Demo các câu lệnh của Git

```
~$ mkdir PlayWithGit
~$ cd PlayWithGit/
PlayWithGit$ mkdir Machine1
PlayWithGit$ cd Machine1/
```

Để dễ theo dõi, mình sẽ thêm một dòng vào file bằng câu lệnh `echo`, và nhiều dòng bằng câu lệnh `cat`.

```
Machine1$ echo "Line 1" > File1.md
Machine1$ cat File1.md
Line 1
Machine1$ echo 'This is the compiled file of the source file
"File1.md"' > File1.pdf
Machine1$ cat File1.pdf
This is the compiled file of the source file "File1.md"
```

1 Demo 1

Dùng Git để quản lý phiên bản cho thư mục Machine1, làm offline trong một máy, chỉ có một nhánh (phiên bản 2 được tạo ra từ phiên bản 1, phiên bản 3 được tạo ra từ phiên bản 2, ...)

Đầu tiên, ta sẽ nhờ Git quản lý phiên bản cho thư mục Machine1:

```
Machine1$ git init
Initialized empty Git repository in /home/kien/PlayWithGit/Machine
Machine1$ ls -a
.  ..  File1.md  File1.pdf .git
```

Thư mục .git chính là thùng chứa các phiên bản của Git. Để tắt Git đi thì bạn

có thể xóa hoặc đổi tên thư mục .git. Hiện giờ, thùng chứa đang rỗng, chưa có phiên bản nào. Phiên bản làm việc thì có file File1.md và file File1.pdf vừa được tạo.

Tra cứu thông tin của một câu lệnh Git, chẳng hạn `git init`:

```
Machine1$ git init --help
```

Xem trạng thái hiện giờ của phiên bản làm việc:

```
Machine1$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    File1.md
    File1.pdf

nothing added to commit but untracked files present (use "git add" to
```

Git báo là trong phiên bản làm việc có 2 file mới. Thường ta chỉ muốn quản lý phiên bản cho file mã nguồn dạng text (File1.md), chứ không muốn quản lý phiên bản cho file kết quả biên dịch (File1.pdf). Để nói với Git là đừng quan tâm đến những file/thư mục nào thì ta sẽ ghi những file/thư mục này vào file .gitignore (có thể ghi theo cú pháp wildcard).

```
Machine1$ echo "*.pdf" > .gitignore
Machine1$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    File1.md

nothing added to commit but untracked files present (use "git add" to
```

OK, Git đã không quan tâm đến file pdf nữa. Nhưng giờ Git báo có thêm file mới trong phiên bản làm việc là .gitignore. Có nên quản lý phiên bản của

file .gitignore? Mình sẽ để bạn suy nghĩ. Nếu không muốn thì bạn thêm dòng .gitignore vào chính file .gitignore. Còn ở đây mình muốn quản lý phiên bản của file .gitignore nên mình sẽ để đó.

Phiên bản mới của thư mục Machine1 sẽ là phiên bản cũ (hiện giờ là không có gì) + những thay đổi nào? Giả sử ta xác định là tạo 2 file File1.md và .gitignore:

```
Machine1$ git add File1.md
Machine1$ git add .gitignore
Machine1$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore
    new file:   File1.md
```

Git báo là các thay đổi đã sẵn sàng để commit (đưa vào thùng chứa phiên bản mới = phiên bản cũ + các thay đổi).

Commit:

```
Machine1$ git commit -m "Create File1.md and .gitignore"
[master (root-commit) 5d31ae0] Create File1.md and .gitignore
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 File1.md
```

Bạn cố gắng ghi message rõ ràng để lúc sau bạn/người khác có thể dễ dàng xem lại và biết là commit (phiên bản) này ứng với thay đổi gì. Ở đây, mình đã setup tên và email trước; còn nếu không thì khi commit sẽ bị báo lỗi và Git sẽ đưa ra 2 câu lệnh để bạn setup tên và email (khi làm việc nhóm, tên và email sẽ rất quan trọng để cho biết commit này là của ai, để có thể qui tội ;-)).

```
Machine1$ git status
On branch master
nothing to commit, working tree clean
```

Xem các phiên bản được lưu lại trong thùng chứa của Git:

```
Machine1$ git log
commit 5d31ae02f2b5b1a7301cd054a2e577132eeaabb8 (HEAD -
> master)
Author: KienTran <4948520+KienTrann@users.noreply.github.com>
Date:   Sun Apr 4 15:31:47 2021 +0700

    Create File1.md and .gitignore
```

Một phiên bản (commit) gồm:

- Nội dung của thư mục mà ta muốn lưu lại phiên bản.
- Các thông tin meta như: author, date, message, con trỏ tới (các) phiên bản cha mà từ đó phiên bản này được tạo ra.

Git sẽ tạo một id duy nhất cho một phiên bản, trong phiên bản này chỉ cần thay đổi một ký tự (ví dụ ở message) thì sẽ ra một id khác. Mặc định thì git log sẽ hiển thị các phiên bản với id, author, date, message.

Trong kết quả hiển thị ở trên: HEAD là gì? master là gì? Master là tên mặc định của nhánh chính, là con trỏ trỏ tới id của commit sau cùng của nhánh (Tại sao lại sau cùng? Vì chỉ cần nắm được commit sau cùng là có thể lôi ra các commit trước đó). Còn HEAD là con trỏ cho biết phiên bản làm việc hiện ứng với phiên bản nào trong thùng chứa. Hiện giờ HEAD trỏ tới phiên bản vừa commit *thông qua con trỏ master*, và khi phiên bản mới được commit thì *cả HEAD và master* đều sẽ được cập nhật để trỏ vào phiên bản mới này (còn nếu để HEAD trỏ trực tiếp vào phiên bản vừa commit thì khi phiên bản mới được commit chỉ có HEAD được cập nhật, còn master không được cập nhật; ta không muốn như vậy).

Bây giờ, ta sẽ lặp lại qui trình ở trên một lần nữa: thay đổi phiên bản làm việc, xác định phiên bản mới sẽ là phiên bản cũ + những thay đổi nào, lưu phiên bản mới vào thùng chứa

```
Machine1$ echo "Line 2" >> File1.md # Để ý: >>
Machine1$ cat File1.md
Line 1
Line 2
Machine1$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git restore <file>..." to discard changes in working director
    modified:   File1.md

no changes added to commit (use "git add" and/or "git commit -
a")
```

Git báo là File1.md đã được chỉnh sửa, chứ không phải là “untracked” như lần trước.

Trong thực tế, ta có thể chỉnh sửa rất nhiều file và không nhớ rõ là đã chỉnh sửa gì. Để nhớ lại là đã chỉnh sửa gì để xác định là sẽ đưa những thay đổi nào vào phiên bản mới cũng như là để viết nội dung của message khi commit:

```
Machine1$ git diff HEAD
diff --git a/File1.md b/File1.md
index 3be9c81..c82de6a 100644
--- a/File1.md
+++ b/File1.md
@@ -1 +1,2 @@
 Line 1
+Line 2
```

Câu lệnh này so sánh sự khác nhau giữa phiên bản làm việc với phiên bản cũ được lưu trong thùng chứa (đang được trỏ tới bởi HEAD). `git diff` còn có thể dùng để so sánh giữa phiên bản làm việc với phiên bản ở vùng chuẩn bị, phiên bản ở vùng chuẩn bị với phiên bản ở thùng chứa, 2 phiên bản ở thùng chứa với nhau, ...; khi cần thì bạn tra cứu thêm thông tin của câu lệnh này.

```
Machine1$ git add File1.md
Machine1$ git commit -m "Add line 2 to File1.md"
[master 00d7b4d] Add line 2 to File1.md
 1 file changed, 1 insertion(+)
Machine1$
Machine1$ git log
commit  00d7b4db8895750ff28d5703e0133e45f611ee18  (HEAD  -
> master)
Author: KienTran <4948520+KienTrann@users.noreply.github.com>
Date:   Sun Apr 4 16:11:12 2021 +0700

    Add line 2 to File1.md
```

```
commit 5d31ae02f2b5b1a7301cd054a2e577132eeaabb8
Author: KienTran <4948520+KienTrann@users.noreply.github.com>
Date:   Sun Apr 4 15:31:47 2021 +0700

    Create File1.md and .gitignore
```

HEAD và master đều đã được cập nhật để trỏ vào phiên bản mới nhất.

Để dễ dàng quan sát kết quả của git log khi có nhiều commit:

```
Machine1$ git log --oneline
00d7b4d (HEAD -> master) Add line 2 to File1.md
5d31ae0 Create File1.md and .gitignore
```

Để lấy ra một phiên bản nào đó từ thùng chứa (chẳng hạn, phiên bản 5d31ae0...) và làm việc:

```
Machine1$ git checkout 5d31ae0
Note: switching to '5d31ae0'.

You are in 'detached HEAD' state. You can look around, make experimen
changes and commit them, and you can discard any commits you make in
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you m
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead t

HEAD is now at 5d31ae0 Create File1.md and .gitignore
```

Hmm ...

```
Machine1$ cat File1.md
Line 1
Machine1$ git log --oneline
```

```
5d31ae0 (HEAD) Create File1.md and .gitignore
```

Mặc định thì `git log` chỉ show ra các phiên bản dựa vào con trỏ HEAD. Để show ra dựa vào tất cả các con trỏ:

```
Machine1$ git log --oneline --all
00d7b4d (master) Add line 2 to File1.md
5d31ae0 (HEAD) Create File1.md and .gitignore
```

Các phiên bản mà không thể lôi ra được từ các con trỏ coi như là sẽ bị mất (vẫn có cách để lấy lại được, nhưng nếu để lâu thì Git sẽ dọn rác và sẽ bị mất thật):

```
Machine1$ echo "Test abcxyz" >> File1.md
Machine1$ cat File1.md
Line 1
Test abcxyz
Machine1$ git add File1.md
Machine1$ git commit -m "Test with File1.md"
git commit -m "Test with File1.md"
[detached HEAD 301a6cb] Test with File1.md
 1 file changed, 1 insertion(+)
Machine1$ git log --oneline --all
301a6cb (HEAD) Test with File1.md
00d7b4d (master) Add line 2 to File1.md
5d31ae0 Create File1.md and .gitignore
Machine1$ git checkout master
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

  301a6cb Test with File1.md

If you want to keep it by creating a new branch, this may be a good ti
to do so with:

 git branch <new-branch-name> 301a6cb

Switched to branch 'master'
Machine1$ git log --oneline --all
00d7b4d (HEAD -> master) Add line 2 to File1.md
5d31ae0 Create File1.md and .gitignore
```

2 Demo 2

Tiếp tục từ demo 1, vẫn làm offline trong một máy, nhưng làm với nhiều nhánh (có phiên bản sẽ được tạo ra bằng cách merge các phiên bản thuộc các nhánh khác nhau)

Bây giờ, ta sẽ viết một chương trình Python mà khi người dùng chạy và không nhập vào tham số dòng lệnh thì sẽ không làm gì cả; còn khi người dùng nhập `feature1` thì sẽ thực hiện tính năng ứng với `feature1`, tương tự cho `feature2`, `feature3`, ...

Đầu tiên, từ phiên bản làm việc, ta sẽ tạo file và viết chương trình mà khi người dùng chạy và không nhập tham số dòng lệnh thì sẽ thông báo là không làm gì cả. Ta sẽ đưa phiên bản mới vào thùng chứa với phiên bản mới = phiên bản gần đây nhất + file mới này.

```
Machine1$ cat > MyProgram.py
import sys

if len(sys.argv) == 1:
    print('Do nothing')
Machine1$ cat MyProgram.py
import sys

if len(sys.argv) == 1:
    print('Do nothing')
Machine1$ python MyProgram.py
Do nothing
Machine1$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    MyProgram.py

nothing added to commit but untracked files present (use "git add" to
Machine1$ git add MyProgram.py
Machine1$ git commit -m "Create MyProgram.py"
[master 0f61082] Create MyProgram.py
 1 file changed, 4 insertions(+)
 create mode 100644 MyProgram.py
Machine1$ git log --oneline --all
```

```
0f61082 (HEAD -> master) Create MyProgram.py
00d7b4d Add line 2 to File1.md
5d31ae0 Create File1.md and .gitignore
```

Hiện giờ, thùng chứa chỉ có một nhánh và ta đang ở nhánh này:

```
Machine1$ git branch
* master
```

Bây giờ, ta sẽ tạo 2 nhánh mới, gọi là feature1 và feature2, để code và test hai tính năng tương ứng mà không làm ảnh hưởng gì đến code hiện tại ở nhánh chính (nhánh master). Sau khi tính năng 1 ở nhánh feature1 đã hoàn thành và không có vấn đề gì thì ta sẽ merge vào nhánh chính. Tương tự với tính năng 2 ở nhánh feature2.

```
Machine1$ git branch feature1
Machine1$ git branch feature2
Machine1$ git branch
  feature1
  feature2
* master
Machine1$ git log --oneline --all
0f61082 (HEAD -> master, feature2, feature1) Create MyProgram.py
00d7b4d Add line 2 to File1.md
5d31ae0 Create File1.md and .gitignore
```

Như có thể thấy, tạo nhánh chẳng qua là tạo một con trỏ và cho trỏ vào id của một phiên bản nào đó, mặc định thì sẽ là id của phiên bản mà HEAD đang trỏ vào.

Bây giờ ta sẽ checkout nhánh feature1, code và test tính năng 1, đưa phiên bản mới mà có thêm tính năng 1 vào thùng chứa.

```
Machine1$ git checkout feature1
Switched to branch 'feature1'
Machine1$ git log --oneline --all
0f61082 (HEAD -> feature1, master, feature2) Create MyProgram.py
00d7b4d Add line 2 to File1.md
5d31ae0 Create File1.md and .gitignore
Machine1$
Machine1$ cat >> MyProgram.py
else:
    if sys.argv[1] == 'feature1':
```

```
        print('Do feature1')
Machine1$ cat MyProgram.py
import sys

if len(sys.argv) == 1:
    print('Do nothing')
else:
    if sys.argv[1] == 'feature1':
        print('Do feature1')
Machine1$ python MyProgram.py feature1
Do feature1
Machine1$ git status
On branch feature1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working director
    modified:   MyProgram.py

no changes added to commit (use "git add" and/or "git commit -
a")
Machine1$ git diff HEAD
diff --git a/MyProgram.py b/MyProgram.py
index ed9a565..94aaea7 100644
--- a/MyProgram.py
+++ b/MyProgram.py
@@ -2,3 +2,6 @@ import sys

 if len(sys.argv) == 1:
        print('Do nothing')
+else:
+       if sys.argv[1] == 'feature1':
+               print('Do feature1')
Machine1$ git add MyProgram.py
Machine1$ git commit -m "Add feature1 to MyProgram.py"
[feature1 9814feb] Add feature1 to MyProgram.py
 1 file changed, 3 insertions(+)
Machine1$ git log --oneline --all
9814feb (HEAD -> feature1) Add feature1 to MyProgram.py
0f61082 (master, feature2) Create MyProgram.py
00d7b4d Add line 2 to File1.md
```

```
5d31ae0 Create File1.md and .gitignore
```

Kế tiếp, ta làm tương tự với nhánh feature2.

```
Machine1$ git checkout feature2
Switched to branch 'feature2'
Machine1$ cat >> MyProgram.py
else:
    if sys.argv[1] == 'feature2':
        print('Do feature2')
Machine1$ cat MyProgram.py
import sys

if len(sys.argv) == 1:
    print('Do nothing')
else:
    if sys.argv[1] == 'feature2':
        print('Do feature2')
Machine1$ python MyProgram.py feature2
Do feature2
Machine1$ git status
On branch feature2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working director
    modified:   MyProgram.py

no changes added to commit (use "git add" and/or "git commit -
a")
Machine1$ git diff HEAD
diff --git a/MyProgram.py b/MyProgram.py
index ed9a565..46378bc 100644
--- a/MyProgram.py
+++ b/MyProgram.py
@@ -2,3 +2,6 @@ import sys

 if len(sys.argv) == 1:
        print('Do nothing')
+else:
+       if sys.argv[1] == 'feature2':
+               print('Do feature2')
```

```
Machine1$ git add MyProgram.py
Machine1$ git commit -m "Add feature2 to MyProgram.py"
[feature2 22e4176] Add feature2 to MyProgram.py
 1 file changed, 3 insertions(+)
Machine1$ git log --oneline --all
22e4176 (HEAD -> feature2) Add feature2 to MyProgram.py
9814feb (feature1) Add feature1 to MyProgram.py
0f61082 (master) Create MyProgram.py
00d7b4d Add line 2 to File1.md
5d31ae0 Create File1.md and .gitignore
```

Để nhìn rõ hơn các nhánh:

```
Machine1$ git log --oneline --all --graph
* 22e4176 (HEAD -> feature2) Add feature2 to MyProgram.py
| * 9814feb (feature1) Add feature1 to MyProgram.py
|/
* 0f61082 (master) Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Tiếp theo, ta sẽ lần lượt merge nhánh feature1 và feature2 vào master. Để merge vào master thì đầu tiên cần checkout master đã.

```
Machine1$ git checkout master
Switched to branch 'master'
Machine1$ git merge feature1
Updating 0f61082..9814feb
Fast-forward
 MyProgram.py | 3 +++
 1 file changed, 3 insertions(+)
Machine1$ git log --oneline --all --graph
* 22e4176 (feature2) Add feature2 to MyProgram.py
| * 9814feb (HEAD -> master, feature1) Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Việc merge nhánh feature1 vào master diễn ra rất nhanh chóng và êm đềm. Kết quả chỉ đơn giản là HEAD -> master được cập nhật để trỏ vào cùng phiên bản mà feature1 đang trỏ tới. Git làm như vậy có đúng mong muốn của bạn

không?

Bây giờ, ta sẽ merge nhánh feature2 vào master.

```
Machine1$ git merge feature2
Auto-merging MyProgram.py
CONFLICT (content): Merge conflict in MyProgram.py
Automatic merge failed; fix conflicts and then commit the result.
```

Git báo là không merge tự động được vì có conflict. Đó là do phiên bản hiện giờ mà master trỏ tới và phiên bản mà feature2 trỏ tới đều được tạo ra từ cùng một phiên bản, nhưng ở cùng file và cùng dòng mỗi bên chỉnh sửa một kiểu; Git không biết nên merge như thế nào mới đúng nên để cho ta là người quyết định cách merge. Nếu xem file "MyProgram.py" trong phiên bản làm việc thì ta thấy Git đã tự động đánh dấu các phần mà xảy ra conflict:

```
Machine1$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   MyProgram.py

no changes added to commit (use "git add" and/or "git commit -
a")
Machine1$
Machine1$ cat MyProgram.py
import sys

if len(sys.argv) == 1:
    print('Do nothing')
else:
<<<<<<< HEAD
    if sys.argv[1] == 'feature1':
        print('Do feature1')
=======
    if sys.argv[1] == 'feature2':
        print('Do feature2')
>>>>>>> feature2
```

Ta sẽ dùng một editor như Nano, Vim, ... để chỉnh sửa file MyProgram.py theo mong muốn merge của ta. Đây là kết quả sau khi đã chỉnh sửa:

```
Machine1$ cat MyProgram.py
import sys

if len(sys.argv) == 1:
    print('Do nothing')
else:
    if sys.argv[1] == 'feature1':
        print('Do feature1')
    elif sys.argv[1] == 'feature2':
        print('Do feature2')
Machine1$ python MyProgram.py
Do nothing
Machine1$ python MyProgram.py feature1
Do feature1
Machine1$ python MyProgram.py feature2
Do feature2
```

Việc còn lại chỉ đơn giản là:

```
Machine1$ git add MyProgram.py
Machine1$ git commit -m "Merge feature2"
[master 00f095c] Merge feature2
Machine1$ git log --oneline --all --graph
*   00f095c (HEAD -> master) Merge feature2
|\
| * 22e4176 (feature2) Add feature2 to MyProgram.py
* | 9814feb (feature1) Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

:-)

3 Demo 3

Tiếp tục từ demo 2, nhưng bây giờ sẽ làm việc nhóm thông qua thùng chứa ở trên mạng

Do mình chỉ có một máy nên sẽ giả lập như này. Trong thư mục PlayWithGit:

- Thư mục Machine1 sẽ ứng với thùng chứa và phiên bản làm việc của một máy
- Thư mục Machine2 sẽ ứng với thùng chứa và phiên bản làm việc của một máy khác
- Thư mục Remote sẽ ứng với thùng chứa trên mạng (không cần có phiên bản làm việc)

```
Machine1$ cd ..
PlayWithGit$ mkdir Remote Machine2
PlayWithGit$ ls
Machine1  Machine2  Remote
PlayWithGit$ cd Remote/
Remote$ git init --bare # Chỉ tạo thùng chứa
Initialized empty Git repository in /home/kien/PlayWithGit/Remote/
```

Hiện giờ, thùng chứa của Machine1 đã có các phiên bản, thùng chứa của Remote thì đang rỗng, còn Machine2 mới chỉ là một thư mục bình thường và rỗng. Bây giờ, ta sẽ đẩy thùng chứa của Machine1 lên thùng chứa trên mạng là Remote. Nhưng trước hết thì Machine1 cần setup thông tin về thùng chứa trên mạng gồm tên (thường đặt là origin, đặt tên khác cũng được) và đường dẫn đến thùng chứa trên mạng (khi làm thực tế với Github thì đường dẫn sẽ có dạng “https://github.com/KienTrann/Remote.git”); vụ setup này chỉ cần làm một lần.

```
Remote$ cd ../Machine1/
Machine1$ git remote add origin ../Remote/
```

Bây giờ, ta đã sẵn sàng để đẩy thùng chứa của Machine1 lên thùng chứa trên mạng Remote. Ta cũng cần chỉ định là muốn đẩy nhánh nào lên. Ý ở đây là trong thùng chứa cục bộ ở mỗi máy của mỗi người, có thể có các nhánh riêng của người đó để test cái này test cái kia. Thường thì ta chỉ muốn đẩy nhánh master lên.

```
Machine1$ git push origin master
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (19/19), 1.53 KiB | 392.00 KiB/s, done.
Total 19 (delta 8), reused 0 (delta 0)
To ../Remote/
```

```
 * [new branch]      master -> master
Machine1$ git log --oneline --all --graph
*   00f095c (HEAD -> master, origin/master) Merge feature2
|\
| * 22e4176 (feature2) Add feature2 to MyProgram.py
* | 9814feb (feature1) Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Ta thấy có một con trỏ mới là `origin/master`, con trỏ này ứng với thùng chứa trên mạng.

Bây giờ, ta sẽ qua Machine2. Nhắc lại: Machine2 hiện là một thư mục bình thường và rỗng. Để setup Machine2 là thư mục được Git quản lý phiên bản với thùng chứa trên mạng Remote:

```
Machine1$ cd ../Machine2/
Machine2$ git clone ../Remote/ .
Cloning into '.'...
done.
Machine2$ ls -a
.  ..  File1.md  .git  .gitignore  MyProgram.py
Machine2$ git log --oneline --all --graph
*  00f095c (HEAD -> master, origin/master, origin/HEAD) Merge featu
|\
| * 22e4176 Add feature2 to MyProgram.py
* | 9814feb Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Bây giờ, ở Machine 2, từ phiên bản làm việc ta sẽ thêm tính năng 3 vào file MyProgram.py, rồi đưa vào thùng chứa phiên bản mới mà có thêm tính năng 3 này.

```
Machine2$ cat >> MyProgram.py
    elif sys.argv[1] == 'feature3':
        print('Do feature3')
Machine2$ cat MyProgram.py
```

```
import sys

if len(sys.argv) == 1:
    print('Do nothing')
else:
    if sys.argv[1] == 'feature1':
        print('Do feature1')
    elif sys.argv[1] == 'feature2':
        print('Do feature2')
    elif sys.argv[1] == 'feature3':
        print('Do feature3')
Machine2$ python MyProgram.py feature3
Do feature3
Machine2$ 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 restore <file>..." to discard changes in working director
    modified:   MyProgram.py

no changes added to commit (use "git add" and/or "git commit -
a")
Machine2$ git add MyProgram.py
Machine2$ git commit -m "Add feature3 to MyProgram.py"
[master 8118957] Add feature3 to MyProgram.py
 1 file changed, 2 insertions(+)
Machine2$ git log --oneline --all --graph
* 8118957 (HEAD -> master) Add feature3 to MyProgram.py
*   00f095c (origin/master, origin/HEAD) Merge feature2
|\
| * 22e4176 Add feature2 to MyProgram.py
* | 9814feb Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Như vậy, master ở thùng chứa cục bộ đã đi trước master ở thùng chứa trên

mạng. Bây giờ, ta sẽ đẩy master ở thùng chứa cục bộ lên thùng chứa trên mạng. Ở Machine2 thì ta chỉ cần dùng `git push` mà không cần truyền thêm `origin master` như ở Machine1, vì câu lệnh `git clone` lúc nãy đã tự động setup là nhánh master ở thùng chứa cục bộ của Machine2 sẽ tương ứng với nhánh master của origin (ở Machine1, ta cũng có thể setup như vậy bằng câu lệnh `git branch --set-upstream-to=origin/master` khi đang ở master).

```
Machine2$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 339 bytes | 339.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To /home/kien/PlayWithGit/Machine2/../Remote/
   00f095c..8118957  master -> master
Machine2$ git log --oneline --all --graph
* 8118957 (HEAD -> master, origin/master, origin/HEAD) Add feature3
*   00f095c Merge feature2
|\
| * 22e4176 Add feature2 to MyProgram.py
* | 9814feb Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Quay lại Machine1. Machine1 vẫn không biết là thùng chứa trên mạng đã được cập nhật bởi Machine2:

```
Machine1$ git log --oneline --all --graph
*   00f095c (HEAD -> master, origin/master) Merge feature2
|\
| * 22e4176 (feature2) Add feature2 to MyProgram.py
* | 9814feb (feature1) Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Giả sử bây giờ Machine1 muốn thêm tính năng 4 vào file MyProgram.py.

Chuyện gì sẽ xảy ra nếu Machine1 cứ cắm cổ code, cập nhật thùng chứa cục bộ rồi mới đẩy lên mạng? Sẽ xảy ra conflict và phải tốn chi phí để giải quyết conflict (nếu không tin thì bạn có thể thử ;-)). Conflict sẽ không xảy ra nếu đầu tiên Machine1 cập nhật thùng chứa trên mạng xuống đã:

```
Machine1$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
From ../Remote
   00f095c..8118957  master     -> origin/master
Machine1$ git log --oneline --all --graph
* 8118957 (origin/master) Add feature3 to MyProgram.py
*   00f095c (HEAD -> master) Merge feature2
|\
| * 22e4176 (feature2) Add feature2 to MyProgram.py
* | 9814feb (feature1) Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Để nhánh master cục bộ bắt kịp với nhánh master trên mạng:

```
Machine1$ git merge origin/master
Updating 00f095c..8118957
Fast-forward
 MyProgram.py | 2 ++
 1 file changed, 2 insertions(+)
Machine1$ git log --oneline --all --graph
* 8118957 (HEAD -> master, origin/master) Add feature3 to MyProgram.
*   00f095c Merge feature2
|\
| * 22e4176 (feature2) Add feature2 to MyProgram.py
* | 9814feb (feature1) Add feature1 to MyProgram.py
|/
* 0f61082 Create MyProgram.py
* 00d7b4d Add line 2 to File1.md
* 5d31ae0 Create File1.md and .gitignore
```

Bây giờ, có thể code được rồi đó.

Để cho tiện thì Git có câu lệnh `git pull` = `git fetch` + `git merge`.