### Hướng dẫn cài đặt và sử dụng GITHUB

## 1. Git là gì?

Git là một phần mềm [VCS](http://wiki.nukeviet.vn/programming:vcs) dùng để quản lý và kiểm tra các phiên bản mã nguồn khác nhau trong quá trình phát triển mã nguồn. Git là thế hệ mới nhất của các phần mềm này. Chúng ta sẽ lựa chọn Github làm nơi lưu trữ code mới cho **III AND 9** vì tính chuyên nghiệp của Github đối với Git.

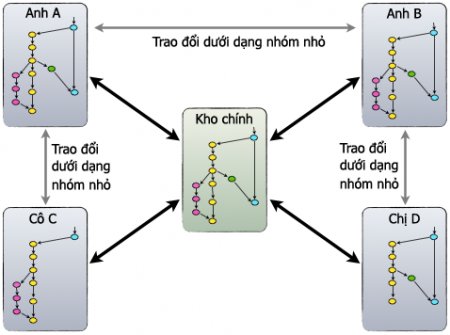
## 2. Tại sao sử dụng Git?

Git có nhiều ưu điểm, đặc biệt nó đã thay đổi cách các lập trình viên chia tách và nhập các nhánh phần mềm. Trong khi với svn, việc này luôn là ác mộng vì các xung đột trong quá trình tách/nhập, còn với Git việc này quá dễ dàng.

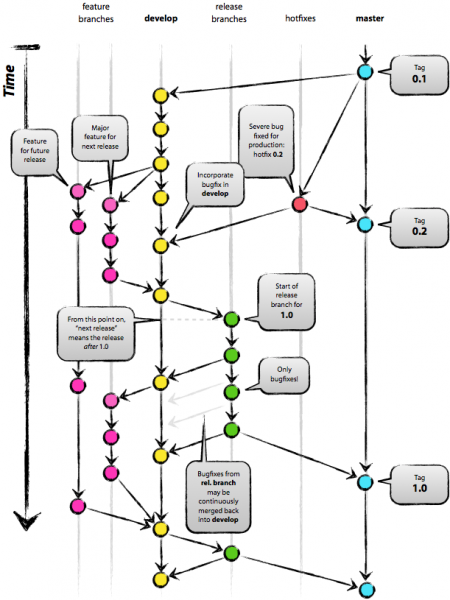
## 3. Đặc tính phi tập trung nhưng mang tính tập trung của Git

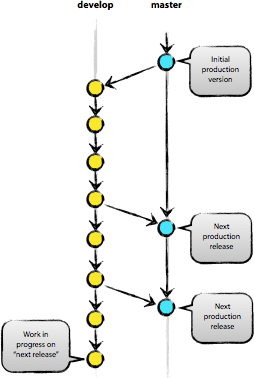
Trong khi mỗi lập trình viên đều sở hữu một kho code đầy đủ thì Git cho thiết lập một kho trung tâm thực sự, tất cả các lập trình viên đều có thể kéo (pull) các thay đổi từ đó về kho của mình và đẩy (push) các thay đổi về kho trung tâm. Tuy nhiên, bên cạnh những mối quan hệ push-pull tập trung, mỗi nhà phát triển cũng có thể kéo (pull) thay đổi từ các đồng nghiệp khác để tạo thành các nhóm nhỏ hơn.

Hình dưới đây cho thấy một dự án với 1 kho code trung tâm và 4 người, họ hình thành 3 nhóm nhỏ hơn là nhóm của Anh A - Anh B, Anh A - Chị C, Anh B - Cô D. Mỗi nhóm nhỏ này có thể phụ trách phát triển một tính năng thuộc hệ thống rồi sau đó mới đẩy về kho trung tâm. Như vậy mô hình này rất hiệu quả cho các dự án lớn.



## 4. Cấu trúc tổ chức kho code



* Hai nhánh (branch) đi suốt chiều dài phát triển code là Master và Develop (xem hình)[](http://wiki.nukeviet.vn/_detail/programming:main_branch.png?id=programming:github_rule).
* Trong đó:
  + Nhánh chính là nhánh Master, nhánh này luôn đảm bảo rằng code được lưu trữ trên đó là phiên bản chính thức mới nhất đang được phát hành.
  + Nhánh Develop là nhánh được cập nhật liên tục các đóng góp của tất cả mọi người ở mọi thời điểm. Nhánh này sẽ tiếp nhận các đóng góp của mọi người gửi đến thông qua việc tiếp nhận Pull Request cũng như tiếp nhận việc nhập các nhánh khác vào (integration branch). Khi nhánh Develop đạt độ chín mùi, nó sẽ được nhập vào nhánh Master đồng thời được dán nhãn phiên bản và quá trình phát hành phiên bản được tiến hành.
* Ngoài 2 nhánh trên, kho code còn có các nhánh hỗ trợ phát triển, và nó chỉ tồn tại trong một giai đoạn nào đó của dự án nhằm phục vụ những mục đích nhất định, gồm có các nhánh sau:
  + Feature branches - Các nhánh tính năng
  + Release branches - Các nhánh hỗ trợ phát hành phiên bản
  + Hotfix branches - Các nhánh hỗ trợ vá lỗi nhanh

**5. Tạo Github account**

Bạn có thể tạo Github account rất dễ dàng, miễn phí và nhanh chóng tại đây [https://github.com/signup/free](https://github.com/signup/free" \t "_blank)

#### 6. Cài đặt Git trên window

Chú ý: các hướng dẫn này dành cho phiên bản Git-1.8.0, các phiên bản khác có thể có một số khác biệt.

Download git tại địa chỉ <http://git-scm.com/download/win> . Sau khi download thành công ta sẽ được file **Git-1.8.0-preview20121022.exe** (tên file có thể khác tùy theo phiên bản).

Để tiến hành cài đặt, nhấp đôi file Git-1.8.0-preview20121022.exe, tại giao diện thứ nhất của chương trình nhấp “Next”, tại giao diện thứ hai nhấp “Next”; tiếp theo, chọn thư mục sẽ cài đặt phần mềm vào hoặc để mặc định và nhấp “Next” … tại các giao diện tiếp theo nếu không có tùy chọn khác hãy nhấp “Next” cho đến khi cài đặt hoàn tất.

#### 7. Làm việc với Git Bash

***7.1 Tạo SSH key***

Khởi động Gitbash

* cd ~/.ssh
* ssh-keygen –t rsa –C “youraddress@email.com”  (Bỏ qua phần đặt tên)
* Mở file “id\_rsa.pub” trong thư mục “C:\Users\<Tên>\.ssh”, copy nội dung vào trong Github account của bạn trên máy chủ: https://github.com/settings/ssh  (Vào Add SSH key)
* Test kiểm tra: ssh –T git@github.com (Nếu nhận được dòng thông báo “Hi username! You’ve successfully authenticated, but Github does not provide shell access.” nghĩa là cài đặt đã thành công.

Xem thêm hướng dẫn cho Windows tại đây: <http://help.github.com/win-set-up-git/>

***7.2 Cài đặt thông tin cá nhân***

* git config --global user.name “Tên của bạn”
* git config --global user.email “Địa chỉ email của bạn”

**8. Đưa file từ client lên Github**

Sau khi Git client được cài đặt và cấu hình đúng, bước tiếp theo là khởi tạo Git trên máy cá nhân của bạn và tạo ra các tương tác thực sự tới Github.

***8.1 Khởi tạo thư mục repo trên máy client (máy của lập trình viên)***

* mkdir C:\<Tên repo>
* cd <Tên repo>
* git init

***8.2 Kết nối tới Github repo***

git remote add origin git@github.com:(Tên repo)  
*Ví dụ tên repo: giaidieu-com:giaidieu.com.git*  
  
***8.3 Luôn cập nhật nội dung từ server trước khi tải nội dung mới từ client lên***

git pull git@github.com:(Tên repo) master  
*Ví dụ tên repo: giaidieu-com/giaidieu.com.git*  
  
***8.4 Kiểm tra trạng thái thay đổi***

git status  
  
***8.5 Đưa những file vào danh sách trước khi commit (bỏ qua nếu không có file mới được tạo)***

git add -A  (-A Tất cả files)  
  
***8.6 Commit những thay đổi trước khi push lên server***

git commit -a -m “Thông tin ngắn gọn về thay đổi này” (-a Tất cả thay đổi)  
  
***8.7 Tải lên server nhánh master***

git push -u origin master   
  
**9. Tạo nhánh (branch) riêng**

***9.1 Xem toàn bộ các nhánh đang có***  
git branch -a  
  
***9.2 Tạo nhánh mới***  
git branch <tên mới>  
  
***9.3 Chuyển nhánh***  
git checkout <tên nhánh>  
  
***9.4 Xóa nhánh***

git branch -d <Tên branch>  
  
**10. Nhập nhánh con vào nhánh hiện tại**

git merge <Tên nhánh>  
  
**11. Liệt kê một số lệnh hay dùng**

* git checkout <tên nhánh hoặc mã hash của commit> : chuyển thư mục hiện tại về một commit trong quá khứ hoặc một nhánh khác
* git checkout -b <tên nhánh> : tạo một nhánh mới và checkout sang nhánh đó
* git branch : hiện danh sách tất cả các branch
* git branch <tên nhánh> : tạo một nhánh mới
* git pull : lấy tất cả các cập nhật mới nhất từ server về
* git push : đưa những thay đổi ở máy hiện tại lên server
* git merge <tên nhánh hoặc mã hash của commit> : ghép những thay đổi ở nhánh (hoặc commit) này sang nhánh hiện tại
* git add <đường dẫn đến file> : cho các file này vào diện chuẩn bị commit (staging)
* git commit [-a] [-m "<Tóm tắt nội dung commit>"] : Commit những thay đổi đã tạo trong nhánh hiện tại. Thêm -a để tự động stage tất cả các file đã thay đổi. Thêm -m "<tóm tắt>" để ghi commit message trực tiếp mà không phải vào giao diện vim (default trên Windows, khó sử dụng với người mới).

*NOTE* Trong Vim, nhấn i để vào chế độ edit, nhấn Escape :wq Enter để ghi file và quit.

* git stash : ghi tất cả các thay đổi chưa commit vào vùng nhớ tạm (stash). Sau lệnh này hiện trạng của repo là clean.
* git stash pop : apply các thay đổi trong stash vào nhánh hiện tại nhưng chưa commit. Hai lệnh này thường dùng để hỗ trợ di chuyển qua lại giữa các nhánh mà không cần commit những công việc đang dang dở.

Ngoài ra còn có một số lệnh mà có thể dùng giao diện đồ họa để thay thế thì tiện hơn (gitk, Github 4 Windows, GitTortoise, ...):

* git status : xem hiện trạng của repo hiện tại (có file nào chưa commit, file nào đã thuộc diện staging,...)
* git diff : so sánh hiện trạng của thư mục hiện tại với commit gần đây nhất
* git whatchanged : hiện lịch sử commit của nhánh hiện tại

### 12. Luồng công việc

1. Clone repository của nhóm (chỉ cần làm một lần):

git clone https://github.com/lewtds/project-g3.git

1. Xem xét việc cần làm, đặt một cái tên gợi nhớ cho công việc đó (trong ví dụ này giả định là fix-footer) sau đó tạo một branch mới dựa trên branch gh-pages với tên đó:

git checkout -b fix-footer

1. Sửa code, viết code, test code và commit trên branch vừa tạo
2. Khi cảm thấy ổn oy thì chuẩn bị push lên server:
   * Tải các thay đổi mới nhất của mọi người trên server:

git pull

* + Checkout branch gh-pages và merge những thay đổi vừa tạo:
  + git checkout gh-pages
  + git merge fix-footer
  + Nếu có conflict thì resolve và commit
  + Push lên server:

git push

1. Quay lại bước 2

### 13. Cách giải quyết conflict:

Conflict xảy ra khi 2 người (hoặc một người nhưng ở 2 nhánh) cùng chỉnh sửa một phần của một file. VD:

(file gốc)

Trăm năm trong cõi người ta

Chữ tài chữ mệnh khéo là ghét nhau

(nhánh a)

Trăm năm trong cõi người ta

Chữ tài chữ mệnh khéo là thích nhau

(nhánh b)

Trăm năm trong cõi người ta

Chữ sinh chữ mệnh khéo là ghét nhau

Khi merge nhánh a vào nhánh b (hoặc ngược lại) thì sẽ xảy ra conflict với message kiểu như thế này:

Auto-merging <tên nhánh>

CONFLICT (content): Merge conflict in <tên file conflict>

Automatic merge failed; fix conflicts and then commit the result.

Conflict sẽ tạo ra tình huống dở dang chưa commit cho repo. Trong file conflict sẽ hiện ra đoạn so sánh để giúp lập trình viên giải quyết conflict:

Trăm năm trong cõi người ta

<<<<<<< HEAD

chữ sinh chữ mệnh khéo là ghét nhau

=======

chữ tài chữ mệnh khéo là thích nhau

>>>>>>> a

Có thể thấy 7 dấu bằng phân cách 2 phiên bản khác nhau. HEAD là phiên bản ở nhánh hiện tại (nhánh b) còn a là phiên bản ở nhánh a. Để resolve chúng ta chọn một phương án phù hợp nhất từ 2 phương án. Rồi xóa những ký hiệu do git để lại trong file (ở đây chúng ta chọn phương án sử dụng cả 2 branch):

Trăm năm trong cõi người ta

Chữ sinh chữ mệnh khéo là thích nhau

Sau đó commit file này (vì lệnh merge không thành công tạo tình huống dở chừng trong repo):

git add kieu

git commit

Git sẽ hiện ra giao diện commit với một commit message được tạo sẵn kiểu ntn:

Merge branch 'a' into b

Conflicts:

kieu

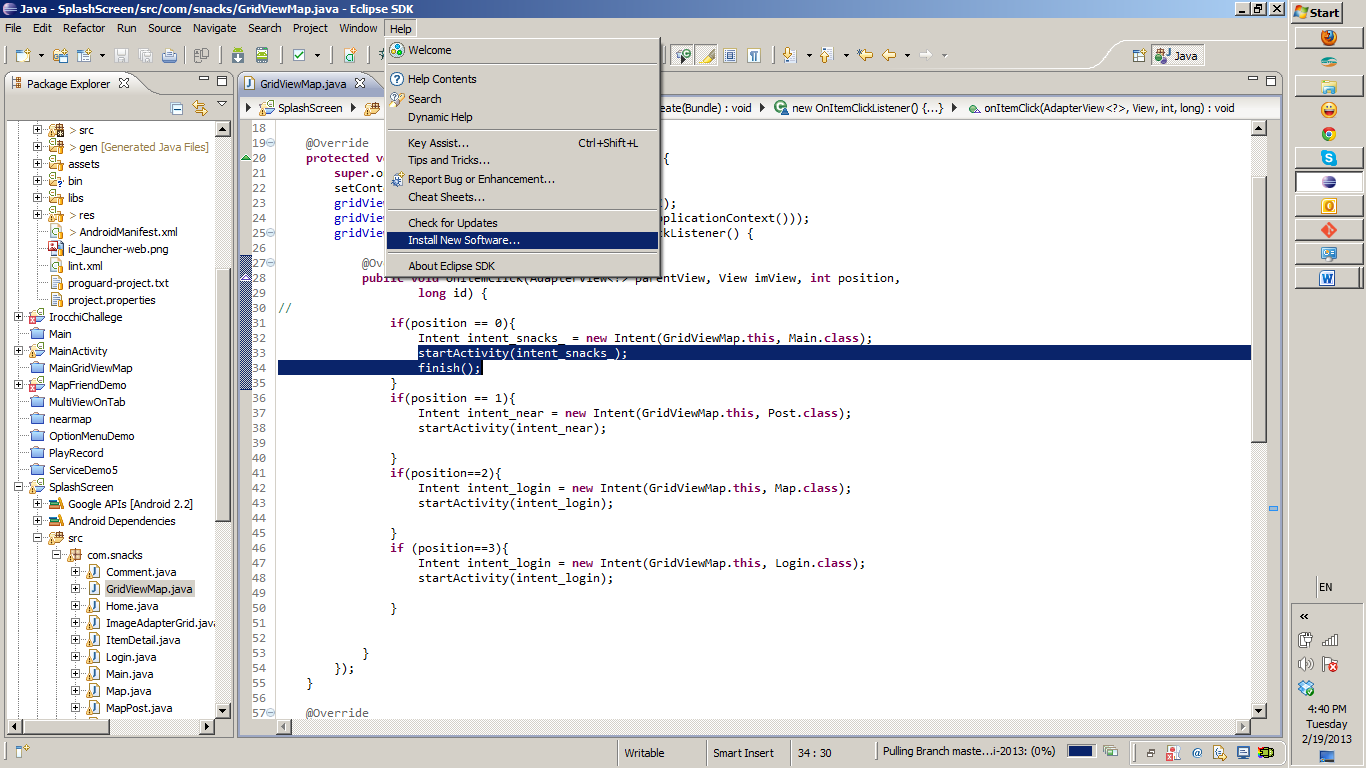
Giữ nguyên hoặc thêm gì nếu muốn và tiếp tục commit (nhấn Escape, sau đó gõ :wq rồi Enter).

* + Các bạn có thể tham khảo thêm bằng tiếng anh các lệnh tại địa chỉ: <https://help.github.com/articles/>
  + Hoặc tiếng việt tại địa chỉ: <http://www-cs-students.stanford.edu/~blynn/gitmagic/intl/vi/ch03.html#_push_ng%C6%B0%E1%BB%A3c_v%E1%BB%9Bi_pull>

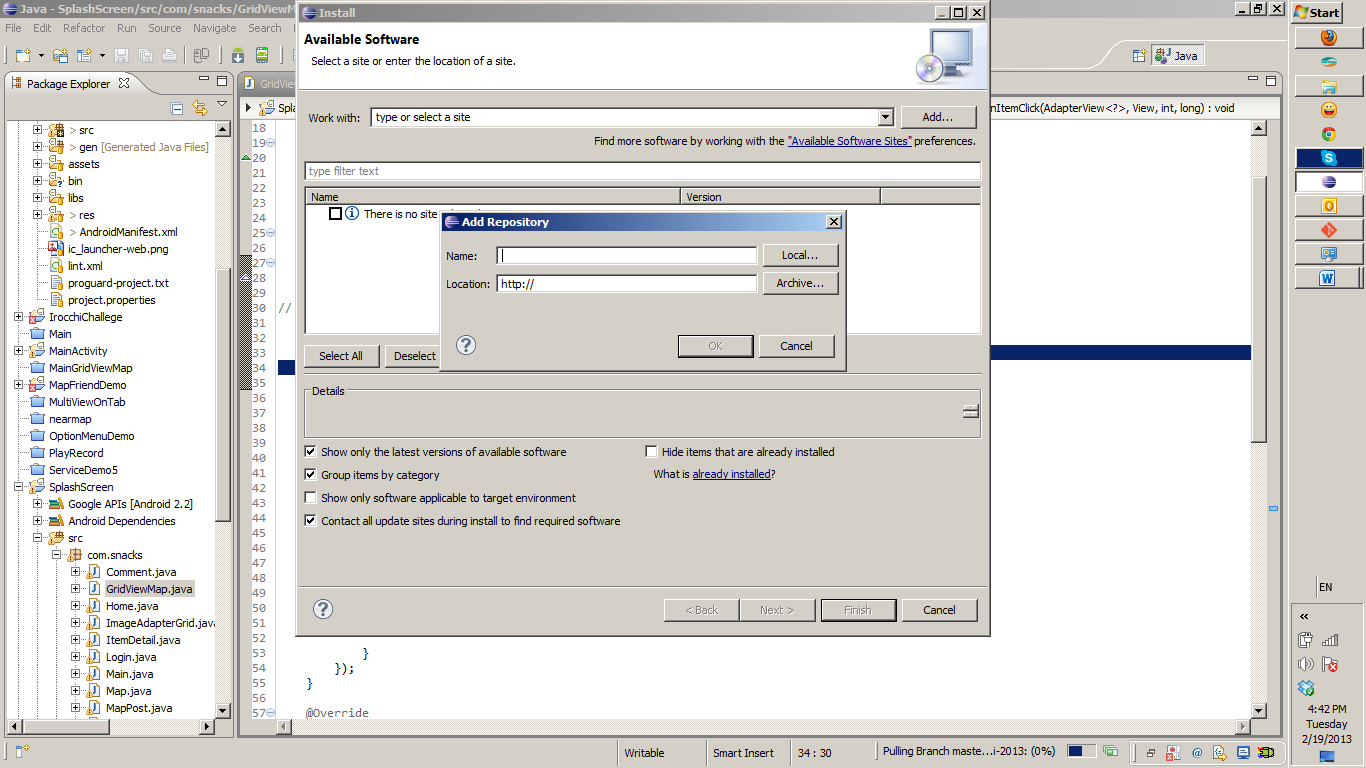
**II. Sử dụng GitHub trên Eclipse:**

1. **Cài đặt Github:**

- Chọn "Help --> Install new software"



- Click vào add, xuất hiện hộp thoại sau:



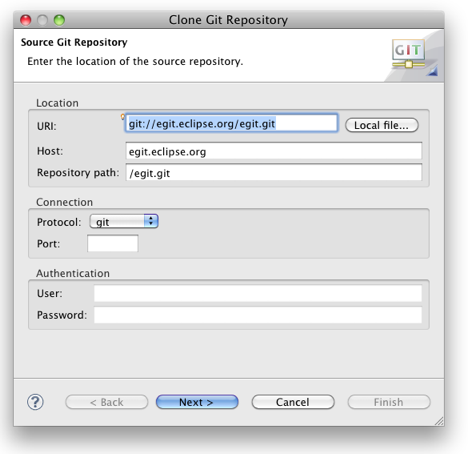
- Tại hộp thoại, ta đặt tên bất kỳ ví dụ: "Eclipse Git plugin" và Location như sau: <http://download.eclipse.org/egit/updates-1.3/>  
  
Rồi sau đó click OK để bắt đầu quá trình cài đặt

- Sau khi thực hiện 1 số thao tác cài đặt, chương trình sẽ yêu cầu người sử dụng Restart bộ Elipse, click vào Restart.

1. **Làm việc những kho dữ liệu từ xa trên Github:**

Sử dụng Wizard Clone Git bạn có thể sao chép kho từ xa bằng cách sử dụng các giao thức có sự vận hành khác nhau.

- Để bắt đầu Import một dự án từ Git ta làm như sau: **Import... > Git > Projects from Git > Next > Clone....** Xuất hiện hộp thoại sau:



Trong đó:

+ URI: Đường dẫn của 1 tập tin hệ thống, nó được tự động đồng bộ với các trường khác.

+ Host: Tên của host từ xa nếu được nhân bản từ tập tin hệ thống.

+ Repository path: Đường dẫn trên tập tin hệ thống.

+ Protocol: Các giao thức.

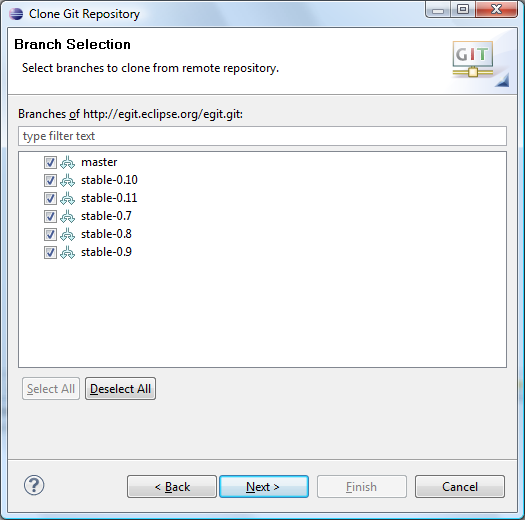
+ Port: Số cổng

+ User: Tài khoản của người sử dụng

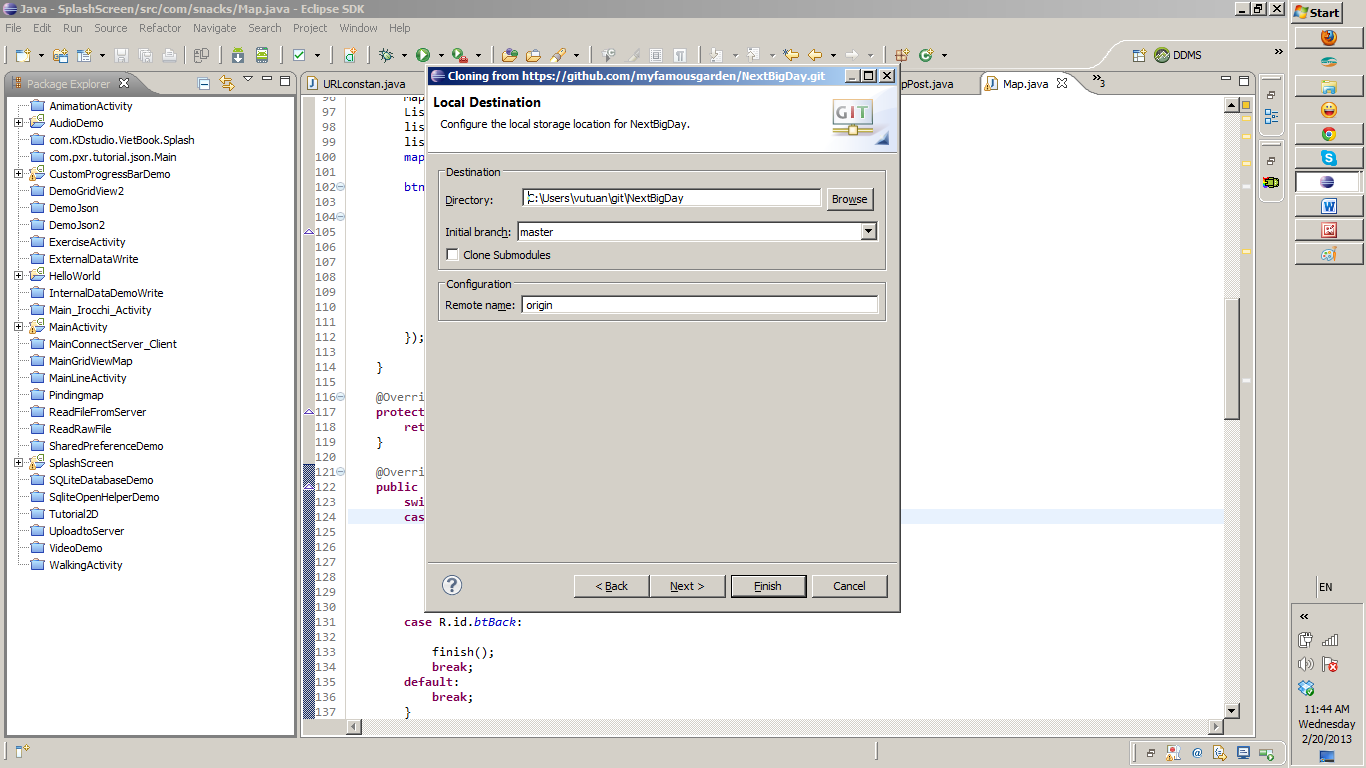
+ PasswordL: Mật khẩu của người sử dụng

**Chú ý:** Nếu firewall vẫn đang bật thì hãy thiết lập cài đặt proxy (**Preferences > General > Network Connections)**. Nhiều proxy HTTP được cấu hình để chặn các URL có chứa một tên người dùng (và / hoặc mật khẩu, do đó nó được đề nghị sử dụng User/ Password.

- Trên trang tiếp theo chọn những project được nhân bản:



- Tiếp theo, xác định nơi bạn muốn lưu trữ:



Trong đó:

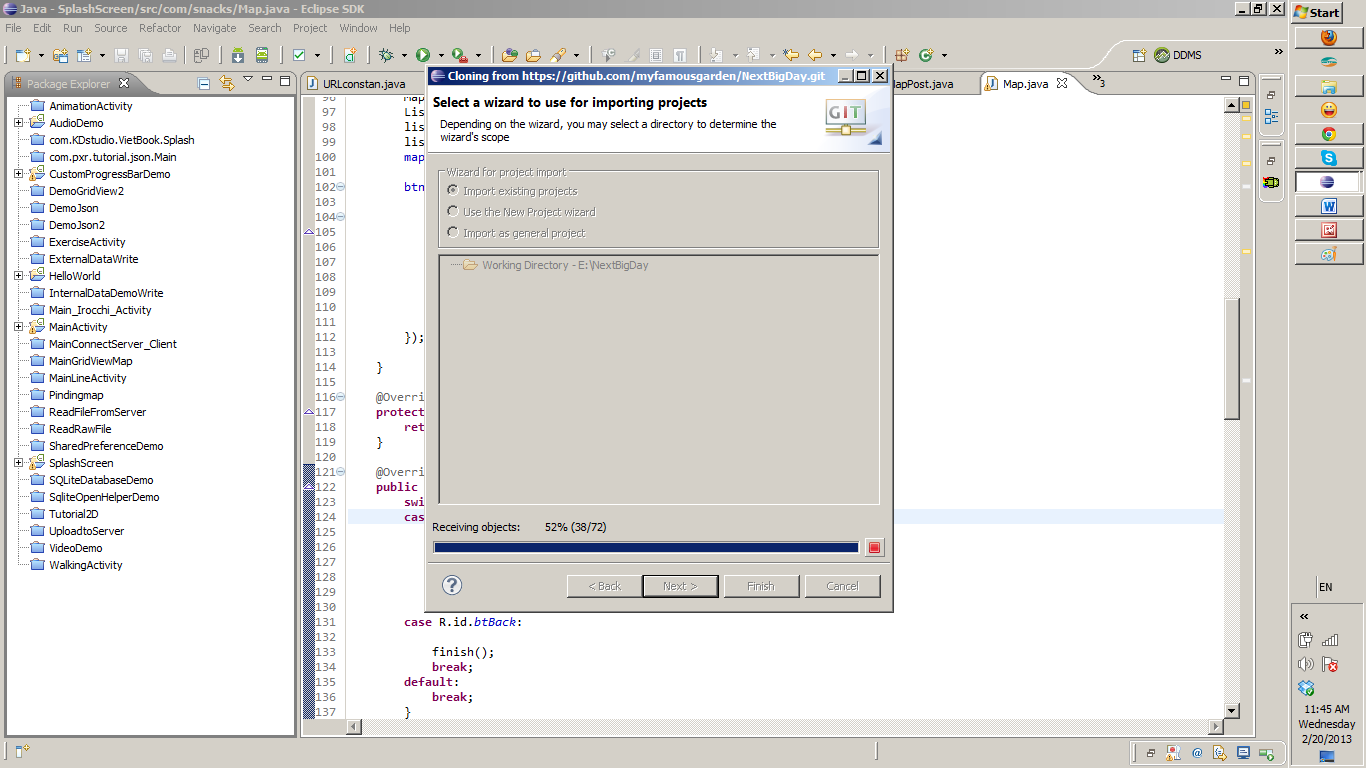
+ Directory: tạo ra thư mục lưu trữ.

+ Initial branch: chọn những Location Branch và kiểm tra.

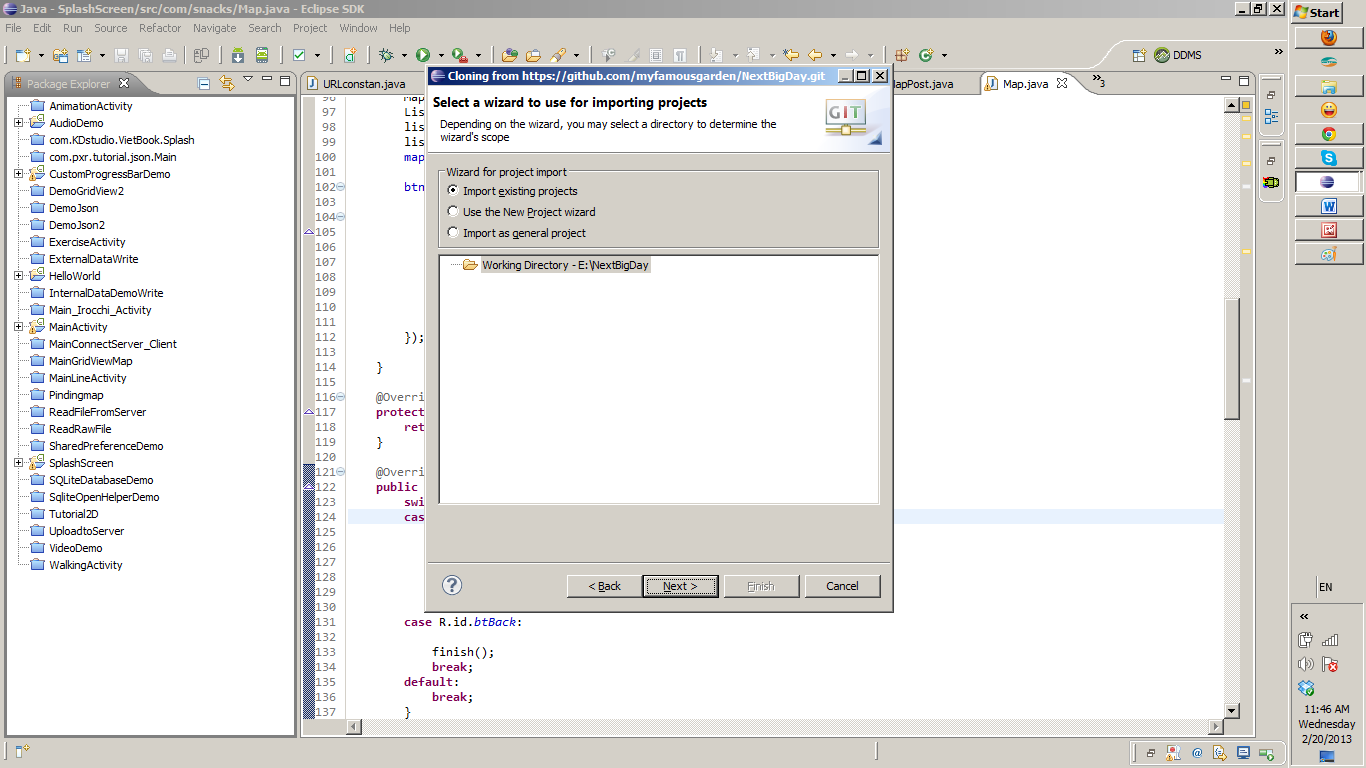
+ Remote name: xác định tên cho việc điều khiển dữ liệu từ xa.

- Ấn Next để tiếp tục quá trình cài đặt.

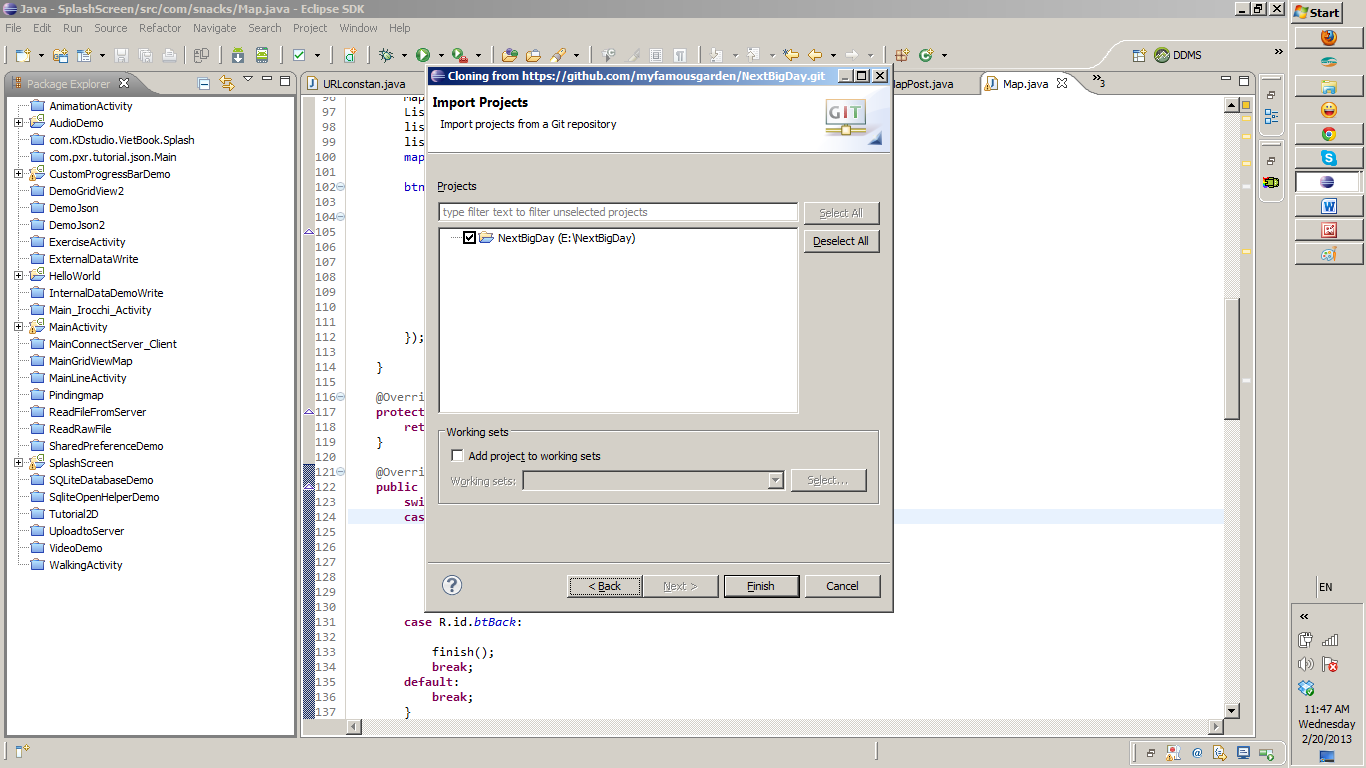
- Tại màn hình kế tiếp, sau quá trình đợi nhân bản



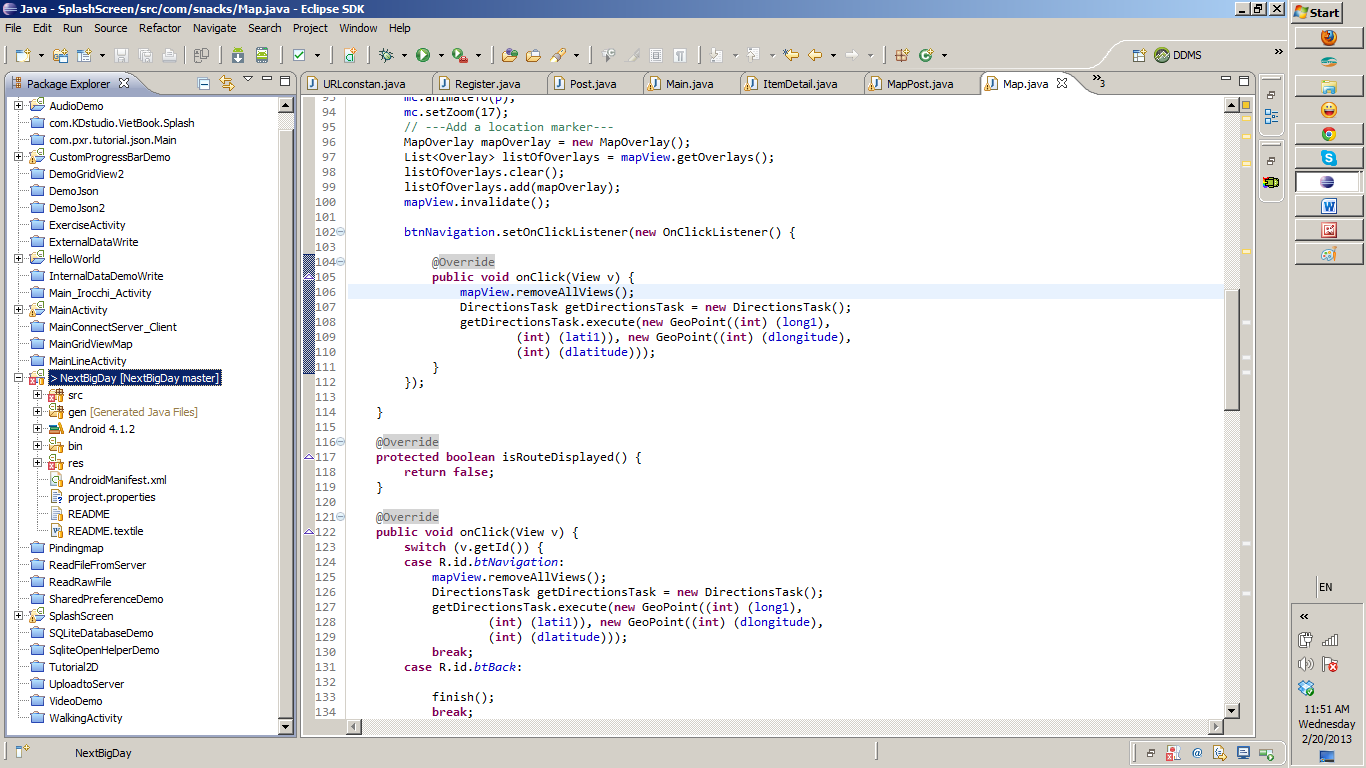
Ta chọn như sau:



- Chọn Next, xuất hiện màn hình thông báo Project sẽ được thêm vào từ file nguồn:



- Ấn Finish để hoàn tất quá trình Import



1. **Các chức năng hay sử dụng của Git trong Eclipse:**

Sau khi đã Import một project, ta sử dụng các chức năng có trong git: Click chuột phải vào Project, chọn Team…

- Chức năng Commit:

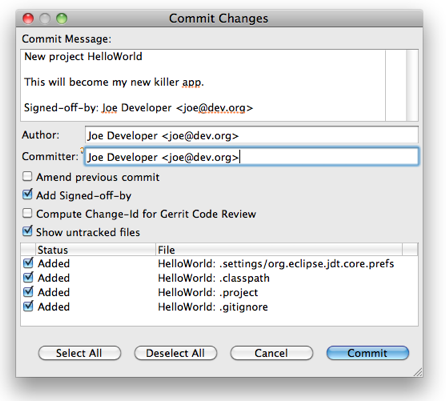
+ Commit Message: Nhập 1 đoạn tin giải thích những thay đổi của bạn.

Theo mặc định, author và commiter được lấy từ các tập tin .gitconfig

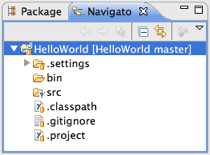
+ Bạn có thể click **Add Signed-off-by** để thêm một *Signed-off-by:* tag.

+ Nếu bạn đag thay đổi commit của 1 author khác, bạn có thể thay đổi trường tác giả để cung cấp tên và địa chỉ email của tác giả.

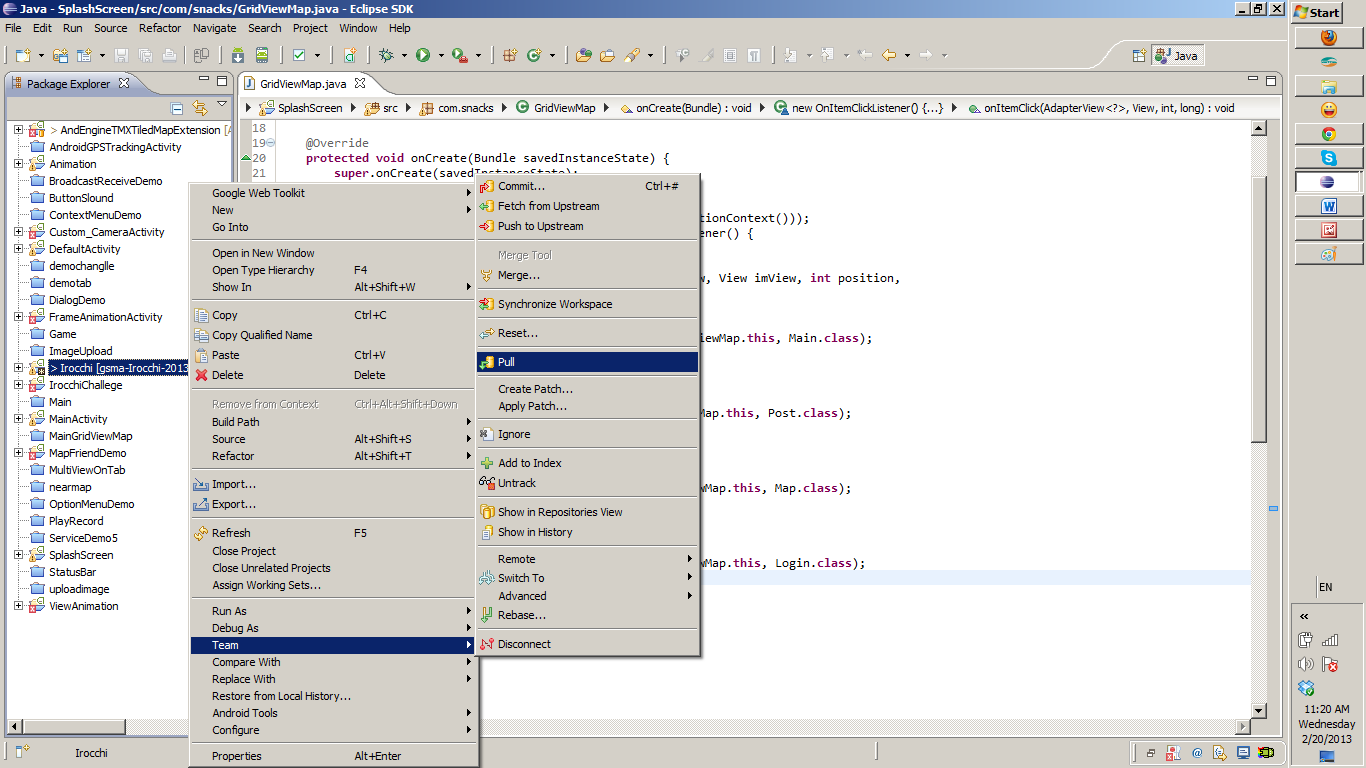
+ Chọn Commit để chuyển lên thay đổi đầu tiên của bạn.



+ Chú ý: Khi tập tin thay đổi sẽ có thông báo như sau:



- Chức năng Pull: lấy về những đoạn code đã thay đổi từ Project nguồn



- Chức năng Reset: nếu bạn không muốn thay đổi hoặc bỏ những thay đổi từ File của mình thì sử dụng **Team->Reset..**

**-** Chức năng Merge: khi lấy về, những thay đổi từ File nguồn và những File theo dõi sẽ được cập nhật. Sau đó, những File theo dõi sẽ được trộn với những thay đổi mới.

**-** Chức năng Show in History: sẽ đưa ra tất cả các thay đổi

**-**-------------------------------------HẾT----------------------------------------------

--fixup <commit> use autosquash formatted message to fixup specified co

mmit

--squash <commit> use autosquash formatted message to squash specified c

ommit

--reset-author the commit is authored by me now (used with -C/-c/--am

end)

-s, --signoff add Signed-off-by:

-t, --template <file>

use specified template file

-e, --edit force edit of commit

--cleanup <default> how to strip spaces and #comments from message

--status include status in commit message template

-S, --gpg-sign[=<key id>]

GPG sign commit

Commit contents options

-a, --all commit all changed files

-i, --include add specified files to index for commit

--interactive interactively add files

-p, --patch interactively add changes

-o, --only commit only specified files

-n, --no-verify bypass pre-commit hook

--dry-run show what would be committed

--short show status concisely

--branch show branch information

--porcelain machine-readable output

--long show status in long format (default)

-z, --null terminate entries with NUL

--amend amend previous commit

--no-post-rewrite bypass post-rewrite hook

-u, --untracked-files[=<mode>]

show untracked files, optional modes: all, normal, no.

(Default: all)

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 1 commit.

# (use "git push" to publish your local commits)

#

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# modified: helloworld.txt

#

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git checkout long

error: Your local changes to the following files would be overwritten by checkou

t:

helloworld.txt

Please, commit your changes or stash them before you can switch branches.

Aborting

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git commit -m ""

Aborting commit due to empty commit message.

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git checkout long

error: Your local changes to the following files would be overwritten by checkou

t:

helloworld.txt

Please, commit your changes or stash them before you can switch branches.

Aborting

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git push origin

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)

Username for 'https://github.com': vietnamquehuongtoi

Password for 'https://vietnamquehuongtoi@github.com':

Everything up-to-date

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git checkout long

error: Your local changes to the following files would be overwritten by checkou

t:

helloworld.txt

Please, commit your changes or stash them before you can switch branches.

Aborting

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git status

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# modified: helloworld.txt

#

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git reset head

Unstaged changes after reset:

M helloworld.txt

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git status

# On branch 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: helloworld.txt

#

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

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git add .

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git commit -m

error: switch `m' requires a value

usage: git commit [options] [--] <filepattern>...

-q, --quiet suppress summary after successful commit

-v, --verbose show diff in commit message template

Commit message options

-F, --file <file> read message from file

--author <author> override author for commit

--date <date> override date for commit

-m, --message <message>

commit message

-c, --reedit-message <commit>

reuse and edit message from specified commit

-C, --reuse-message <commit>

reuse message from specified commit

--fixup <commit> use autosquash formatted message to fixup specified co

mmit

--squash <commit> use autosquash formatted message to squash specified c

ommit

--reset-author the commit is authored by me now (used with -C/-c/--am

end)

-s, --signoff add Signed-off-by:

-t, --template <file>

use specified template file

-e, --edit force edit of commit

--cleanup <default> how to strip spaces and #comments from message

--status include status in commit message template

-S, --gpg-sign[=<key id>]

GPG sign commit

Commit contents options

-a, --all commit all changed files

-i, --include add specified files to index for commit

--interactive interactively add files

-p, --patch interactively add changes

-o, --only commit only specified files

-n, --no-verify bypass pre-commit hook

--dry-run show what would be committed

--short show status concisely

--branch show branch information

--porcelain machine-readable output

--long show status in long format (default)

-z, --null terminate entries with NUL

--amend amend previous commit

--no-post-rewrite bypass post-rewrite hook

-u, --untracked-files[=<mode>]

show untracked files, optional modes: all, normal, no.

(Default: all)

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git commit -m "dsfgsdg "

[master c108d6c] dsfgsdg

1 file changed, 1 insertion(+), 1 deletion(-)

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git push origin

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)

Username for 'https://github.com': vietnamquehuongtoi

Password for 'https://vietnamquehuongtoi@github.com':

Counting objects: 5, done.

Writing objects: 100% (3/3), 273 bytes, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/vietnamquehuongtoi/IIITP.git

0f9db1c..c108d6c master -> master

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git checkout long

Switched to branch 'long'

Computer@COMPUTER-PC /d/test/IIITP (long)

$ git status

# On branch long

# 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: helloworld.txt

#

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

Computer@COMPUTER-PC /d/test/IIITP (long)

$ git add .

Computer@COMPUTER-PC /d/test/IIITP (long)

$ git commit -m

error: switch `m' requires a value

usage: git commit [options] [--] <filepattern>...

-q, --quiet suppress summary after successful commit

-v, --verbose show diff in commit message template

Commit message options

-F, --file <file> read message from file

--author <author> override author for commit

--date <date> override date for commit

-m, --message <message>

commit message

-c, --reedit-message <commit>

reuse and edit message from specified commit

-C, --reuse-message <commit>

reuse message from specified commit

--fixup <commit> use autosquash formatted message to fixup specified co

mmit

--squash <commit> use autosquash formatted message to squash specified c

ommit

--reset-author the commit is authored by me now (used with -C/-c/--am

end)

-s, --signoff add Signed-off-by:

-t, --template <file>

use specified template file

-e, --edit force edit of commit

--cleanup <default> how to strip spaces and #comments from message

--status include status in commit message template

-S, --gpg-sign[=<key id>]

GPG sign commit

Commit contents options

-a, --all commit all changed files

-i, --include add specified files to index for commit

--interactive interactively add files

-p, --patch interactively add changes

-o, --only commit only specified files

-n, --no-verify bypass pre-commit hook

--dry-run show what would be committed

--short show status concisely

--branch show branch information

--porcelain machine-readable output

--long show status in long format (default)

-z, --null terminate entries with NUL

--amend amend previous commit

--no-post-rewrite bypass post-rewrite hook

-u, --untracked-files[=<mode>]

show untracked files, optional modes: all, normal, no.

(Default: all)

Computer@COMPUTER-PC /d/test/IIITP (long)

$ git commit -m "nhan thay doi tu Vinh"

[long b66288c] nhan thay doi tu Vinh

1 file changed, 1 insertion(+), 1 deletion(-)

Computer@COMPUTER-PC /d/test/IIITP (long)

$ git push

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)

Username for 'https://github.com': vietnamquehuongtoi

Password for 'https://vietnamquehuongtoi@github.com':

Everything up-to-date

Computer@COMPUTER-PC /d/test/IIITP (long)

$ git push origin long

Username for 'https://github.com': vietnamquehuongtoi

Password for 'https://vietnamquehuongtoi@github.com':

Counting objects: 5, done.

Writing objects: 100% (3/3), 279 bytes, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/vietnamquehuongtoi/IIITP.git

\* [new branch] long -> long

Computer@COMPUTER-PC /d/test/IIITP (long)

$ git checkout master

Switched to branch 'master'

Computer@COMPUTER-PC /d/test/IIITP (master)

$ git merge long

Auto-merging helloworld.txt

CONFLICT (content): Merge conflict in helloworld.txt

Automatic merge failed; fix conflicts and then commit the result.

Computer@COMPUTER-PC /d/test/IIITP (master|MERGING)

$