**CHỦ ĐỀ 1**

**PHẦN MỀM GIT**

1. **GIỚI THIỆU PHẦN MỀM GIT**



1. Git là một trong những hệ thống kiểm soát phiên bản dùng để theo dõi các thay đổi trong tập tin, nó được dùng để thay đổi trong các tập tin bất kỳ và nó được sử dụng chính trong các lĩnh vực phát triển [kỹ thuật phần mềm](https://freelancervietnam.vn/ky-thuat-phan-mem-la-gi-sinh-vien-hoc-ky-thuat-phan-mem-se-lam-gi/).
2. Git được hiểu là hệ thống điều khiển về mặt cơ bản nó là một trình theo dõi nội dung vì thế Git có thể sử dụng được để lưu trữ nội dung chủ yếu sử dụng để lưu trữ các mã do tính năng khác mà nó cung cấp.
3. Git được coi là một hệ thống điều khiển phân tán có tốc độ xử lý nhanh những vẫn đảm bảo được toàn vẹn dữ liệu và hỗ trợ hiệu quả cao cho các workflow phân tán. Giống như các hệ thống quản lý phiên bản phân tán khác, mọi thư mục Git trên máy tính đều là một kho chứa lịch sử . Gồm có kho lưu trữ từ xa trong máy chủ và một kho lưu trữ cục bộ được lưu trữ trong máy tính của nhà phát triển.

**GIT sẽ giúp bạn:**

* Lưu lại được các phiên bản khác nhau của mã nguồn dự án phần mềm
* Khôi phục lại mã nguồn từ một phiên bản bất kỳ
* Dễ dàng so sánh giữa các phiên bản
* Phát hiện được ai đã sửa phần nào làm phát sinh lỗi
* Khôi phục lại tập tin bị mất
* Dễ dàng thử nghiệm, mở rộng tính năng của dự án mà không làm ảnh hưởng đến phiên bản chính (master branch)
* Giúp phối hợp thực hiện dự án trong nhóm một cách hiệu quả.

1. **Các phiên bản của phần mềm**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **phiên bản** | **Ngày phát hành ban đầu** | **Phiên bản (bản vá) mới nhất** | **Ngày phát hành (của bản vá)** | **Những thay đổi đáng chú ý** |
| 0,99 | 2005-07-11 | 0,99,9n | 2005-12-15 |  |
| 1,0 | 2005-12-21 | 1.0.13 | 2006-01-27 |  |
| 1.1 | 2006-01-08 | 1.1.6 | 2006-01-30 |  |
| 1,2 | 2006-02-12 | 1.2.6 | 2006-04-08 |  |
| 1,3 | 2006-04-18 | 1.3.3 | 2006-05-16 |  |
| 1,4 | 2006-06-10 | 1.4.4.5 | 2008-07-16 |  |
| 1,5 | 2007-02-14 | 1.5.6.6 | 2008-12-17 |  |
| 1,6 | 2008-08-17 | 1.6.6.3 | 2010-12-15 |  |
| 1,7 | 2010-02-13 | 1.7.12.4 | 2012-10-17 |  |
| 1,8 | 2012-10-21 | 1.8.5.6 | 2014-12-17 |  |
| 1,9 | 2014-02-14 | 1.9.5 | 2014-12-17 |  |
| 2.0 | 2014-05-28 | 2.0.5 | 2014-12-17 |  |
| 2.1 | 2014-08-16 | 2.1.4 | 2014-12-17 |  |
| 2,2 | 2014-11-26 | 2.2.3 | 2015-09-04 |  |
| 2.3 | 2015-02-05 | 2.3.10 | 2015-09-29 |  |
| 2,4 | 2015-04-30 | 2.4.12 | 2017-05-05 |  |
| 2,5 | 2015-07-27 | 2.5.6 | 2017-05-05 |  |
| 2,6 | 2015-09-28 | 2.6.7 | 2017-05-05 |  |
| 2,7 | 2015-10-04 | 2.7.6 | 2017-07-30 |  |
| 2,8 | 2016-03-28 | 2.8.6 | 2017-07-30 |  |
| 2,9 | 2016-06-13 | 2.9.5 | 2017-07-30 |  |
| 2,10 | 2016-09-02 | 2.10.5 | 2017-09-22 |  |
| 2,11 | 2016-11-29 | 2.11.4 | 2017-09-22 |  |
| 2,12 | 2017-02-24 | 2.12.5 | 2017-09-22 |  |
| 2,13 | 2017-05-10 | 2,13,7 | 2018-05-22 |  |
| 2,14 | 2017-08-04 | 2,14,6 | 2019-12-07 |  |
| 2,15 | 2017-10-30 | 2.15.4 | 2019-12-07 |  |
| 2,16 | 2018-01-17 | 2,16,6 | 2019-12-07 |  |
| 2,17 | 2018-04-02 | 2,17,6 | 2021-03-09 |  |
| 2,18 | 2018-06-21 | 2.18.5 | 2021-03-09 |  |
| 2,19 | 2018-09-10 | 2,19,6 | 2021-03-09 |  |
| 2,20 | 2018-12-09 | 2,20,5 | 2021-03-09 |  |
| 2,21 | 2019-02-24 | 2.21.4 | 2021-03-09 |  |
| 2,22 | 2019-06-07 | 2,22,5 | 2021-03-09 |  |
| 2,23 | 2019-08-16 | 2,23,4 | 2021-03-09 |  |
| 2,24 | 2019-11-04 | 2,24,4 | 2021-03-09 |  |
| 2,25 | 2020-01-13 | 2,25,5 | 2021-03-09 | Dễ dàng quản lý thanh toán thưa thớt |
| 2,26 | 2020-03-22 | 2.26.3 | 2021-03-09 | * Giao thức phiên bản 2 hiện là mặc định * Một số thủ thuật cấu hình mới * Nội dung cập nhật đối với git Slice-checkout |
| 2,27 | 2020-06-01 | 2.27.1 | 2021-03-09 |  |
| 2,28 | 2020-07-27 | 2.28.1 | 2021-03-09 | * Giới thiệu init.defaultBranch * Bộ lọc Bloom đường dẫn đã thay đổi |
| 2,29 | 2020-10-19 | 2.29.3 | 2021-03-09 | * Hỗ trợ SHA-256 thử nghiệm * Refspec phủ định * Thủ git shortlogthuật mới |
| 2,30 | 2020-12-27 | 2.30.2 | 2021-03-09 | * Userdiff cho cập nhật PHP, Rust, cập nhật CSS * Tập lệnh hoàn thành dòng lệnh (trong Contrib /) đã biết rằng "git stash show" nhận các tùy chọn "git diff". |
| 2,31 | 2021-03-15 | 2.31.1 | 2021-04-02 | * git difftoolthêm --skip-totùy chọn * --format cải tiến để máy có thể đọc được * git pull cảnh báo để chỉ định rebase hoặc hợp nhất |
| 2,32 | 2021-06-06 |  |  |  |
| **2,33** | 2021-08-16 |  |  |  |
| **Chú thích:** | | | | |

1. **Giấy phép**

**Git** là [phần mềm quản lý mã nguồn](https://vi.wikipedia.org/w/index.php?title=Ph%E1%BA%A7n_m%E1%BB%81m_qu%E1%BA%A3n_l%C3%BD_m%C3%A3_ngu%E1%BB%93n&action=edit&redlink=1) [phân tán](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A2n_t%C3%A1n&action=edit&redlink=1) được phát triển bởi [Linus Torvalds](https://vi.wikipedia.org/wiki/Linus_Torvalds) vào năm [2005](https://vi.wikipedia.org/wiki/2005), ban đầu dành cho việc phát triển [nhân Linux](https://vi.wikipedia.org/wiki/H%E1%BA%A1t_nh%C3%A2n_Linux). Hiện nay, Git trở thành một trong các phần mềm quản lý [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n) phổ biến nhất. Git là phần mềm mã nguồn mở được phân phối theo giấy phép công cộng [GPL2](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_C%C3%B4ng_c%E1%BB%99ng_GNU).

**Giấy phép Công cộng GNU** ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): *GNU General Public License*, viết tắt **GNU GPL** hay chỉ **GPL**) là [giấy phép phần mềm tự do](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_ph%E1%BA%A7n_m%E1%BB%81m_t%E1%BB%B1_do) được sử dụng rộng rãi, đảm bảo cho người dùng cuối tự do chạy, nghiên cứu, sửa đổi và chia sẻ phần mềm. Giấy phép ban đầu được viết bởi [Richard Stallman](https://vi.wikipedia.org/wiki/Richard_Stallman) của [Quỹ Phần mềm Tự do](https://vi.wikipedia.org/wiki/Qu%E1%BB%B9_Ph%E1%BA%A7n_m%E1%BB%81m_T%E1%BB%B1_do) (FSF) cho [Dự án GNU](https://vi.wikipedia.org/wiki/GNU), và cấp cho người nhận chương trình máy tính quyền của [Định nghĩa Phần mềm Tự do](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%8Bnh_ngh%C4%A9a_Ph%E1%BA%A7n_m%E1%BB%81m_T%E1%BB%B1_do&action=edit&redlink=1). GPL là giấy phép copyleft, có nghĩa là tác phẩm phái sinh chỉ có thể được phân phối theo các điều khoản cấp phép tương tự. Đây là sự phân biệt đối với [giấy phép phần mềm tự do](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_ph%E1%BA%A7n_m%E1%BB%81m_t%E1%BB%B1_do) cho phép, trong đó [giấy phép BSD](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_BSD) và [Giấy phép MIT](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_MIT) được sử dụng rộng rãi là ví dụ. GPL là giấy phép copyleft đầu tiên để sử dụng chung.

Trong lịch sử, gia đình giấy phép GPL là một trong những giấy phép phần mềm phổ biến nhất trong lĩnh vực phần mềm tự do và nguồn mở. Các chương trình phần mềm miễn phí nổi bật được cấp phép theo GPL bao gồm [nhân Linux](https://vi.wikipedia.org/wiki/H%E1%BA%A1t_nh%C3%A2n_Linux) và [Bộ biên dịch GNU](https://vi.wikipedia.org/wiki/B%E1%BB%99_tr%C3%ACnh_d%E1%BB%8Bch_GNU) (GCC). David A. Wheeler cho rằng copyleft được cung cấp bởi GPL là rất quan trọng đối với sự thành công của các hệ thống dựa trên Linux, giúp các lập trình viên đóng góp cho hạt nhân sự đảm bảo rằng công việc của họ sẽ mang lại lợi ích cho toàn thế giới. các công ty phần mềm sẽ không phải trả lại cho cộng đồng.

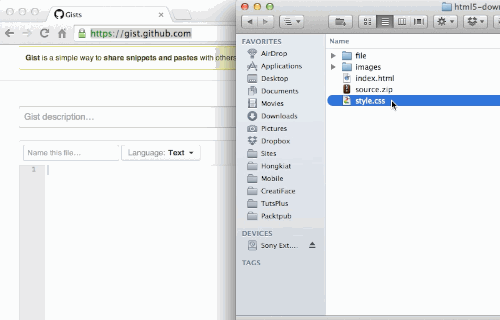
Trong năm 2007, phiên bản thứ ba của giấy phép (GNU GPLv3) đã được phát hành để giải quyết một số vấn đề nhận thức với phiên bản thứ hai (GNU GPLv2) đã được phát hiện trong quá trình sử dụng lâu dài của nó. Để giữ cho giấy phép cập nhật, giấy phép GPL bao gồm một điều khoản "bất kỳ phiên bản sau" tùy chọn, cho phép người dùng lựa chọn giữa các điều khoản gốc hoặc các điều khoản trong các phiên bản mới như được [FSF](https://vi.wikipedia.org/wiki/Qu%E1%BB%B9_Ph%E1%BA%A7n_m%E1%BB%81m_T%E1%BB%B1_do) cập nhật. Các nhà phát triển có thể bỏ qua nó khi cấp phép phần mềm của họ; ví dụ hạt nhân Linux được cấp phép theo GPLv2 mà không có mệnh đề "bất kỳ phiên bản nào sau này".

**IV. Các tính năng, đặc điểm , kiến trúc của phần mềm git**

1. **Tính năng**

### ****1. Kéo và thả code với Git****

Gist là 1 phần riêng biệt của Github, cho phép lưu trữ các đoạn code. Cũng có thể duyệt và tìm một số lượng lớn các đoạn mã của nhiều ngôn ngữ khác nhau. Sử dụng Gist hoàn toàn dễ dàng và phải trực quan. Cách có thể thêm code trực tiếp từ các tệp tin trong máy tính của mình .Chỉ cần kéo và thả các tệp từ máy tính vào Gist, các đoạn code trong các tệp sẽ được sao chép ngay lập tức.



### 2. Tạo một thư mục thông qua Giao diện Web

Trong khi nhiều người trong chúng ta có thể quản lý repo Github thông qua ứng dụng Github miễn phí. Ngoài ra, Github cũng đã xây dựng những gì họ gọi là WebFlow. Nó cho phép chúng ta quản lý repo thông qua giao diện web của Github.



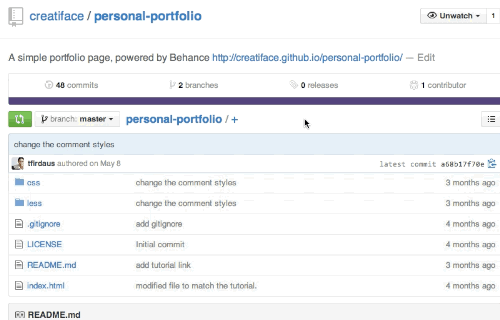
### 3. Sử dụng Git URL Shortener

Ngày nay, mọi người thích chia sẻ mọi thứ từ hình ảnh, trạng thái và tin tức của họ trên mạng xã hội. Nếu bạn là người dùng Github, bạn cũng có thể muốn chia sẻ repo Github của mình. Tuy nhiên, URL của repo đôi khi quá dài để chia sẻ, ví dụ như Twitter chỉ chấp nhận URL chứa tối đa 140 ký tự.Chắc chắn có rất nhiều tùy chọn để rút ngắn URL ví dụ như [Bit.ly](http://bit.ly/) và [Goo.gl](http://goo.gl/), nhưng tại sao không cân nhắc sử dụng 1 trong các chức năng tương tự như vậy của Github là [Git.io](http://git.io/)? [Git.io](https://git.io/) sẽ rút ngắn URL của repo Github của bạn.



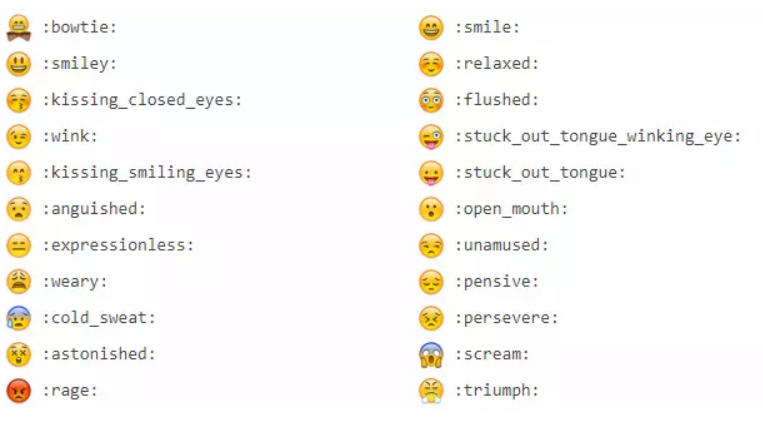
### 4. Tìm kiếm tập tin và file

Bên cạnh việc tạo các tệp mới, bạn cũng có thể điều hướng và tìm kiếm nhanh chóng qua các tệp trong bất kỳ repo nào. Tính năng này rất hữu ích nhưng lại không rõ ràng vì nó bị ẩn và chỉ hoạt động khi dùng phím tắt.Nhấn phím T để kích hoạt Trình tìm kiếm tệp. Nhấn ↑ và ↑ nhảy qua các tệp lên và xuống. Hoặc, nhập tên tệp để chọn một tệp cụ thể mà bạn đã có.



### 5. Sử dụng Github Emoji

Tương tự như Facebook hay một số mạng xã hội, Github cũng tích hợp 1 số Emoji để khi review, comment các đoạn code trên Github đỡ nhàm chán .Bạn có thể tìm tất cả các ký tự Emoji và mã trong [Bảng Cheat Emoji](https://www.webpagefx.com/tools/emoji-cheat-sheet/).



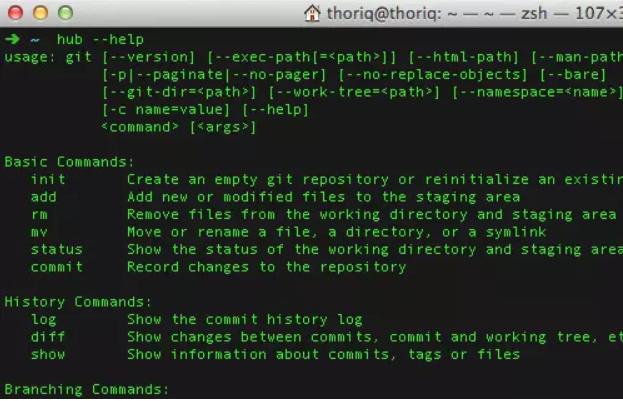
### 6. Sử dụng Github CLI

Trong khi hầu hết mọi người thích làm việc bằng cách sử dụng GUI, vẫn có một số người thích sử dụng CLI (Giao diện dòng lệnh). Github CLI được bắt đầu với từ khoá hub. Github CLI bổ sung các cú pháp có thể được sử dụng cùng với các cú pháp của git. Danh sách đầy đủ các tính năng có thể được tìm thấy tại [Hub](https://github.com/github/hub) repo. Ví dụ:

$ hub clone rtomayko/tilt

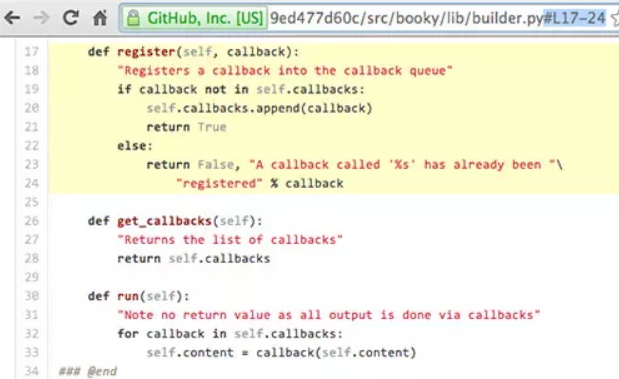
sẽ expands thành:

$ git clone git://github.com/rtomayko/tilt.git



### 7. Linking Lines

Đôi khi, bạn có thể muốn chia sẻ và chỉ ra các dòng cụ thể trong 1 file bất kỳ trên git repo của bạn . Github cho phép bạn làm điều này bằng cách thêm tham số #L tiếp theo là số dòng ở cuối URL của file (xem ví dụ bên dưới).

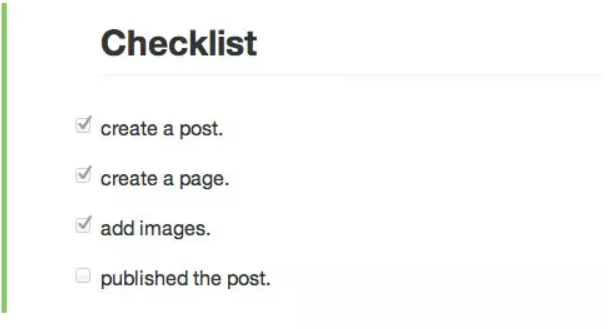


### 8. Task Checklist

Github tích hợp sẵn rất nhiều các cú pháp markdown . Bạn có thể thêm một list các checkboxes trong pull request bằng cách sử dụng - [] hoặc - [x] để biểu thị một mục đã chọn. Xin lưu ý rằng checkboxes sẽ chỉ xuất hiện trong một list các item; các [] dấu hiệu phải được bắt đầu với một dấu gạch ngang -. Đây là một ví dụ:

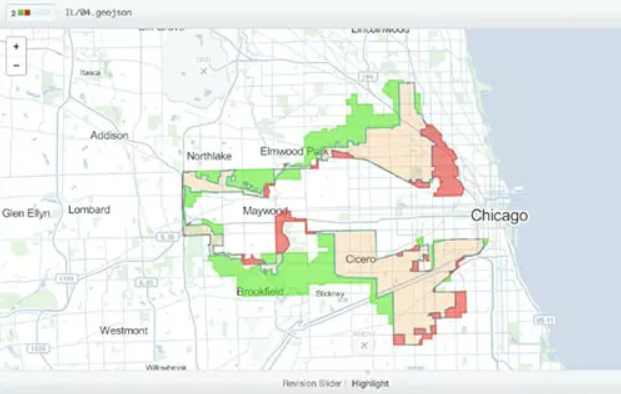
* [x] create a post.
* [x] create a page.
* [x] add images.
* [ ] published the post.

Sau khi create pull sẽ trông như thế này:



### 9. Maps, CSV và 3D Rendering

Gihub hỗ trợ file **CSV**. Nếu bạn include một tệp .csv, Github sẽ hiển thị tệp CSV của bạn thành định dạng dữ liệu bảng. Nó thậm chí còn cho phép bạn tìm kiếm thông qua nó. Ngoài CSV, Github cũng sẽ tự động hiển thị Bản đồ với định dạng **geoJSON** và 3D với phần mở rộng **STL** .



### 10. Get Octodex

Cuối cùng nhưng không kém phần quan trọng, bạn có biết rằng Github có nhiều phiên bản linh vật của nó, Octocat? Google có Doodle, trong khi Github có [Octodex](https://octodex.github.com/). Octodex là một tập hợp các phiên bản sáng tạo thay thế cho Octocat. Ở đó, bạn có thể tìm thấy Labtocat, Femalecodertocat, Octoliberty, Spidertocat, Megacat, và một loạt các Octocats thú vị khác. Bạn có thể sử dụng Octodex làm hình đại diện cá nhân.

1. **Kiến trúc phần mềm**

## 1. Kiến trúc phần mềm là gì

Một phần mềm (software) được tạo nên bởi rất nhiều các thành phần khác nhau. Mỗi thành phần lại nắm một vai trò và nhiệm vụ cụ thể. Để biết cách xây dựng, thay đổi, hay nâng cấp software như thế nào, chúng ta cần đến KTPM.

Vậy, ta có thể hiểu, kiến trúc phần mềm (Software Architecture) là kiến trúc giúp các Software Engineer dễ dàng hình dung các thành phần của phần mềm và cách chúng móc xích để tạo nên sản phẩm.

## 2. Vai trò của kiến trúc phần mềm

KTPM đóng vai trò quan trọng trong việc đảm bảo chất lượng, hiệu quả vận hành và sự thành công của một sản phẩm phần mềm. Nếu không nắm chắc và liên tục rà soát, xem xét lỗ hổng trong KTPM, rồi đưa ra những biện pháp cải thiện phù hợp, các developer sẽ dễ gặp phải những hậu quả nghiêm trọng như phần mềm bị hack, vận hành kém,…

Hiện nay, có nhiều mẫu KT được sử dụng để xây dựng phần mềm. KTPM không bị bó buộc trong một mẫu nhất định. Thay vào đó, các Engineer sẽ kết hợp nhiều mẫu kiến trúc để tạo nên một hệ thống hoàn chỉnh.



Kiến trúc phần mềm mô tả cấu trúc của một hệ thống

### Ưu điểm của mẫu kiến trúc phần mềm:

* Các mẫu KTPM giúp việc đưa ra quyết định và quản lý mô hình của phần mềm trở nên dễ dàng hơn.
* Cung cấp giải pháp có thể tái sử dụng đã được kiểm chứng cho các vấn đề khi xây dựng hệ thống
* Cơ hội tiết kiệm chi phí lớn và giảm rủi ro nếu sử dụng kiến ​​trúc phần mềm.
* Đạt được các tiêu chuẩn chất lượng cao hơn.

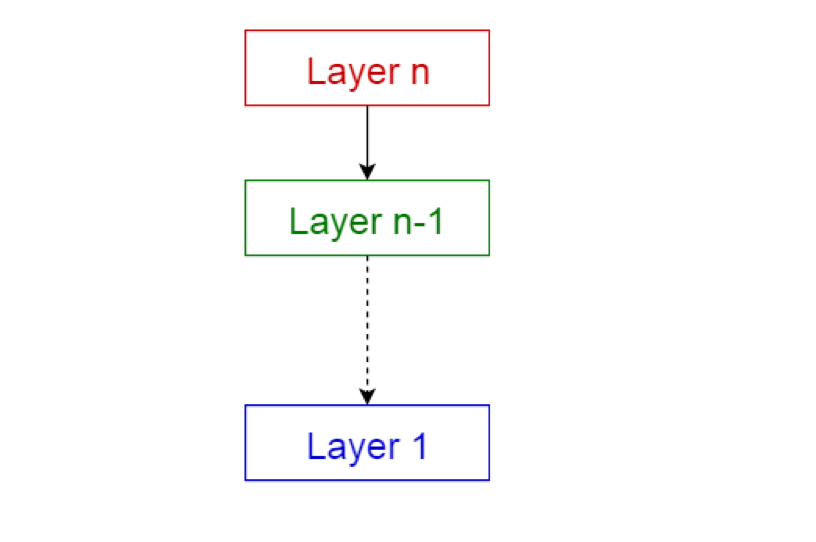
## 3. Các mẫu kiến trúc phần mềm phổ biến

Để tổ chức các hệ thống phần mềm, bạn có thể sử dụng các mẫu KTPM dưới đây:

### 3.1. Kiến trúc phân lớp – Layered (n-tier) Architecture

Mẫu KTPM này thường được sử dụng trong các hệ thống có thể được phân tách thành các nhóm gồm nhiều công việc nhỏ. Đặc điểm của mẫu này là:

* Các lớp khác nhau được xác định trong kiến ​​trúc. Nó bao gồm lớp bên ngoài và bên trong.
* Thành phần của lớp ngoài quản lý các hoạt động giao diện người dùng.
* Các thành phần thực thi giao diện hệ điều hành ở lớp bên trong.
* Các lớp bên trong là lớp ứng dụng, lớp tiện ích và lớp lõi.



Kiến trúc phân lớp

### 3.2. Kiến trúc hướng sự kiện – Event-driven architecture

Mẫu KT hướng sự kiện được sử dụng nhiều cho các sản phẩm phần mềm có khả năng scale (mở rộng quy mô) mạnh mẽ. Mẫu KTPM này rất dễ sử dụng. Dù là xây dựng các phần mềm nhỏ, đơn giản hay các phần mềm lớn, phức tạp, Engineer đều có thể cân nhắc sử dụng mẫu này.

KT hướng sự kiện được tạo thành từ các thành phần xử lý sự kiện (event processing components) giúp nhận và xử lý các sự kiện một cách không đồng bộ.

### 3.3. Kiến trúc hướng đối tượng – Objects-based Architecture

Đây là một mô hình KT dựa trên việc phân chia công việc cho một ứng dụng hoặc hệ thống thành các đối tượng có thể tái sử dụng và tự vận hành. KT hướng đối tượng xem một hệ thống phần mềm như một tập hợp các thực thể được gọi là các đối tượng.

1. **Đặc điểm**

Khi vào trang web chính thức của Git, chúng ta sẽ thấy ngay các đặc điểm chính của Git như: fast-version-control (quản lý phiên bản nhanh), distributed-even-if-your-workflow-isnt (phân tán cho dù quy trình làm việc của bạn không cần đến), distributed-is-the-new-centralized (phân tán cũng là một cách tập trung mới), local-branching-on-the-cheap (phân nhánh local với chi phí thấp), everything-is-local (mọi công việc đều thực hiện local). Hay có thể nói Git có các đặc điểm chính sau:

### Branching & Merging

* Nói về Git là phải nói về mô hình phân nhánh của Git (branching model). Git cho phép và khuyến khích chúng ta nên có nhiều nhánh cục bộ có thể làm việc hoàn toàn độc lập với nhau. Và việc tạo mới, gộp lại hay xóa các kết quả trong quá trình phát triển chỉ mất vài giây.
* Đáng chú ý là khi ta đưa vào một kho điều khiển từ xa (remote repository), ta không phải đưa vào tất cả các nhánh. Ta có thể chọn chỉ chia sẻ một, một vài hoặc tất cả trong số các nhánh đó. Điều này giúp cho mọi người cảm thấy thoải mái để tiếp tục thử nghiệm những ý tưởng mới mà không phải lo lắng về việc phải lập kế hoạch làm thế nào và khi nào họ sẽ gộp vào hoặc chia sẻ các nhánh cho người khác. Git làm cho quá trình này trở nên vô cùng dễ dàng.

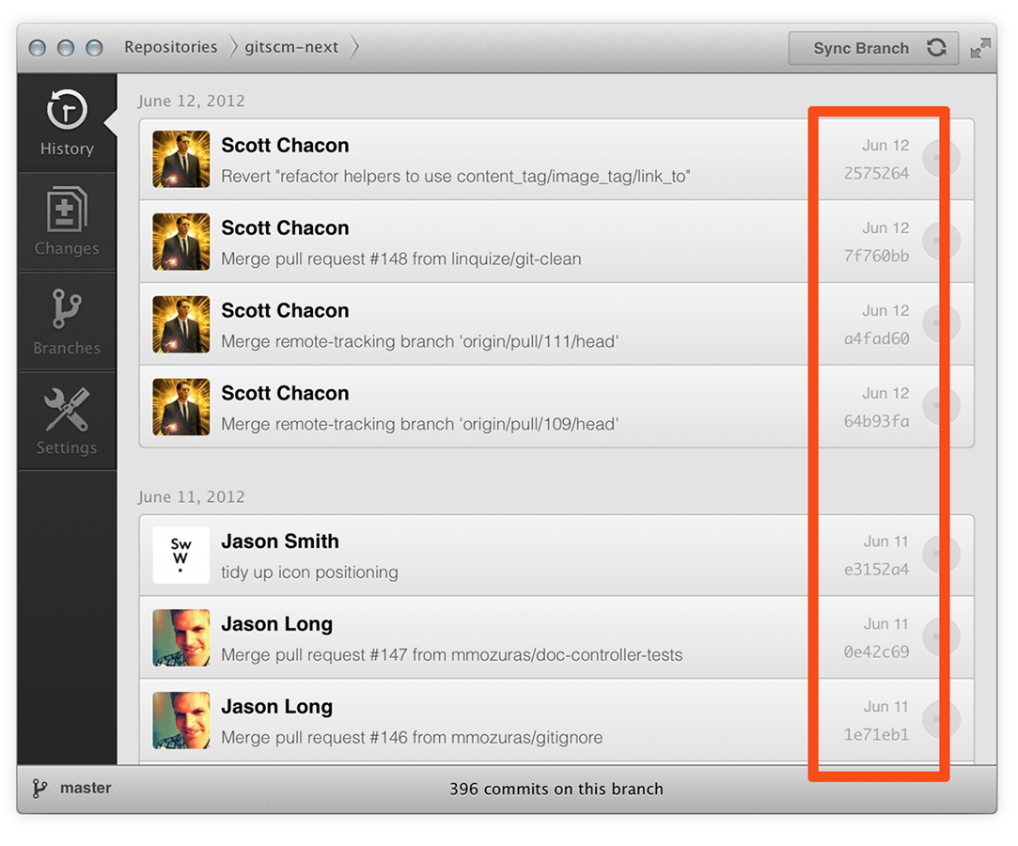
### Nhỏ gọn và nhanh chóng – Small & Fast

* Hầu hết các hoạt động trên Git được thực hiện cục bộ, điều đó giúp Git có một lợi thế rất lớn về tốc độ xử lý, nhất là trên các hệ thống tập trung thường xuyên phải giao tiếp với một máy chủ ở đâu đó.
* Git được xây dựng để làm việc trên Linux kernel, có nghĩa là Git có khả năng xử lý hiệu quả các kho lưu trữ (repository) lớn ngay từ giai đoạn đầu. Tốc độ và hiệu suất là mục tiêu thiết kế chính của Git.

### Phân tán - Distributed

* Distributed: Một trong những tính năng hấp dẫn nhất của Distributed SCM mà Git có đó là phân tán. Điều này có nghĩa là thay vì thực hiện “checkout” source code hiện tại, ta có thể “clone” toàn bộ kho lưu trữ (repository).
* Nhiều bản sao lưu (Multiple Backups): Điều này có nghĩa là ngay cả khi bạn đang sử dụng quy trình làm việc tập trung, mỗi người dùng đều có một bản sao lưu đầy đủ từ máy chủ. Bản sao này có thể được đẩy lên thay thế cho bản chính trên máy chủ trong trường hợp có sự cố xảy ra. Rõ ràng, nếu bạn có nhiều bản sao thì việc phục hồi sẽ an toàn và dễ thành công hơn rất nhiều.
* Workflow: Vì bản chất của Git là phân tán và có hệ thống nhánh tuyệt vời nên hầu như các quy trình công việc (workflow) nào cũng có thể được thực hiện tương đối dễ dàng trên Git.

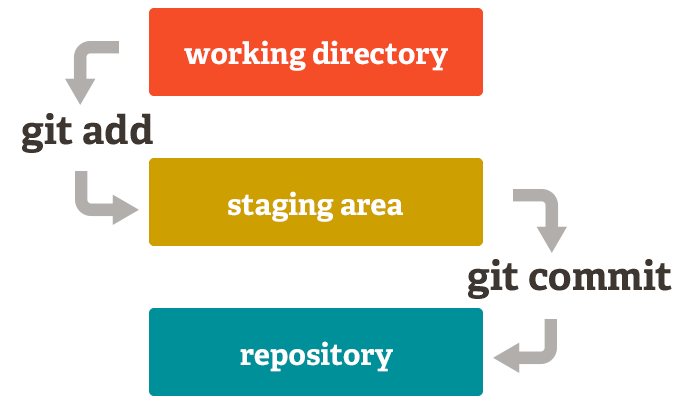
### Bảo đảm dữ liệu – Data Assurance



* Mô hình dữ liệu mà Git sử dụng đảm bảo tính toàn vẹn cho từng bit của dự án. Nó có cơ chế để mỗi tập tin và việc hoàn tất được kiểm tra kỹ lưỡng. Không thể lấy bất cứ thứ gì ra khỏi Git ngoài các bit chính xác mà chúng ta đưa vào.
* Chúng ta cũng không thể thay đổi tập tin, ngày tháng, thông điệp hoàn tất (commit message) hay bất cứ dữ liệu nào trong kho của Git mà không thay đổi ID của chúng sau đó. Điều này có nghĩa là nếu bạn có một commit ID, bạn có thể yên tâm là dự án của bạn chính xác như khi nó đã được commit. Trong khi đó hầu hết các hệ thống kiểm soát tập trung đều không mặc định cung cấp tính toàn vẹn như vậy.

### Staging Area

* Không giống như các hệ thống khác, Git có một vùng gọi là "staging area" hay "index". Đây là một khu vực trung gian nơi các giao dịch được định dạng và kiểm tra lại trước khi hoàn tất.
* Git có thể nhanh chóng chạy các tập tin và hoàn tất chúng mà không cần phải hoàn tất tất cả các tập tin được sửa đổi trong thư mục làm việc hoặc phải liệt kê chúng trên command line trong quá trình thực hiện.

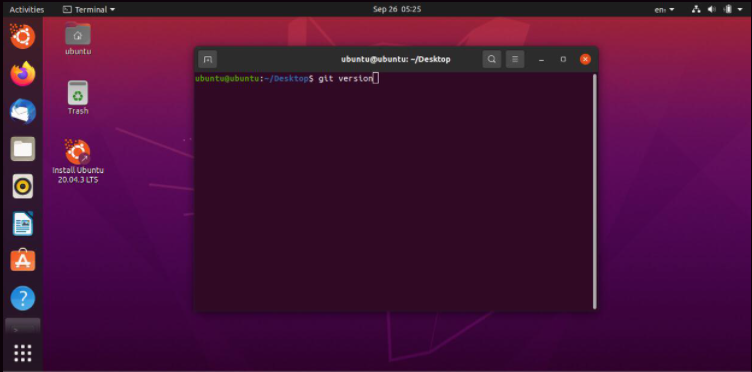


* Điều này cho phép ta chỉ thực hiện các phần của một tập tin được sửa đổi, cập nhật các thay đổi trong tập tin khi cần.
* Git cũng có thể không sử dụng tính năng này khi ta không muốn – chỉ cần thêm ‘-a’ vào lệnh commit để thêm tất cả thay đổi của tất cả các tập tin vào “staging area”.

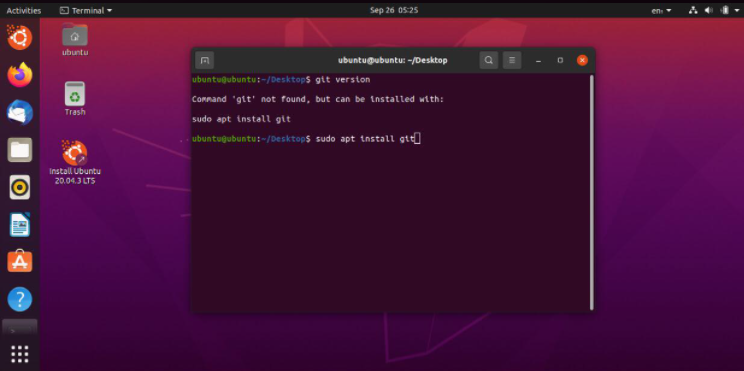


* 1. **Cài đặt phần mềm và cách sử dụng**
     1. **Cài đặt**

­­B1: gõ lệnh “git version” để kiểm tra xem đã cài đặt git chưa



B2: Gõ “sudo apt intall git” để tiến hành cài git



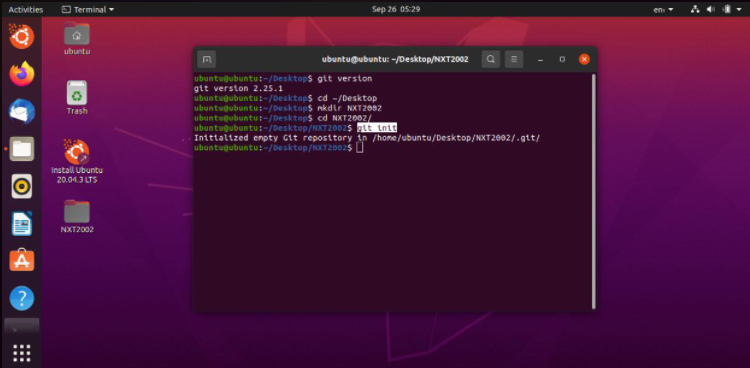
B3: Tạo một kho chứa git tại máy “NXT2002”

Code :

cd~/Desktop

Mkdir NXT2002

Cd NXT2002/



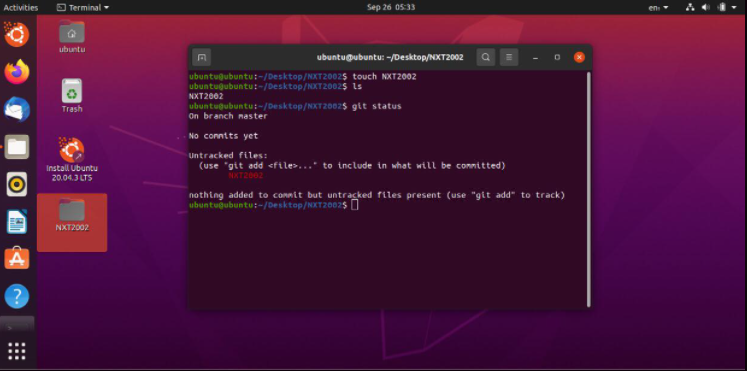
B4: tạo file mới vào git repo, git add

Code:

Touch NXT2002

Is

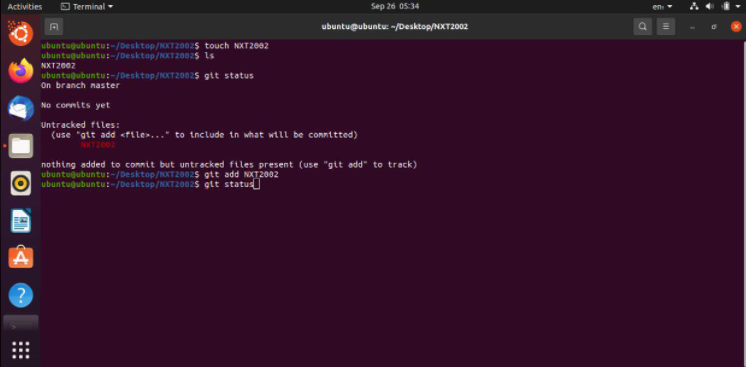
Gõ “git status” để xem file nào mà git biet nó tồn tại



B5: những file chưa được theo dõi bởi git, ta dùng “git add” để đưa những file mình muốn git theo dõi vào vùng theo dõi

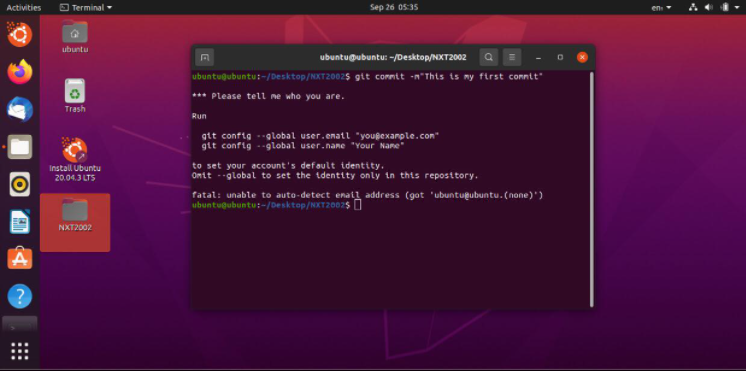
Code: git add NXT2002

Dùng “git add” để kiểm tra lại



B6: tạo commit đầu bằng cách dùng câu lệnh: git commit –m “tin nhắn của bạn về commit”

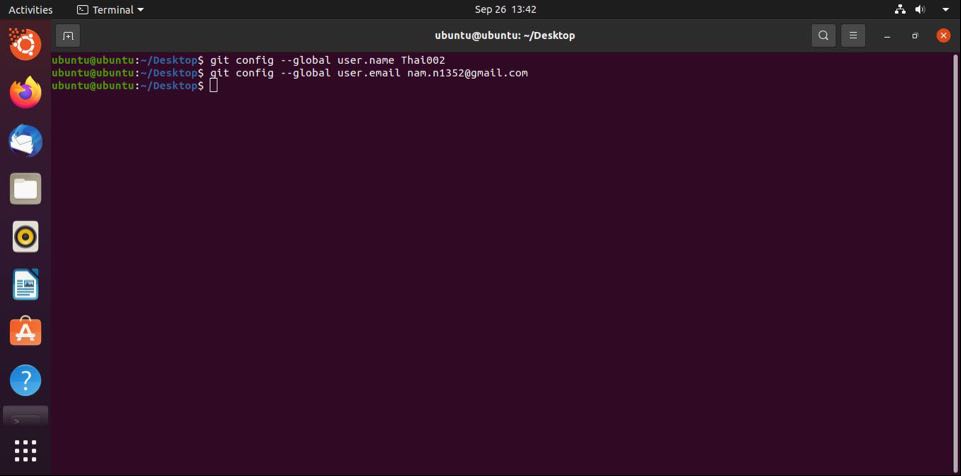
Vd: git commit –m “this is my firt commit”



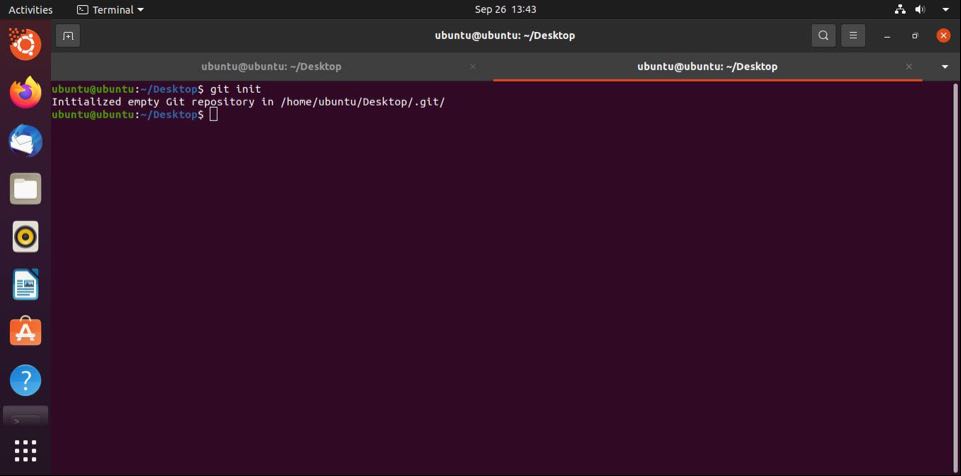
* + 1. **Cách sử dụng**

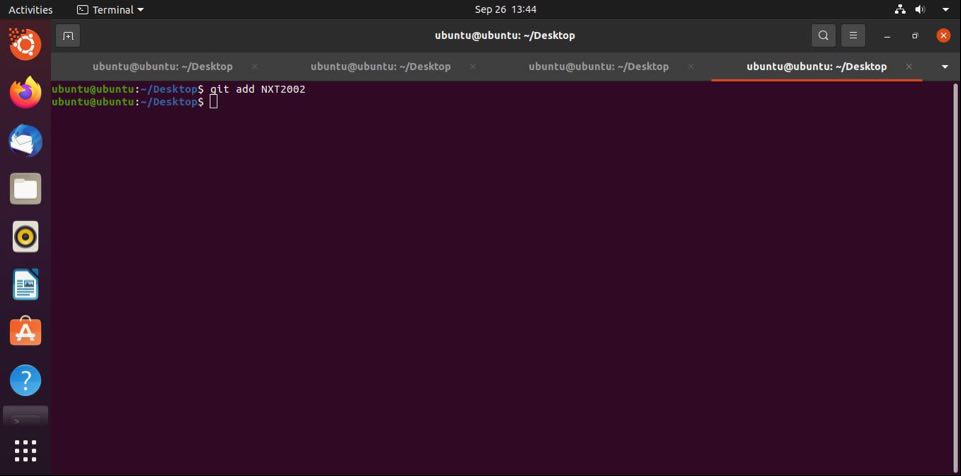
**-** Một trong các lệnh được dùng nhiều nhất là git config, có thể được cấu hình cho cá nhân với các giá trị của emails, thuật toán cho diff, username và file format

*git config --global user.email*

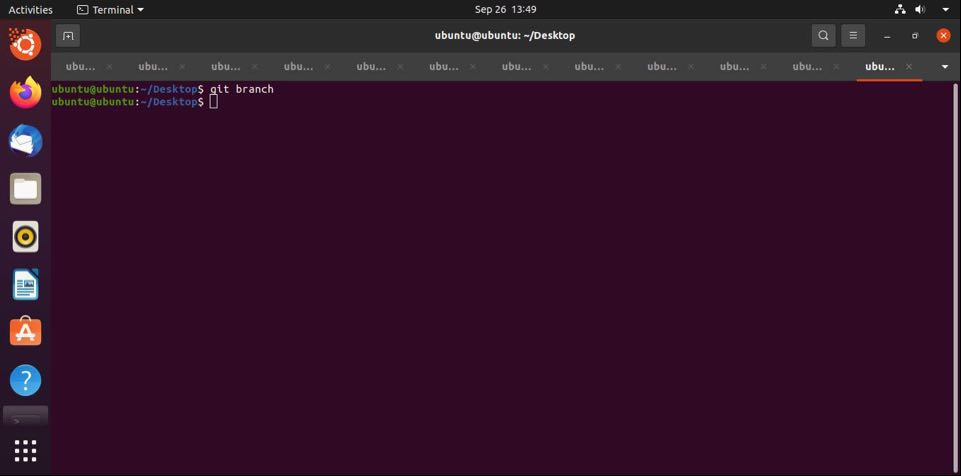


*- git init*  
Lệnh này được dùng để tạo GIT repository



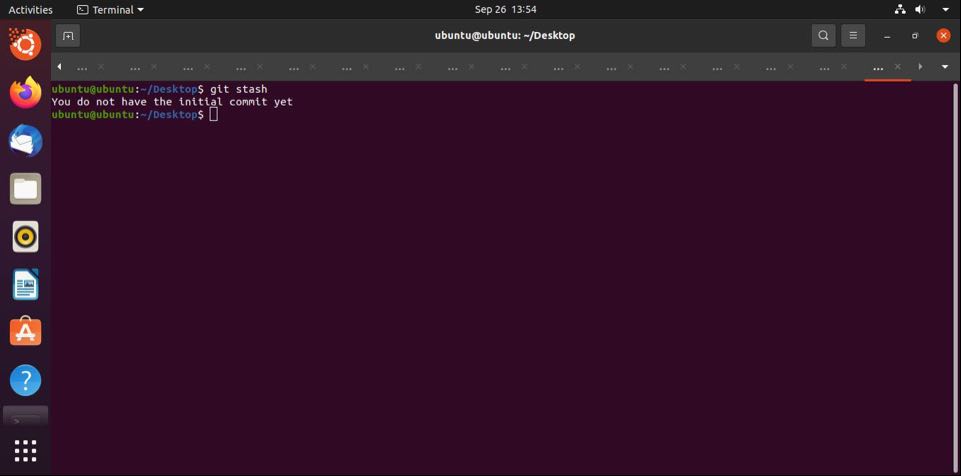
- *git add temp.txt*Lệnh git add có thể được dùng để thêm file vào index. Ví dụ, command sau có thể thêm một file có tên temp.txt vào thư mục local vào index 

*- git branch*  
Lệnh git branch có thể được dùng để liệt kê, tạo hoặc xóa branches. Để liệt kê tất cả các branchs đang có trong repositories

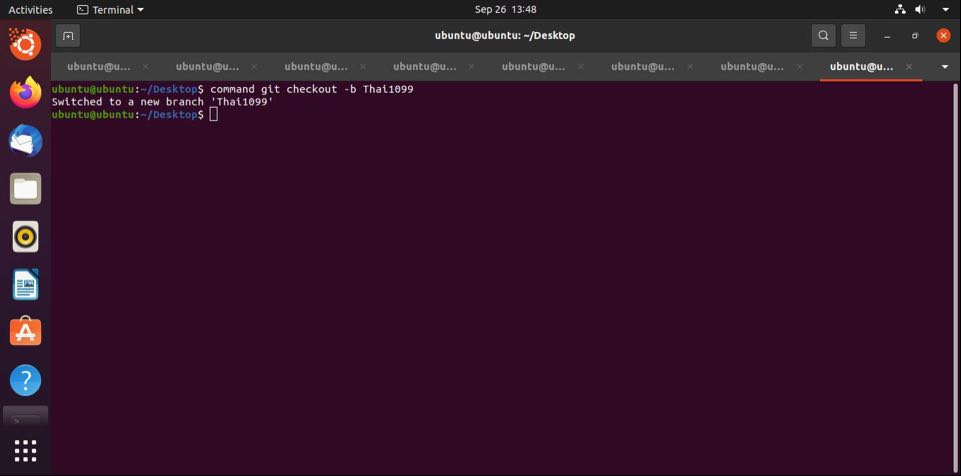


*- git stash*

Là một trong số ít lệnh cơ bản ít được biết đến, git stash giúp bạn lưu lại thay đổi mà chưa được commited ngay lập tức, nhưng tạm thời



*- command git checkout -b <branch-name>*Lệnh git checkout có thể được dùng để tạo branched hoặc chuyển giữa chúng. Ví dụ, lệnh sau tạo một brach mới và chuyển tới nó



- *git gc*  
Để tối ưu repository với garbage collection, sẽ dọn dẹp các file không cần thiết và tối ưu chúng

