

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



# BÁO CÁO

## TỔNG QUAN SỬ DỤNG GITHUB

*Sinh viên thực hiện:*

<b>MSSV</b>	<b>Họ tên</b>
19520524	- Phan Vỹ Hào
19520843	- Trần Xuân Phú
19520576	- Lê Văn Hùng
19520758	- Trần Đình Nam

**MÔN HỌC KỸ NĂNG NGHỀ NGHIỆP**

**SS004.K23**

**TP. HỒ CHÍ MINH, 2020**



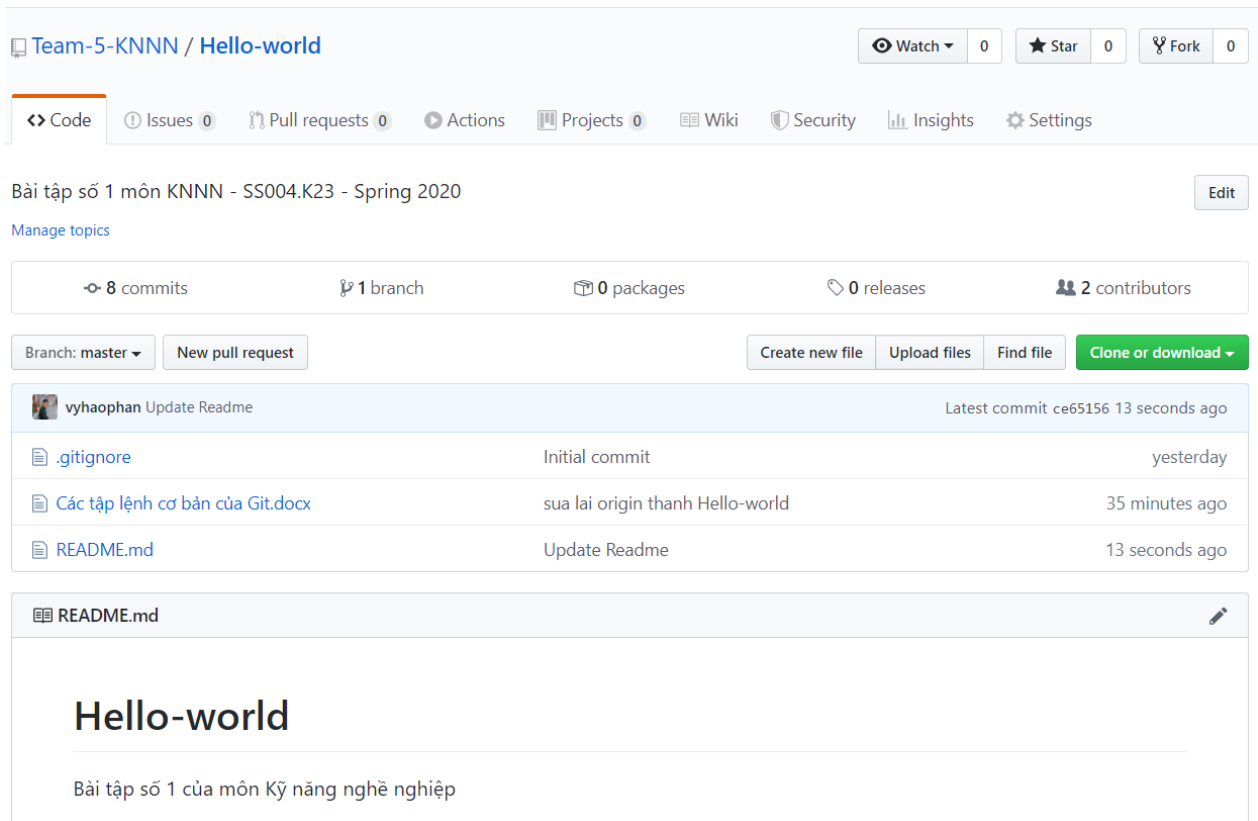
# MỤC LỤC

Chương 1. Giao diện của một project GITHUB .....	4
Phần 1:.....	4
Phần 2:.....	5
Phần 3:.....	6
Phần 4:.....	7
Chương 2. Các lệnh được dùng trong Github .....	8
Thiết lập chứng thực cá nhân.....	8
Tạo một kho chứa Git.....	8
Sao chép một kho chứa đã tồn tại .....	8
Nhánh trong git .....	8
Chuyển nhánh .....	9
Cập nhật thay đổi .....	9
Cập nhật lên server .....	9
Gộp nhánh.....	10
Xem lại lịch sử commit .....	10
Gộp commit.....	11
Pull từ remote repository.....	11

## DANH SÁCH HÌNH ẢNH

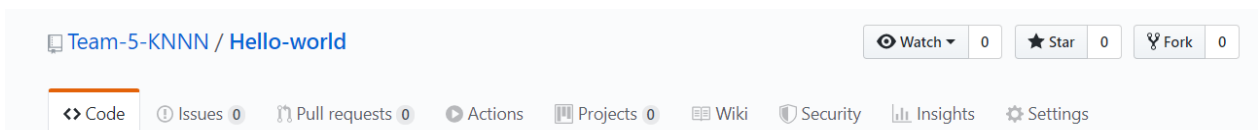
Hình ảnh 1 Giao diện của một repository .....	4
Hình ảnh 2 Thanh chức năng của Github.....	4
Hình ảnh 3 Các chức năng tương tác của người dùng.....	5
Hình ảnh 4 Các tập tin trong repo .....	6
Hình ảnh 5 File Readme .....	7

# Chương 1. Giao diện của một project GITHUB



Hình ảnh 1 Giao diện của một repository

## **Phần 1:**



Hình ảnh 2 Thanh chức năng của Github

**Code:** Từ đây, bạn có thể xem các nhánh (*branch*) khác nhau đang được thực hiện, cũng như khi ai đó thực hiện một **Commit** (đây là loại giống như lưu một tập tin). Tùy thuộc vào cách thiết lập kho lưu trữ, bạn cũng có thể tạo **Branch** của riêng mình và thực hiện các **Commits** của riêng mình ở đó.

**Issues:** Các vấn đề được sử dụng để theo dõi mã thông báo, lỗi, yêu cầu tính năng ... Khi các issues được tạo ra, chúng sẽ xuất hiện trong một danh sách có thể tìm kiếm và lọc được.

**Pull requests:** Yêu cầu kéo giúp bạn cộng tác trên code với người khác. Khi các yêu cầu kéo được tạo, chúng sẽ xuất hiện trong danh sách có thể tìm kiếm và lọc được

**Action:** Có thể chọn một quy trình công việc để xây dựng, kiểm tra và triển khai code của bạn. Thực hiện đánh giá code, quản lý chi nhánh và xử lý vấn đề theo cách bạn muốn.

**Projects:** Cho phép sắp xếp các Issues của bạn với Projects board.

Bạn có biết bạn có thể quản lý các dự án ở cùng một nơi bạn giữ code của mình không? Thiết lập Projects Board trên GitHub để hợp lý hóa và tự động hóa quy trình làm việc của bạn.

**Wiki:** GitHub Wikis là một cách đơn giản để cho người khác đóng góp nội dung. Bất kỳ người dùng GitHub nào cũng có thể tạo và chỉnh sửa các trang để sử dụng cho tài liệu, ví dụ, hỗ trợ hoặc bất cứ điều gì bạn muốn.

**Security:** Có thể tìm hiểu những thông tin, thông báo về bảo mật của Projects.

**Insights:**

**Settings:** Cài đặt những phần làm việc với Projects: Options, Manage access, Branches, Webhooks, Notifications,...

## ***Phần 2:***

Bài tập số 1 môn KNNN - SS004.K23 - Spring 2020

Edit

[Manage topics](#)

8 commits

1 branch

0 packages

0 releases

2 contributors

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

*Hình ảnh 3 Các chức năng tương tác của người dùng*

**Commits:** thao tác để ghi lại lịch sử việc thêm, thay đổi file hay thư mục vào repository.

Khi thực hiện commit, trong repository sẽ tạo ra commit (hoặc revision) mới ghi lại sự khác biệt từ trạng thái đã commit lần trước với trạng thái hiện tại.

Commit này đang được chứa tại repository, các commit nối tiếp với nhau theo thứ tự thời gian. Bằng việc lần theo commit thì có thể biết được lịch sử thay đổi trong quá khứ.

**Branch:** Nhánh có thể hiểu như là một không gian làm việc (workspace), Trong một project sẽ luôn có một nhánh chính (mặc định) gọi là master. Tính năng được tạo ra trong các nhánh phụ sẽ được hợp nhất lại vào master khi đã làm xong, hành động này gọi là merge.

**Packages:** là một dịch vụ lưu trữ gói phần mềm cho phép bạn lưu trữ các gói phần mềm của mình một cách riêng tư hoặc công khai và sử dụng các gói làm phụ thuộc trong các dự án của bạn.

**Releases:** là cách đóng gói và cung cấp phần mềm của GitHub cho người dùng của bạn. Bạn có thể nghĩ về nó như một sự thay thế cho việc sử dụng các phần tải xuống để cung cấp phần mềm

**“Branch: Master”:** Nhánh làm việc và hiển thị trên màn hình làm việc với Projects.




**Create new file:** tạo file mới.

**Upload files:** tải files lên.

**Find file:** tìm kiếm file sẵn có trên Projects.

**Clone or download:** dùng để tải mã nguồn từ trên github (của công ty/ của bạn bè/ người khác) về máy tính, tải về máy bằng ZIP hoặc dùng chức năng `git clone` của git scm. Đây là chức năng thường được các coder bấm vào đầu tiên :D

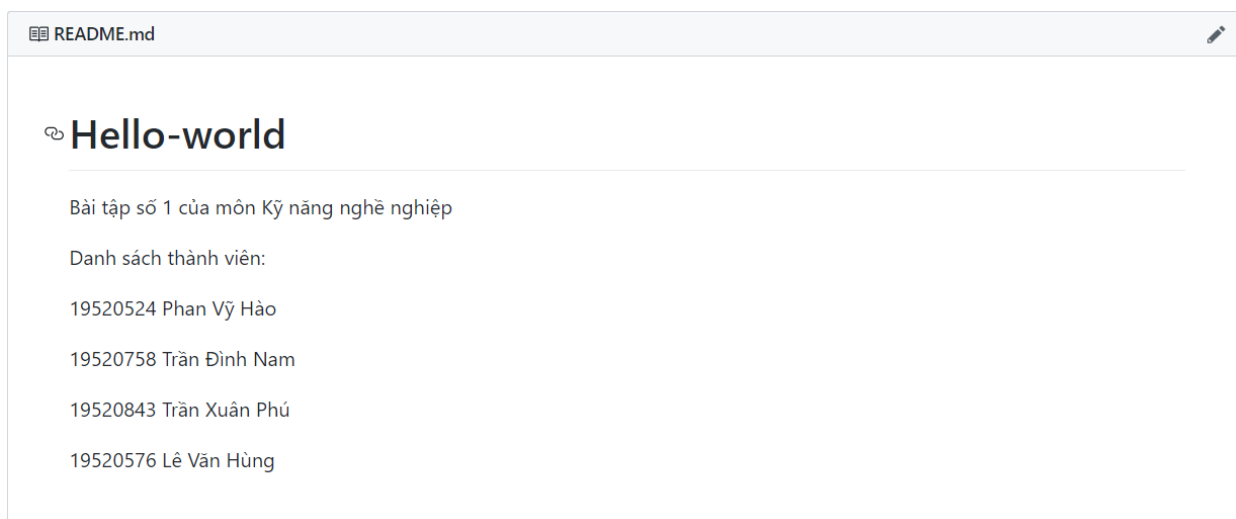
### ***Phần 3:***

vyhaophan Update Readme		Latest commit ce65156 13 seconds ago
 <a href="#">.gitignore</a>	Initial commit	yesterday
 <a href="#">Các tập lệnh cơ bản của Git.docx</a>	sua lai origin thanh Hello-world	35 minutes ago
 <a href="#">README.md</a>	Update Readme	13 seconds ago

Hình ảnh 4 Các tập tin trong repo

Lịch sử các lần update or commit lên Project. Hỗ trợ người sử dụng biết được thành viên trong nhóm đã develop và commit những gì ( thời gian, lịch sử commit, mã commit).

## ***Phần 4:***



*Hình ảnh 5 File Readme*

File này dùng để giới thiệu về dự án, là thứ để người khác đọc để hiểu bạn đang làm gì với dự án trên repo của bạn.



## **Chương 2. Các lệnh được dùng trong Github**

### ***Thiết lập chứng thực cá nhân***

```
git config --global user.name "user Name"
```

```
git config --global user.email "username@gmail.com"
```

Lưu ý:

--global được sử dụng để áp dụng cho tất cả các projects. Nếu không sử dụng --global thì settings sẽ chỉ dùng cho riêng project đó.

### ***Tạo một kho chứa Git***

```
git init
```

Lệnh này sẽ tạo nên một thư mục mới có tên là **.git**, thư mục **.git** này sẽ chứa mọi thiết lập về Git cũng như toàn bộ thông tin về kho chứa. Thư mục **.git** này sẽ cho phép bạn theo dõi dự án của bạn trong Git.

### ***Sao chép một kho chứa đã tồn tại***

```
git clone https://github.com/user/repository.git
```

Lệnh này sẽ tạo ra một thư mục mới có tên giống trên của repo.

### ***Nhánh trong git***

Ta có thể tạo ra nhiều nhánh (branch) khác nhau khi sử dụng Git:

Câu lệnh tạo mới một branch:

```
git branch <name_branch>
```

Câu lệnh chuyển và tạo mới:

```
git branch -b <name_branch>
```

Câu lệnh dùng để kiểm tra branch hiện tại:

```
git branch
```

## ***Chuyển nhánh***

Bạn cần phải check out một nhánh trước khi thay đổi source code. Để checkout một nhánh, có thể dùng câu lệnh:

```
git checkout <name_branch>
```

## ***Cập nhật thay đổi***

Sau khi bạn thực hiện các thao tác thay đổi khác đi so với ban đầu: thêm mới, sửa, xoá files,... Bạn cần phải cập nhật lên Staging Area( khu vực trung gian giữa máy tính của bạn và Local Repository). Câu lệnh được dùng khi muốn cập nhật files:

```
git add . (Cập nhật hết tất cả)
```

```
git add <file_name> (Cập nhật một file cụ thể)
```

Bạn còn có thể loại bỏ một file ra khỏi Staging Area để file đó không bị commit theo thì sử dụng câu lệnh:

```
git reset HEAD <file name>
```

Sau khi file đã được cập nhật lên Staging Area, nếu muốn đưa thông tin thay đổi lên Local Repository thì sử dụng câu lệnh:

```
git commit -m "Message"
```

Note: "Message" là ghi chú những thay đổi của bạn khi bạn cập nhật file để người khác có thể biết bạn đã sửa đổi những gì.

## ***Cập nhật lên server***

Sau khi những thông tin lên Local Repository bằng lệnh commit, nếu muốn cập nhật lên trên server chung thì phải sử dụng câu lệnh:

```
git push Hello-world <name_branch> (Hello-world là tên của repo)
```

Nếu chưa tồn tại remote trên server thì bạn cần phải thực hiện một bước nữa trước khi push lên server, bước này là bước add một remote mới vào server:

```
git remote add Hello-world <remote_url>
```

Sau khi add remote thì ta có thể push lên server:

```
git push Hello-world <name_branch>
```

## ***Gộp nhánh***

Sau một thời gian cập nhật các file và push lên git trên branch mới, bây giờ mình cần ghép (merge) code lại vào nhánh gốc (master). Trước tiên, cần phải checkout ra khỏi branch hiện tại cần gộp để vào branch master, sau đó thì dùng lệnh merge để ghép branch mới vào master:

Sau khi cập nhật các file và push lên một nhánh (branch) mới, bước tiếp theo là ghép (merge) các code lại vào nhánh gốc (master).

Đầu tiên ta phải check out ra khỏi branch cần gộp vào master:

```
git checkout master
```

Sau đó gộp branch mới vào master bằng câu lệnh:

```
git merge <new_branch>
```

## ***Xem lại lịch sử commit***

Nếu bạn muốn biết những thông tin( người commit, ngày giờ, lời nhắn) của những lần commit đó thì bạn có thể sử dụng câu lệnh:

```
git log
```

Bạn có thể thêm tham số -p vào để hiện chi tiết thông tin mỗi lần commit:

```
git log -p
```

Ngoài ra bạn có thể sử dụng thêm một số tùy chọn xem log khác để quá trình đọc log được tối ưu hơn:

`--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.

## **Xem thay đổi trước khi push**

Nếu bạn muốn biết có những thay đổi nào xảy ra giữa branch hiện tại và branch trước đó thì có thể sử dụng câu lệnh:

```
git diff
```

## ***Gộp commit***

```
git rebase -i HEAD~
```

 (Sau dấu ~ là số commit bạn muốn gộp)

Một cửa sổ trình soạn thảo sẽ hiện ra sau khi bạn gõ câu lệnh. Thay đổi ký tự pick của dòng các dòng sau dòng đầu thành s rồi lưu lại/kết thúc. Khi đó, trình soạn thảo để chỉnh sửa giải thích commit thiết lập cho commit sau khi đã tổng hợp sẽ được hiển thị, nên hãy chỉnh sửa lưu lại/kết thúc.

## ***Pull từ remote repository***

Khi bạn muốn gộp những thay đổi mới trên server với nhánh hiện tại trên máy tính bạn thì sử dụng câu lệnh

```
git pull Hello-world master
```