ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO

MÔN HỌC: CÁC THÀNH PHẦN PHẦN MỀM

NỘI DUNG: Tìm hiểu về git và github

Họ và tên **: Nguyễn Thị Thắng**

Mã sinh viên **: 21002174**

Lớp **: K66A5 – Khoa học dữ liệu**

Năm học 2022-2023

MỤC LỤC

I. Git

1. Git là gì?

Git là một hệ thống kiểm soát phân tán mã nguồn phân tán (Open Source Distributed Version Control System/ hay DVCS). Các VCS là những hệ thống ghi lại những thay đổi của file xuyên suốt dự án và mỗi sự thay đổi thì sẽ được lưu trữ thành một phiên bản. VCS cũng là hệ thống cho phép các lập trình viên có thể lưu trữ nhiều phiên bản của các mã nguồn được nhân bản **(clone)** từ kho chứa các mã nguồn **(repository).**

Nói một cách đơn giản, hệ thống này sẽ quản lý các mã nguồn trong dự án của bạn. Trong một dự án thường sẽ có nhiều dev cùng làm việc với nhau, do đó Git tồn tại để giúp các lệnh của người này không xung đột với lệnh của người khác. Hơn nữa, trong quá trình hình thành và phát triển sản phẩm, các yêu cầu đầu vào có thể thay đổi và chúng ta sẽ phải tìm về các phiên bản cũ của code để tiến hành chỉnh sửa.

2. Ưu điểm của git

- **Quản lý code và lịch sử thay đổi**: Hãy tưởng tượng nếu bạn phát triển một tính năng mới và tạo ra một đống bug. Lúc này bạn hối hận và muốn quay đầu. Nếu không có hệ thống này, bạn sẽ phải quay lại từ đầu để dò bug và chỉnh lại cho đúng. Git sẽ giúp bạn hoàn thành những công đoạn đó chỉ với một vài dòng lệnh.
- **Tiết kiệm thời gian**: Ngoài tính năng giúp lập trình viên chỉnh sửa thay đổi dễ dàng, hệ thống này còn đóng vai trò quan trọng trong phát triển tính năng mới. Khi phát triển tính năng mới, bạn có thể copy mã nguồn gốc để làm backup lỡ tính năng không được như mong muốn. Tuy nhiên, nếu bạn muốn phát triển thêm 2-3 tính năng nữa thì việc này chắc chắn sẽ rất mất thời gian. Do đó lúc này bạn sẽ cần đến Git để tối giản hoá quá trình này
- **Giúp làm việc nhóm hiệu quả hơn**: Một dự án sẽ bao gồm nhiều người đến từ nhiều bộ phận khác nhau. Do đó, việc dẫm lên chân nhau khi thực hiện dự án là hoàn toàn có thể xảy ra. Hệ thống này sẽ đóng vai trò quản lý hệ thống và giúp giảm thiểu mâu thuẫn giữa các bộ phận

3.Các thành phần của git

Git bao gồm hai thành phần: Repository và Branch

Repository

Nói một cách dễ hiểu thì repository là một kho chứa các mã của dự án được tạo ra bởi Git, và thường được gọi tắt là Repo. Mỗi một Repo sẽ có hai cấu trúc dữ liệu chính là Object Store và Index, được lưu trữ ẩn trong một file có đuôi .git.
Thông thường, có hai loại Repo phổ biến nhất như sau:

- Local Repo: Đây là loại Repo sẽ được lắp đặt trực tiếp trên máy tính của lập trình viên và sẽ được đồng bộ hoá với Remote thông qua lệnh của Git.
- Remote Repo: Với Remote Repo, các kho sẽ được lưu trữ trực tuyến trên các server chuyên dụng, điển hình như GitHub, BitStacker, v.v..

Branch

Trong một dự án, Leader sẽ là người phụ trách giao việc cho các thành viên trong team. Tuy nhiên, việc tiến hành các công việc này cùng 1 lúc là điều không thể vì chắc chắn sẽ có hiện tượng đụng code. Do đó, tính năng Branch có trong Git cho phép các leader có thể chia riêng mỗi nhiệm vụ một branch riêng. Các branch này hoạt động riêng lẻ và không ảnh hưởng lẫn nhau, đồng thời chúng cũng ghi lại những sửa đổi trong cùng một khoảng thời gian, cực kỳ hữu dụng khi chúng ta triển khai nhiều nhiệm vụ cùng 1 lúc.

Ngoài ra, khi bạn tạo một Repo ở trên Git thì hệ thống này sẽ mặc định tạo cho bạn 1 branch master và lúc này mọi thay đổi của bạn sẽ được ghi trên branch master đó.

4. Một số khái niệm cần biết

- **Commit**

Một commit đại diện cho một thời điểm cụ thể trong lịch sử dự án của bạn. Sử dụng lệnh commit kết hợp với lệnh **git add** để cho git biết những thay đổi bạn muốn lưu vào local repository.

- **Checkout**

Sử dụng lệnh `git checkout` để chuyển giữa các branch. Chỉ cần nhập git checkout theo sau là tên của branch bạn muốn chuyển đến hoặc nhập git checkout master để trở về branch chính (master branch).

- **Fetch**

Lệnh `git fetch` tìm nạp các bản sao và tải xuống tất cả các tệp branch vào máy tính của bạn. Sử dụng nó để lưu các thay đổi mới nhất vào kho lưu trữ của bạn. Nó có thể tìm nạp nhiều branch cùng một lúc.

- **Fork**

Một fork là một bản sao của một kho lưu trữ (repository). Các lập trình viên thường tận dụng lợi ích của fork để thử nghiệm các thay đổi mà không ảnh hưởng đến dự án chính.

- **Head**

Các commit ở đầu của một branch được gọi là head. Nó đại diện cho commit mới nhất của repository mà bạn hiện đang làm việc.

- **Index**

Bất cứ khi nào bạn thêm, xóa hoặc thay đổi một file, nó vẫn nằm trong chỉ mục cho đến khi bạn sẵn sàng commit các thay đổi. Nó như là khu vực tổ chức (stagging area) cho Git. Sử dụng lệnh `git status` để xem nội dung của index của bạn.

- **Master**

Master là nhánh chính của tất cả các repository của bạn. Nó nên bao gồm những thay đổi và commit gần đây nhất.

- **Merge**

Lệnh `git merge` kết hợp với các yêu cầu kéo (pull requests) để thêm các thay đổi từ nhánh này sang nhánh khác.

- **Origin**

Origin là phiên bản mặc định của repository. Origin cũng đóng vai trò là bí danh hệ thống để liên lạc với nhánh chính.

Lệnh `git push origin master` để đẩy các thay đổi cục bộ đến nhánh chính.

- **Pull**

Pull requests thể hiện các đề xuất thay đổi cho nhánh chính. Nếu bạn làm việc với một nhóm, bạn có thể tạo các pull request để yêu cầu người bảo trì kho lưu trữ xem xét các thay đổi và hợp nhất chúng.

Lệnh `git pull` được sử dụng để thêm các thay đổi vào nhánh chính.

- **Push**

Lệnh `git push` được sử dụng để cập nhật các nhánh từ xa với những thay đổi mới nhất mà bạn đã `commit`.

- **Remote**

Một Remote (kho lưu trữ từ xa) là một bản sao của một chi nhánh. Remote giao tiếp ngược dòng với nhánh gốc (origin branch) của chúng và các Remote khác trong kho lưu trữ.

5. Cơ chế hoạt động của git

Sự khác biệt chính giữa Git và bất kỳ VCS nào khác (bao gồm Subversion…) là cách Git nghĩ về dữ liệu của nó.

Về mặt khái niệm, hầu hết các hệ thống khác đều lưu trữ thông tin dưới dạng danh sách các thay đổi dựa trên file. Các hệ thống này (CVS, Subversion, Perforce, Bazaar, v.v.) coi thông tin chúng lưu giữ dưới dạng một tập hợp các file và những thay đổi được thực hiện đối với mỗi file theo thời gian.

Git không nghĩ đến hoặc lưu trữ dữ liệu của mình theo cách này. Thay vào đó, Git coi thông tin được lưu trữ là một tập hợp các snapshot – ảnh chụp toàn bộ nội dung tất cả các file tại thời điểm.

Mỗi khi bạn "commit", Git sẽ "chụp" và tạo ra một snapshot cùng một tham chiếu tới snapshot đó. Để hiệu quả, nếu các tệp không thay đổi, Git sẽ không lưu trữ lại file — chỉ là một liên kết đến tệp giống file trước đó mà nó đã lưu trữ.

Đây là điểm khác biệt quan trọng giữa Git và gần như tất cả các VCS khác. Nó khiến Git phải xem xét lại hầu hết mọi khía cạnh của kiểm soát phiên bản mà hầu hết các hệ thống khác đã sao chép từ thế hệ trước. Điều này làm cho Git giống như một hệ thống tệp nhỏ với một số công cụ cực kỳ mạnh mẽ được xây dựng trên nó, thay vì chỉ đơn giản là một VCS.

6. Các lệnh git cơ bản

- **git config**

Tác dụng : Để set user name và email của bạn trong main configuration file.
Cách xài : Để kiểm tra tên và kiểu email trong cấu hình dùng
git config--global user.name và ***git config --global user.email***.
Để set email hoặc tên mới ***git config--global user.name = "Hải Nguyễn"*** và ***git config--global user.email = "hainguyen@gmail.com"***

- **git init**

Tác dụng : Khởi tạo 1 git repository 1 project mới hoặc đã có.
Cách dùng: ***git init*** trong thư mục gốc của dự án.

- **git clone**

Tác dụng: Copy 1 git repository từ remote source.
Cách dùng: ***git clone <:clone git url:>***

- **git status**

Tác dụng: Để check trạng thái của những file bạn đã thay đổi trong thư mục làm việc. VD: Tất cả các thay đổi cuối cùng từ lần commit cuối cùng.

Cách dùng: ***git status*** trong thư mục làm việc.

- **git add**

Tác dụng: Thêm thay đổi đến stage/index trong thư mục làm việc.

Cách dùng: ***git add***

- **git commit**

Tác dụng: commit nghĩa là một action để Git lưu lại một snapshot của các sự thay đổi trong thư mục làm việc. Và các tập tin, thư mục được thay đổi đã phải nằm trong Staging Area. Mỗi lần commit nó sẽ được lưu lại lịch sử chỉnh sửa của code kèm theo tên và địa chỉ email của người commit. Ngoài ra trong Git bạn cũng có thể khôi phục lại tập tin trong lịch sử commit của nó để chia cho một branch khác, vì vậy bạn sẽ dễ dàng khôi phục lại các thay đổi trước đó.

Cách dùng: ***git commit -m "Đây là message, bạn dùng để note những thay đổi để sau này dễ dò lại"***

- **git push/git pull**

Tác dụng: Push hoặc Pull các thay đổi đến remote. Nếu bạn đã added và committed các thay đổi và bạn muốn đẩy nó lên hoặc remote của bạn đã update và bạn apply tất cả thay đổi đó trên code của mình.

Cách dùng: ***git pull <:remote:> <:branch:>; git push <:remote:> <:branch:>***

- **git branch**

Tác dụng: liệt kê tất cả các branch (nhánh).

Cách dùng: ***git branch*** hoặc ***git branch -a***

- **git checkout**

Tác dụng: Chuyển sang branch khác

Cách dùng: ***git checkout <: branch:>*** hoặc ***** _ git checkout -b <: branch:>*** nếu bạn muốn tạo và chuyển sang một chi nhánh mới.

- **git stash**

Tác dụng: Lưu thay đổi mà bạn không muốn commit ngay lập tức.

Cách dùng: ***git stash*** trong thư mục làm việc của bạn.

- **git merge**

Tác dụng: Merge 2 branch lại với nahu.

Cách dùng: Chuyển tới branch bạn muốn merge rồi dùng ***git merge <:branch_ban_muon_merge:>***

- **git reset**

Tác dụng: Bạn đã đưa một tập tin nào đó vào Staging Area nhưng bây giờ bạn muốn loại bỏ nó ra khỏi đây để không phải bị commit theo.

Cách dùng: ***git reset HEAD tên_file***

- **git remote**

Tác dụng: Để check remote/source bạn có hoặc add thêm remote

Cách dùng: ***git remote*** để kiểm tra và liệt kê. Và ***git remote add <: remote_url:>*** để thêm.

- **git add**

Tác dụng: Để đưa một tập tin vào Staging Area

Cách dùng: ***git add tên_file*** hoặc muốn thêm hết file của thư mục thì ***git add all***

II. Github

1. Github là gì?

GitHub là một hệ thống quản lý dự án và phiên bản code, hoạt động giống như một mạng xã hội cho lập trình viên. Các lập trình viên có thể clone lại mã nguồn từ một repository và Github chính là một dịch vụ máy chủ repository công cộng, mỗi người có thể tạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có thể làm việc.

GitHub là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn Git cho các dự án phần mềm. **Github có đầy đủ những tính năng của Git**, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.

2. Lợi ích của github với lập trình viên

Quản lý source code dễ dàng

Khi bạn tạo một repo, toàn bộ source code của repo đó được lưu trên GitHub. Tại đây, bạn có thể coi lại quá trình mình đã làm việc thông qua các comment sau mỗi lần commit. Và cái hay ở đây, là nhiều người có thể cùng làm một repo.

Lợi ích đầu tiên, chính là bạn biết được ai đã commit và commit cái gì. Tiếp theo, source của bạn có thể phát triển theo nhiều nhánh. Nguyên tắc làm việc với các nhánh như thế này: Bạn có thể rẽ nhiều nhánh để phát triển project. Nhưng cuối cùng, bạn phải merge lại vào nhánh MASTER để ra được project hoàn chỉnh.

Tracking sự thay đổi qua các version

Khi có nhiều member cùng thực hiện một dự án thì khá là phức tạp để theo dõi revisons – ai thay đổi cái gì, lúc nào và mấy cái files đó được stored ở đâu. Đừng lo vì GitHub đã tính đến chuyện này giúp bạn, bằng cách luôn lưu lại những thay đổi bạn đã push lên repository. Cũng tương tự với Microsoft Word hay Google Drive, bạn có một lịch sử phiên bản để phòng trường hợp các phiên bản trước đó bị mất hay không được lưu.

Markdown

Markdown là một cách định dạng text trên web. Bạn có thể chỉnh sửa cách hiển thị của document, format từ như định dạng **in đậm** hay *in nghiêng*, thêm hình và tạo list những thứ bạn có thể làm với Markdown. Hầu hết, Markdown chỉ là đoạn text đơn thuần với những ký tự đặc biệt chèn vào, như # hay *. Trong GitHub thì bạn có thể sử dụng Mardown ở những nơi: Git, Comments tại Issues và Pull Requests, các file có đuôi .md hay .markdown extension.

Github giúp chứng tỏ bạn là ai

Chẳng thể phủ nhận những lời hay ý đẹp bạn viết trong CV là cần thiết. Nhưng Source code luôn là minh chứng tốt nhất để thể hiện bạn là developer thực thụ. Có thể nói, 1 phần GitHub "nho nhỏ" trong CV có thể đánh bóng vị trí của bạn, nổi bật hơn những ứng cử viên khác. Đối với nhà tuyển dụng, GitHub cũng giống như một chiếc máy liar-detech – phân biệt real developer với những kẻ "faker".

Hãy đầu tư cho mình một tài khoản Github thật ấn tượng và đưa đường dẫn vào trong CV, chẳng nhà tuyển dụng nào lại dại dột mà bỏ qua bạn đâu.

Có rất nhiều công ty lớn trên thế giới xem đây là một yêu cầu trong quy trình tuyển dụng của họ. Nếu bạn có nhiều đóng góp cho cộng đồng hoặc có nhiều sản phẩm trên Github, sẽ là một lợi thế rất lớn so với các ứng viên khác. Vì bằng cách đăng tải các project của mình lên đây, bạn đã tạo cho mình một profile cá nhân vô cùng đáng tin cậy.

Vì khi nhìn vào đó, nhà tuyển dụng sẽ biết được ngay thế mạnh của bạn là gì, và khả năng coding của bạn thế nào.

Github giúp cải thiện kỹ năng code, thậm chí là tracking bug

Có hàng ngàn hàng vạn cách để học, học trên Github sẽ là một ý kiến không tồi trong thời đại này. Với hàng vạn open source projects, hàng trăm ngàn contributors, hàng tỉ commit mỗi ngày thì chỉ bằng việc xem. So sánh, học tập từ những thay đổi đó đã đem lại cho bạn hàng tá điều hay để cải thiện kỹ năng code của bản thân mình.

"Bug tracking" là một tính năng được GitHub tích hợp vào để đơn giản hóa quá trình "tìm và diệt bọ". Để hiểu được quy trình thì những gì bạn cần làm là mở dashboard của từng project lên và filter các thông tin. Sau đó, các câu hỏi sẽ được

hệ thống, sắp xếp theo mức độ phổ biến, thời gian update hay tương tại. Phần mềm này cũng có giao diện khá mượt nên luôn được xếp hạng cao trong cộng đồng IT dev.

Github là một kho tài nguyên tuyệt vời

Với chức năng Explore, bạn có thể theo dõi, tìm kiếm những open source projects theo đúng technology pattern mà bạn ưa thích. Github hỗ trợ code search không kể nó ở dưới dạng một project riêng biệt hay là website. Ngoài ra, nền tảng này cũng có SEO khá tốt nên người dùng có thể tìm kiếm bất kỳ code string nào được chia sẻ public.