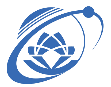
****

**Trường đại học Công nghệ thông tin, ĐHQG-HCM**

**🙞🕮🙜**

**BÀI BÁO CÁO**

Môn học: Kỹ năng nghề nghiệp

Lớp: SS004.K23-VN

**Đề tài:**

**TÌM HIỂU VỀ GIT, GITHUB**

**VÀ MỘT SỐ VẤN ĐỀ LIÊN QUAN**



**Nhóm 11**

**Thành viên nhóm:**

1. Thái Minh Triết (nhóm trưởng)

MSSV: 19522397

1. Bùi Thắng Lợi

MSSV: 19521768

1. Chu Hà Thảo Ngân

MSSV: 19521882

1. Tiêu Kim Hảo

MSSV: 19521480

**🙛Năm học: 2019 - 2020🙙**

Mục lục

[Chương I. Giới thiệu về Git, Github và giao diện 4](#_Toc35181651)

[1. Giới thiệu về Git 4](#_Toc35181652)

[a. Git là gì ? 4](#_Toc35181653)

[b. Một số khái niệm cơ bản 4](#_Toc35181654)

[c. Lịch sử phát triển của Git 6](#_Toc35181655)

[2. Giới thiệu về Github 8](#_Toc35181656)

[a. Vài nét về Github 8](#_Toc35181657)

[b. Các tính năng thuộc phạm vi của Github 9](#_Toc35181658)

[c. Làm quen với giao diện cơ bản của Github 9](#_Toc35181659)

[Chương II. Các tập lệnh cơ bản trong Git 13](#_Toc35181660)

[1. Các tập lệnh thiết lập Git 13](#_Toc35181661)

[2. Các thao tác cơ bản 13](#_Toc35181662)

[3. Các thao tác trên Branch 14](#_Toc35181663)

[4. Các thao tác trên remote repository 15](#_Toc35181664)

[Chương III. Vai trò của Git và Githhub trong công việc 16](#_Toc35181665)

[Chương IV. 20 mẹo để làm chủ Git và GitHub 16](#_Toc35181666)

Danh mục hình ảnh

[Hình 1: Logo của Git 4](#_Toc35181637)

[Hình 2: Quy trình công việc trên Remote Repository và Local Repository 4](#_Toc35181638)

[Hình 3: Sơ đồ object trong thư mục Git 5](#_Toc35181639)

[Hình 4: Logo các website và phần mêm lưu trữ Git. 6](#_Toc35181640)

[Hình 5: Lịch sử các phiên bản của Git (tính đến 24/2/2019) 7](#_Toc35181641)

[Hình 6: Logo của Github 8](#_Toc35181642)

[Hình 7: Bản đồ số lượng người dùng Github trên thế giới 8](#_Toc35181643)

[Hình 8: Giao diện cơ bản của Github 9](#_Toc35181644)

[Hình 9: Thanh tab Github 10](#_Toc35181645)

[Hình 10: Thanh mô tả và topic của Repository 11](#_Toc35181646)

[Hình 11: Thanh commits, branch, packages, releases, contributors 11](#_Toc35181647)

[Hình 12: Thanh tác vụ Github 12](#_Toc35181648)

[Hình 13: Bảng quản lý các file trong repository 12](#_Toc35181649)

[Hình 14: Bảng README.md 12](#_Toc35181650)

1. Giới thiệu về Git, Github và giao diện
   1. Giới thiệu về Git
      1. Git là gì ?



Hình 1: Logo của Git

**Git** là hệ thống quản lý phiên bản phân tán (distributed version control system) được phát triển bởi Linus Torvalds vào năm 2005. Nhờ Git, việc quản lý code và làm việc nhóm của các developer trở nên đơn giản, thuận tiện hơn. Hiện nay, Git trở thành một trong các phần mềm quản lý mã nguồn phổ biến nhất và được sử dụng rộng rãi trong các công ty công nghệ thông tin.

**Địa chỉ website**: http://git-scm.com/

* + 1. Một số khái niệm cơ bản
* **Repository (hay repo):** là nơi chứa cơ sở dữ liệu (database) của tất cả những thông tin cần thiết để duy trì và quản lý các sửa đổi và lịch sử của dự án.   
  Có 2 loại repository:
* **Local repository**: Là repository được cài đặt trên máy tính của lập trình viên, repo này sẽ đồng bộ hóa với remote repo bằng các lệnh của git.
* **Remote repository**: Là repo được cài đặt trên server chuyên dụng, điển hình hiện nay là Github.

Hình 2: Quy trình công việc trên Remote Repository và Local Repository

* **Object store**: là trái tim của Git, nó chứa dữ liệu nguyên gốc (original data files), các file log ghi chép quá trình sửa đổi, tên người tạo file, ngày tháng và các thông tin khác. Git có bốn loại object là:
* **Blobs**: là file nhị phân có thể chứa được mọi loại dữ liệu bất kể là dữ liệu của chương trình gì.
* **Trees**: lớp đại diện cho thông tin thư mục như thông tin định danh của blob, đường dẫn, chứa một ít metadata chứa thông tin cấu trúc và các thư mục nhỏ có trong thư mục đó.
* **Commits:** Chứa metadata có thông tin về mọi thứ như tên tác giả, người tải lên (commiter), ngày tải lên, thông tin log...
* A close up of a sign

  Description automatically generated**Tags:** gán nhãn cho dễ nhớ và dễ sử dụng.

Hình 3: Sơ đồ object trong thư mục Git

* **Index:** là file nhị phân động và tạm thời miêu tả cấu trúc thư mục của toàn bộ Repo và trạng thái của dự án được thể hiện thông qua commit và tree tại một thời điểm nào đó trong lịch sử phát triển. Do đó Git còn là một hệ thống truy tìm nội dung (content tracking system).

Index không chứa nội dung file mà chỉ dùng để truy tìm (track) những thứ mà người dùng muốn commit.

* **Branch**: Branch được tạo ra dùng để phân nhánh và ghi lại luồng của lịch sử. Branch đã phân nhánh sẽ không ảnh hưởng đến branch khác nên có thể tiến hành nhiều thay đổi đồng thời trong cùng 1 repository. Các branch sau khi chỉnh sửa có thể tổng hợp lại thành 1 branch bằng cách sử dụng câu lệnh merge.
* **Xung đột (conflict):** Xung đột xảy ra khi có 2 sự thay đổi trong một dòng code và máy tính không thể tự quyết định dòng code nào là “đúng”. Đúng ở đây có nghĩa là “ý đồ của lập trình viên”.
* **Commit:** Để ghi lại việc thêm/thay đổi file hay thư mục vào repository thì sẽ thực hiện thao tác gọi là Commit. Khi thực hiện commit, trong repository sẽ tạo ra commit (hoặc revision) đã ghi lại sự khác biệt từ trạng thái đã commit lần trước với trạng thái hiện tại.

Commit đang được chứa trong repository, các commit nối tiếp với nhau theo thứ tự thời gian. Bằng việc lần theo commit này từ trạng thái mới nhất thì có thể biết được lịch sử thay đổi trong quá khứ hoặc nội dung thay đổi đó.

* + 1. Lịch sử phát triển của Git

Phiên bản mới nhất của Git là phiên bản 2.25.1, được cập nhật vào ngày 17/2/2020.

Các hệ điều hành được hỗ trợ cài đặt Git gồm có: Windows, Linux, Unix, macOS, ...

Hiện nay đã có nhiều website và phần mềm cho phép lưu trữ (miễn phí và có phí) Git: [BerliOS](https://vi.wikipedia.org/w/index.php?title=BerliOS&action=edit&redlink=1), [GitHub](https://vi.wikipedia.org/wiki/GitHub), [GitLab](https://vi.wikipedia.org/w/index.php?title=GitLab&action=edit&redlink=1), , [Sourceforge](https://vi.wikipedia.org/wiki/SourceForge), [GNU Savannah](https://vi.wikipedia.org/w/index.php?title=GNU_Savannah&action=edit&redlink=1), [GitEnterprise](https://vi.wikipedia.org/w/index.php?title=GitEnterprise&action=edit&redlink=1).



Hình 4: Logo các website và phần mêm lưu trữ Git.

A screenshot of a cell phone

Description automatically generated

Hình 5: Lịch sử các phiên bản của Git (tính đến 24/2/2019)

* 1. Giới thiệu về Github

Hình 6: Logo của Github

**Website**: github.com.

* + 1. Vài nét về Github

**GitHub** là một dịch vụ cung cấp [kho lưu trữ mã nguồn](https://vi.wikipedia.org/w/index.php?title=Kho_l%C6%B0u_tr%E1%BB%AF_m%C3%A3_ngu%E1%BB%93n&action=edit&redlink=1) [Git](https://vi.wikipedia.org/wiki/Git_(ph%E1%BA%A7n_m%E1%BB%81m)) dựa trên nền web cho các dự án phát triển phần mềm. GitHub cung cấp cả phiên bản trả tiền lẫn miễn phí cho các tài khoản. Các dự án [mã nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_ngu%E1%BB%93n_m%E1%BB%9F) sẽ được cung cấp kho lưu trữ miễn phí.

Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Nhiều nhà phát triển đã bắt đầu xem Github là một sự thay thế cho sơ yếu lý lịch khi mà một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên.

Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn 35 triệu kho mã nguồn, làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới.

A picture containing snow, map, sky, text

Description automatically generated

Hình 7: Bản đồ số lượng người dùng Github trên thế giới

* + 1. Các tính năng thuộc phạm vi của Github

GitHub chủ yếu được sử dụng để lưu trữ mã nguồn phần mềm, nhưng cũng thường được sử dụng với nhiều loại tập tin như Final Cut hoặc các tài liệu Word.

Ngoài mã nguồn, thì Gihub còn hỗ trợ các định dạng và tính năng sau:

* Track (truy xuất) Issue
* Xem trước và so sánh định dạng file PSD gốc của Photoshop với các phiên bản trước đó
* Lồng nhiệm vụ đa danh sách
* Tài liệu và wiki
* Code Snippets (bằng cách sử dụng tên miền phụ Gist)
* Tích hợp thư mục
* Emojis
* Trực quan hóa dữ liệu thông tin địa lý
* 3D render files
* Biểu đồ Gantt
* …
  + 1. Làm quen với giao diện cơ bản của Github



Hình 8: Giao diện cơ bản của Github



Hình 9: Thanh tab Github

* Tab **Code**: chứa các tính năng cơ bản quan trọng trong project như: tạo file mới, tạo branch, các thao tác trên file và thư mục,…..
* Tab **Issues**: Issues giúp cho người dùng truy xuất theo dõi các lỗi, những cải tiến hay những yêu cầu khác.
* Tab **Pull requests:** Pull request giúp ta thông báo cho người khác về những thay đổi mà mình đã đẩy (push) lên repository. Khi một pull request được tạo, những người đóng góp sẽ xem được những thay đổi, thảo luận về những sửa đổi và có thể đẩy lên những commit tiếp theo nếu cần thiết.
* Tab **Actions:** Github Actions giúp cho quy trình công việc của phần mềm diễn ra tự động một cách đơn giản. Giúp người dùng đánh giá các code, quản lý branch và xử lý các issue theo ý muốn của bản thân.
* Tab **Projects:** Bảng dự án có thể được sử dụng để tùy chỉnh quy trình công việc đề phù hợp với mục đích của người sử dụng.
* Tab **Wiki:** Ta có thể tổ chức,bố trií các file tài liệu dưới dạng wiki, nhờ đó những người khác có thể sử dụng và đóng góp vào project.
* Tab **Security:** người tạo repository có thể thực hiện các thao tác giúp tăng tính bảo mật cho các file lưu trữ. Github security có các tính năng: tạo cảnh báo, tạo khuyến cáo bảo mật và tạo điều khoản bảo mật.
* Tab **Insights:** Insight thể hiện dữ liệu các hoạt động của trên repository như commit, pulse, network, … dưới dạng biểu đồ trực quan, chi tiết, sinh động.
* Tab **Settings:** Đóng vai trò quan trọng trong việc quản lý các hoạt động trong project. Settings bao gồm các tính năng: tùy chọn cài đặt, cấp quyền truy cập, quản lý branch, webhooks, cài đặt thông báo, các công cụ tích hợp và dịch vụ, deploy keys, nội dung bảo mật, cấp quyền hành động.



Hình 10: Thanh mô tả và topic của Repository

Mô tả giúp người dùng hiểu rõ mục đích của Repository.

Topics phân loại các chủ đề của Repository và giúp nó được dễ dàng tìm kiếm, khám phá.



Hình 11: Thanh commits, branch, packages, releases, contributors

* **Commits:** ghi lại chi tiết lịch sử các lần chỉnh sửa/thêm/xóa file hoặc thư mục trong repository.
* **Branch**: thống kê các branch được sử dụng trong repository
* **Packages:** Github packages là dịch vụ tổ chức các gói phần mềm, nó cho phép người dùng tổ chức và quản lý các gói phần mềm riêng tư hoặc công cộng.
* **Releases**: Git được thiết kế giúp hỗ trợ chức năng tạo tag và releases.
* **Tag**: Tag là chức năng đặt tên một cách đơn giản của Git, nó cho phép ta xác định một cách rõ ràng các phiên bản mã nguồn (code) của dự án. Ta có thể coi tag như một branch không thay đổi được. Một khi nó được tạo (gắn với 1 commit cụ thể) thì ta không thể thay đổi lịch sử commit ấy được nữa.
* **Releases:** Mục đích tạo release là để chia sẻ đóng gói ứng dụng, cùng các ghi chú phát hành và các link tới các file tài liệu ứng dụng cho mọi người trong team, công ty có sửa dụng.

Release dựa trên Git tag, nó đánh dấu một điểm cụ thể ở lịch sử repository của ta. Các release được sắp xếp theo thời gian chúng được tạo trên GitHub.

* **Cotributors:** thống kế số lượng người đóng góp và các lượt đóng góp của từng thành viên cho project dưới dạng biểu đồ.



Hình 12: Thanh tác vụ Github

* ***Branh: master*:** cho phép người dùng chuyển đổi giữa các branch hoặc tạo một branch mới. Tên mặc định của branch khi vừa tạo repository là *master.*
* ***New pull request*:** cho phép tạo một pull request mới từ branch đã chọn.
* ***Create new file*:** tạo một file mới trong một repository.
* ***Upload file*:** tải lên một file mới từ máy tính cá nhân.
* ***Find file*:** tìm kiếm tên các file trong repository.
* ***Clone or download*:** cho phép người dùng tạo bản sao của repository hoặc download về máy dưới dạng file .zip.



Hình 13: Bảng quản lý các file trong repository

Lưu trữ các thư mục hoặc file trong repository, cho phép các contributor chỉnh sửa (file text), xem lịch sử chỉnh sửa của file hoặc tải về.

Bảng gồm 3 cột: tên file/thư mục, mô tả hành động commit, thời gian lần sửa đổi cuối.



Hình 14: Bảng README.md

Để giúp cộng đồng hiểu rõ hơn về nội dung, mục đích của project, các contributor có thể thêm mô tả công việc và nội dung chi tiết ở file README.md trong repository Github. Sau khi được chỉnh sửa, toàn bộ nội dung sẽ được cập lên giao diện chung của project.

1. Các tập lệnh cơ bản trong Git
   1. Các tập lệnh thiết lập Git

* **Thiết lập tên người dùng / địa chỉ email**

$ git config --global user.name <username>

$ git config --global user.email <mailaddress>

* 1. Các thao tác cơ bản
* **Tạo repository**

$ git init

* **Sao chép remote repository**

$ git clone <url>

* **Hiển thị danh sách của file đã được chỉnh sửa**

$ git status

* **Thêm file vào index**

$ git add <file>

* **Thêm tất cả các file**

$ git add .

* **Commit những thay đổi**

$ git commit -m “<commit message>”

* **Xoá file / thư mục**

$ git rm <file>

* **Thay đổi tên file / thư mục**

$ git mv <oldfilename> <newfilename>

* **Hiển thị danh sách commit**

$ git log

* **Xem thay đổi của file hiện tại**

$ git diff

* 1. Các thao tác trên Branch
* **Hiển thị danh sách branch**

$ git branch

* **Hiển thị danh sách local và remote branch**

$ git branch -a

* **Tạo branch**

$ git branch <branch name>

* **Xoá branch**

$ git branch -d <branch name>

* **Xoá remote branch**

$ git push origin --delete <branch name>

* **Thay đổi tên branch**

$ git branch -m <oldbranch> <newbranch>

* **Chuyển sang một branch (switch to a branch)**

$ git checkout <branch name>

* **Tạo branch mới và chuyển sang branch đó**

$ git checkout -b <branch name>

* **Sao chép remote branch và chuyển sang remote branch đó**

$ git checkout -b <branch name> origin/<branch name>

* **Merge một branch vào branch đang sử dụng**

$ git merge <branch name>

Ví dụ: Muốn merge issue01 vào branch master

Trước đó ta switch vào branch master

$ git checkout master

$ git merge issue01

* **Merge một branch vào target branch**

$ git merge <source branch> <target branch>

* 1. Các thao tác trên remote repository
* **Thêm remote**

$ git remote add <name> <url>

* **Xoá remote**

$ git remote rm <remote\_name>

* **Đổi tên remote**

$ git remote rename <old\_remotename> <new\_remotename>

* **Hiển thị danh sách remote repository**

$ git remote -v

* **Thay đổi remote’s URL từ SSH sang HTTPS**

$ git remote set-url origin https://github.com/USERNAME/REPOSITORY.git

* **Thay đổi remote’s URL từ HTTPS sang SSH**

$ git remote set-url origin git@github.com:USERNAME/REPOSITORY.git

* **Push branch lên remote repository**

$ git push origin <branch name>

* **Xoá remote branch**

$ git push origin --delete <branch name>

* **Push – Chia sẻ những thay đổi lên remote**

$ git push

$ git push -u origin <branch name>

* **Fetch – Lấy những thay đổi trên remote về local repo (không merge vào source code)**

$ git fetch origin <branch name>

* **Pull - Lấy những thay đổi trên remote về local repo (đồng thời merge vào source code)**

$ git pull origin <branch name>

1. Vai trò của Git và Githhub trong công việc

GitHub cung cấp [kho lưu trữ mã nguồn](https://vi.wikipedia.org/w/index.php?title=Kho_l%C6%B0u_tr%E1%BB%AF_m%C3%A3_ngu%E1%BB%93n&action=edit&redlink=1) [Git](https://vi.wikipedia.org/wiki/Git_(ph%E1%BA%A7n_m%E1%BB%81m)) dựa trên nền web cho các dự án phát triển phần mềm. Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Hiện nay, hầu như các dự án lớn hay nhỏ người ta đều sử dụng GitHub. Thậm chí nhiều nhà phát triển đã bắt đầu xem nó là một sự thay thế cho sơ yếu lý lịch và một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên.

Github cho phép người dùng có thể thảo luận, quản lý, tạo ra các kho, đóng góp cho kho của người dùng khác, và xem lịch sử thay đổi mã, đặc biệt với sự ra đời của phần mềm quản lý mã nguồn Git. Điểm khác biệt lớn nhất của Git với các phần mềm khác là nó có khả năng tách nhánh (branch). Nhờ vậy, Git hỗ trợ rất tốt cho teamwork, những việc như phân chia task, tổng hợp code trở nên dễ dàng,… giúp người dùng tiết kiệm thời gian và tối ưu hoá công việc.  
 Ví dụ, cùng một lúc, chúng ta thường có một team làm tính năng mới, một vài team khác nâng cấp các tính năng hiện có, và một người fix bug.

Nếu không dùng Git, khả năng cao là mọi người sẽ làm việc giẫm chân nhau, những task sắp hoàn thành sẽ bị trì hoãn. Ngoài ra, tất cả mọi task lớn nhỏ sẽ buộc phải hoàn thành hết trước khi deploy, bởi vì chỉ cần 1 task vẫn đang dang dở, cả phần mềm có thể bị sập.

1. 20 mẹo để làm chủ Git và GitHub

* **Clone gần như bất cứ thứ gì**
* **Pull thường xuyên**
* **Commit sớm và thường xuyên**
* **Tự nhận xét các commit của bạn một cách khách quan**
* **Chỉ push khi những thay đổi của bạn đã được test**
* **Branch tự do**
* **Hợp nhất cẩn thận**
* **Stash trước khi thay đổi branch**
* **Sử dụng gists để chia sẻ snippet và paste**
* **Khám phá GitHub**
* **Đóng góp cho các dự án mã nguồn mở**
* **Sử dụng editor và IDE có tích hợp Git**
* **Tạo nhánh mới trên repository**
* **Theo dõi các dự án**
* **Theo dõi bạn bè**
* **Gửi pull request**
* **Tạo và giải quyết các vấn đề**
* **Chuyển các repo cũ lên Git**
* **Sử dụng Markdown**
* **Luyện tập viết README cho các dự án**

(Nguồn: Techtalk - 20 mẹo để làm chủ Git và GitHub)