

Phát Triển Phần Mềm Mã Nguồn Mở

NGUYỄN HẢI TRIỀU

Bộ môn Kỹ thuật phần mềm
Khoa Công nghệ thông tin
ĐH Nha Trang

Chương 1: Tổng quan về phần mềm mã nguồn mở

- ① Phần mềm và vấn đề bản quyền phần mềm
- ② Mô hình phát triển phần mềm mã nguồn mở
 - Bài tập
- ③ Cách đóng góp vào phát triển phần mềm mã nguồn mở

Mục tiêu

- ① Hiểu về kiến thức thế nào là phần mềm nguồn mở.
- ② Khi phát triển phần mềm nguồn mở thì cần phải tuân theo nguyên tắc nào
- ③ Biết một số giấy phép về nguồn mở
- ④ Biết cách đóng góp vào phát triển phần mềm mã nguồn mở
- ⑤ Tham gia vào cộng đồng mã nguồn mở
- ⑥ Một số phần mềm mã nguồn mở thông dụng
- ⑦ Biết cách sử dụng hệ thống quản lý mã nguồn Git

Thuật ngữ

- Copyright, close source, commercial software, proprietary software
- Copyleft, open source, free software, free operation system
- Freeware, shareware
- Public domain, license, GPL, version, distribution terms
- Git, MIT, BSD, . . .

① Phần mềm và vấn đề bản quyền phần mềm

② Mô hình phát triển phần mềm mã nguồn mở

③ Cách đóng góp vào phát triển phần mềm mã nguồn mở

- Khi một phần mềm được tạo ra thì nó thuộc một chủ sở hữu nào đó (lập trình viên, công ty phần mềm, etc.)
- Chủ sở hữu phần mềm:
 - ▶ có toàn quyền trên phần mềm
 - ▶ quyết định mức độ sử dụng và khai thác của những người khác trên phần mềm mà họ là chủ sở hữu thông qua việc cấp giấy phép sử dụng (**license**) phần mềm.
- Khi muốn sử dụng một phần mềm, bạn phải có một **giấy phép sử dụng phần mềm** đó.

Giấy phép sử dụng phần mềm

Là một *bản hợp đồng/thỏa thuận* giữa người sử dụng và chủ sở hữu phần mềm, quy định về những khả năng mà người sử dụng có thể có được trên phần mềm mà người đó được cấp giấy phép sử dụng:

- cách thức người dùng có được phần mềm: mua, thuê viết, download
- mức độ người dùng khai thác và sử dụng phần mềm
- mục đích: thương mại, nghiên cứu, phi lợi nhuận
- dùng trên bao nhiêu máy, số lượng người dùng
- khả năng người dùng có thể có trên phần mềm: phân phối lại, truy cập vào mã nguồn, etc

Những nguyên tắc cơ bản của luật bản quyền

- Công ước Berne: bản quyền gắn kết với sự thể hiện của ý tưởng.
- Bản quyền không bảo hộ ý tưởng mà chỉ bảo hộ sự thể hiện của ý tưởng đó.
- Không ai ngoài tác giả có quyền tạo ra các tác phẩm kế thừa từ tác phẩm gốc có bản quyền.
- Thời gian được bản quyền bảo vệ: phụ thuộc vào quốc gia. Ví dụ: Hoa Kỳ: min(70 năm sau khi tác giả mất, 95 năm từ khi tác phẩm phát hành, 120 từ ngày tạo ra sản phẩm).
- sau khi hết thời hạn bảo vệ các tác phẩm có bản quyền đi vào môi trường công cộng (public domain)
- Tác phẩm làm thuê (work for hire), bao gồm cả dịch thuật: bản quyền thuộc về người thuê hoặc tác giả của bản gốc (người ủy quyền).

Copyright

- *bản quyền* là một khái niệm pháp luật được chính phủ ban hành, giao cho người sáng tác (creator) của một tác phẩm nào đó những quyền riêng biệt trên tác phẩm của mình nhằm bảo vệ quyền tác giả.
- “quyền sao chép” (the right of copy)
- một dạng của sở hữu trí tuệ



Copyleft

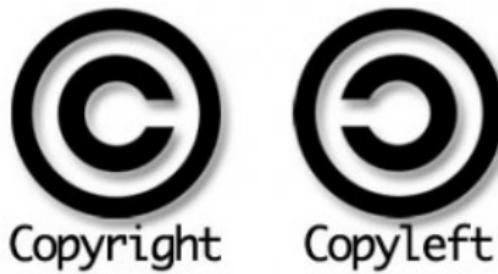
- Copyleft là một phương pháp tổng quát cho việc làm cho một chương trình/phần mềm trở nên tự do (free) và yêu cầu tất cả các phiên bản được chỉnh sửa hoặc mở rộng của nó cũng phải tự do.
- Copyleft phát biểu rằng bất cứ người nào phân phối lại phần mềm, có hay không thay đổi, phải để lại sự tự do sao chép và thay đổi.
- Copyleft bảo đảm cho **MỌI NGƯỜI** có sự tự do trên phần mềm.



copyleft

Copyleft một phần mềm

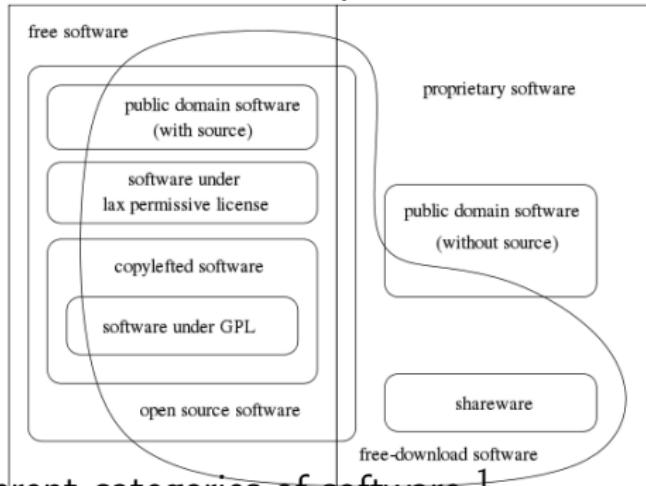
- Thừa nhận có bản quyền (copyrighted)
- Thêm vào những phần liên quan đến việc phân phối.
- Copyleft dùng copyright để bảo đảm sự tự do của người sử dụng.
- Copyleft là một cách dùng copyright trên một chương trình (nó không có nghĩa là bỏ đi copyright)



Phân loại phần mềm

Dựa vào 3 tiêu chí để phân loại:

- khả năng phân phối lại (Distribution Possibility)
- truy cập mã nguồn (Accessibility to source code)
- phí sử dụng (Fee)



Hình 1: The different categories of software¹

¹originally by Chao-Kuei

<https://www.gnu.org/philosophy/categories.html>

Phần mềm thương mại (Commercial Software)

- Bản quyền của phần mềm thương mại chỉ cho phép người sử dụng khai thác phần mềm theo những ràng buộc đã ghi rõ trong giấy phép.
- Bản quyền loại này bị hạn chế.
- Trong trường hợp xuất hiện lỗi phần mềm, người sử dụng không có cách nào khác hơn là phải chờ cho đến khi chủ sở hữu phần mềm sửa đổi chúng.
 - ▶ Người sử dụng không có một phương tiện nào để thúc đẩy tiến trình cập nhật và sửa chữa lỗi của các phần mềm thương mại.
 - ▶ Các nhà sản xuất phần mềm đôi khi không sẵn lòng làm việc đó hoặc thực hiện chúng với thời gian rất lâu hay đôi khi người sử dụng phải trả thêm tiền cho các bản cập nhật.

Phần mềm miễn phí hay trả phí một phần

- Phần mềm miễn phí ([freeware](#)) và phần mềm trả một phần ([shareware](#)) **KHÔNG** là phần mềm mã nguồn mở.
- Phần mềm miễn phí và phần mềm trả một phần:
 - ▶ Vẫn là các phần mềm [có chủ sở hữu](#).
 - ▶ Được phân phối một cách tự do.
- Phần mềm trả một phần thì sau một khoảng thời gian đã định người sử dụng phải trả tiền nếu như muốn được phép sử dụng tiếp.

Phong trào phần mềm tự do

- Nhằm tạo ra những Phần mềm tự do ([Free Software](#)): là những phần mềm mà người dùng có thể tự do chia sẻ, nghiên cứu và sửa đổi chúng.
- Dự án [GNU](#): được khởi xướng bởi Richard M. Stallman vào năm 1984
 - ▶ Viết tắt của “[GNU's Not UNIX](#)”
 - ▶ Nhằm thay thế hệ điều hành Unix với tính năng tự do
- Richard Stallman thành lập Quỹ phần mềm tự do (FSF - Free Software Foundation) năm 1985 để hỗ trợ phong trào phần mềm tự do.



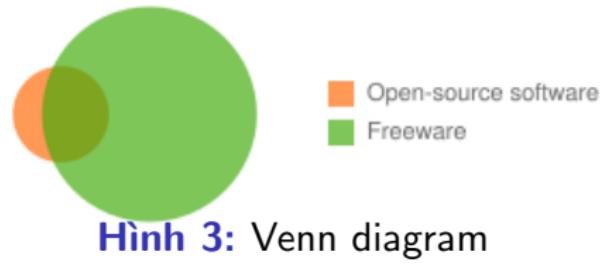
Hình 2: Biểu tượng của dự án GNU

Phần mềm tự do

- Phần mềm tự do (PMTD) đề cập đến sự tự do, không đề cập đến vấn đề chi phí/giá cả:
- Sự tự do bao gồm 4 yếu tố:
 - ▶ Tự do **thực thi** chương trình cho bất kỳ mục đích gì.
 - ▶ Tự do **nghiên cứu cách thực thi** của chương trình và sửa đổi chúng cho mục đích của bạn.
 - ▶ Tự do **phân phối lại** phần mềm cho người khác.
 - ▶ Tự do **cải tiến** chương trình và phân phối cải tiến của bạn cho cộng đồng.

Sáng kiến mã nguồn mở

- OSI (Open Source Initiative)²
- Là tổ chức phi lợi nhuận được thành lập năm 1998 bởi Eric Raymond và Bruce Perens
- Thay thế khái niệm Phần mềm tự do (Free software) bằng khái niệm Phần mềm mã nguồn mở (Open Source Software) để tránh sự hiểu nhầm:
 - ▶ Ý nghĩa tự do với miễn phí của từ “free” tiếng Anh.
 - ▶ Phần mềm tự do là không thương mại.



²www.opensource.org

Phần mềm mã nguồn mở

Một phần mềm mã nguồn mở nếu nó hội đủ các yếu tố cơ bản sau:

- Nó được phân phối đến người sử dụng **cùng với mã nguồn** của nó mà chúng có thể bị sửa đổi.
- Nó có thể được **phân phối lại** mà không bị một ràng buộc nào khác.
- Chúng ta có thể phân phối cả những thay đổi mà chúng ta đã thực hiện trên mã nguồn gốc.

Tác giả giữ bản quyền (copyright) đối với mã nguồn và phân phối mã nguồn dưới một giấy phép định nghĩa những gì được làm (hoặc không được làm) đối với mã nguồn.

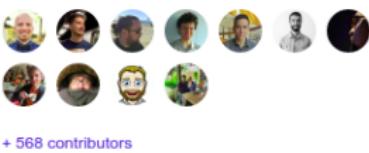
Phần mềm mã nguồn mở

10 điều kiện mà các điều khoản phân phối phần mềm mã nguồn mở phải tuân theo:

- ① phân phối lại tự do
- ② mã nguồn
- ③ các sản phẩm kế thừa
- ④ tính toàn vẹn của mã nguồn của tác giả
- ⑤ không phân biệt đối xử người dùng
- ⑥ không phân biệt đối xử lĩnh vực áp dụng
- ⑦ phân phối giấy phép
- ⑧ giấy phép không được dành riêng biệt cho 1 sản phẩm
- ⑨ giấy phép không được hạn chế các sản phẩm khác
- ⑩ giấy phép phải trung lập với công nghệ

Lợi ích của phần mềm nguồn mở

Contributors 579



Languages



- Phần mềm mã nguồn mở **được phát triển bởi một cộng đồng** nhiều người nhờ đó có thể tìm ra các lỗi một cách dễ dàng
 - ▶ Là điểm **mạnh nhất** của phần mềm mã nguồn mở.
 - ▶ Mỗi người, với khả năng có hạn của mình có thể **xem xét** và **cải tiến** các công việc được thực hiện bởi những người bạn khác.
 - ▶ Mỗi thành viên chỉ tập trung vào phần thuộc lĩnh vực chuyên sâu của mình.
- Cách **phân phối** của phần mềm mã nguồn mở giúp nhiều người có điều kiện **tiếp cận** với chúng hơn.

Hình 4: Contributors to Laravel

So sánh một số loại phần mềm phổ biến

Bảng 1: So sánh một số loại phần mềm

Các phần mềm phổ biến	Phân phối lại	Truy cập vào mã nguồn	Miễn phí
Phần mềm thương mại	không	không	không
Phần mềm miễn phí	đôi khi	không	có
Phần mềm trả phí một phần	đôi khi	không	không
Phần mềm mã nguồn mở	có	có	có

Giấy phép mã nguồn mở (GPMNM)

- Là các giấy phép bản quyền dành cho các phần mềm máy tính trong đó đặt ra các quy định buộc người sử dụng phần mềm đó phải tuân theo.
- Giấy phép mã nguồn mở được sử dụng cho các phần mềm nguồn mở.
- Giấy phép mã nguồn mở xác nhận về bản quyền của tác giả gốc đối với phần mềm, tuy nhiên được đưa thêm các điều khoản để các hành vi phân phối, sửa đổi, sao chép... các phần mềm này trở thành hợp pháp.

Phân loại GPMNM

- Những giấy phép **không quy định bất cứ sự hạn chế nào** trong việc sử dụng mã nguồn:
 - ▶ Apache Software License v.1.1, BSD License, Intel Open Source License for CDSA/CSSM, MIT License, Sun Industry Standards Source License, W3C Software Notice and License.
- Những giấy phép **quy định một số hạn chế** trong việc sử dụng mã nguồn:
 - ▶ Common Public License v .1.0, GNU General Public License v.2.0, IBM Public License v .1.0, Mozilla Public License v.1.0 and v.1.1, Open Software License v .1.1, Python Software Foundation License v .2.1.1, Sun Public License v .1.0

GNU GPL

- GNU GPL là giấy phép phần mềm tự do phổ biến nhất, ban đầu được thiết kế bởi Richard Stallman cho dự án GNU.
- Là giấy phép [copyleft](#).
- Tác giả gốc giữ bản quyền, và [cho phép](#) người dùng các quyền.



GNU GPL

- Giấy phép công cộng GPL yêu cầu một cách tường minh những phần mềm kế thừa từ các phần mềm được phân phối dưới giấy phép GPL chỉ có thể được phép phân phối lại các điều khoản của giấy phép GPL.
- Một số phần mềm sử dụng GNU GPL: Ubuntu, Wordpress, Joomla, GIMP, Octave, LibreOffice–core, MPV ...



BSD License

Giấy phép BSD cho phép **sử dụng và phân phối lại** mã nguồn và sản phẩm, có hoặc không có sửa đổi, miễn là tuân thủ các yêu cầu sau:

- Phải **giữ nguyên thông báo bản quyền** của sản phẩm.
- Phải kèm theo 2 thông báo: Danh sách các **điều kiện** và **từ chối trách nhiệm**.
- Không được sử dụng tên dự án hay tên nhà phân phối vào **mục đích quảng bá bản thân** nếu không được cho phép.
- Hiện tại giấy phép BSD mới **tương thích** với **GPL**



BSD License

Một số phần mềm sử dụng giấy phép BSD: FreeBSD, NetBSD, OpenBSD.



FreeBSD



Mozilla Public License

- Đặc điểm của MPL là lai giữa giấy phép BSD có chỉnh sửa và giấy phép công cộng GNU
- Giấy phép được xem là một copyleft yêu: mã nguồn nếu được sao chép hoặc thay đổi theo giấy phép MPL phải giữ nguyên giấy phép MPL
- Tương thích với các giấy phép khác

MOZILLA PUBLIC LICENSE VERSION 2.0

1. DEFINITIONS

1.1. "Contributor"

means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

Hình 5: MPL³

³<https://www.mozilla.org/en-US/MPL/2.0/>

Mozilla Public License

- Không giống như những giấy phép copyleft mạnh, mã nguồn được cấp phép theo MPL có thể được kết hợp với các tập tin thương mại trong một chương trình
- Một số phần mềm sử dụng giấy phép MPL: Mozilla Firefox, Mozilla Thunderbird



MIT License

- MIT là một giấy phép phần mềm tự do được phát hành bởi Học viện Công nghệ Massachusetts, được hội đồng MIT X sử dụng.
- Tương tự như giấy phép BSD, MIT là giấy phép nguồn mở xưa nhất, được dùng trong rất nhiều dự án nguồn mở.
- Ban đầu được soạn thảo cho X Window System (dự án tạo giao diện đồ họa cho người dùng ở các máy tính kết nối mạng vào năm 1984)



Hình 6: MIT License⁴

⁴<https://opensource.org/licenses/MIT>

MIT License

- giấy phép này rất **mềm dẻo và tương thích** với hầu hết các dạng của giấy phép nguồn mở.
- Một số phần mềm sử dụng giấy phép MIT: Ruby on Rail, jQuery, X Window System, Node.js, ...



.Org Foundation



Apache License

- Giấy phép Apache **không yêu cầu** bản sửa đổi của phần mềm phải được phân phối dưới cùng giấy phép với bản gốc, **cũng không yêu cầu** bản sửa đổi phải được phân phối dưới dạng mã nguồn mở.
- Giấy phép Apache **không yêu cầu** trích dẫn toàn bộ giấy phép vào sản phẩm hay tệp tin đính kèm bản phân phối, mà chỉ cần thêm phần thông báo có chứa đường link tới website chứa giấy phép.



Hình 7: Apache License⁵

⁵<https://www.apache.org/licenses/LICENSE-2.0>

Apache License

- Phần mềm sử dụng giấy phép Apache: Openoffice, Apache HTTP Server



So sánh các giấy phép phổ biến

Rights granted	Public domain	Permissive FOSS license (e.g. BSD license)	Copyleft FOSS license (e.g. GPL)	Freeware/Shareware/Freemium	Proprietary license	Trade secret
Copyright retained	No	Yes	Yes	Yes	Yes	Very strict
Right to perform	Yes	Yes	Yes	Yes	Yes	No
Right to display	Yes	Yes	Yes	Yes	Yes	No
Right to copy	Yes	Yes	Yes	Often	No	Lawsuits are filed by the owner against copyright infringement the most
Right to modify	Yes	Yes	Yes	No	No	No
Right to distribute	Yes	Yes, under same license	Yes, under same license	Often	No	No
Right to sublicense	Yes	Yes	No	No	No	No
Example software	SQLite, ImageJ	Apache web server, ToyBox	Linux kernel, GIMP, OBS	Irfanview, Winamp, League of Legends	Windows, the majority of commercial video games and their DRMs, Spotify, xSplit, TIDAL.	Server-side Cloud computing programs and services, forensic applications, and other line-of-business work.

Hình 8: Software licenses according to Mark Webbink⁶

⁶https://en.wikipedia.org/wiki/Software_license

- ① Phần mềm và vấn đề bản quyền phần mềm
- ② Mô hình phát triển phần mềm mã nguồn mở
- ③ Cách đóng góp vào phát triển phần mềm mã nguồn mở

Mô hình phát triển phần mềm mã nguồn mở

- ① Một PMMNM là một phần mềm, vì thế nó được phát triển trong một dự án phát triển phần mềm, với một ngoại lệ: là **dự án được phát triển bởi một nhóm hoặc cộng đồng** mà các thành viên của nhóm có thể chưa bao giờ gặp nhau.
- ② Câu hỏi đặt ra: các quy trình công nghệ phần mềm (CNPM) hay **quy trình phát triển phần mềm truyền thống** có ứng dụng được vào cho **phát triển PMMNM** hay không?

Mô hình phát triển phần mềm truyền thống

- Đòi hỏi tính **chặt chẽ** trong các công đoạn quản lý, thiết kế và xây dựng như:
 - Quản lý ai là người viết các phần mã lệnh, phương pháp mà họ tích hợp các gói mã lệnh
 - Định nghĩa rõ ràng một cấu trúc quản lý
 - Xây dựng một kế hoạch chính xác về lịch phát hành mã lệnh

Mô hình phát triển PMMNM

- Không có một thiết kế ban đầu rõ ràng, không có một quy trình quản lý chính thức
- Nguyên tắc căn bản: “Viết mã lệnh thường xuyên, phát hành thường xuyên” nhờ vào cộng đồng phát triển lớn.

The screenshot shows the GitHub repository page for TensorFlow. It highlights several key features:

- Commits:** 93,791 commits, with the most recent one being "yesterday".
- Tags:** tensorflow, machine-learning, python, neural-network, ml, distributed.
- Readme:** Apache-2.0 License.
- Releases:** TensorFlow 2.3.0 (Latest), released 25 days ago, with +115 releases.
- Packages:** No packages published.
- Used by:** 5,000+ users, with a total of 89,921 users.

Mô hình phát triển PMMNM

- Đây là **mô hình tăng trưởng**: tự phát triển khi phần mềm đạt đến một số chức năng cơ bản nào đó. Mô hình phát triển gồm 2 giai đoạn:
 - ① Phần mềm **chưa đủ** các chức năng để có thể hấp dẫn các lập trình viên khác. Cần một số tài trợ về tài chính để có thể đạt đến điểm có thể sử dụng được, sẽ chuyển sang giai đoạn tăng trưởng.
 - ② Nhận được **thêm nhiều** chức năng mới và **các gói sửa lỗi** từ **cộng đồng**.

So sánh hai mô hình

Tài nguyên cho phát triển một phần mềm: lập trình viên, công cụ phát triển (máy tính, ...), kênh phân phối, kỹ thuật viên.

Bảng 2: So sánh mô hình phát triển PM truyền thống và PMMNM

Phát triển PM truyền thống	PMMNM
Khan hiếm và tốn kém, vì thế cần quản lý chặt chẽ	Lập trình viên là tài nguyên
Cần xây dựng môi trường để bảo vệ tài nguyên này	Sử dụng hạ tầng cơ sở (ví dụ máy tính) sẵn có. Phân phối qua Internet

Người phát triển PMMNM

Câu hỏi đặt ra: Ai sẽ phù hợp để phát triển phần mềm mã nguồn mở? **Động cơ** của người tham gia phát triển PMMNM là gì?

- Về **lợi ích kinh tế** đối với lập trình viên: **không đáng kể**
- Phần lớn PMMNM là kết quả của:
 - ▶ niềm **đam mê** lập trình
 - ▶ kết quả của một số **dự án** trong các chương trình đại học
 - ▶ vì **lợi ích cộng đồng**
- Một số công ty dùng PMMNM để
 - ▶ **Phát hành sản phẩm nhanh hơn** nhờ sử dụng lại PMMNM
 - ▶ Thâm nhập thị trường đã bị thống trị bởi công ty khác

Môi trường phát triển PMMN

Môi trường phát triển PMMN cần cung cấp các chức năng sau:

- ① Các kênh truyền thông (communication channel)
- ② Các cơ sở dữ liệu về lỗi (Bug database)
- ③ Hệ thống quản lý mã nguồn (Version control)

Communication channel

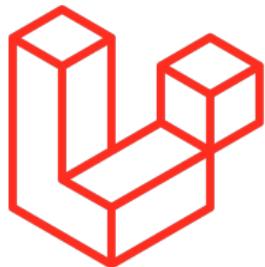
Gồm các thành phần như: Website, Mailing list, Bug Tracker, IRC, Wiki, Newsletters, Files bundled with code. Cung cấp các thông tin như:

- Mô tả thông tin về phần mềm mã nguồn mở (mục tiêu dự án, bản phân phối mới nhất, tài liệu thiết kế, kế hoạch và lịch trình tương lai)
- Chuẩn lập trình
- Quyền sở hữu tập tin/module
- Danh sách lỗi đang mở (và đóng)
- Cách thức để lấy mã, đóng góp vào mã nguồn
- Liên kết tới những kênh giao tiếp khác

Communication channel

Thông thường mỗi dự án mã nguồn mở lớn sẽ có một **website** đưa thông tin về dự án. Ví dụ như

- Laravel <https://laravel.com/>
- MPV <https://mpv.io/>
- Tensorflow <https://www.tensorflow.org/>
- Node.js <https://nodejs.org/en/>



Bug database

Đối với một PMMNM, lỗi (bugs) là không tránh khỏi, cần có phương tiện để người dùng thông báo lỗi

Sử dụng mailing list có hạn chế:

- Dễ bị mất,
- Lập trình viên mới không biết các lỗi trước đây.



Bug database

Lưu lỗi vào cơ sở dữ liệu có những lợi thế:

- Dễ dàng tìm kiếm lỗi
- Dùng cho các mục đích khác nữa: yêu cầu tính năng, cải tiến, bản vá lỗi
- Ví dụ: Bugzilla⁷, Mantis⁸, Trac⁹, Google Code¹⁰, github-issues¹¹



⁷<https://www.bugzilla.org/>

⁸https://www.mantisbt.org/bugs/my_view_page.php

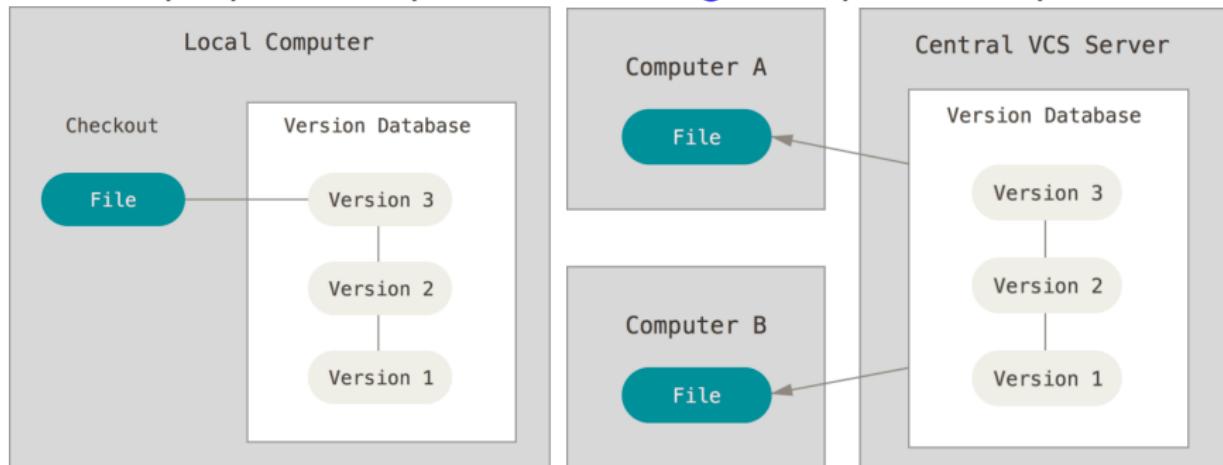
⁹<https://pypi.org/project/Trac/>

¹⁰<https://code.google.com/>

¹¹<https://github.com/>

Hệ thống quản lý phiên bản–Version Control Systems

- Lưu trữ mã nguồn
- Được dùng để theo vết tất cả các công việc và các thay đổi trong một tập hợp các tập tin.
- Cho phép nhiều lập trình viên cùng nhau phát triển phần mềm

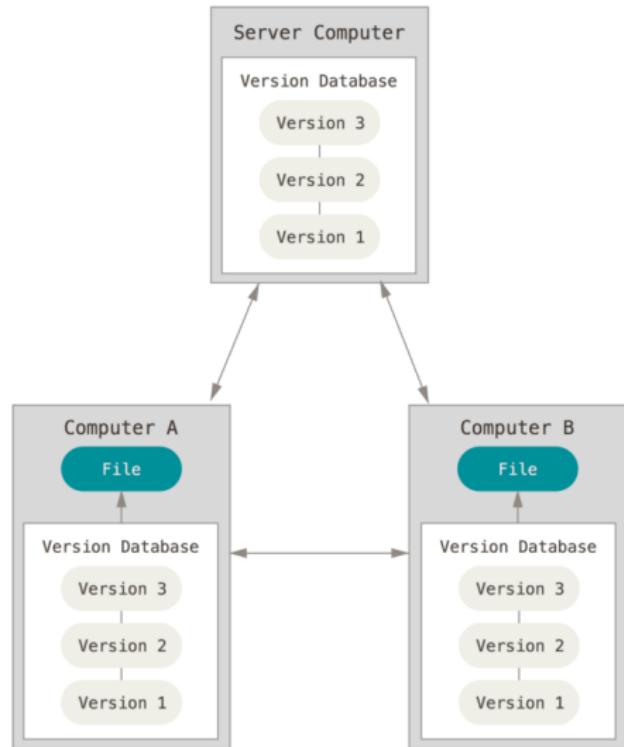


Hệ Thống Quản Lý Phiên Bản Phân Tán–Distributed Version Control Systems

Hệ Thống Quản Lý Phiên Bản Phân Tán–DVCSs khắc phục được các nhược điểm của Hệ Thống Quản Lý Phiên Bản trước đó.

- Trong các DVCSs (ví dụ như [Git](#), Mercurial, Bazaar hay Darcs) các máy khách sao chép toàn bộ kho chứa (repository) ⇒ luôn có những bản sao đầy đủ của tất cả dữ liệu.
- DVCSs còn giải quyết được bài toán quản lý nhiều kho chứa từ xa, khi đó còn có thể cộng tác với nhiều nhóm người khác nhau theo những cách khác nhau trong cùng một dự án.

Hệ Thống Quản Lý Phiên Bản Phân tán (DVCSs)



Hình 9: Distributed version control¹²

[https://git-scm.com/book/en/v2/
Getting-Started-About-Version-Control](https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control)

Distributed Version Control Systems–Git

Git¹³ là một hệ thống quản lý phiên bản phân tán được viết bởi Linus Torvalds (người tạo ra và phát triển chính của Linux kernel) vào năm 2005. SV có thể tham khảo cách cài đặt Git:

- trên hệ điều hành Linux bằng lệnh sau trong terminal

\$ sudo apt install git-all

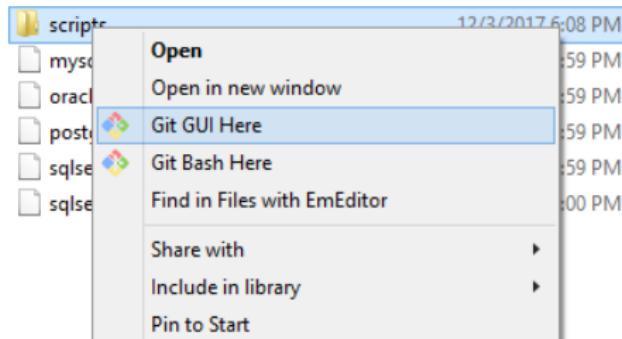
```
trieunh@trieunh:~$ sudo apt install git-all
[sudo] password for trieunh:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cvs cvspc dh-elpa-helper elpa-dash elpa-ghub elpa-git-commit elpa-let-alist
  elpa-magit elpa-magit-popup elpa-treepy elpa-with-editor emacs
  emacs-bin-common emacs-common emacs-el emacs-gtk git-cvs git-daemon-run
  git-doc git-el git-email git-gui git-mediawiki git-svn gitk gitweb
  libcommon-sense-perl libdate-time-format-iso8601-perl libdbd-sqlite3-perl
  libdbi-perl libdigest-bubblebabble-perl libdigest-hmac-perl
  libemail-valid-perl libjson-perl libjson-xs-perl libm17n-0
  libmediawiki-api-perl libnet-dns-perl libnet-dns-sec-perl
  libnet-domain-tld-perl libnet-ip-perl libotf0 libperl4-corelibs-perl
  libserf-1 libsvn-perl libsvn1 libterm-readkey-perl
  libtypes-serialiser-perl libutf8proc2 libyaml-libyaml-perl libyaml-perl
  m17n-db runit
Suggested packages:
  mksh rcs emacs-common-non-dfsg ncurses-term mediawiki subversion
  libldb-perl libnet-daemon-perl libsql-statement-perl m17n-docs
  libyaml-shell-perl gawk
The following NEW packages will be installed:
```

¹³<https://git-scm.com/>

Distributed Version Control Systems–Git

```
trieunh@trieunh:~$ git --version
git version 2.25.1
```

- trên Windows tại <https://git-scm.com/download/win>



Hình 10: Tham khảo cài đặt chi tiết¹⁴

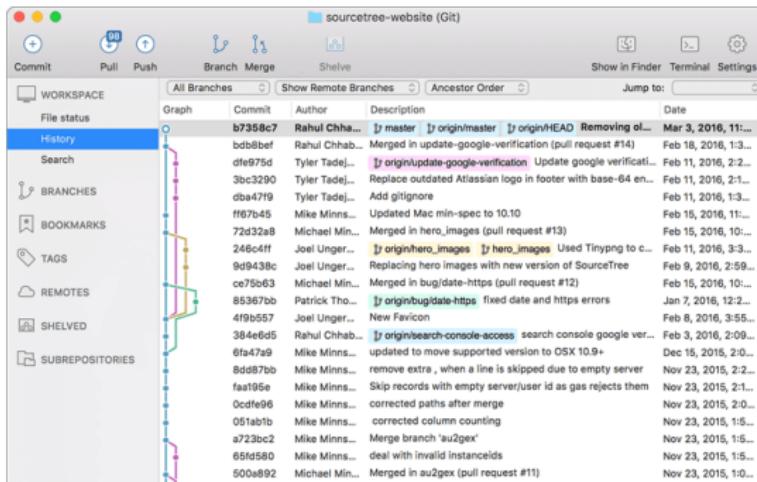
¹⁴<https://o7planning.org/vi/11707/>

huong-dan-cai-dat-va-cau-hinh-git-tren-windows

Git-GUI Clients

Các công cụ để thao tác với Git

- Terminal của Linux hoặc Cmd, GitBash của Windows.
- dùng các GUI hỗ trợ như SourceTree (Platforms: Mac, Windows. Price: Free. License: Proprietary)



Ưu điểm: giao diện trực quan, dễ hiểu, có thể thêm nhiều tài khoản từ Git, Bitbucket.

Git-GUI Clients

text_classification: All files - gitk

File Edit View Help

Local uncommitted changes, not checked in to index

- fix bugs for class: KHAC
- use lsd model for topic modeling
- add new function for topic modeling
- fix bug for pyntstaller
- Hyperparameter tuning + modify static function
- add file name to save model
- add static function
- fix bug for pyntstaller
- code function for predict example
- show accuracy of model

SHA1: 3171a1745154e2f8677819962957800b3e73f645

Find commit containing: Exact All fields

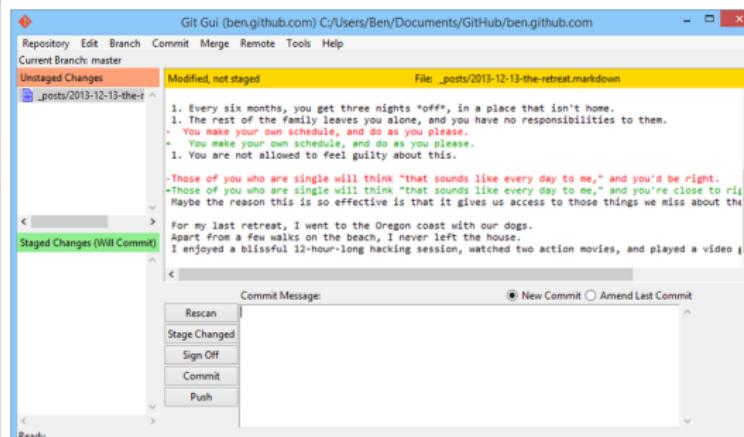
Search Old version New version Lines of context: 3 Ignore space change

```

diff --git a/app.py b/app.py
index 2272e7a..b7d4175 100755
@@ -14,6 +14,8 @@ import csv, os.path, pathlib
from sklearn.externals import joblib
import numpy as np
from model.lda_svm import LDASVMModel
+import numpy as np

db = Database('train.db')
class TextClassificationPredict(object):
@@ -62,7 +64,7 @@ class TextClassificationPredict(object):
    print("using trained model")
except:
    clf = model.clf.fit(df['feature1'], df_train.target)
    model_path = os.path.join('./save_model', 'svm_model_ss.pkl')
    save_model(ss.pkl, 'svm_model_ss.pkl')

```

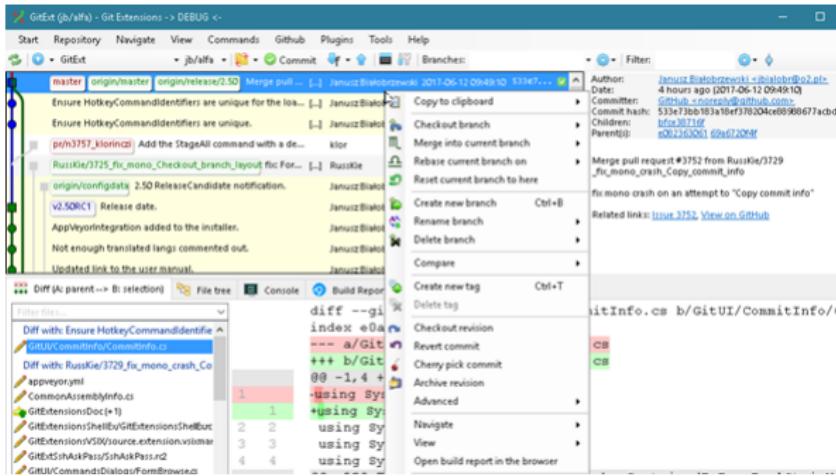


(a) gitk

(b) git-gui

Hình 11: Sử dụng gitk và git-gui có sẵn khi cài đặt Git

Git-GUI Clients



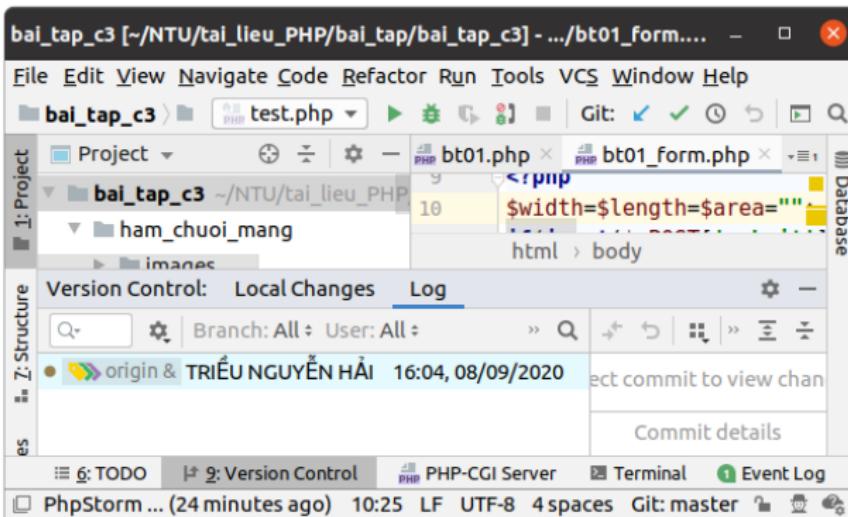
Hình 12: Git Extensions (Platforms: Linux, Mac, Windows. Price: Free. License: GNU GPL)

Ngoài ra, có thể tham khảo một số Git-GUI Clients khác tại <https://git-scm.com/downloads/guis>

Git-GUI Clients

Các công cụ để thao tác với Git

- dùng một số editor như PHP storm, Visual Studio Code, Pycharm, Ruby Mine . . .



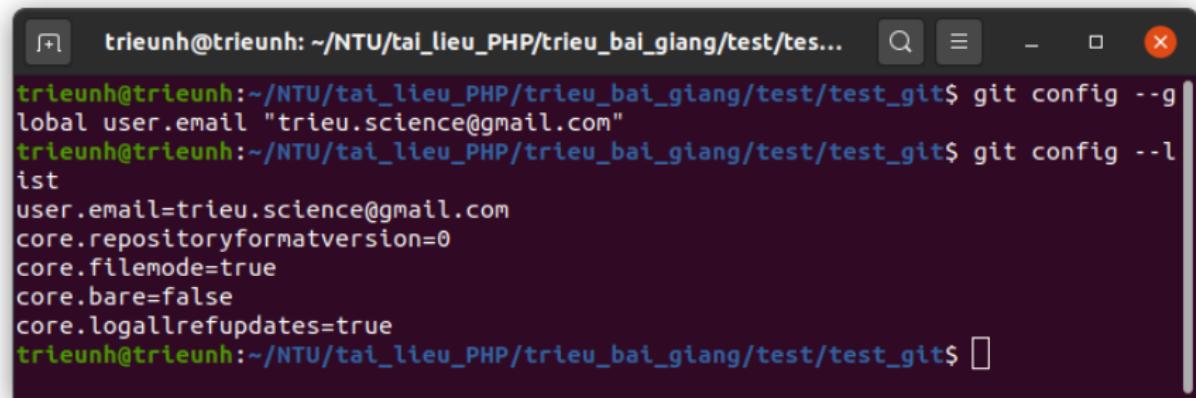
Hình 13: VCS trong PHPstrom

Các lệnh Git cơ bản

Sử dụng [terminal](#) hoặc [Cmd](#):

- **git config –global user.name <username>** hoặc
- **git config –global user.email <email>**

→ config [username](#) và [email](#) đã đăng ký trên Git servers

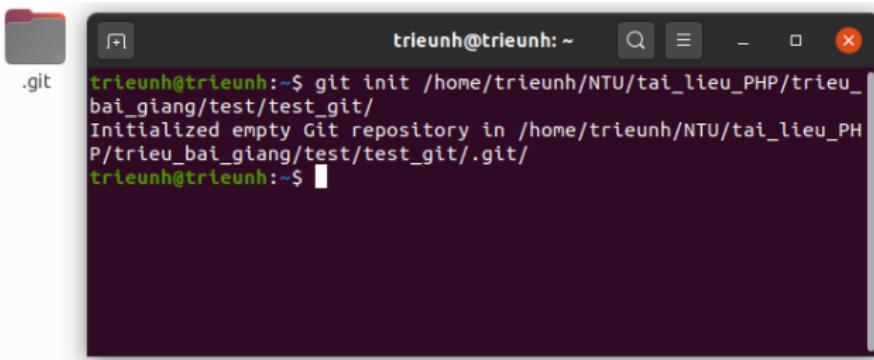


```
trieunh@trieunh: ~/NTU/tai_lieu_PHP/trieu_bai_giang/test/test_git$ git config --global user.name "trieu.science@gmail.com"
trieunh@trieunh: ~/NTU/tai_lieu_PHP/trieu_bai_giang/test/test_git$ git config --list
user.name=trieu.science@gmail.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
trieunh@trieunh: ~/NTU/tai_lieu_PHP/trieu_bai_giang/test/test_git$ 
```

Các lệnh Git cơ bản

Sử dụng [terminal](#) hoặc [Cmd](#):

- **git init** → khởi tạo một kho Git hoàn toàn mới và bắt đầu theo dõi một thư mục hiện có.



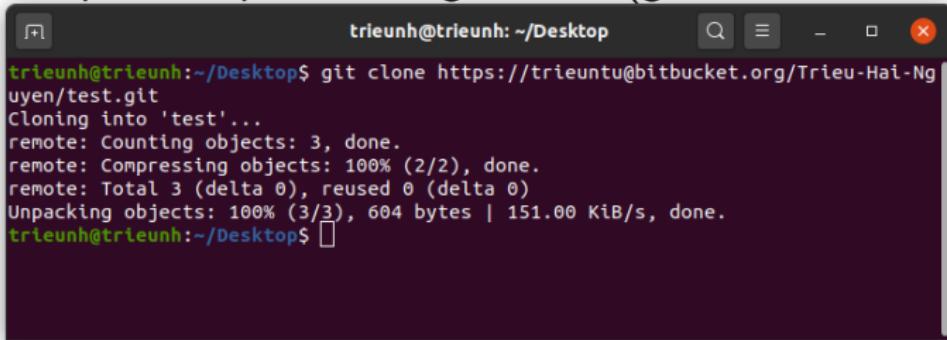
A screenshot of a terminal window titled "trieunh@trieunh: ~". The window shows the command "git init /home/trieunh/NTU/tai_lieu_PHP/trieu_bai_giang/test/test_git/" being run, followed by the message "Initialized empty Git repository in /home/trieunh/NTU/tai_lieu_PHP/trieu_bai_giang/test/test_git/.git/". The terminal has a dark background and light-colored text. A ".git" folder icon is visible on the desktop to the left of the terminal window.

Hình 14: Khởi tạo một local Git repository rỗng tại thư mục `test_git`

Lệnh trên sẽ tạo ra thư mục mới có tên `.git` chứa tất cả các tập tin cần thiết cho repository.

Các lệnh Git cơ bản

- **git clone /path/to/repository** → Lệnh này được sử dụng để copy một project từ Local Repository đến một thư mục khác
- **git clone username@serverAddress: /path/to/project** → Sao chép một repo từ một git server (github, bitbucket)



A screenshot of a terminal window titled "trieunh@trieunh: ~/Desktop". The terminal displays the following command and its execution:

```
trieunh@trieunh:~/Desktop$ git clone https://trieuntu@bitbucket.org/Trieu-Hai-Nguyen/test.git
Cloning into 'test'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), 604 bytes | 151.00 KiB/s, done.
trieunh@trieunh:~/Desktop$
```

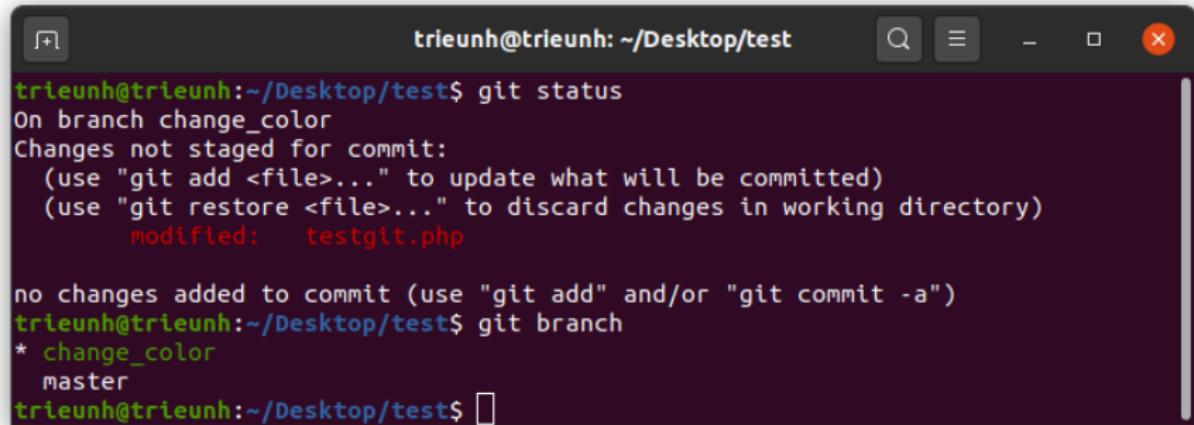
Hình 15: git clone

<https://trieuntu@bitbucket.org/Trieu-Hai-Nguyen/test.git>

Ví dụ ở Fig.15 sẽ sao chép repo **test** từ server **bitbucket.org**. Để đi đến repo vừa tải, dùng lệnh **cd test/**

Các lệnh Git cơ bản

- **git status** → lệnh này sẽ kiểm tra tình trạng có bất kì thay đổi gì ở local trong source hay không. Nếu có thì tiến hành commit đầy source lên trên Git server.



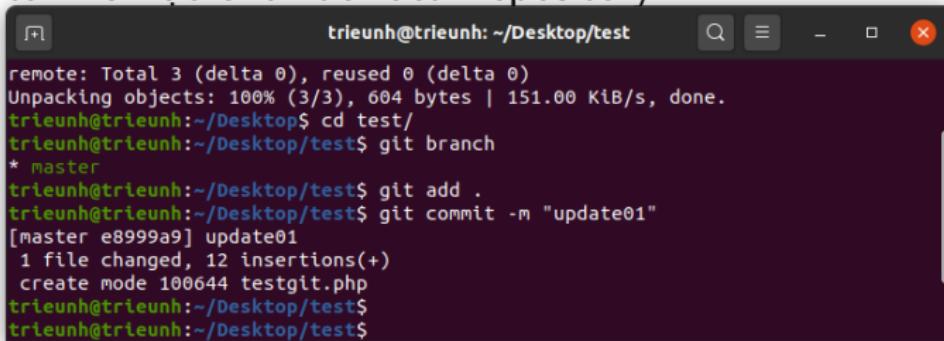
```
trieunh@trieunh: ~/Desktop/test$ git status
On branch change_color
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   testgit.php

no changes added to commit (use "git add" and/or "git commit -a")
trieunh@trieunh: ~/Desktop/test$ git branch
* change_color
  master
trieunh@trieunh: ~/Desktop/test$
```

Hình 16: Ví dụ sử dụng lệnh kiểm tra **git status** ở local khi thay đổi testgit.php

Các lệnh Git cơ bản

- **git add .** → Khi thay đổi trong repo như thêm mới, sửa, xoá files, ... Lệnh này sẽ cập nhật toàn bộ lên Staging Area (khu vực chờ).
- **git commit -m 'nội_dung'** → Sau lệnh add, lệnh **commit** sẽ lưu từ khu vực chờ vào local repository.



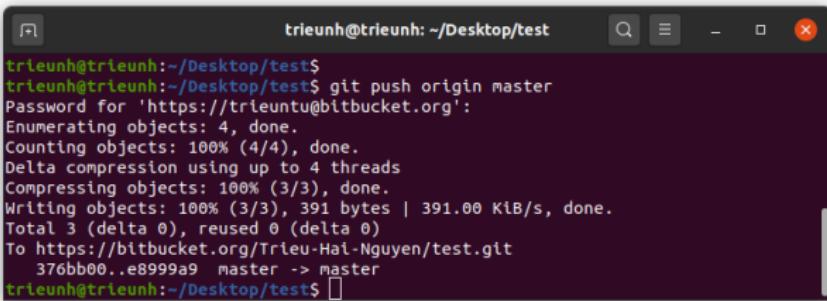
```
trieunh@trieunh: ~/Desktop/test
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), 604 bytes | 151.00 KiB/s, done.
trieunh@trieunh:~/Desktop$ cd test/
trieunh@trieunh:~/Desktop/test$ git branch
* master
trieunh@trieunh:~/Desktop/test$ git add .
trieunh@trieunh:~/Desktop/test$ git commit -m "update01"
[master e8999a9] update01
 1 file changed, 12 insertions(+)
  create mode 100644 testgit.php
trieunh@trieunh:~/Desktop/test$
trieunh@trieunh:~/Desktop/test$
```

Hình 17: git commit -m "update01"

Lúc này, ở Local Repository đã được cập nhật thông tin của file **testgit.php**

Các lệnh Git cơ bản

- **git push origin <name_branch>** → đưa những thay đổi từ local repository lên remote repository của server (ví dụ như Bitbucket, GitHub). Mặc định `name_branch` đang ở nhánh `master`



```
trieunh@trieunh:~/Desktop/test$ git push origin master
Password for 'https://trieunh@bitbucket.org':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 391 bytes | 391.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://bitbucket.org/Trieu-Hai-Nguyen/test.git
    376bb00..e8999a9  master -> master
trieunh@trieunh:~/Desktop/test$
```

Ngoài ra, nếu chưa tồn tại `remote` trên server thì cần phải add mới một `remote` trước rồi mới push:

- **git remote add origin <remote_url>** (`remote_url` là link đến project trên server, trong ví dụ này là: `https://trieuntu@bitbucket.org/Trieu-Hai-Nguyen/test.git`)
- **git push origin master**

Các lệnh Git cơ bản

Xem lại logs trực tiếp bằng lệnh `gitk` hoặc từ UI của git server

The image displays two side-by-side screenshots illustrating how to view commit history.

(a) gitk: A screenshot of the `gitk` graphical interface. The main window shows a log of commits for the `master` branch. The first commit is labeled "Initial Commit". Below the log, there is a detailed view of the first commit, showing the SHA1 ID (`e8999a9b70e6d15d35cad0a41f2ca093cce2ea5e`), author (`trieunh <trieu.science@gmail.com>`), date (`2020-09-08 23:02:24`), and committer (`TRIỀU NGUYỄN HÀI <trieu.science@gmail.com>` at `2020-09-08 16:04:13`). The bottom pane shows the raw diff of the first commit, which added a new file `testgit.php` containing the following PHP code:

```

new file mode 100644
index 0000000..5337b04
@@ -0,0 +1,12 @@
<!DOCTYPE html>
<html lang="en">
<head>
+   <meta charset="UTF-8">
+   <title>Git</title>
</head>
<body>
</body>
</html>

```

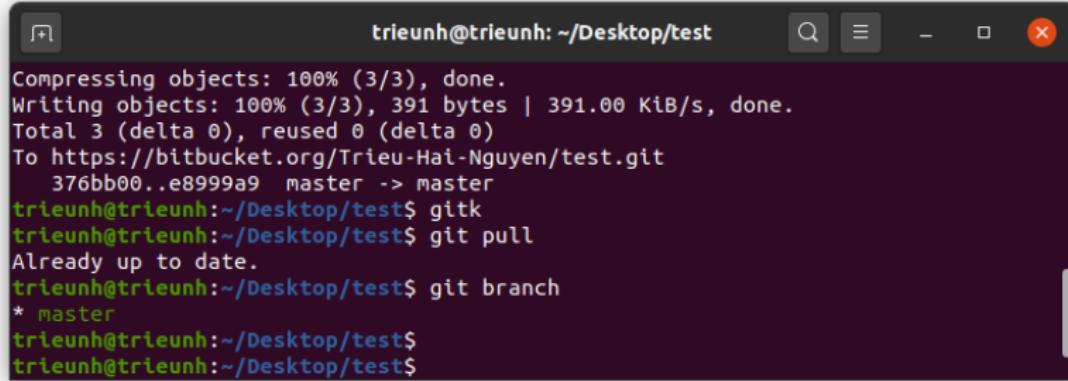
(b) bitbucket-commits: A screenshot of the Bitbucket UI showing the commit history for a repository named "test". The sidebar on the left has "Commits" selected. The main area displays a table of commits:

Author	Commit	Message
TRIỀU NGUYỄN HÀI	e8999a9	update01
TRIỀU NGUYỄN HÀI	376bb00	Initial commit

Hình 18: Xem lại lịch sử commits

Các lệnh Git cơ bản

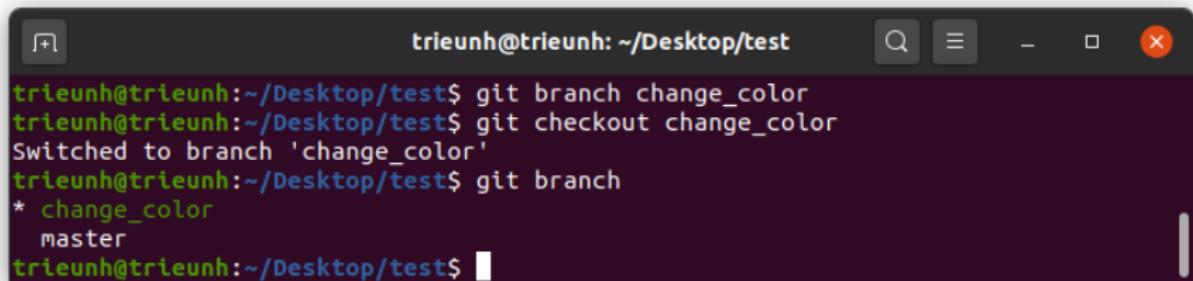
- **git pull** → lệnh này dùng để lấy những thay đổi từ source code trên remote server về working directory và merge với code hiện tại trên máy. Trường hợp không thể merge tự động được thì phải merge thủ công.
- **git branch** → lệnh này dùng để kiểm tra branch hiện tại. Khi sử dụng Git, chúng ta có thể tạo ra nhiều nhánh (branch) khác nhau.



```
trieunh@trieunh: ~/Desktop/test
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 391 bytes | 391.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://bitbucket.org/Trieu-Hai-Nguyen/test.git
  376bb00..e8999a9 master -> master
trieunh@trieunh:~/Desktop/test$ gitk
trieunh@trieunh:~/Desktop/test$ git pull
Already up to date.
trieunh@trieunh:~/Desktop/test$ git branch
* master
trieunh@trieunh:~/Desktop/test$ 
trieunh@trieunh:~/Desktop/test$
```

Các lệnh Git cơ bản

- **git branch <name_branch>** → tạo mới một branch mới với tên nhánh <name_branch> bất kỳ
- **git checkout <name_branch>** → trước khi muốn thay đổi source code, điều đầu tiên cần phải làm là checkout một nhánh.
- **git checkout master** → quay lại nhánh master



```
trieunh@trieunh: ~/Desktop/test$ git branch change_color
trieunh@trieunh: ~/Desktop/test$ git checkout change_color
Switched to branch 'change_color'
trieunh@trieunh: ~/Desktop/test$ git branch
* change_color
  master
trieunh@trieunh: ~/Desktop/test$
```

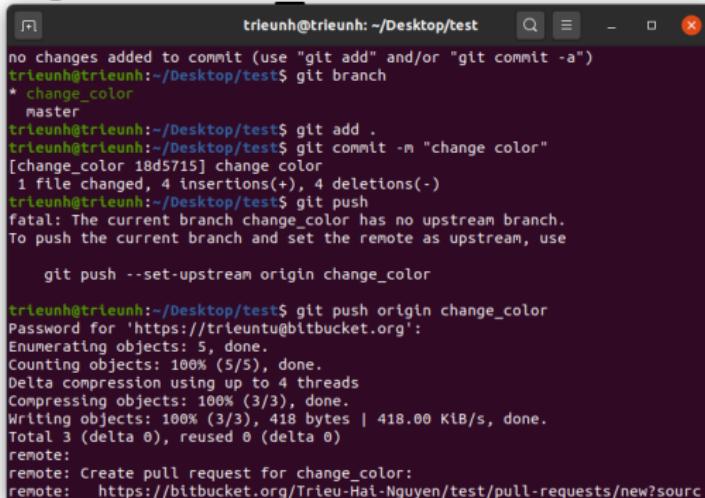
A screenshot of a terminal window titled "trieunh@trieunh: ~/Desktop/test". The terminal shows the execution of several git commands. First, it runs "git branch change_color", which creates a new branch named "change_color". Then, it runs "git checkout change_color", which switches the working directory to this new branch. Finally, it runs "git branch" again to show the current branches, where "change_color" is marked with an asterisk (*), indicating it is the active branch. The terminal has a dark background with light-colored text and standard window controls at the top.

Hình 19: Tạo ra nhánh mới có tên là [change_color](#) và đi đến nhánh đó.

Các lệnh Git cơ bản

Nếu thay đổi code ở một nhánh mới, thực hiện tuần tự lại các bước sau để đẩy code lên Git server

- **git checkout <name_branch>**
- **git add .**
- **git commit -m "change color"**
- **git push origin <name_branch>**



```
trieunh@trieunh:~/Desktop/test$ git branch
* change_color
  master
trieunh@trieunh:~/Desktop/test$ git add .
trieunh@trieunh:~/Desktop/test$ git commit -m "change color"
[change_color 18d5715] change color
 1 file changed, 4 insertions(+), 4 deletions(-)
trieunh@trieunh:~/Desktop/test$ git push
fatal: The current branch change_color has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin change_color

trieunh@trieunh:~/Desktop/test$ git push origin change_color
Password for 'https://trieunh@bitbucket.org':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 418 bytes | 418.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create pull request for change_color:
remote: https://bitbucket.org/Trieu-Hai-Nguyen/test/pull-requests/new?source=
```

Các lệnh Git cơ bản

Sử dụng lệnh `gitk` để xem lại các thay đổi lên server

The screenshot shows the gitk application window titled "test: All files - gitk". The main pane displays a commit history:

SHA1 ID	Author	Date
18d57153f234a6c4368bdd92deef661eb9791540	trieunh <trieu.science@trieunh>	2020-09-09 10:23
remotes/origin/change_color	trieunh <trieu.science@trieunh>	2020-09-08 23:59
remotes/origin/master	TRIỀU NGUYỄN HÀI <tri...	2020-09-08 16:46
Initial commit		

Below the commits, there is a search bar with "Find" and "commit containing:" fields, and a status bar showing "Row 1 / 3".

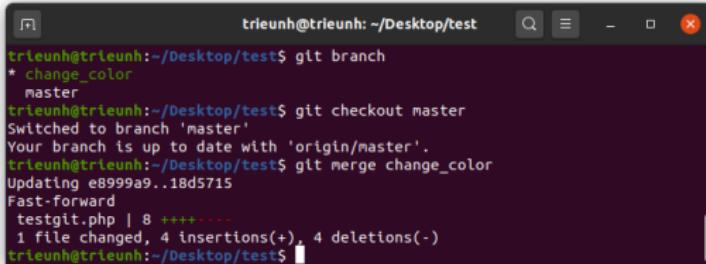
The bottom half of the window shows a diff view between the "change_color" commit and the "Initial commit". The diff highlights changes in the file "testgit.php":

```
</head>
<body>
-<?php
-echo "hello world"
-?>
+<font style="color: #ff0a07">
+<?php echo "hello world";?>
+<font>
</body>
-</html>
```

Các lệnh Git cơ bản

Hiện tại source ở nhánh **master** vẫn chưa thay đổi. Sau khi người quản lý dự án kiểm tra source ở nhánh **change_color** đạt yêu cầu thì tiến hành hợp nhất sources giữa 2 nhánh bằng các lệnh

- **git checkout master** → để đưa nhánh **change_color** về nhánh **master**
- **git merge [branch name]** → lệnh này sẽ merge **branch master** có trước đó vào vị trí nhánh hiện tại **change_color** theo phương thức fast-forward.



```
trieunh@trieunh:~/Desktop/test$ git branch
* change_color
  master
trieunh@trieunh:~/Desktop/test$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
trieunh@trieunh:~/Desktop/test$ git merge change_color
Updating e8999a9..18d5715
Fast-forward
 testgit.php | 8 ++++++--
 1 file changed, 4 insertions(+), 4 deletions(-)
trieunh@trieunh:~/Desktop/test$
```

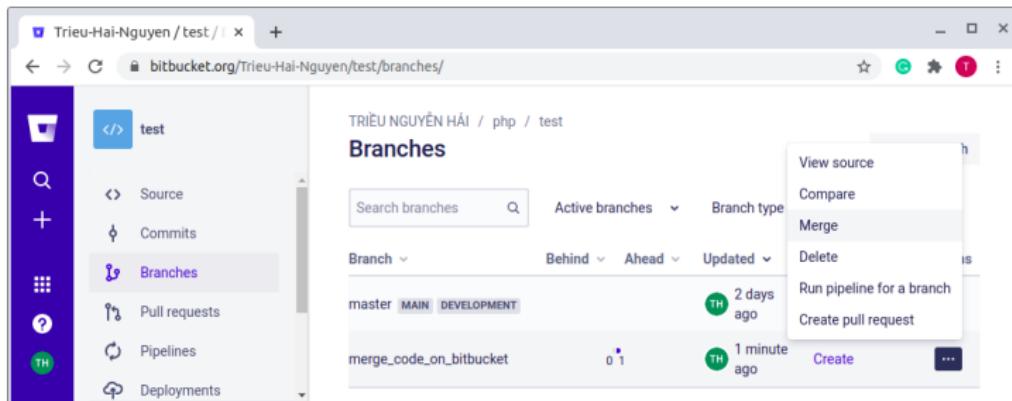
Các lệnh Git cơ bản

Author	Commit	Message
TH TRIỀU NGUYỄN HẢI	24e8239	MERGED Merged change_color into master
TH TRIỀU NGUYỄN HẢI	18d5715	change color
TH TRIỀU NGUYỄN HẢI	e8999a9	update01
TH TRIỀU NGUYỄN HẢI	376bb00	Initial commit

Hình 20: Kết quả sau khi merged hai nhánh `change_color` và `master`

Các lệnh Git cơ bản

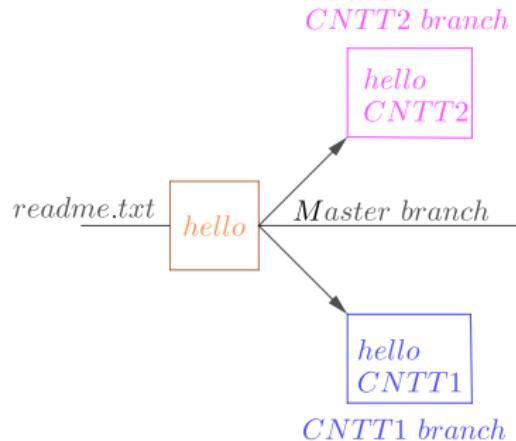
Ngoài ra chúng ta có thể merge trực tiếp từ UI của các Git servers như Github, Bitbucket ...



Hình 21: Ví dụ merge nhánh `merge_code_on_bitbucket` vào nhánh `master` trên `bitbucket`

Các lệnh Git cơ bản

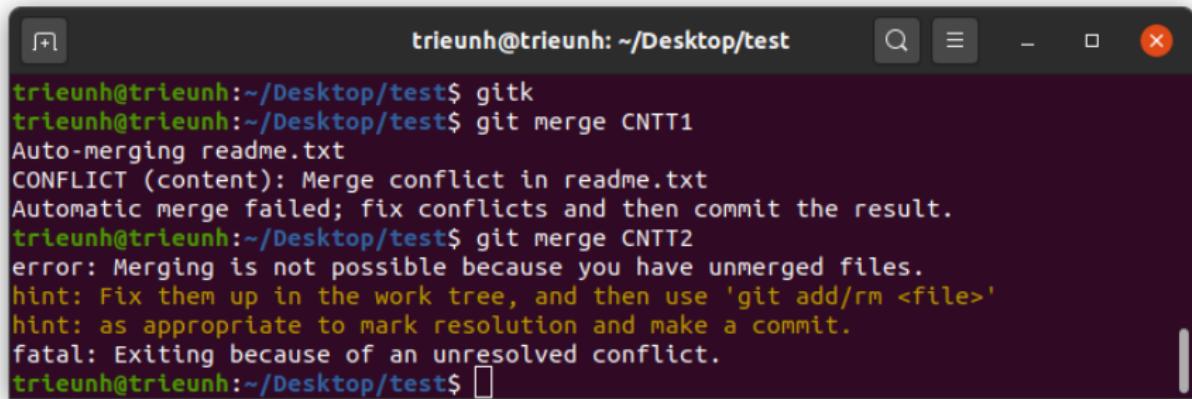
Lưu ý, thông thường khi merge source sẽ xuất hiện độ đụng độ giữa các branches làm việc song song. Ví dụ



Hình 22: Nhánh master hiện tại có file **readme.txt** với nội dung **hello**. Khi làm việc song song, nhánh CNTT1 được thêm vào hàng thứ hai là **CNTT1**, tương tự thêm **CNTT2** cho nhánh thứ 2

Các lệnh Git cơ bản

Khi tiến hành merge nhánh CNTT1 và CNTT2 sẽ xuất hiện lỗi đụng độ



A screenshot of a terminal window titled "trieunh@trieunh: ~/Desktop/test". The terminal shows the following command sequence and output:

```
trieunh@trieunh:~/Desktop/test$ gitk
trieunh@trieunh:~/Desktop/test$ git merge CNTT1
Auto-merging readme.txt
CONFLICT (content): Merge conflict in readme.txt
Automatic merge failed; fix conflicts and then commit the result.
trieunh@trieunh:~/Desktop/test$ git merge CNTT2
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'.
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
trieunh@trieunh:~/Desktop/test$
```

Hình 23: Lỗi: Merge conflict in readme.txt

Nguyên nhân là do cùng một dòng số hai trong file `readme.txt`, chúng ta thêm vô hai thành phần khác nhau là **CNTT1** và **CNTT2**

Các lệnh Git cơ bản

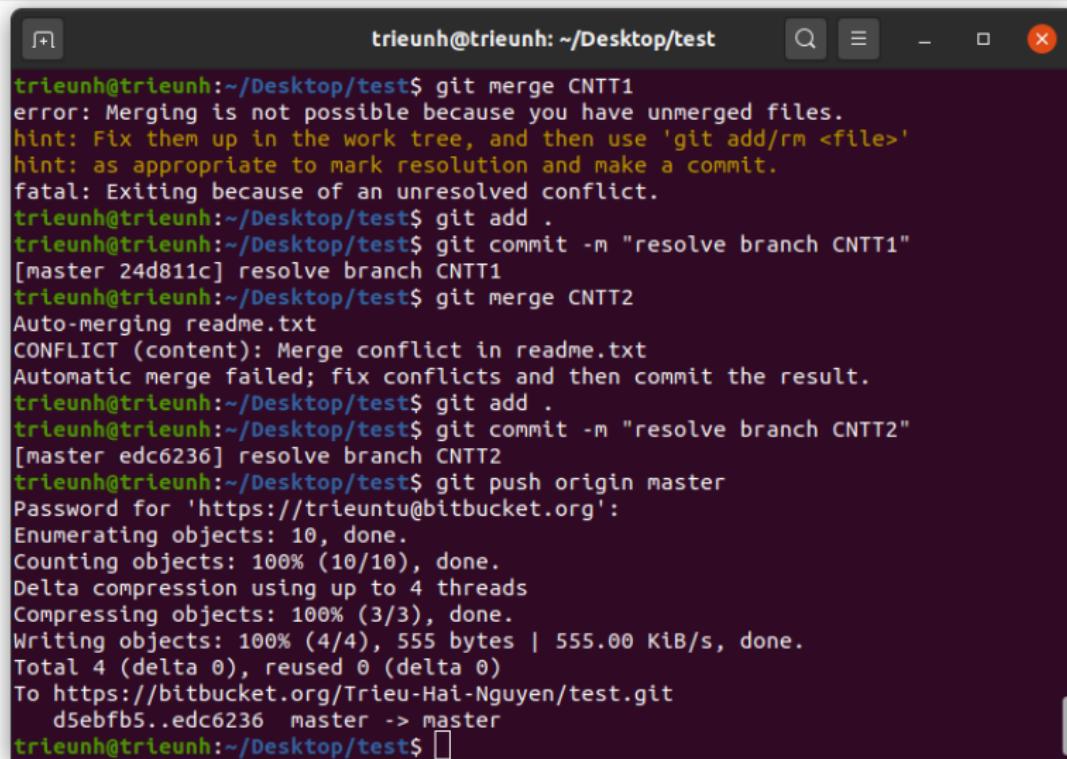
Lúc này trong file *readme.txt* sẽ có dạng như thế này

```
1 hello
2 <<<<< HEAD
3 CNTT1
4 =====
5 CNTT2
6 >>>>> CNTT2
```

Hướng giải quyết:

- chỉ cần xóa các kí tự ở chỗ có xung đột mà Git chèn vào: <<<< HEAD/ ===== / >>>> CNTT2.
- tiến hành add, commit lại và push source lên server.

Các lệnh Git cơ bản

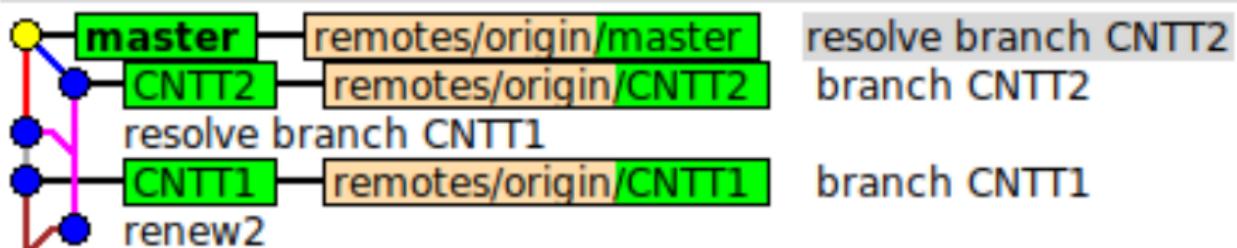


```
trieunh@trieunh:~/Desktop/test$ git merge CNTT1
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
trieunh@trieunh:~/Desktop/test$ git add .
trieunh@trieunh:~/Desktop/test$ git commit -m "resolve branch CNTT1"
[master 24d811c] resolve branch CNTT1
trieunh@trieunh:~/Desktop/test$ git merge CNTT2
Auto-merging readme.txt
CONFLICT (content): Merge conflict in readme.txt
Automatic merge failed; fix conflicts and then commit the result.
trieunh@trieunh:~/Desktop/test$ git add .
trieunh@trieunh:~/Desktop/test$ git commit -m "resolve branch CNTT2"
[master edc6236] resolve branch CNTT2
trieunh@trieunh:~/Desktop/test$ git push origin master
Password for 'https://trieunh@bitbucket.org':
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 555 bytes | 555.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://bitbucket.org/Trieu-Hai-Nguyen/test.git
    d5ebfb5..edc6236  master -> master
trieunh@trieunh:~/Desktop/test$
```

Hình 24: Các bước giải quyết xung đột giữa các branches

Các lệnh Git cơ bản

Kết quả



Hình 25: Đã hợp nhất được hai nhánh song song CNTT1 và CNTT2

Đóng góp vào repository thông qua Git

Các bước sử dụng Git ở trên có phạm vi áp dụng cho một dự án nhất định. Vậy câu hỏi đặt ra là đối với **dự án mã nguồn mở**, một thành viên bất kì trong cộng đồng quan tâm đến dự án đó thì sẽ **đóng góp vào repository** đó thông qua Git như thế nào? Câu trả lời phổ biến đối với các hệ thống DVCSs là:

- ① Tiến hành **fork repository** dự án nguồn mở đó từ trang gốc của dự án.
- ② Sử dụng các lệnh git cơ bản ở trên để **tạo branch** trên local của bạn, **đóng góp vào repo**.
- ③ Sau khi xong thì sử dụng **pull request** đến repository dự án nguồn mở.
- ④ Đợi người quản trị của dự nguồn mở xem, xét duyệt, và **merge** nếu đáp ứng được yêu cầu đề ra.

Fork repository

Bước 1: Ví dụ sử dụng Git server là Bitbucket để fork repository

The screenshot shows a Bitbucket repository page for 'trieu-hai-nguyen/test'. The left sidebar has navigation links for Repository, Workspace, Project, Snippet, IMPORT, and GET TO WORK. Under GET TO WORK, there are buttons for Clone this repository, Create a branch, Create a pull request, Compare branches or tags, and Fork this repository. The main content area shows repository details: Last updated 3 hours ago, Open pull requests 0, Branches 6, Watchers 1, Forks 1, Version control system Git, and Access level Admin. It also includes instructions for cloning the repository via git clone https://trieu-hai-nguyen/test.git and using Sourcetree. A file list table is partially visible at the bottom.

Branch

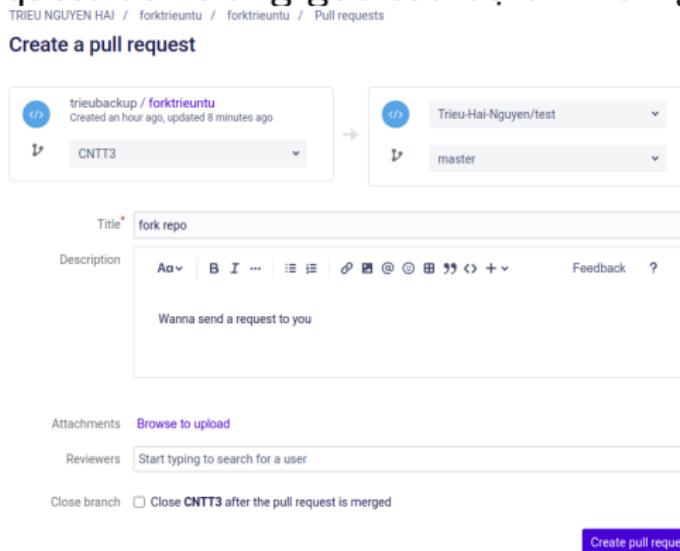
Bước 2: Tạo nhánh ở local để đóng góp vào repo

Author	Commit	Message	
TH TRIEUV NGUYEN HAI	0c07551	fork repo	↳ CNTT3
TH TRIỀU NGUYỄN HẢI	edc6236	MERGED resolve branch CNTT2	
TH TRIỀU NGUYỄN HẢI	24d811c	MERGED resolve branch CNTT1	
TH TRIỀU NGUYỄN HẢI	c70707f	branch CNTT2	
TH TRIỀU NGUYỄN HẢI	d5ebfb5	renew2	
TH TRIỀU NGUYỄN HẢI	fe85ac7	branch CNTT1	
TH TRIỀU NGUYỄN HẢI	698eb7e	parallel_CNTT2	

Pull request

Bước 3:

- mở lại repository của dự án mã nguồn mở trên Git server mà bạn đã fork về ([Pull requests](#) → [create pull request](#))
- tạo Pull request đến trang gốc của dự án mã nguồn mở



Hình 26: Ví dụ tạo Pull requests từ Git server Bitbucket.

The project maintainer's side

Bước 4: Yêu cầu từ thành viên sẽ [gửi đến](#) người quản trị để xét duyệt xem có được [merge source](#) của bạn không. Nếu [được chấp nhận](#) thì bạn chính thức trở thành một [contributor](#) của dự án.

TRIỀU NGUYỄN HẢI / php / test

Pull requests

Create pull request

A screenshot of a GitHub pull request interface. At the top, there is a search bar labeled "Search pull requests" with a magnifying glass icon, and dropdown menus for "Open", "Author", and "Target branch". A button labeled "Create pull request" is visible. Below this, a message says "I'm reviewing". The main area shows a single pull request card. The card has a user icon with the letters "TH", the text "fork repo → master", the author name "TRIEU NGUYEN HAI", the pull request number "#1", and the creation and update times ("created 4 minutes ago, updated 4 minutes ago"). To the right of the card, it says "No reviewers". At the bottom of the card, there are tabs for "Summary", "Activity", "Reviewers", and "Builds".

Hình 27: Người quản trị sẽ nhận được tin nhắn pull request như trên.

The project maintainer's side

Kết quả sau khi được chấp nhận

TRIỀU NGUYỄN HẢI / php / test

Commits

Author	Commit	Message
TRIỀU NGUYỄN HẢI	23993b3	MERGED Merged in trieubackup/forktrieuntu/CNTT3 (pull request #1) for...
TRIỀU NGUYỄN HẢI	0c07551	fork repo
TRIỀU NGUYỄN HẢI	edc6236	MERGED resolve branch CNTT2
TRIỀU NGUYỄN HẢI	24d811c	MERGED resolve branch CNTT1
TRIỀU NGUYỄN HẢI	c70707f	branch CNTT2
TRIỀU NGUYỄN HẢI	d5ebfb5	renew2

Bài tập thực hành

- Sinh viên chia làm các nhóm (tối đa 5 người/1 nhóm), trong đó có một người làm quản lý dự án (team leader). Người quản lý dự án sẽ tạo một project trên một Git server bất kỳ mô phỏng một dự án mã nguồn mở. Các thành viên còn lại có nhiệm vụ tạo ra các branchs và thực hiện đóng góp của mình vào dự án. Người quản lý dự án có nhiệm vụ duyệt các đóng góp và merge source vào nhánh master.
- Ví dụ, team leader tạo ra file *readme.txt* trong đó chứa tên của trưởng nhóm, các thành viên khác có thể đóng góp tên của mình vào file đó.

- 1 Phần mềm và vấn đề bản quyền phần mềm
- 2 Mô hình phát triển phần mềm mã nguồn mở
- 3 **Cách đóng góp vào phát triển phần mềm mã nguồn mở**

Contributing to open source software

- Trong các dự án phát triển phần mềm mã nguồn mở, các nhà phát triển và người dùng từ khắp mọi nơi trên thế giới cùng nhau tham gia phát triển. Vậy câu hỏi đặt ra là **làm sao để đóng góp vào các dự án mã nguồn mở?**.
- Nhiều người thường có suy nghĩ rằng đóng góp cho các dự án nguồn mở thường như **quá khó khăn và tốn thời gian**.
- Tuy nhiên, **bất cứ ai cũng có thể đóng góp**, để lại dấu ấn của mình vào sự phát triển của phần mềm mã nguồn mở.
- Đóng góp ở đây **không chỉ đơn thuần là trực tiếp tham gia vào code một dự án nguồn mở**. Có nhiều cách đóng góp như:
 - ① **Tìm lỗi** của phần mềm nguồn mở bạn quan tâm (Find Bugs)
 - ② **Tìm hiểu và sửa** các vấn đề hiện có (Explore existing Issues)
 - ③ Giúp **cải thiện** các tài liệu liên quan đến dự án (Help improve documentation)
 - ④ Đề xuất các **tính năng mới** cho phần mềm (Suggest new features)

Find Bugs

Báo cáo các lỗi xảy ra đối với phần mềm là cách đơn giản và nhanh nhất để đóng góp vào sự phát triển của phần mềm.

- Cài đặt và test phần mềm. Trong quá trình sử dụng thỉnh thoảng sẽ xuất hiện một số lỗi.
- Cần nhanh chóng báo cáo lỗi đến “project's issue tracker”. Lưu ý trước khi báo cáo lỗi, cần tìm hiểu xem lỗi của bạn gấp phải đã được báo cáo chưa, hoặc một số cách khắc phục lỗi đã được cộng đồng phát triển hướng dẫn.
- Đôi khi, các lỗi xảy ra không đến từ mã nguồn mà lại đến từ tài liệu hướng dẫn (tài liệu hướng dẫn lỗi thời, sai chính tả, ý nghĩa cách dụng, . . .), chúng cần được báo cáo đến hệ thống theo dõi của phần mềm.

Explore existing Issues

Khi tìm được dự án mã nguồn mở mà bạn quan tâm muốn đóng góp, có thể **khám phá kho lưu trữ Github** của dữ án. Đầu tiên, hãy đọc tài liệu hướng dẫn và đi đến đến các **issues tab**:

- Ở đó bạn có thể thấy được các vấn đề phát sinh được người dùng, cộng đồng phát triển báo cáo
- hãy chọn những **vấn đề đang mở (Open)** phù hợp với chuyên môn của bạn nhất và giải quyết nó
- nếu giải quyết được issue đó, người quản lý dự án sẽ xem xét và **merge code** của bạn vào dự án. Bạn sẽ **trở thành một "Contributor"**.

Help improve documentation

Rất ít nhà phát triển sẵn sàng viết tài liệu hướng dẫn, nhưng nó rất quan trọng đối với sự thành công của các dự án nguồn mở. Bạn có thể đóng góp tài liệu:

- **cho nhà phát triển.** Đối với các dự án liên quan đến lập trình, chẳng hạn như thư viện hoặc framework, hãy giải thích trường hợp sử dụng chung của code.
- **cho người sử dụng.** Bạn có thể viết hướng dẫn sử dụng phần mềm với phiên bản hiện tại cho người dùng dễ dàng sử dụng.

Suggest new features

- Nếu bạn có thể **đề xuất các tính năng hữu ích cho phần mềm** thì đừng ngần ngại gửi yêu cầu. Lưu ý, trước khi đề xuất một tính năng mới cần tìm hiểu xem đề xuất tương tự đã được gửi đến dự án mã nguồn mở chưa.
- Có thể **đề xuất** tính năng về **code** hoặc **tài liệu** hướng dẫn liên quan đến dự án