Để cho việc cài đặt trở nên dễ dàng, hầu hết các bản phân phối đều cung cấp các giải pháp quản lý phần mềm bậc cao. Đây là các phần mềm thực hiện các thao tác quản lý phần mềm thông qua rpm hoặc dpkg, tuy nhiên tự động hóa một phần các công việc liên quan đến tải gói và cài đặt gói. Một điểm khác biệt của các phần mềm này so với rpm và dpkg là chúng cho phép quản lý cùng một lúc nhiều phần mềm. Các bản phân phối dựa trên rpm thường dùng dpkg hoặc urpmi. Các bản phân phối dựa trên debían sử dụng apt.

7.8.2. Các kho phần mềm

Để có thể cài đặt các phần mềm một cách tự động, các công cụ nói trên cần có khả năng tải được các gói phần mềm về hệ thống để rồi sau đó cài đặt. Như vậy các phần mềm này cần có một danh sách các kho phần mềm. khi có nhu cầu sẽ tải các phần mềm cần thiết về. Để tải các gói phần mềm, có thể sử dụng giao thức http, ftp, truy cập tệp trực tiếp hoặc thậm chí P2P. Các kho phần mềm thông thường chứa một hoặc nhiều kho phần mềm của các bản phân phối khác nhau. Dể cho quá trình tìm kiếm các gói được nhanh chóng, thông tin về các tệp chứa trong kho phần mềm được tập hợp tạo ra một tệp tiêu đề. Các công cụ cài đặt sẽ dựa vào các tiêu đề này để tìm kiếm tệp cần thiết. Các nhà phân phối duy trì trên các server các phiên bản repo phù hợp với các bản phân phối của mình. Với mỗi bản phân phối, có các repo con clio các phần mềm đã được dịch, các phần mềm để thử nghiệm và các phần mềm còn đang được phát triển. Các công cụ bậc cao còn hỗ trợ việc tải các tệp từ các kho phần niềm bằng các giao thức khác khác nhau.

7.8.3. Các công cụ tương tác

Trên cơ sở các công cụ bậc cao, các bản phân phối cung cấp các công cụ với giao diện tương tác để có thể quản lý các phần mềm. Ví dụ yumex, aptitude, synaptic,.... Việc sử dụng các công cụ này thuận tiện hơn, tuy nhiên về bản chất không có gì khác so với việc sử dụng công cụ bậc cao.

7.9. Sử dụng Git trong các dự án phần mềm

7.9.1. Giới thiệu về GIT

Git là một hệ thống kiểm soát phiên bản phân tán mã nguồn mở phổ biến hiện nay. Nó có thể giúp cho chúng ta lưu lại các phiên bản của những lần thay đổi vào mã nguồn và có thể dễ dàng khôi phục lại và người khác có quyền truy cập mã nguồn họ có thể xem các thay đổi ở từng phiên bản. Cơ chế lưu trữ của Git là nó sẽ tạo một snapshot trên mỗi tập tin và thư mục sau khi commit, từ đó có thể tái sử dụng lại..

Một số định nghĩa cơ bản cần trong Git:

- Thư mục làm việc là thư mục mà bạn có nội dung cần git quản lý.
- Một commit là ảnh chụp nhanh các thay đổi của thư mục làm việc của bạn và nó được theo dõi bằng cách sử dụng hàm **40 character SHA1 hash.**

- Index là nơi ảnh chụp nhanh các thay đổi của thư mục làmm việc của bạn trước khi commit. Nó khá quan trọng vì nó nằm giữa thư mục làm việc và một commit khác.
- Branch trong git nó là một con trỏ đến một commit cụ thể.

7.9.2. Cài đặt Git

Gói `git` có trong hầu hết các bản phân phối Linux. Để kiểm tra xem gói `git` có khả dụng trên hệ thống chưa bằng cách chạy lệnh sau:

```
[root@localhost ~]# git --version

-bash: git: command not found
```

Nếu gói `git` không có trên hệ thống, có thể dễ dàng cài đặt như sau:

- Đối với bản phân phối Debian/Ubuntu :

Chạy lệnh sau để cài đặt gói `git` :

```
root@ubuntuserver:~# apt-get install git
Reading package lists.. Done
Building dependency tree
Reading state informattion.. Done
git is alreading thenewest version (1:2.20.1-2ubuntu1).
…
```

- Đối với bản phân phối RHEL/CentOS:

Chạy lệnh bên dưới để thực hiện cài đặt gói git:

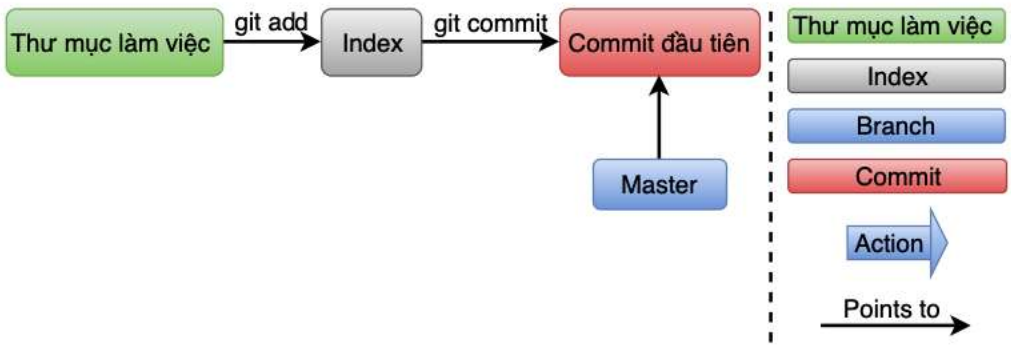
```
[root@localhost ~]# yum -y install git
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base:mirrors.viethosting.com
* extra: mirrors.viethosting.com
* updates: mirrors.viethosting.com
base
extras
updates
…
```

Để kiểm tra xem gói `git` cài đặt thành công chạy lệnh sau:

```
[root@localhost ~] git –versrion
Git version 1.8.3.1
```

7.9.3. Sử dụng Git

7.9.3.1. Git index



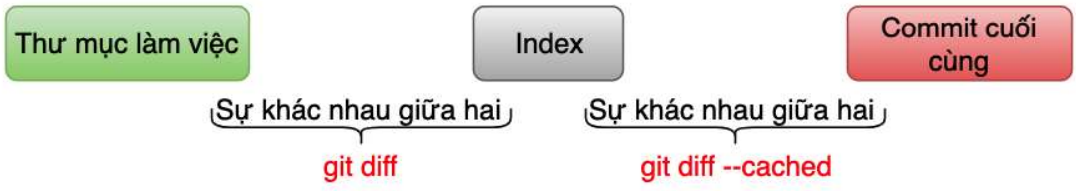
Hình 7.3. Git index

Index nằm giữa thư mục làm việc hiện thời và các commit khác. Khi thực hiện `git add` nghĩa là sao chép một ảnh chụp nhanh của thư mục làm việc vào index, tiếp theo khi chạy lệnh `git commit` thì bạn đã sao chép điều tương tự từ index để tạo một commit mới. Lệnh `git status` là một lệnh rất hữu ích vì nó cho chúng ta biết sự khác biệt giữa thư mục làm việc, index và commit trước đó.

Ví dụ sau sẽ thêm file `img` vào thư mục làm việc, chạy lệnh `git status` để thấy sự khác biệt:

```
[root@localhost ~]# git status
On branch master
Your branch is up to date with 'origin/master'.
Changes not staged for commit:
 (use "git add <file>..." to update what will be committed)
 (use "git checkout -- <file>…" to discard changes in working directory)
        modified: huong-dan-su-dung-git-co-ban/index.md
Untracked files:
    (use "git add <file>…" to include in what will be committed)
          huong-dan-su-dung-git-co-ban/img/
no changes added to commit (use "git add" and/or "git commit -a")
```

Lệnh `git diff` cũng tương tự như `git status`, nhưng nó cho thấy sự khác biệt giữa các commit khác nhau và giữa thư mục làm việc và index. `git diff – cached` cho chúng ta thấy sự khác biệt trong index so với commit cuối cùng.

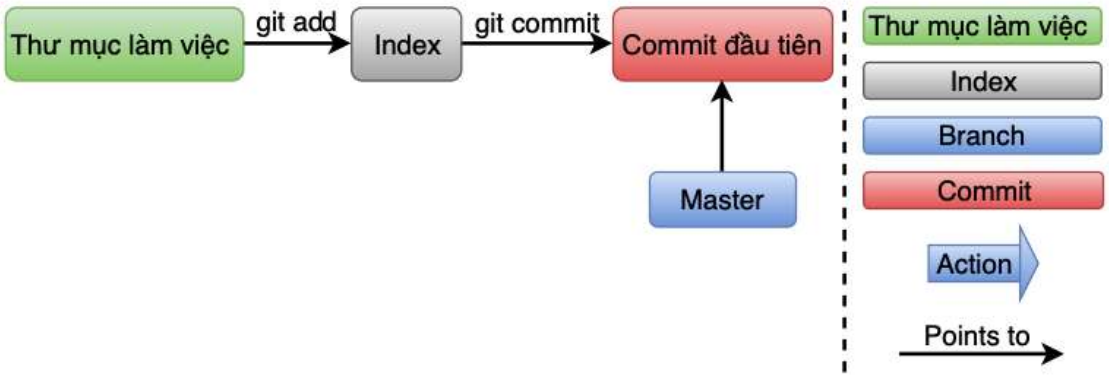


Hình 7.4. Sự khác biệt giữa index và commit

7.9.3.2. Branch

Branch là một con trỏ đến các commit khác nhau. Khi tiến hành commit lần đầu trong repository thì git sẽ tạo ra một branch có tên là master. Vì thế những lần commit sau sẽ được thêm vào branch master cho đến khi chuyển đổi branch.

Mô hình minh họa khi chúng ta làm việc trên một project đã có một commit từ trước:



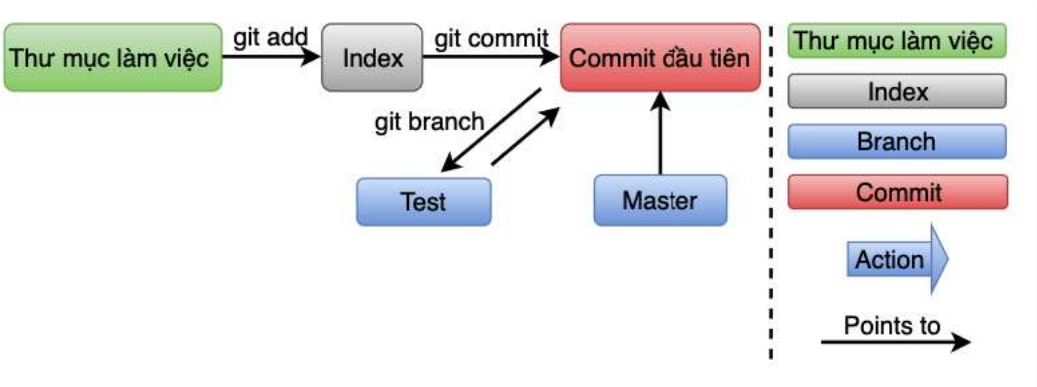
Hình 7.5. Branch

Trong `git` thì chúng ta có thể tự do tạo một branch mới chúng ta sử dụng lệnh `git branch` và trỏ đến commit hiện tại

Ví dụ: Cần tạo một branch có tên `test` chạy lệnh sau:

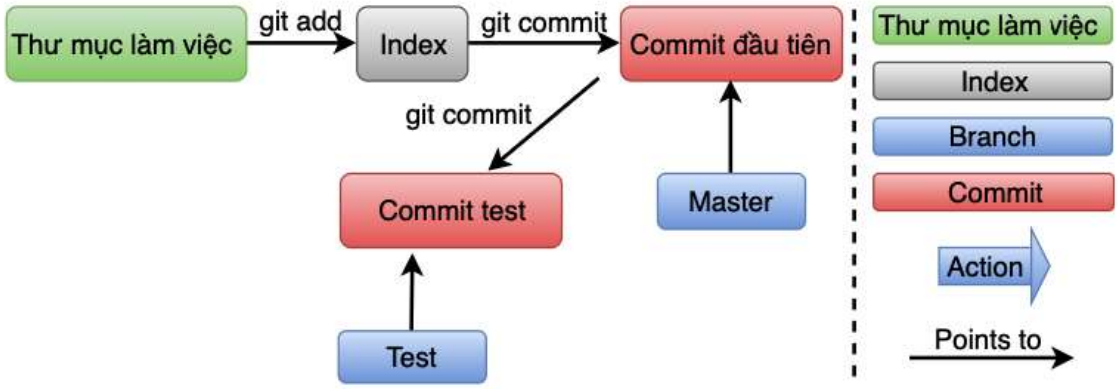
```
[root@localhost project_euler]# git branch test
```

Kết quả sẽ như hình bên dưới:



Hình 7.6. Branch có tên test

Branch có tên `test` đã được tạo, gõ lệnh `git branch` chúng ta sẽ thấy có 2 branch tên `master` và `test` nhưng con trỏ vẫn cho biết nơi làm việc ở branch `master`. Branch `test` khởi đầu từ commit tiếp theo (kế thừa master từ commit tiếp theo trở về trước).



Hình 7.7. Branch Test và Master

Để có thể thay đổi và bắt đầu làm việc trên một branch nhất định cần phải thực hiện lệnh sau đây `git checkout [name]` .

Ví dụ: cần chuyển qua làm việc trên branch tên `test` chạy lệnh như sau:

```
[root@localhost project_euler]# git checkout test

Switched to branch 'test'
```

7.9.3.3. Remote Respository

Mục đích của việc sử dụng remote repository là cho phép bất cứ ai ở những địa điểm khác nhau cũng có thể đóng góp các thay đổi mới cho repository.

Để kiểm tra tên của remote chúng ta chạy lệnh sau:

[root@localhost project_euler]# git remote -v

```
origin https://github.com/ngocdang1999/project_euler (fetch)
origin https://github.com/ngocdang1999/project_euler (push)
```

Chúng ta thấy repository khi chúng ta đã clone đều được đặt tên là origin, và mỗi repository đều có hai trạng thái:

- **fetch**: Dùng để lấy dữ liệu về từ server.
- **push**: Dùng để gửi dữ liệu lên server.

Qua đây giúp chúng ta hiểu về lệnh `git push origin master` là chúng ta có thể push các thay đổi trên mã nguồn ở máy lên remote tên là origin với branch master.

Khi chúng ta muốn thay đổi tên remote mặc định (origin) sang một tên khác cho chúng ta dễ dàng quản lý.

Ví dụ: cần đổi từ tên `origin` sang tên `blogd` thì sẽ thực hiện như sau:

```
[root@localhost project_euler]# git remote rename origin dang
[root@localhost project_euler]# git remote -v
```

```
dang    https://github.com/ngocdang1999/project_euler (fetch)
dang    https://github.com/ngocdang1999/project_euler (push)
```

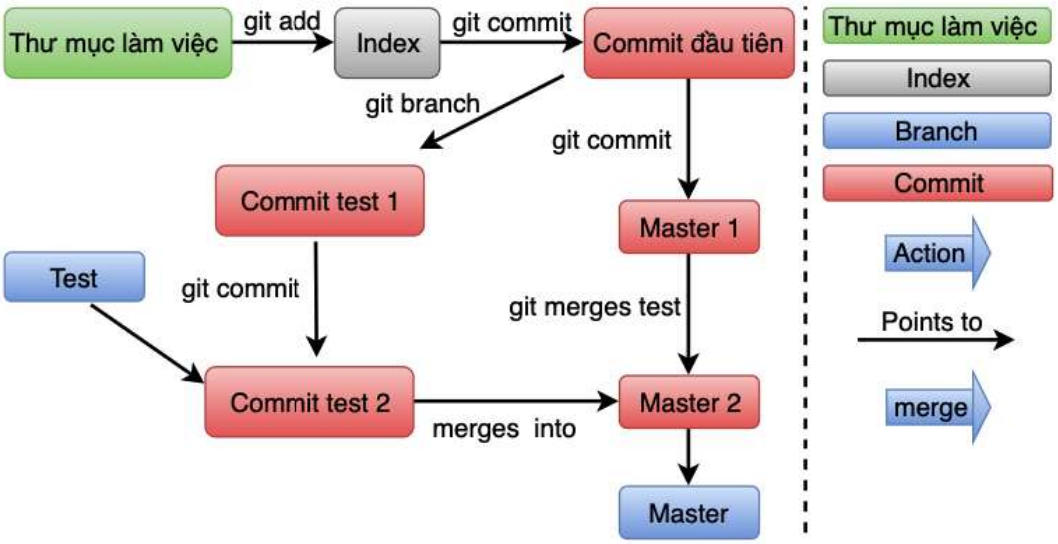
Sau khi đã đổi tên remote thì khi chúng ta thực hiện push chúng ta cần thực hiện lệnh như sau `git push [name remote] master` để có thể push các thay đổi trên mã nguồn ở máy lên remote repository.

Nếu chúng ta muốn thêm một remote để có thể lấy dữ liệu hoặc thêm các thay đổi trên mã nguồn máy lên remote thì chúng ta sử dụng lệnh `git remote add [name_remote] [URL]`

7.9.3.4. Merge

Trong một dự án chúng ta sẽ có nhiều thành viên tham gia thực hiện thì mỗi người sẽ đảm nhận một nhiệm vụ riêng nhưng đến khi hoàn thành thì chúng ta cần phải tiến hành đồng bộ hóa các nhiệm vụ của tất cả các thành viên. `git` có tính năng giúp cho chúng ta có thể thực hiện di chuyển các con trỏ nhánh cần thiết để có thể hoàn thành dự án.

Hình minh họa bên dưới giúp chúng ta merge commit test vào commit master và bao gồm cả những thay đổi tồn tại trong test:



Hình 7.8. Merge

7.9.3.5. Sử dụng git

Sử dụng git trên hệ thống cục bộ

Sử dụng git để tạo project hay một thư mục đã có sẵn để nhập vào git. Để có thể khởi tạo một kho chứa từ thư mục có sẵn đầu tiên chúng ta cần cần chạy lệnh sau:

```
[root@localhost ~]# mkdir website && cs website
[root@localhost website]# git init
Initialized empty Git repository in /root/website/.git/
```

Tiếp theo thực hiện tạo nội dung bằng cách sử dụng lệnh echo để thêm nội dung vào file index.html.

```
[root@localhost website]# echo 'Hello, world!' > index.html
```

Chạy lệnh sau để thêm nội dung vào `git index` :

```
[root@localhost website]# git add index.html
```

Thực hiện commit để thay đổi commit mới:

```
[root@localhost website]# git commit -m "Newweb"

[master (root-commit) 02bb945] Newweb

 Committer: root <root@localhost.localdomain>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

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

    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)

 create mode 100644 index.html
```

Sử dụng git trên máy chủ

Để bắt đầu một kho lưu trữ git mới trên máy chủ chúng ta chạy lệnh sau

```
[root@localhost ~]# mkdir mywebsite.git

[root@localhost ~]# cd mywebsite.git

[root@localhost mywebsite.git]# git init --bare --shared

Initialized empty shared Git repository in /root/mywebsite.git/
```

Tạo hook của bạn sẽ kiểm tra mã vào thư mục web thực tế của bạn.

```
[root@localhost mywebsite.git]# cat > hooks/post-receive

#!/bin/sh
```

```
GIT_WORK_TREE=/path/to/webroot/of/mywebsite git checkout -f

[root@localhost website.git]# chmod +x hooks/post-receive
```

Quay lại hệ thống cục bộ

Thêm thư mục từ xa vào cấu hình cục bộ:

```
[root@localhost website]# git remote add web ssh://root@192.168.70.188/home/user/mywebsite.git
```

Đẩy nội dung của kho lưu trữ cục bộ vào kho từ xa:

```
[root@localhost website]# git push web +master:refs/heads/master

The authenticity of host '192.168.70.188 (192.168.70.188)' can't be established.

ECDSA key fingerprint is SHA256:YIGfm3ouDO4t5tzyksDxs1J1CB5hlOMdctL21iNNcRM.

ECDSA key fingerprint is MD5:74:53:22:9f:b0:be:e8:e5:f5:fa:83:6e:63:a8:62:c5.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '192.168.70.188' (ECDSA) to the list of known hosts.

Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

fatal: Could not read from remote repository.

Please make sure you have the correct access rights

and the repository exists.
```

Sau đó, thay đổi mọi thứ cục bộ và để tải lên các thay đổi, chỉ cần thực hiện:

```
[root@localhost website]# git push web

warning: push.default is unset; its implicit value is changing in

Git 2.0 from 'matching' to 'simple'. To squelch this message

and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple
```

```
See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
fatal: Could not read from remote repository.
Please make sure you have the correct access rights
and the repository exists.
[root@secondarydns website]# git remote -v
web     ssh://root@192.168.70.188/home/user/mywebsite.git (fetch)
web     ssh://root@192.168.70.188/home/user/mywebsite.git (push)
```

7.9.4. Các lệnh thường dùng trong git

Để có thể tạo một repository mới chúng ta chạy lệnh sau:

```
git init
```

Khi muốn sao chép (clone) một repository trên máy cục bộ chúng ta thực thi lệnh:

```
git clone /đường-dẫn-đến/repository/
```

Khi một repository trên một máy chủ khác, thực thi dòng lệnh như sau:

```
git clone username@ipadd:/đường-dẫn-đến/repository
```

Ví dụ: Chúng ta cần thực hiện sao chép một repository từ máy chủ "https://github.com/phamminhthai/project_euler" chúng ta chạy lệnh sau:

```
[root@localhost ~]# git clone https://github.com/phamminhthai/project_euler
Cloning into 'project_euler'...
remote: Enumerating objects: 157, done.
remote: Total 157 (delta 0), reused 0 (delta 0), pack-reused 157
Receiving objects: 100% (157/157), 414.60 KiB | 324.00 KiB/s, done.
Resolving deltas: 100% (46/46), done.
```

Sử dụng lệnh git để thêm các thay đổi vào chỉ mục index sử dụng lệnh bên dưới:

```
git add <tên-tập-tin>

git add *
```

Ví dụ: Thực hiện thêm một file "/test-git/test.txt" vào repository chúng ta vừa sao chép về sau đó thêm nó vào chỉ mục index:

```
[root@localhost project_euler]# cd

[root@localhost ~]# cd project_euler/

[root@localhost project_euler]# cat > test-git/test.txt

Hello Blogd.net

[root@localhost project_euler]# git add *
```

Để commit những thay đổi các chỉ mục index chúng ta chạy lệnh sau:

```
git commit -m "Ghi chú Commit"
```

Ví dụ: Sau khi chúng ta thực hiện thêm một file vào chỉ mục index tiếp theo chúng ta cần commit các thay đổi thực hiện như sau:

```
[root@localhost project_euler]# git commit -m "Them thu muc test-git vao repository"

[master a176e65] Them thu muc test-git vao repository

 Committer: root <root@localhost.localdomain>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

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

    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)

 create mode 100644 test-git/test.txt
```

Khi ta cần push các thay đổi khi các thay đổi đang nằm tại HEAD của bản sao cục bộ đang làm việc. Để có thể push các thay đổi đến repository remote chúng ta thực thi như sau:

```
git push origin master
```

Ví dụ: Khi đã commit thành công chúng ta sẽ push các thay đổi lên repository như bên dưới:

```
[root@localhost project_euler]# git push origin master

Username for 'https://github.com': nguyenngocdangabcd@gmail.com

Password for 'https://nguyenngocdangabcd@gmail.com@github.com':

Counting objects: 5, done.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (4/4), 349 bytes | 0 bytes/s, done.

Total 4 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/ngocdang1999/project_euler

   9240ce8..a176e65  master -> master
```

Như kết quả trên chúng ta đã thực hiện push thành công lên repository.

Để có thể tạo một nhánh mới có tên là "test" và chuyển qua nhánh đó từ master thực hiện như sau:

```
[root@localhost project_euler]# git checkout -b test

Switched to a new branch 'test'
```

Khi chúng ta cần trở lại nhánh master từ một nhánh khác thực thi như sau:

```
[root@localhost project_euler]# git checkout master

Switched to branch 'master'
```

Để có thể xóa một nhánh và nhánh đó có tên là "test" chúng ta chạy lệnh sau:

```
[root@localhost project_euler]# git branch -d test

Deleted branch test (was a176e65).
```

Để cập nhật một repository cục bộ của bạn và commit mới nhất chúng ta chạy lệnh sau:

```
[root@localhost project_euler]# git pull

Already up-to-date.
```

Nếu chúng ta muốn hủy tất cả thay đổi và các commit cục bộ, về fetch lịch sử gần đây nhất từ máy chủ và trỏ đến nhánh master cục bộ như sau:

```
[root@localhost project_euler]# git fetch origin

[root@localhost project_euler]# git reset --hard origin/master

HEAD is now at a176e65 Them thu muc test-git vao repository
```

Sau khi chúng ta thay đổi tên của một thư mục con trong thư mục làm việc hiện tại chúng ta cần chạy lệnh `git add -u` để git giúp chúng ta update tự động tên đã thay đổi.

Lệnh `git status` hiển thị trạng thái của cây làm việc, bao gồm sự khác biệt giữa chỉ mục và cam kết hiện tại.

Lệnh `git log` giúp chúng ta có thể hiển thị nhật ký commit.

```
[root@localhost project_euler]# git log

commit a176e651e7dac92296b29ce3f9a5f6a0f66d5c94

Author: root <root@localhost.localdomain>

Date:   Wed Sep 11 18:29:18 2019 +0700

    Them thu muc test-git vao repository

commit 9240ce8fc3bc65bf672d570f46546069d8380ebd

Author: phamminhthai <pmthai@uneti.edu.vn>

Date:   Mon Nov 26 20:34:32 2018 +0700

commit 470b0b4131fe7d7cbf2df8c601aed0c543b9fa34

Author: phamminhthai <pmthai@uneti.edu.vn>

Date:   Fri Oct 12 00:20:26 2018 +0700
```

Lệnh `git clean` dùng để loại bỏ các tập tin không bị theo dõi khỏi thư mục làm việc.

Chỉ định vị trí bug nhập vào chúng ta có một số tùy chọn sau:

- Đánh dấu vị trí bắt đầu chạy thực thi lệnh sau: git bisect start.

- Đánh dấu vị trí tốt thực thi lệnh sau: git bisect good.
- Đánh dấu vị trí bị hỏng thực thi lệnh sau: git bisect bad.
- Chạy tự động: git bisect run.

Dùng để di chuyển hoặc xóa file chúng ta chạy lệnh bên dưới:

- Lệnh git mv dùng để di chuyển file.
- Lệnh git rm dùng để xóa file.

Sử dụng git để kiểm soát phiên bản phân tán mã nguồn mở. Đồng thời giúp cho chúng ta có thể lưu lại các phiên bản của tất cả các lần thay đổi có thể dễ dàng khôi phục lại dễ dàng. Ngoài ra chúng ta cũng có thể cung cấp quyền truy cập mã nguồn cho những người thành viên trong nhóm để triển khai công việc dễ dàng.

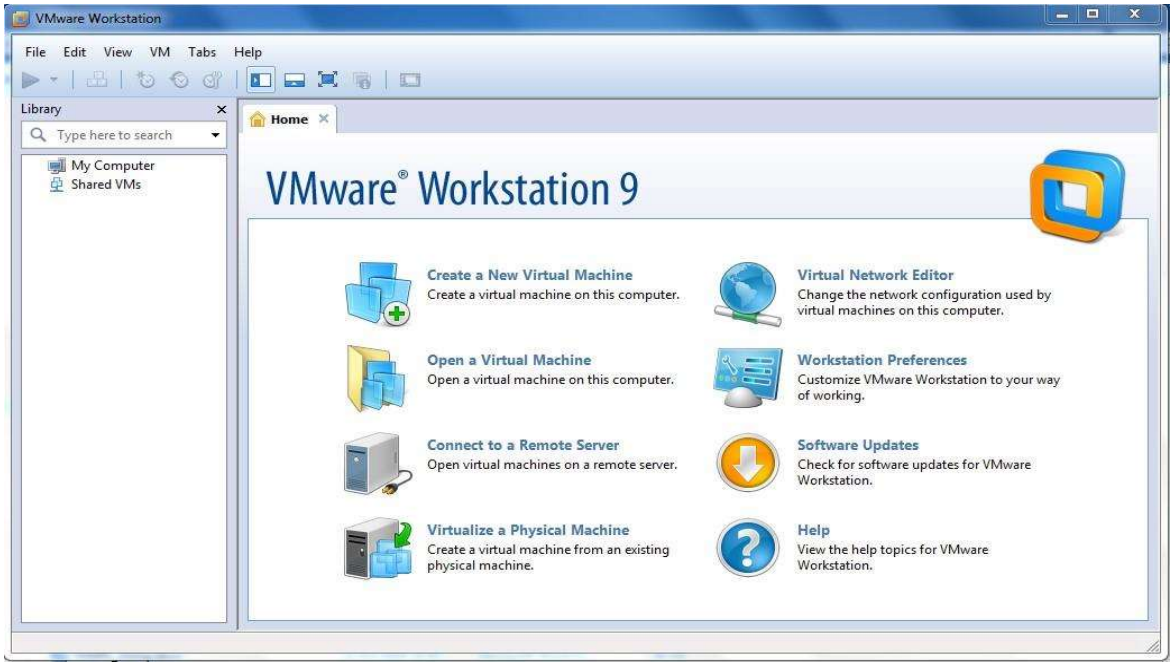
7.10. Bài thực hành LAB 3: QUẢN TRỊ TÀI NGUYÊN VÀ DỊCH VỤ

LƯU Ý: TẤT CẢ SINH VIÊN THỰC HIỆN TRÊN FILE IMAGE CỦA HỆ ĐIỀU HÀNH CENTOS 7 ĐÃ CÀI TỪ CÁC BUỔI THỰC HÀNH TRƯỚC TẠI THƯ MỤC ĐÃ TẠO TẠI Ổ SINHVIEN.

PHẦN 1: MỞ FILE IMAGE CỦA HỆ ĐIỀU HÀNH CENTOS 7

Bước 1: Khởi động phần mềm máy ảo VMware

Sau khi khởi động VMware như hình sau



Bước 2: Mở file image của hệ điều hành CentOS 7 đã cài

Trên màn hình chính VMware chọn File ->Open…

Chọn đến file image VMX của CentOS 7