cả những thay đổi liên quan thành một snapshot để commit một lần tới project history.

# git add

- Ví dụ: Tại thư mục thùng chứa example, tạo tập tin hello.txt
  - Thêm tập tin hello.txt vào staging area để chuẩn bị commit
- Ghi chú: để đưa tập tin này ra khỏi staging area (unstage) như ban đầu dùng lệnh "git rm --cache <tập-tin>" hoặc "git reset HEAD <tập-tin>"

```
Command Prompt

C:\Users\dhthanh\Desktop\example>git add hello.txt

C:\Users\dhthanh\Desktop\example>git status
On branch master

Initial commit

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

    new file:   hello.txt
```

# git commit

- Lệnh này để commit các staged snapshot tới project history

```
git commit -m "<chú-thích>"
```

- Ví dụ: commit thay đổi ở staging area tới thùng chứa

```
Command Prompt

C:\Users\dhthanh\Desktop\example>git commit -m "create hello.txt file"
[master (root-commit) da0e646] create hello.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hello.txt

C:\Users\dhthanh\Desktop\example>git status
On branch master
nothing to commit, working tree clean
```



# git commit --amend

- Lệnh này cho phép ta kết hợp những thay đổi đã được đưa vào staging area vào commit gần nhất trước đó và thay thế commit đó với snapshot kết quả.
- Ghi chú: không nên sử dụng commit --amend khi push code vào thùng chứa chung (public repository), vì lệnh này thật sự sẽ tạo một commit mới hoàn toàn và commit trước đó sẽ bị xóa khỏi project history.

# git status

- Lệnh này dùng để xem trạng thái của working directory và staging area.
- Ví dụ: Chỉnh sửa tập tin hello.txt và thêm tập tin hello2.txt vào thung chứa. Khi đó lệnh git status sẽ cho kết quả như sau:

Command Prompt

```
C:\Users\dhthanh\Desktop\example>git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git checkout -- <file>..." to discard changes in working directory)
```

```
    modified:   hello.txt
```

```
Untracked files:
```

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

```
    hello2.txt
```

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

# git log

- Lệnh này hiển thị lịch sử commit

```
Command Prompt

C:\Users\dhthanh\Desktop\example>git log
commit da0e646ccc688361d00b94272f808560eeaab315c
Author: thanh duong <thanh.dh@ou.edu.vn>
Date:   Sat Nov 26 09:24:26 2016 +0700

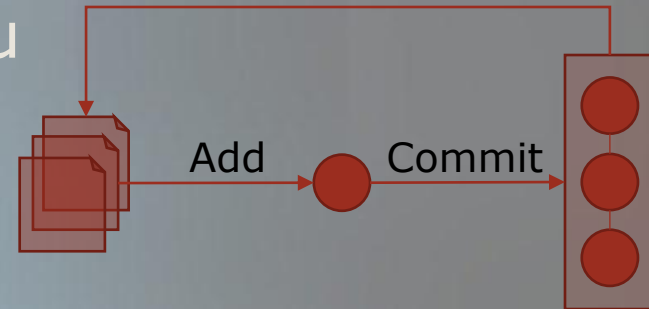
    create hello.txt file

C:\Users\dhthanh\Desktop\example>git log --oneline
da0e646 create hello.txt file
```

# git checkout

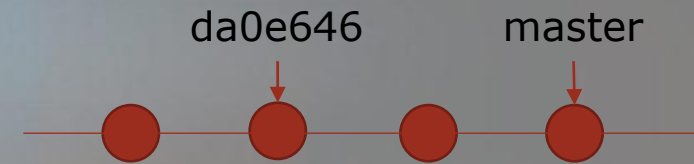
- Lệnh này thực hiện 3 chức năng khác nhau
  - Checkout tập tin trở lại phiên bản gần nhất

```
git checkout <file>
```



- Checkout commit đưa toàn bộ working directory về commit đó

```
git checkout <commit>
```



Ghi chú: **git checkout <commit> <file>** để đưa file về phiên bản trong commit chỉ định

- Checkout branch để chuyển qua một branch chỉ định

```
git checkout <branch>
```

# git checkout

- Ví dụ: Sau khi chỉnh sửa file hello.txt thì có vấn đề gì đó muốn quay lại trạng thái lúc chưa chỉnh sửa file này, ta làm như sau:

```
Command Prompt

C:\Users\dhthanh\Desktop\example>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   hello.txt

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

C:\Users\dhthanh\Desktop\example>git checkout hello.txt

C:\Users\dhthanh\Desktop\example>git status
On branch master
nothing to commit, working tree clean
```

# git checkout

- Ví dụ: Hiện tại thùng chứa có 2 commit như bên dưới

```
Select Command Prompt

C:\Users\dhthanh\Desktop\example>git log
commit 27ba26ec3b8c28afb17c3630b52c4c549849bd3
Author: thanh duong <thanh.dh@ou.edu.vn>
Date: Sat Nov 26 10:08:10 2016 +0700

    Add hello2.txt

commit da0e646cc688361d00b94272f808560eeaab315c
Author: thanh duong <thanh.dh@ou.edu.vn>
Date: Sat Nov 26 09:24:26 2016 +0700

    create hello.txt file
```



# git checkout

- Ta muốn đưa toàn bộ working directory về commit lúc chỉ làm việc với hello.txt

```
ca. Select Command Prompt
C:\Users\dhthanh\Desktop\example>git checkout da0e646cc688361d00b94272f808560eeaab315c
Note: checking out 'da0e646cc688361d00b94272f808560eeaab315c'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b <new-branch-name>

HEAD is now at da0e646... create hello.txt file

C:\Users\dhthanh\Desktop\example>git status
HEAD detached at da0e646
nothing to commit, working tree clean
```



# git branch

- Nhánh (branch) đại diện cho một đường phát triển độc lập.
- Lệnh git branch cho phép tạo, đổi tên, xóa hoặc xem danh sách nhánh.
  - git branch: xem danh sách các nhánh
  - git branch <tên-nhánh>: tạo một nhánh
  - Git branch -d <tên-nhánh>: xóa nhánh chỉ định
  - Git branch -D <tên-nhánh>: xóa nhánh chỉ định, ngay cả khi có những thay đổi chưa được merge
  - Git branch -m <tên-nhánh>: đổi tên nhánh hiện tại thành <tên-nhánh>

# git branch

- Ví dụ: Tạo nhánh tên là staging từ nhánh master.

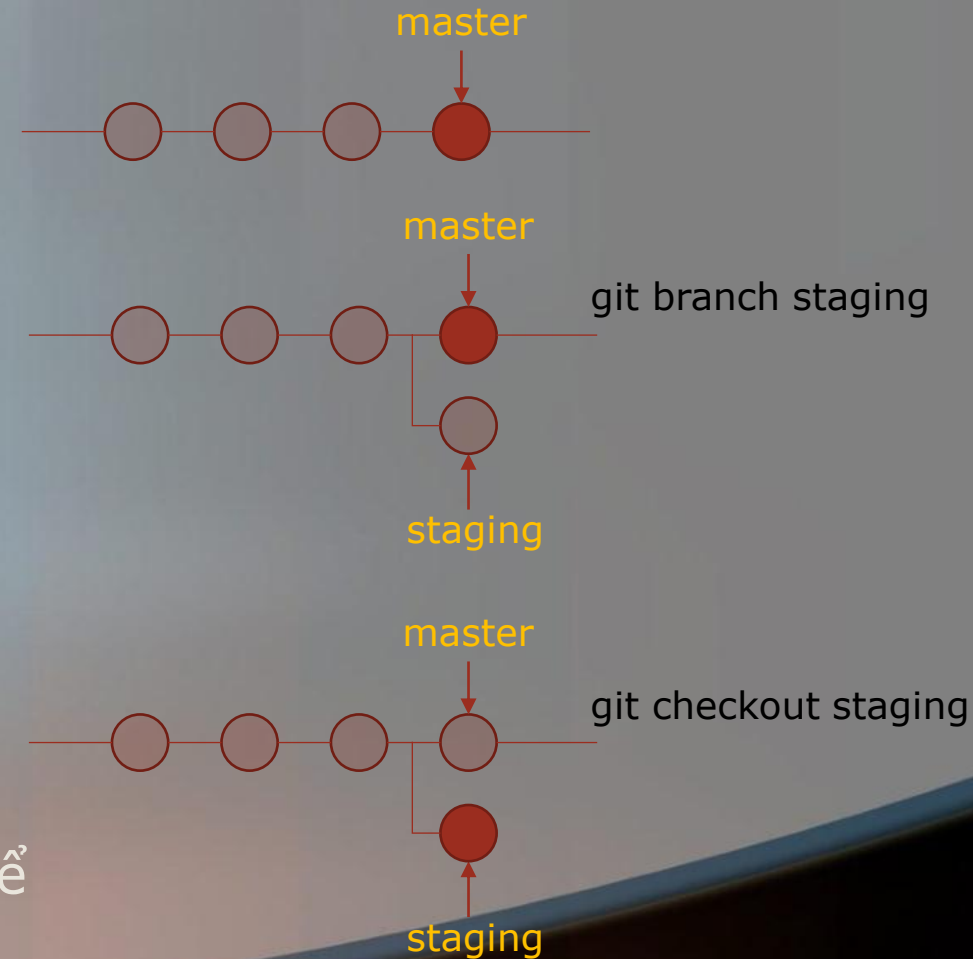
```
C:\Users\dhthanh\Desktop\example>git branch
* master

C:\Users\dhthanh\Desktop\example>git branch staging

C:\Users\dhthanh\Desktop\example>git branch
* master
  staging

C:\Users\dhthanh\Desktop\example>git checkout staging
Switched to branch 'staging'
```

- Ghi chú: dùng lệnh "git checkout <tên-nhánh>" để chuyển đến một nhánh chỉ định



# git branch

- Ghi chú: 2 cách dùng sau đây là tương đương

```
git branch staging  
git checkout staging
```



```
git checkout -b staging
```

- Lệnh “git checkout -b <branch>” sẽ chuyển đến nhánh <branch> nếu đã tồn tại, ngược lại sẽ tạo nhánh <branch> và chuyển đến nó.

# git branch

- Đứng tại nhánh staging, ta đổi tên nhánh này thành develop như sau:

```
C:\Users\dhthanh\Desktop\example>git branch
master
* staging

C:\Users\dhthanh\Desktop\example>git branch -m "develop"

C:\Users\dhthanh\Desktop\example>git branch
* develop
master
```

- Ghi chú: lệnh git branch là xem danh sách nhánh, và nhánh nào hiện màu xanh nghĩa là đang đứng tại nhánh đó.

# git branch

- Tại nhánh develop tạo tập tin hello\_develop.txt, thực hiện add và commit vào nhánh develop.

```
C:\Users\dhthanh\Desktop\example>git branch
* develop
  master

C:\Users\dhthanh\Desktop\example>git add hello_develop.txt

C:\Users\dhthanh\Desktop\example>git commit -m "Create hello_develop.txt at develop"
[develop 9d03fc4] Create hello_develop.txt at develop
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello_develop.txt
```

# git branch

- Chuyển sang nhánh master và thực hiện xóa nhánh develop

- Nếu sử dụng "git branch -d develop" sẽ có thông báo lỗi như bên dưới do nhánh develop có thay đổi chưa được merge vào

- Nếu chắc chắn muốn xóa nhánh mà không cần merge thay đổi trên develop thì dùng "git branch -D develop" thay thế

```
C:\Users\dhthanh\Desktop\example>git checkout master  
Switched to branch 'master'
```

```
C:\Users\dhthanh\Desktop\example>git branch  
develop  
* master
```

```
C:\Users\dhthanh\Desktop\example>git branch -d develop  
error: The branch 'develop' is not fully merged.  
If you are sure you want to delete it, run 'git branch -D develop'.
```

```
C:\Users\dhthanh\Desktop\example>git branch -D develop  
Deleted branch develop (was 9d03fc4).
```

```
C:\Users\dhthanh\Desktop\example>git branch  
* master
```



# git merge

- Lệnh này cho phép tích hợp sự thay đổi trên một nhánh vào một nhánh nào đó.
  - Merge một nhánh chỉ định vào nhánh hiện tại
- Merge một nhánh chỉ định vào nhánh hiện tại và luôn tạo ra một merge commit

```
git merge <tên-nhánh>
```

```
git merge --no-ff <tên-nhánh>
```



# git merge

- Ví dụ: Tạo nhánh feature-test từ master và chuyển đến làm việc trên nhánh này.

```
C:\Users\dhthanh\Desktop\example>git branch
* master

C:\Users\dhthanh\Desktop\example>git checkout -b feature-test
Switched to a new branch 'feature-test'

C:\Users\dhthanh\Desktop\example>git branch
* feature-test
  master
```

# git merge

- Tại nhánh feature-test, tiến hành chỉnh sửa tập tin hello.txt, thêm một tập tin test.txt và commit tất cả thay đổi vào nhánh

```
C:\Users\dhthanh\Desktop\example>git status
On branch feature-test
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   hello.txt

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

        test.txt

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

C:\Users\dhthanh\Desktop\example>git add hello.txt test.txt
C:\Users\dhthanh\Desktop\example>git commit -m "Edit hello.txt and create test.txt"
[feature-test 1d8f4c4] Edit hello.txt and create test.txt
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 test.txt
```

# git merge

- Bây giờ merge những thay đổi từ nhánh feature-test và nhánh master

```
C:\Users\dhthanh\Desktop\example>git checkout master
Switched to branch 'master'

C:\Users\dhthanh\Desktop\example>git merge feature-test
Updating 06c9539..1d8f4c4
Fast-forward
 hello.txt | 3 ++-
 test.txt  | 0
2 files changed, 2 insertions(+), 1 deletion(-)
create mode 100644 test.txt
```

# git merge

- Ví dụ: Chuyển về nhánh feature-test chỉnh sửa test.txt và commit thay đổi vào nhánh.

```
C:\Users\dhthanh\Desktop\example>git status
On branch feature-test
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test.txt

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

C:\Users\dhthanh\Desktop\example>git add test.txt

C:\Users\dhthanh\Desktop\example>git commit -m "test merge --no-ff"
[feature-test af92131] test merge --no-ff
1 file changed, 1 insertion(+)
```

# git merge

- Ta sẽ dùng merge --no-ff để merge nhánh feature-test vào master như sau
- Ta thấy có một merge commit đã được tạo bên cạnh commit trong nhánh feature-test

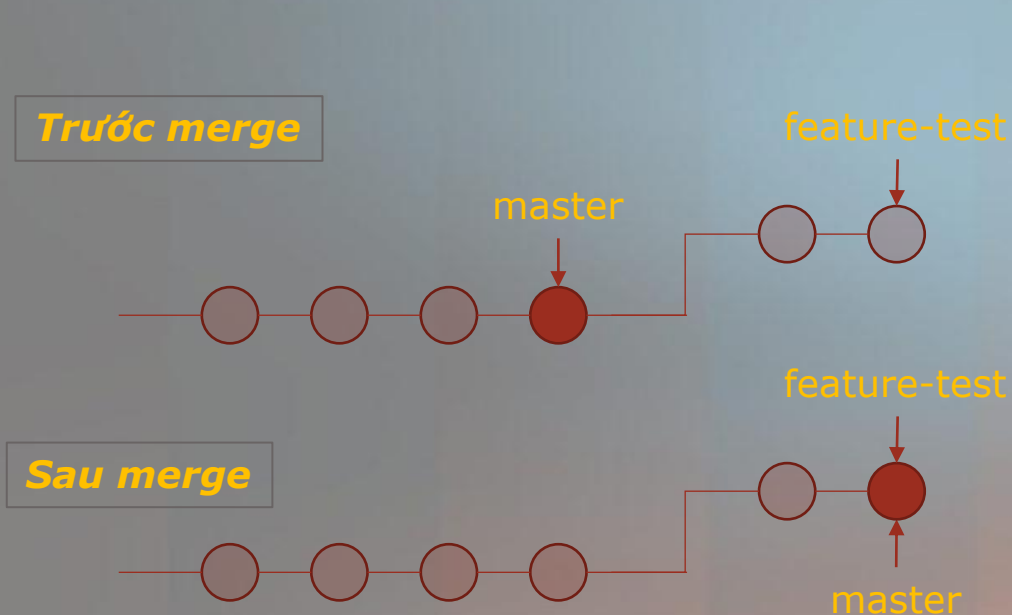
```
C:\Users\dhthanh\Desktop\example>git log --oneline
1d8f4c4 Edit hello.txt and create test.txt
06c9539 Revert "resolve conflict"
585e674 resolve conflict
c9e7d1c Update 3
ada0ff9 update 2
818a546 Update 1
```

```
C:\Users\dhthanh\Desktop\example>git merge --no-ff feature-test
Merge made by the 'recursive' strategy.
 test.txt | 1 +
 1 file changed, 1 insertion(+) █
```

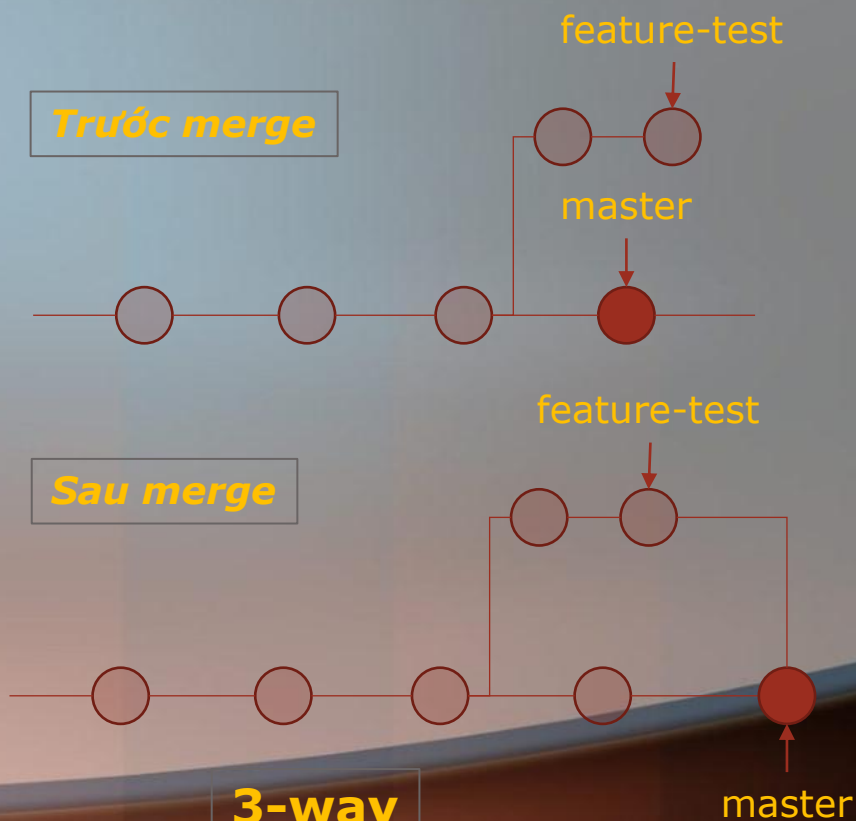
```
C:\Users\dhthanh\Desktop\example>git log --oneline
d32b385 Merge branch 'feature-test'
af92131 test merge --no-ff
1d8f4c4 Edit hello.txt and create test.txt
06c9539 Revert "resolve conflict"
585e674 resolve conflict
c9e7d1c Update 3
ada0ff9 update 2
818a546 Update 1
```

# git merge

- Tùy vào cấu trúc thừng chứa mà git merge sẽ lựa chọn thuật toán merge phù hợp: fast-forward hoặc 3-way



**Fast-forward**



**3-way**



# Resolve conflict

- Nếu hai nhánh chỉnh sửa cùng một phần của tập tin thì khi merge có khả năng xảy ra xung đột, git không thể nào quyết định được phiên bản nào sẽ được sử dụng khi xung đột xảy ra, nên ta cần giải quyết vấn đề này thủ công.
- Chú ý: xung đột khi merge chỉ xảy ra trong merge theo cách 3-way.



# Resolve conflict

- Ví dụ: Tại nhánh master chỉnh sửa tập tin test.txt thêm dòng "master" vào cuối tập tin. Tương tự tại nhánh feature-test cũng chỉnh sửa tập tin test.txt và thêm dòng "feature-test" ở cuối tập tin

```
C:\Users\dhthanh\Desktop\example>git branch
feature-test
* master

C:\Users\dhthanh\Desktop\example>git add test.txt

C:\Users\dhthanh\Desktop\example>git commit -m "The last line: master"
[master 11c1017] The last line: master
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\dhthanh\Desktop\example>git checkout feature-test
Switched to branch 'feature-test'

C:\Users\dhthanh\Desktop\example>git add test.txt

C:\Users\dhthanh\Desktop\example>git commit -m "The last line: feature-test"
[feature-test 43ccaa9] The last line: feature-test
1 file changed, 2 insertions(+), 1 deletion(-)
```

# Resolve conflict

- Chuyển sang nhánh master và tiến hành merge nhánh feature-test vào, ta sẽ thấy kết quả conflict như sau:

```
C:\Users\dhthanh\Desktop\example>git checkout master
Switched to branch 'master'

C:\Users\dhthanh\Desktop\example>git merge feature-test
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\dhthanh\Desktop\example>git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

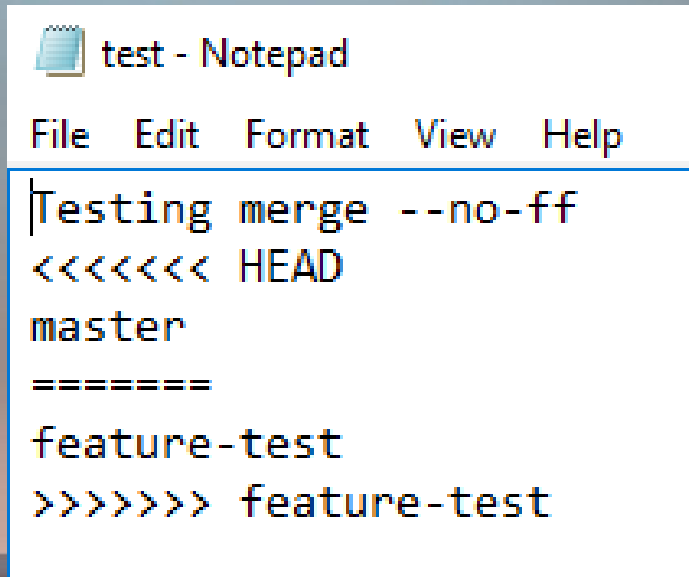
Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   test.txt

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

# Resolve conflict

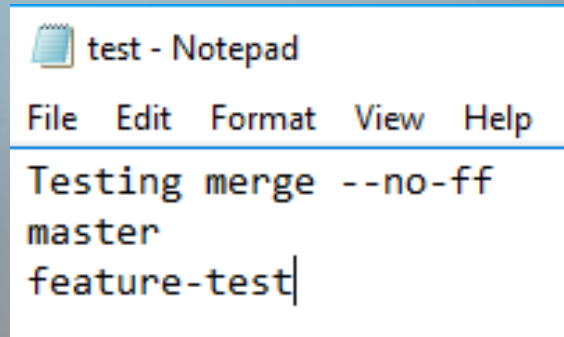
- Mở tập tin test.txt ta sẽ thấy kết quả giống như sau.
  - Phần nằm giữa "<<<<<< HEAD" và "======" là nội dung trong nhánh master
  - Phần nằm giữa "======" và ">>>>>> feature-test" là nội dung trong nhánh feature-test



```
test - Notepad
File Edit Format View Help
Testing merge --no-ff
<<<<<< HEAD
master
=====
feature-test
>>>>>> feature-test
```

# Resolve conflict

- Ta cần quyết định giữ lại nội dung nào, xóa đi các ký hiệu trên, và thực hiện git add, commit thì đã giải quyết được xung đột. Ở đây là muốn giữ lại nội dung cả hai nhánh nên nội dung tập tin như sau



```
C:\Users\dhthanh\Desktop\example>git add test.txt  
  
C:\Users\dhthanh\Desktop\example>git commit -m "resolve conflict with feature-test branch"  
[master d99ed79] resolve conflict with feature-test branch  
  
C:\Users\dhthanh\Desktop\example>git status  
On branch master  
nothing to commit, working tree clean
```

# git remote

- Lệnh này cho phép tạo, xem, xóa kết nối với các thùng chứa khác.

- Xem danh sách remote connection kết nối tới các thùng chứa khác

```
git remote
```

- Tạo kết nối với remote repository

```
git remote add <name> <url>
```

- Hủy kết nối với remote repository

```
git remote remove <name>
```

- Đổi tên kết nối remote connection

```
git remote rename <old-name> <new-name>
```

# git remote

- Ví dụ: Tạo kết nối tên origin thủng chứa example với thủng chứa tutorial trên GitHub.

```
C:\Users\dhthanh\Desktop\example>git remote add origin https://github.com/thanhduong-ou/tutorial.git

C:\Users\dhthanh\Desktop\example>git remote
origin

C:\Users\dhthanh\Desktop\example>git remote -v
origin https://github.com/thanhduong-ou/tutorial.git (fetch)
origin https://github.com/thanhduong-ou/tutorial.git (push)
```

- Ghi chú: lệnh “git remote -v” xem danh sách kết nối, cùng với url của nó.

# git clone

- Lệnh này tạo một thùng chứa mới bằng cách sao chép từ một thùng chứa đã tồn tại.
  - Sao chép vào thư mục hiện hành

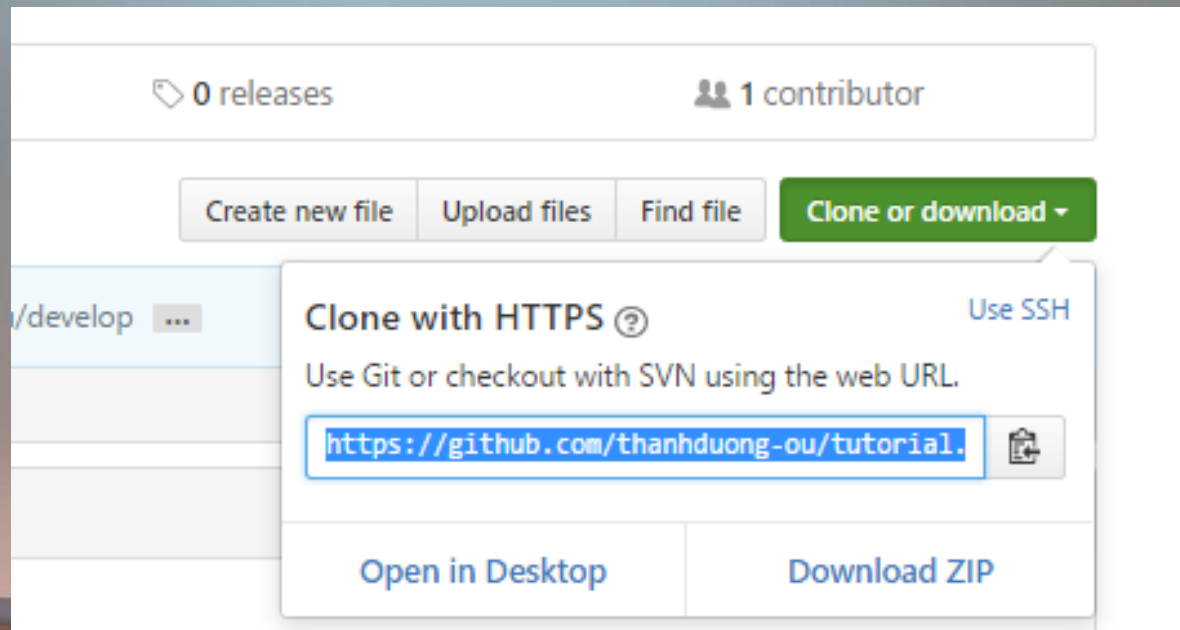
```
git clone <thùng-chứa-nguồn>
```
  - Sao chép vào thư mục chỉ định

```
git clone <thùng-chứa-nguồn> <thư-mục>
```
- Tham số <thùng-chứa-nguồn> là vị trí thùng chứa muốn sao chép, nó có thể là một thùng chứa local hoặc thùng chứa remote được truy cập thông qua HTTP hoặc SSH.



# git clone

- Ví dụ: Ta sẽ tạo thùng chứa bằng cách clone thùng chứa tutorial trên GitHub.
- Tại thùng chứa tutorial, click button “Clone or download” và copy link <https://github.com/thanhduong-ou/tutorial.git>



# git clone

- Tại Destop tạo thư mục demo và sẽ clone thùng chứa tutorial vào thư mục này như sau

```
C:\Users\dhthanh\Desktop>mkdir demo

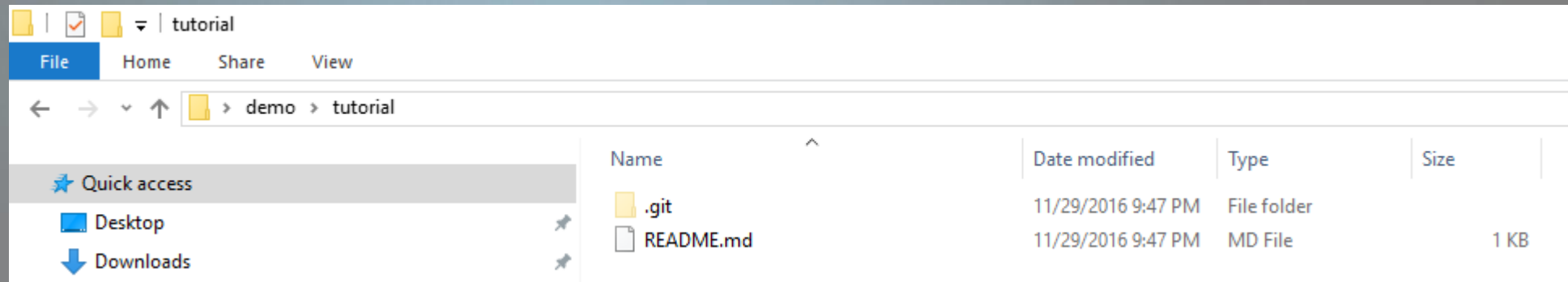
C:\Users\dhthanh\Desktop>cd demo

C:\Users\dhthanh\Desktop\demo>git clone https://github.com/thanhduong-ou/tutorial.git
Cloning into 'tutorial'...
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), done.
```

- Do thùng chứa tutorial là public nên git không yêu cầu ta nhập username và password GitHub . Trong trường hợp thùng chứa private git sẽ yêu cầu thông tin này.

# git clone

- Vào demo/tutorial sẽ thấy kết quả như sau



# git clone

- Khi thực thi git clone, thì nó sẽ tự động tạo một kết nối remote trở tới thùng chứa gốc. Điều này giúp ta dễ dàng tương tác với thùng chứa trung tâm.
- Mở tập tin demo/tutorial/.git/config sẽ có phần config remote giống như sau

```
[remote "origin"]  
    url = https://GitHub .com/thanhduong-ou/tutorial.git  
    fetch = +refs/heads/*:refs/remotes/origin/*
```

# git fetch

- Lệnh này dùng để import các commit từ một thùng chứa remote vào thùng chứa local. Các commit kết quả sẽ được lưu trong các nhánh remote
  - Lấy tất cả các nhánh từ thùng chứa remote

```
git fetch <remote>
```

- Lấy nhánh được chỉ định từ thùng chứa remote

```
git fetch <remote> <branch>
```

# git fetch

- Ví dụ: Thực hiện fetch tất cả các nhánh từ thùng chứa tutorial

```
C:\Users\dhthanh\Desktop\demo>cd tutorial

C:\Users\dhthanh\Desktop\demo\tutorial>git branch
* master

C:\Users\dhthanh\Desktop\demo\tutorial>git fetch

C:\Users\dhthanh\Desktop\demo\tutorial>git checkout develop
Branch develop set up to track remote branch develop from origin.
Switched to a new branch 'develop'

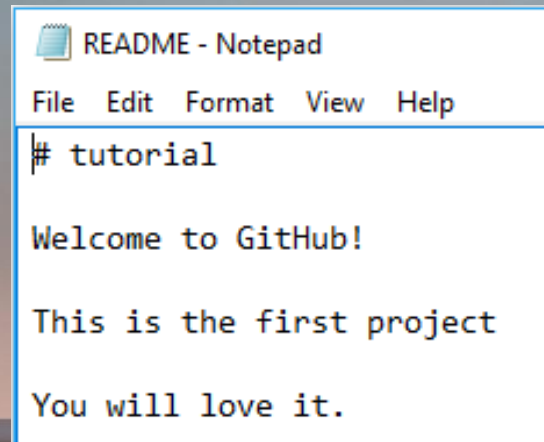
C:\Users\dhthanh\Desktop\demo\tutorial>git branch
* develop
  master
```

# git pull

- Lệnh này tương đương thực hiện tuần tự hai lệnh: git fetch + git merge

```
git pull <remote>
```

- Ví dụ: Mở tập tin demo/tutorial/README.md hiện tại có nội dung như sau



```
README - Notepad
File Edit Format View Help
# tutorial

Welcome to GitHub!

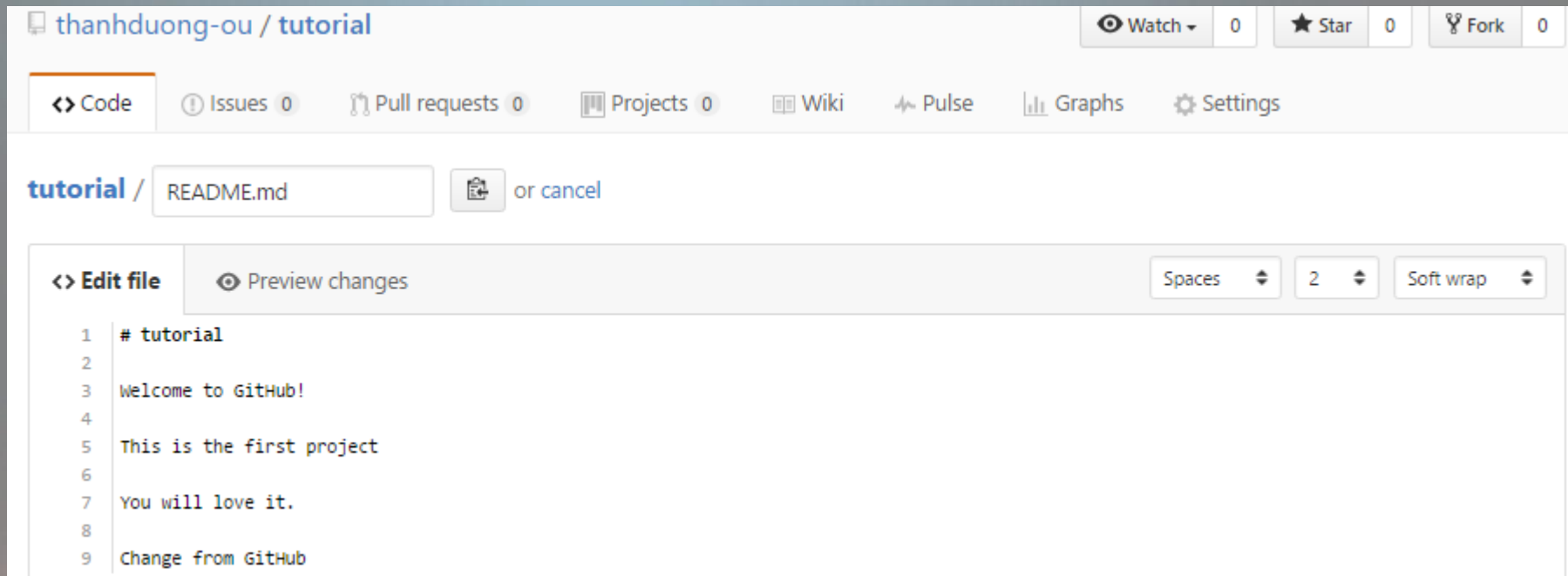
This is the first project

You will love it.
```



# git pull

- Trên GitHub ta chỉnh sửa tập tin này (nhánh master) như bên dưới và commit vào thùng chứa:

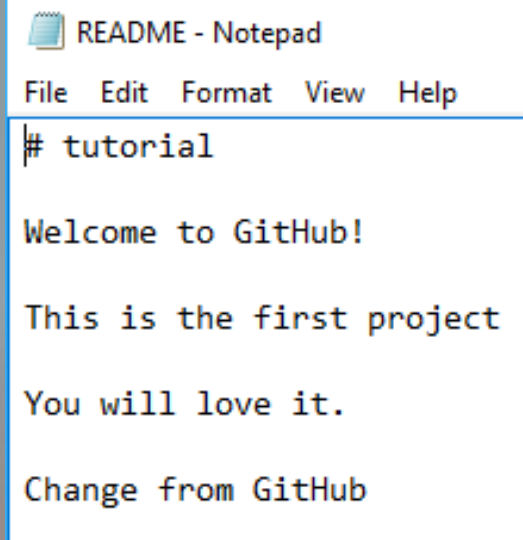


# git pull

- Bây giờ ta lấy những thay đổi từ thùng chứa remote và merge vào nhánh master hiện tại trên local như sau:

```
C:\Users\dhthanh\Desktop\demo\tutorial>git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

C:\Users\dhthanh\Desktop\demo\tutorial>git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/thanhduong-ou/tutorial
* branch                master       -> FETCH_HEAD
   00b8f80..cce4bcf      master       -> origin/master
Updating 00b8f80..cce4bcf
Fast-forward
 README.md | 2 ++
1 file changed, 2 insertions(+)
```



```
README - Notepad
File Edit Format View Help
# tutorial

Welcome to GitHub!

This is the first project

You will love it.

Change from GitHub
```

# git push

- Lệnh này dùng chuyển các commit từ thùng chứa local đến thùng chứa remote
  - Push nhánh chỉ định vào <remote>

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

- Push tất cả các nhánh local tới <remote> chỉ định

```
git push <remote> --all
```

# git push

- Ví dụ: Chỉnh sửa tập tin demo/tutorial/README.md như sau

```
README - Notepad
File Edit Format View Help
# tutorial

Welcome to GitHub!

This is the first project

You will love it.

Change from GitHub

Change from Local|
```

```
C:\Users\dhthanh\Desktop\demo\tutorial>git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

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

# git push

- Ta thực hiện commit thay đổi vào vào thùng chứa local và push lên thùng chứa remote.

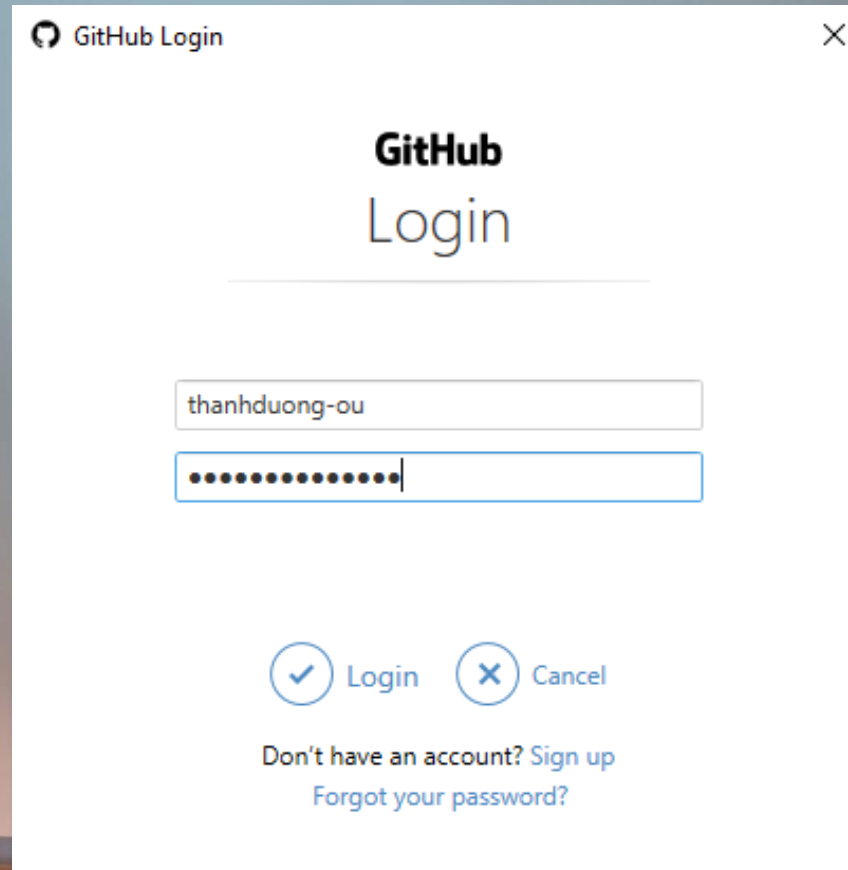
```
C:\Users\dhthanh\Desktop\demo\tutorial>git add README.md

C:\Users\dhthanh\Desktop\demo\tutorial>git commit -m "Change README.md at local"
[master c305f42] Change README.md at local
 1 file changed, 2 insertions(+)

C:\Users\dhthanh\Desktop\demo\tutorial>git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 300 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To https://github.com/thanhduong-ou/tutorial.git
   cce4bcf..c305f42  master -> master
```

# git push

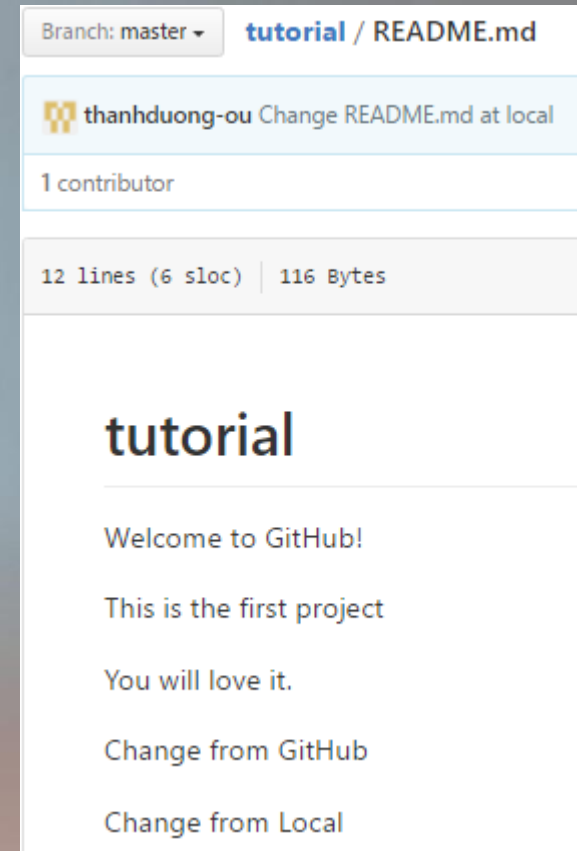
- Chú ý khi thực hiện lệnh git push thì git yêu cầu ta đăng nhập tài khoản GitHub



The image shows a GitHub Login dialog box. At the top left is the GitHub logo and the text "GitHub Login". At the top right is a close button (X). In the center, the text "GitHub Login" is displayed. Below this is a horizontal line. There are two input fields: the first contains the username "thanhduong-ou", and the second contains a password represented by dots. At the bottom, there are two buttons: "Login" with a checkmark icon and "Cancel" with an X icon. Below these buttons, there are two links: "Don't have an account? Sign up" and "Forgot your password?".

# git push

- Truy cập vào thùng chứa GitHub ta sẽ thấy thay đổi của tập tin README.md trên nhánh master

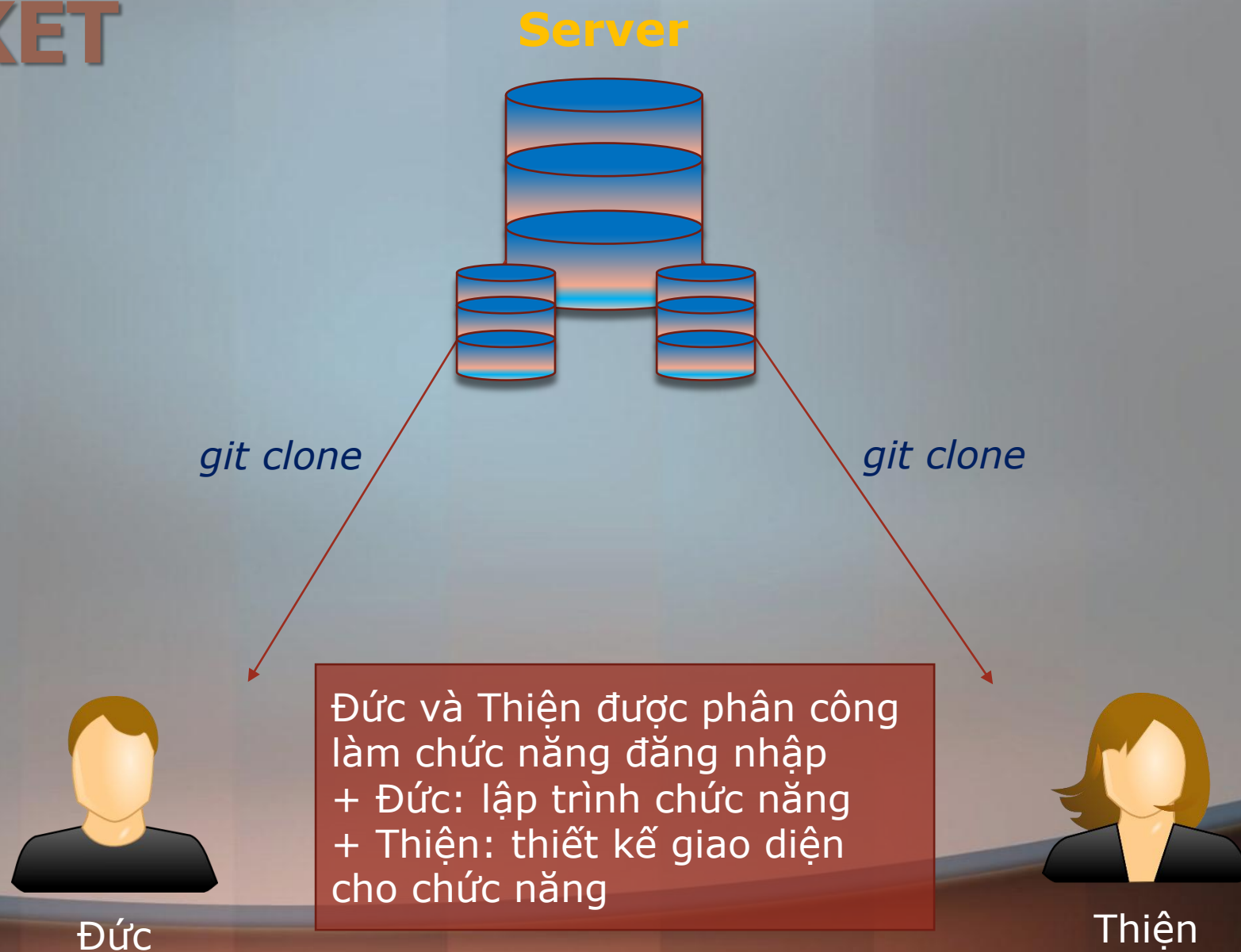




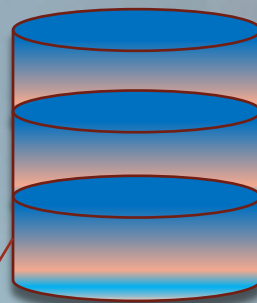
# Tập tin .gitignore

- Trong project thường sẽ có những tập tin, thư mục tự động sinh ra có thể do trình biên dịch, framework sinh ra và những tập tin/thư mục này không cần thiết/không quan trọng để đưa vào git, chẳng hạn:
  - Thư mục bin trong các project C#
  - Các tập tin biên dịch .class, .pyc, ...
  - Các tập tin ẩn hệ thống: .DS\_Store, Thumbs.db
- khi đó ta có thể dùng tập tin .gitignore đặt tại thư mục gốc của thùng chứa để bỏ qua những tập tin/thư mục đó.

# TỔNG KẾT



Server



*git clone*

*git clone*

master

feature-login

master

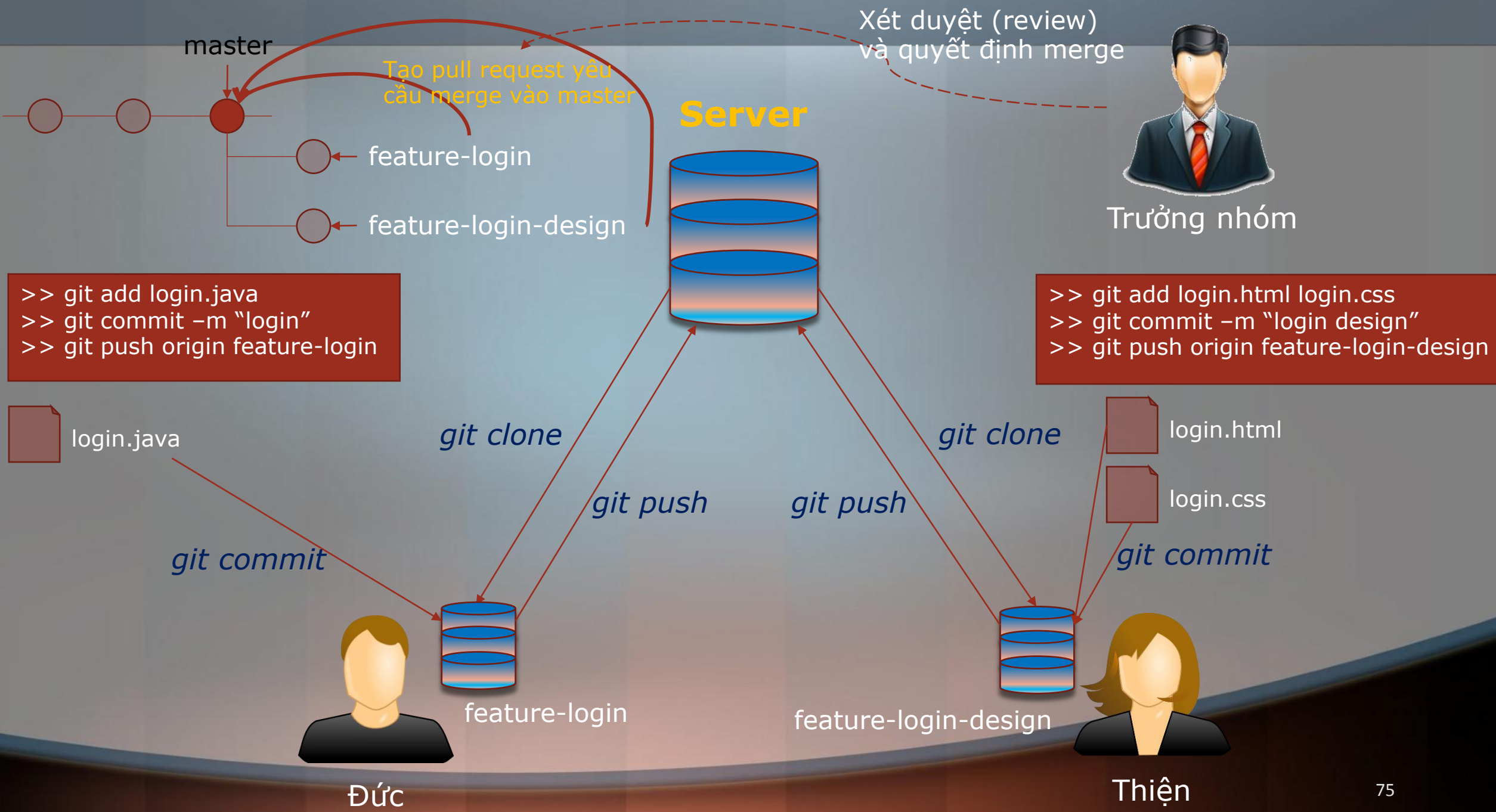
feature-login-design

master



Đức

Thiện



master

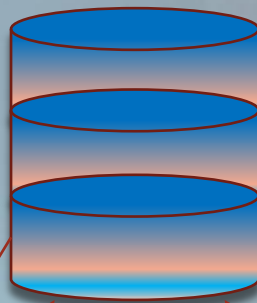
Tạo pull request yêu cầu merge vào master

Xét duyệt (review) và quyết định merge



Trưởng nhóm

Server



feature-login

feature-login-design

```
>> git add login.java
>> git commit -m "login"
>> git push origin feature-login
```

```
>> git add login.html login.css
>> git commit -m "login design"
>> git push origin feature-login-design
```



login.java

git commit

git clone

git push

git clone

git push



login.html



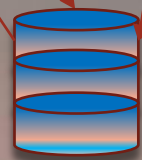
login.css

git commit



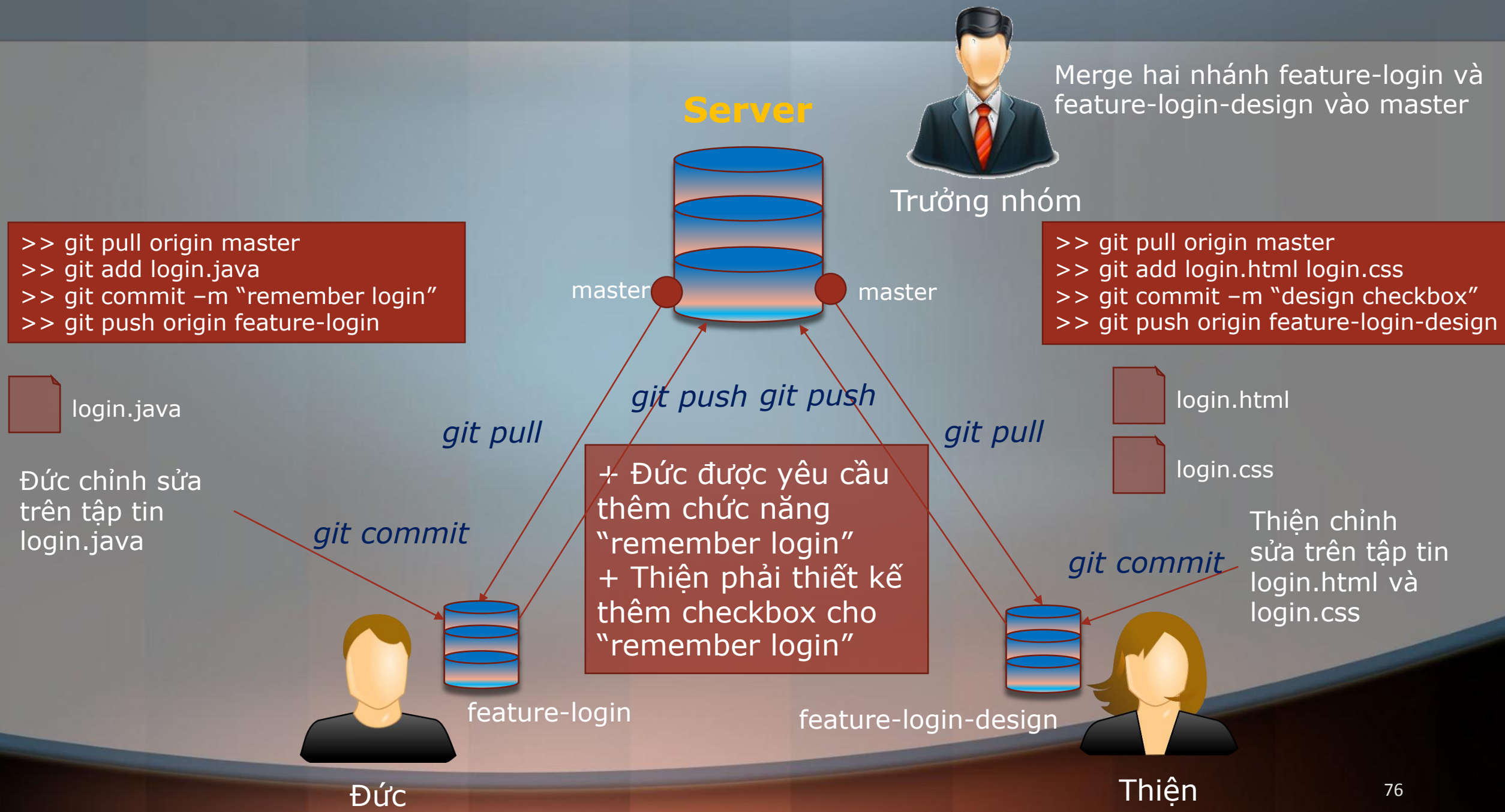
feature-login

Đức



feature-login-design

Thiên



Merge hai nhánh feature-login và feature-login-design vào master

Trưởng nhóm

```
>> git pull origin master
>> git add login.java
>> git commit -m "remember login"
>> git push origin feature-login
```

```
>> git pull origin master
>> git add login.html login.css
>> git commit -m "design checkbox"
>> git push origin feature-login-design
```

login.java

Đức chỉnh sửa trên tập tin login.java

login.html  
login.css

Thiện chỉnh sửa trên tập tin login.html và login.css

+ Đức được yêu cầu thêm chức năng "remember login"  
+ Thiện phải thiết kế thêm checkbox cho "remember login"

Đức

Thiện

# Q&A