2017-01-20 Git版本控制

==========
參考講義：20161112-Big Data之處理與分析實務班.pdf
參考網站(概念)：http://dylandy.github.io/Easy-Git-Tutorial/
參考網站(概念)：http://ithelp.ithome.com.tw/articles/10140055
參考網站(概念)：https://ihower.tw/git/remote.html
==========

每個專案下面都有Repository=>程式碼的倉庫，Commit的集合
（包含修改時間跟敘述的基本資訊）

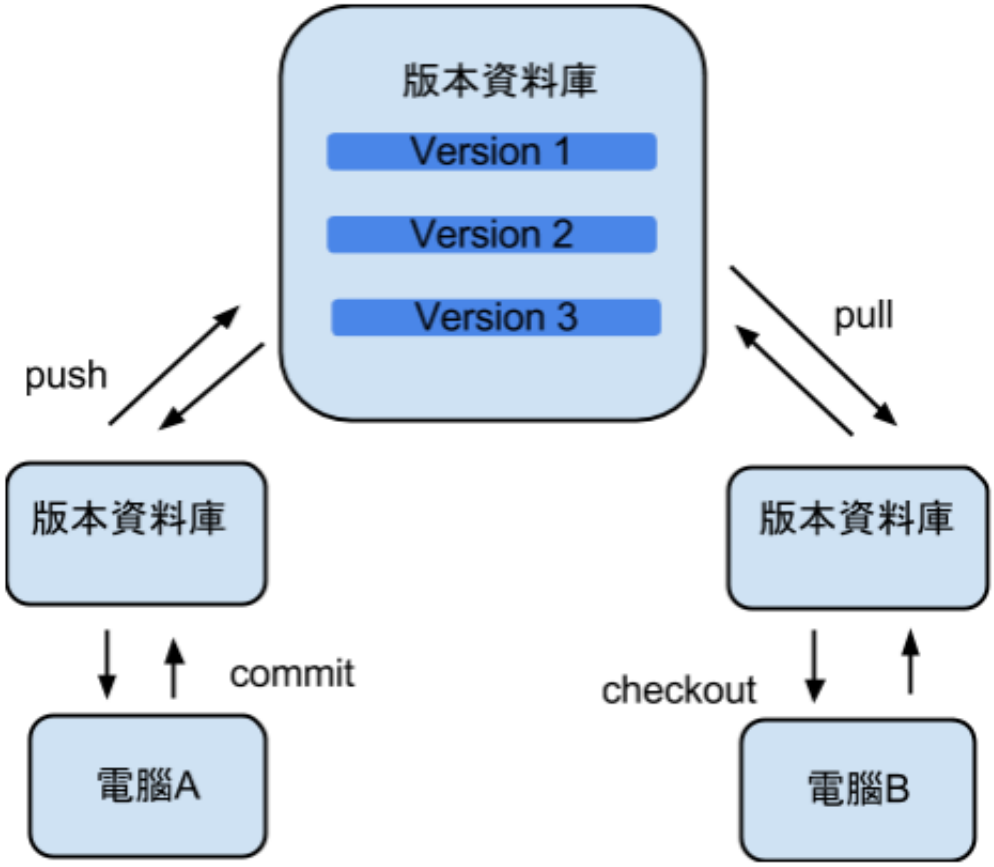
英文：Repository(倉庫、貯藏室)

版本控制都必須自行下Commit指令後，才會生成新版本(Version)

在自己的本機上做版本控管，把結果存在上面

CVS集中式

Distributed VCS
分散式版本控制系統讓本地端也擁有完整的Repository，就沒有上述集中式的問題，即時沒網路，照常可以commit和看history log，也不用擔心server備份。例如Git、Mercurial(Hg)、Bazaar等就是屬於分散式版本控制系統。



Mac:Git安裝指令==>brew install git

```
Mac-10:~ mac$ brew install git
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 1 tap (homebrew/core).
==> New Formulae
cnats                          mingw-w64-binutils             opencoarrays
imagemagick@6                  molecule                       watchexec
==> Updated Formulae
abcde                          gitup                          pdftoedn
afl-fuzz                       gnupg-pkcs11-scd               pius
ansible                        godep                          pngcrush
antigen                        groonga                        poppler
aria2                          gtk+3                          pyenv
armor                          h2o                            rocksdb
arping                         htmlcleaner                    shadowsocks-libev
aws-sdk-cpp                    httrack                        speedtest_cli
awscli                         hyperscan                      sshguard
certbot                        ios-webkit-debug-proxy         swaks
dbt                            iso-codes                      swiftformat
deis                           jenkins                        tcpkali
deisctl                        kawa                           terragrunt
diff-pdf                       kobalt                         tomcat
docker-compose                 libgcrypt                      unshield
docker-machine                 libgtop                        vim
```

```
flow                           mongoose                       yash
fossil                         openshift-cli                  youtube-dl
fwup                           orientdb
fzf                            pdf2htmlex
==> Renamed Formulae
scala210 -> scala@2.10                    scala211 -> scala@2.11

==> Downloading https://homebrew.bintray.com/bottles/git-2.11.0.yosemite.bottle.
######################################################################## 100.0%
==> Pouring git-2.11.0.yosemite.bottle.tar.gz
==> Caveats
Bash completion has been installed to:
  /usr/local/etc/bash_completion.d

zsh completion has been installed to:
  /usr/local/share/zsh/site-functions

zsh functions have been installed to:
  /usr/local/share/zsh/site-functions

Emacs Lisp files have been installed to:
  /usr/local/share/emacs/site-lisp/git
==> Summary
🍺  /usr/local/Cellar/git/2.11.0: 1,452 files, 32.4M
Mac-10:~ mac$
```

==========
查看Git的版本
==========
指令：git —version

```
Mac-10:~ mac$ git --version
git version 2.11.0
```

==========
Git initial指令
建立Repository
==========
指令：先cd到要放Repository的路徑，下 git init repo

```
Mac-10:HSU mac$ git init repo
Initialized empty Git repository in /Volumes/HSU/repo/.git/
Mac-10:HSU mac$ cd repo
Mac-10:repo mac$ ls -a
.       ..      .git
```

版本：對於版本來說，需要知道什麼人，做什麼版本，所以什麼人很重要！！！！

==========
Git config指令
==========

```
Mac-10:repo mac$ git config
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-regex]
    --get-all             get all values: key [value-regex]
    --get-regexp          get values for regexp: name-regex [value-regex]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value_rege
x]
    --add                 add a new variable: name value
    --unset               remove a variable: name [value-regex]
    --unset-all           remove all matches: name [value-regex]
    --rename-section      rename section: old-name new-name
    --remove-section      remove a section: name
    -l, --list            list all
    -e, --edit            open an editor
    --get-color           find the color configured: slot [default]
    --get-colorbool       find the color setting: slot [stdout-is-tty]

Type
    --bool                value is "true" or "false"
    --int                 value is decimal number
    --bool-or-int         value is --bool or --int
    --path                value is a path (file or directory name)

Other
    -z, --null            terminate values with NUL byte
    --name-only           show variable names only
    --includes            respect include directives on lookup
    --show-origin         show origin of config (file, standard input, blob, com
mand line)
```

==========
Git 提交使用者資訊
==========

指令：git config —global user.name ‘xxxxxx'
指令：git config —global user.email ‘xxxxxx@xxx.com’

```
Mac-10:repo mac$ git config -l
credential.helper=osxkeychain
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
```

```
Mac-10:repo mac$ git config --global user.name 'hsuyuming'
Mac-10:repo mac$ git config --global user.email 'abego452@gmail.com'
Mac-10:repo mac$ git config -l
credential.helper=osxkeychain
user.name=hsuyuming
user.email=abego452@gmail.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
Mac-10:repo mac$
```

==========

Git 提交變化至Repository

==========

Step1.在含有(.git的資料夾下)建立一個檔案

Step2.新增檔案至Staging Area的狀態

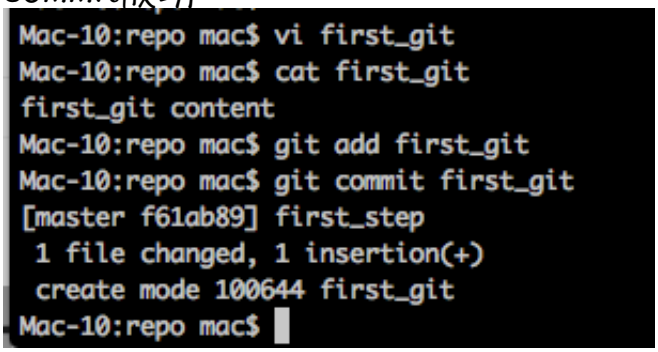
指令：git add fileName (檔案新增或更新不會自動提交&可選擇性的提交檔案)

指令：git commit (-m後面接的是comment(註解) => git commit -m “xxxxxxx”，提交變化 至Repository)

```
Mac-10:repo mac$ vi first_git
Mac-10:repo mac$ cat first_git
first_git content
Mac-10:repo mac$ git add first_git
Mac-10:repo mac$ git commit first_git
```

打註解

commit成功



Step3.更改檔案內容，然後再新增更動後的檔案到Staging Area的狀態，最後提交Commit

指令：git add .　『 . 』的意思是資料夾中所有的檔案

指令：git commit -a 『 -a 』的意思是提交所有檔案變更

```
Mac-10:repo mac$ cat first_git
first_git content
Mac-10:repo mac$ vi first_git
Mac-10:repo mac$ cat first_git
first_git content
new add content
Mac-10:repo mac$ git add .
Mac-10:repo mac$ git commit -a
[master 6f2e211] second version
 1 file changed, 1 insertion(+)
Mac-10:repo mac$
```

```
second version
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#       modified:   first_git
#
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

==========

Git 查詢Log

==========

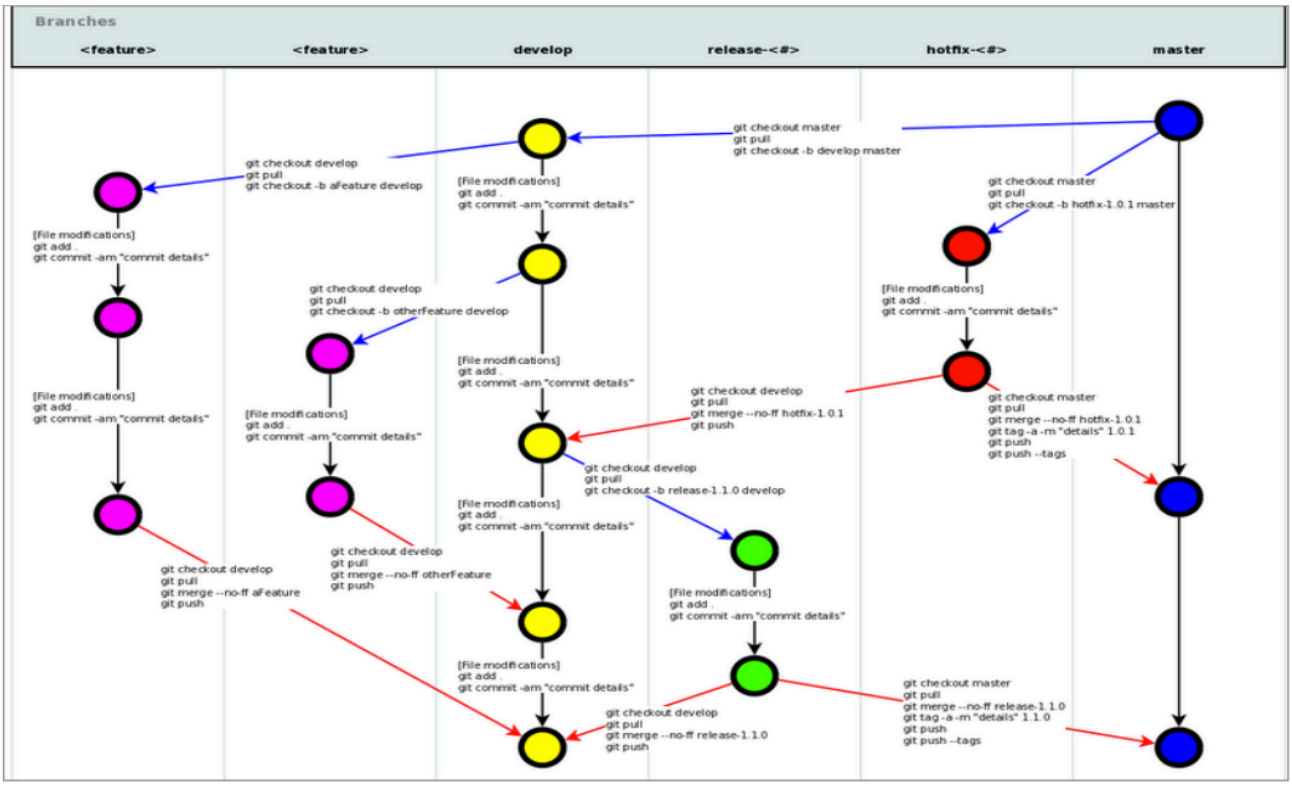
指令：git log

==========
Git 取消新增至staging area的檔案
==========
指令：git reset HEAD file_name

==========
Git 比較版本差異
==========
目的：會做這件事情就是因為可能會有檔案衝突的問題
指令：gif diff xxxxx yyyy

DevOps (開發和營運)
敏捷式開發：適合時常修改的開發專案

==========
Git Branch分支圖
==========



==========

Git Branch指令

==========

指令（建立分支還沒跳過去）：git branch branch_name

指令（從master跳到分支）：git checkout branch_name

```
Mac-10:repo mac$ git branch
* master
Mac-10:repo mac$ ls
0246    0247
Mac-10:repo mac$  git branch new_feature
Mac-10:repo mac$ git branch
* master
  new_feature
Mac-10:repo mac$ git checkout new_feature
Switched to branch 'new_feature'
Mac-10:repo mac$ git branch
  master
* new_feature
Mac-10:repo mac$ ls
0246    0247
Mac-10:repo mac$
```

==========

Git Branch指令2

==========

指令（建立分支並直接跳過去）：git checkout -b branch_name

```
Mac-10:repo mac$ git branch
* master
  new_feature
Mac-10:repo mac$ git checkout -b cool_feature
Switched to a new branch 'cool_feature'
Mac-10:repo mac$ git branch
* cool_feature
  master
  new_feature
Mac-10:repo mac$
```

==========

Git 刪除Branch

==========

前提：先回到主線Master

指令：git branch -D branch_name

```
Mac-10:repo mac$ git branch
* cool_feature
  master
  new_feature
Mac-10:repo mac$ git checkout master
Switched to branch 'master'
Mac-10:repo mac$ git checkout cool_feature
Switched to branch 'cool_feature'
Mac-10:repo mac$ git branch
* cool_feature
  master
  new_feature
Mac-10:repo mac$ git checkout master
Switched to branch 'master'
Mac-10:repo mac$ git branch -D cool_feature
Deleted branch cool_feature (was ebbba91).
Mac-10:repo mac$ git beanch
git: 'beanch' is not a git command. See 'git --help'.

Did you mean this?
        branch
Mac-10:repo mac$ git branch
* master
  new_feature
Mac-10:repo mac$
```

==========

Git 合併分支

==========

重點：不建議改同一個檔案！！！！！

前提：先回到分支，並在分支的狀態下 修改檔案，修改後git add . + git commit -a .確認後，回到主線，進行主線和分支的合併

指令：git merge branch_name

```
Mac-10:repo mac$ git branch
* master
  new_feature
Mac-10:repo mac$ git checkout new_feature
Switched to branch 'new_feature'
new version
Mac-10:repo mac$ git branch
  master
* new_feature
Mac-10:repo mac$ ls
0246    0247
Mac-10:repo mac$ ls 0246
0246
Mac-10:repo mac$ vi 0246
Mac-10:repo mac$ cat 0246
a version1
edit at 0250
edit at 0253
Mac-10:repo mac$ vi 0246
Mac-10:repo mac$ git status
On branch new_feature
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working direct

        modified:   0246

no changes added to commit (use "git add" and/or "git commit -a")
Mac-10:repo mac$ git add .
Mac-10:repo mac$ git commit -a
error: There was a problem with the editor 'vi'.
Please supply the message using either -m or -F option.
```

```
Mac-10:repo mac$ git status
On branch new_feature
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   0246

Mac-10:repo mac$ git checkout master
M       0246
Switched to branch 'master'
Mac-10:repo mac$ git merge new_feature
Already up-to-date.
Mac-10:repo mac$ ls
0246    0247
Mac-10:repo mac$ cat 0246
a version1
edit at 0250
edit at 0253
edit ar 2017-02-21 1416
Mac-10:repo mac$
```

==========

Git 製造衝突

==========

發生情形：在branch的狀態下，先改檔案並且commit後，但先不合併，在切回主線，更改主線的內容並且commit後，在使用merge的時候就會出現版本衝突的情形發生

```
Mac-10:repo mac$ git branch
* master
  new_feature
Mac-10:repo mac$ ls
0246    0247
Mac-10:repo mac$ cat 0246
a version1
edit at 0250
edit at 0253
edit ar 2017-02-21 1416
Mac-10:repo mac$ vi 0246
Mac-10:repo mac$ git checkout new_feature
M       0246
a version1
Switched to branch 'new_feature'
Mac-10:repo mac$ git branch
  master
* new_feature
Mac-10:repo mac$ ls
0246    0247
Mac-10:repo mac$ cat 0246
a version1
edit
edit at 0250
edit at 0253
edit ar 2017-02-21 1416
edit at 2017-02-21 1425
Mac-10:repo mac$ vu 0236
-bash: vu: command not found
Mac-10:repo mac$ vu 0246
-bash: vu: command not found
Mac-10:repo mac$ vi 0246
Mac-10:repo mac$ git add .
Mac-10:repo mac$ git commit -a
```

```
Mac-10:repo mac$ git commit -a
[new_feature 925b15a] edit at 0227
 1 file changed, 3 insertions(+)
Mac-10:repo mac$ git checkout master
Switched to branch 'master'
Mac-10:repo mac$ git branch
* master
  new_feature
Mac-10:repo mac$ cat 0246
a version1
edit at 0250
edit at 0253
Mac-10:repo mac$ vi 0246
Mac-10:repo mac$ ls
0246    0247
Mac-10:repo mac$ vi 0246
Mac-10:repo mac$ git merge new_feature
Updating ebbba91..925b15a
error: Your local changes to the following files would be overwritten by merge:
        0246
Please commit your changes or stash them before you merge.
Aborting
Mac-10:repo mac$ git add .
Mac-10:repo mac$ git commit -a
[master 77bedae] edit
 1 file changed, 1 insertion(+)
Mac-10:repo mac$ git merge new_feature
Auto-merging 0246
CONFLICT (content): Merge conflict in 0246
Automatic merge failed; fix conflicts and then commit the result.
Mac-10:repo mac$
```

==========

創建Github

==========

Step1：產生自己的ssh公鑰

- 指令：ssh-keygen -t rsa -b 4096 -C “xxx@xxx.com”（之後一直按Enter到完成）

Step2：加入私鑰

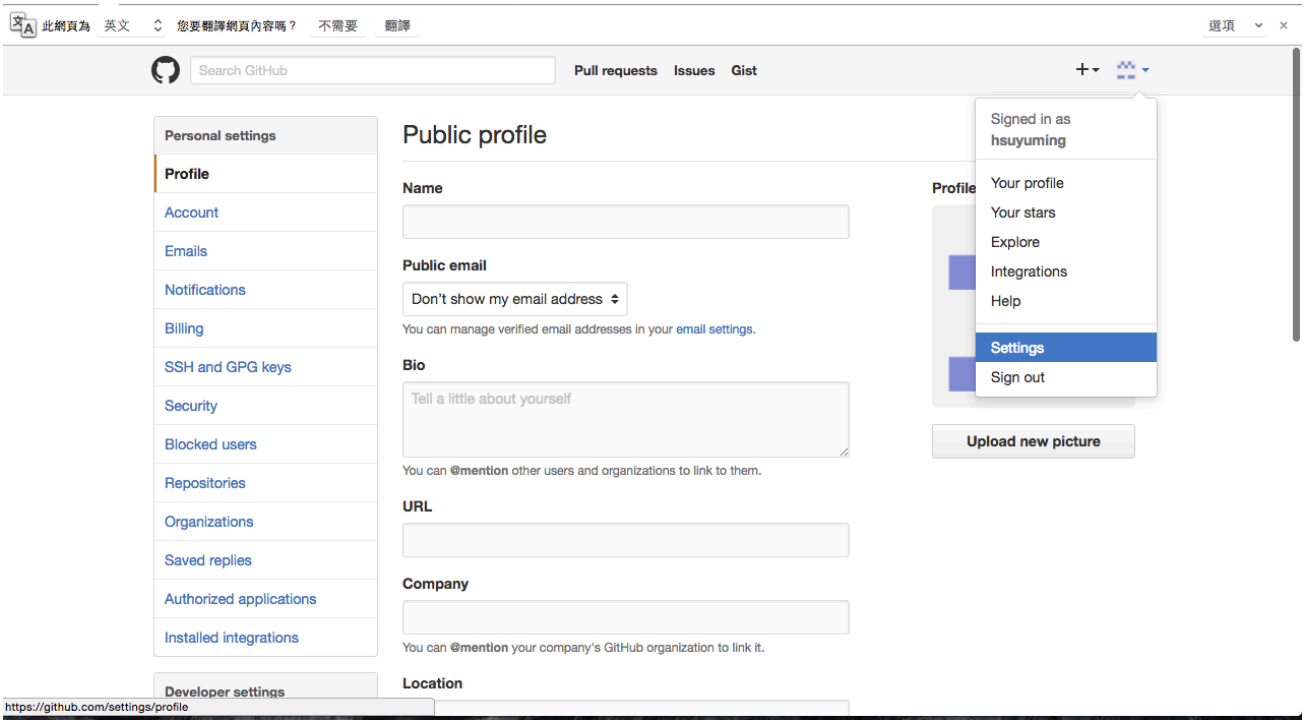
- 指令：ssh-agent -s
- 指令：eval $(ssh-agent -s)
- 指令：ssh-add ~/.ssh/id_rsa

Step3：複製公鑰內容

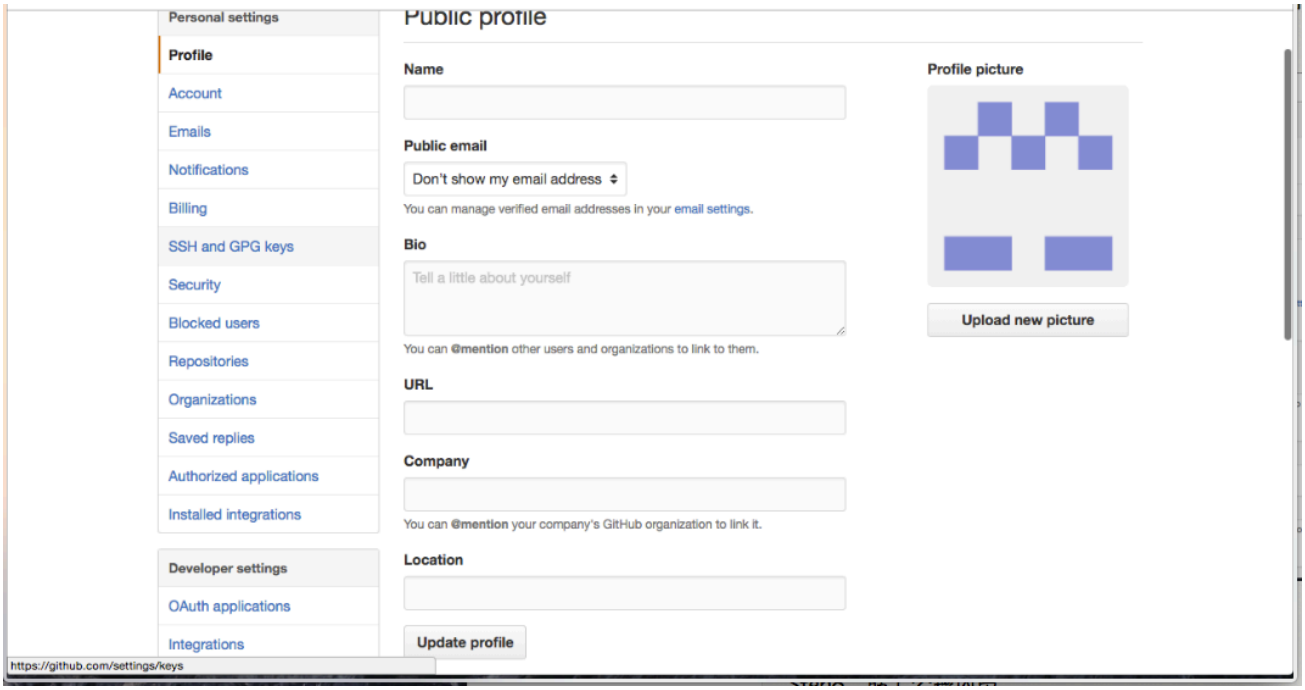
- 指令：cd /Users/xxxxx/.ssh/id_rsa.pub（複製內文）

Step4：註冊Github帳號

Step5：登入Github點選Setting



Step6：點選SSH keys



Step7：按下Add SSH key

Step8：貼上公鑰內容

==========

測試Github連線

==========

指令：ssh -T git@github.com

```
Mac-10:repo mac$ ssh -T git@github.com
The authenticity of host 'github.com (192.30.253.113)' can't be established.
RSA key fingerprint is [redacted]
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.253.113' (RSA) to the list of know
n hosts.
Hi hsuyuming! You've successfully authenticated, but GitHub does not provide she
ll access.
Mac-10:repo mac$
```

==========

上傳repository至Github

==========

Step1：先在自己github的帳號的repository創建一個新的repository

Step2：位於有.git的資料夾層，輸入指令，將檔案上傳

・指令：git remote add origin git@github.com:xxxxx/xxx.git

・指令：git push -u origin master

```
Mac-10:repo mac$ git remote add origin git@github.com:hsuyuming/test.git
Mac-10:repo mac$ git push -u origin master
Warning: Permanently added the RSA host key for IP address '192.30.253.112' to t
he list of known hosts.
Counting objects: 15, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (15/15), 1.20 KiB | 0 bytes/s, done.
Total 15 (delta 0), reused 0 (delta 0)
To github.com:hsuyuming/test.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
Mac-10:repo mac$ git branch
* master
  new_feature
```

==========

下載repository至Github

==========

Step1：先在本機創立一個目錄

Step2：cd進入該目錄

Step3：git clone https://github.com/xxxxxx/xxxx.git

==========

Github Pull 指令說明（從遠端更新）

==========

說明：實際作用是先 git fetch 遠端的 branch，然後與本地端的 branch 做 merge，產生一個 merge commit 節點

補充：所謂的"遠端"預設叫做*origin*，當你有多個不同遠端伺服器時，就會取不同名子了。

指令：`git pull origin master`

==========

Github Push - 將 Commit 送出去

==========

說明：實際的作用是將本地端的 master branch 與遠端的 master branch 作 fast-forward 合併。如果出現 [rejected] 錯誤的話，表示你必須先作 pull。

預設不會 push tags 資訊： git push –tags

指令：`git push 或 git push origin master`

==========

刪除origin

==========

指令：git remote remove origin

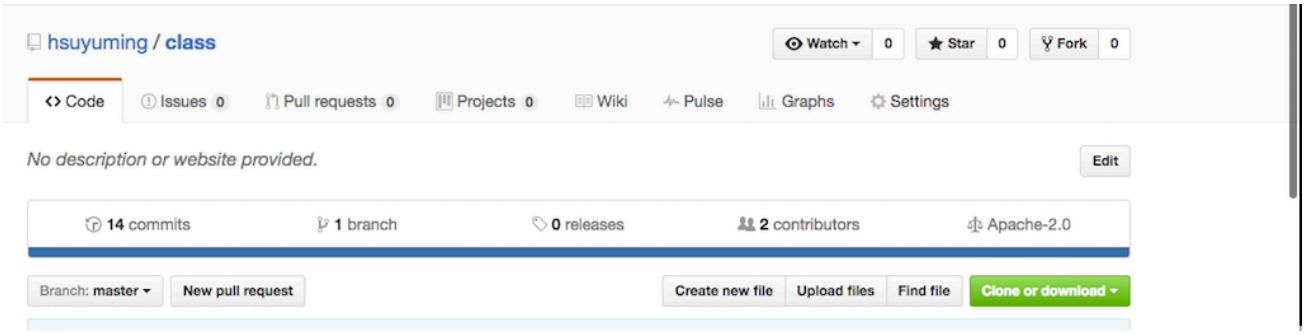
==========

刪除repository

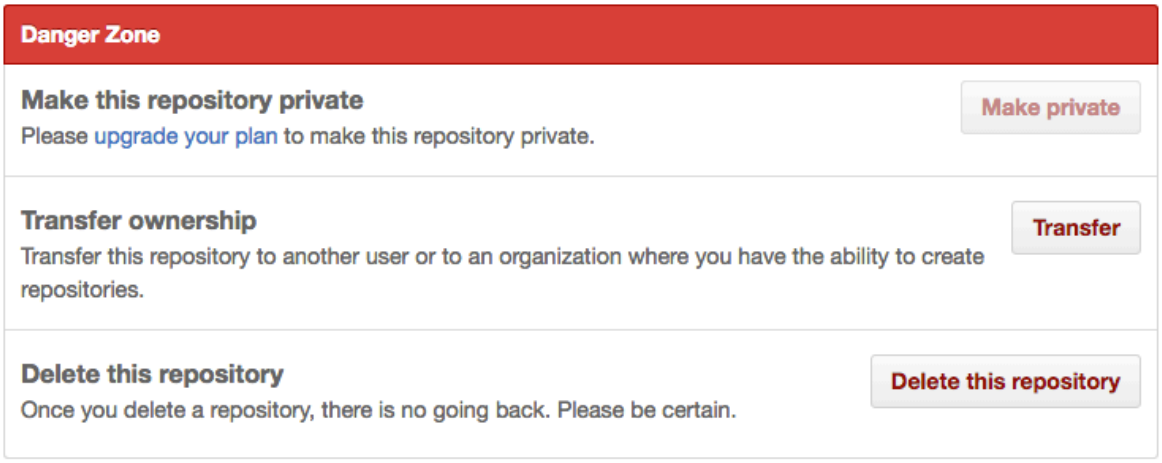
==========

Step1：進入要刪除的repository

Step2：點選setting



Step3：往下滑看到Danger Zone的區塊，按下Delete this repository



Step4：輸入要刪除的repository，github確定你刪除的是吻合的驗證動作