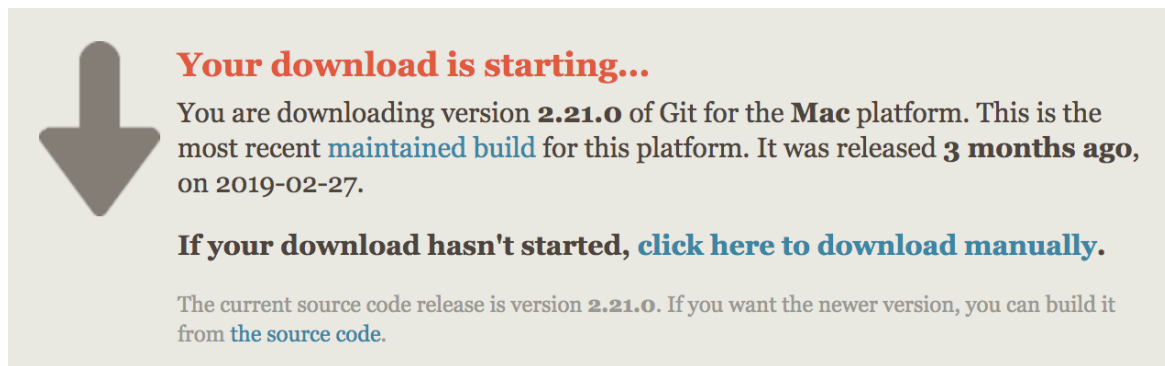


安裝在 Mac OSX 作業系統

在 Mac 上安裝 Git 有幾種做法：

1. 從官方網站下載 Mac 專屬版本的 Git



Git Official Website

網站：<https://git-scm.com/download/mac>

只要下載檔案、點開、執行，基本上這樣就可順利完成了。

2. 使用 Homebrew 軟體來安裝 Git



Homebrew

雖然 Mac 作業系統出廠的時候已經有很多軟體或工具了，但對開發者來說很多工具還是得自己想辦法下載甚至得從原始程式碼進行編譯、安裝。而 Homebrew 這個工具就是用來補足這個缺口，它有點像在 Linux 的 apt-get 之類的安裝工具，通常只要一行指令就可完成下載、編譯、安裝。如果您本身也是開發者，建議可考慮使用 Homebrew 來安裝軟體。

網址：https://brew.sh/index_zh-tw.html

安裝方式很簡單，就是只要照著它網站上的那行複製起來：

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

在終端機視窗貼上並執行就可以了。

Homebrew 安裝完成後，請一樣在終端機下執行這行指令：

```
$ brew install git
```

意思就是請 Homebrew 幫你安裝 Git 這個軟體，按下 Enter 鍵之後，剩下的就交給 Homebrew 去煩惱了。

安裝在 Windows 作業系統

要在 Windows 作業系統上安裝 Git，請到官方網站下載合適的版本：



Your download is starting...

You are downloading the latest (**2.21.0**) **32-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **3 months ago**, on 2019-02-26.

If your download hasn't started, [click here to download manually](#).

Other Git for Windows downloads

Git for Windows Setup

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Git for Windows Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

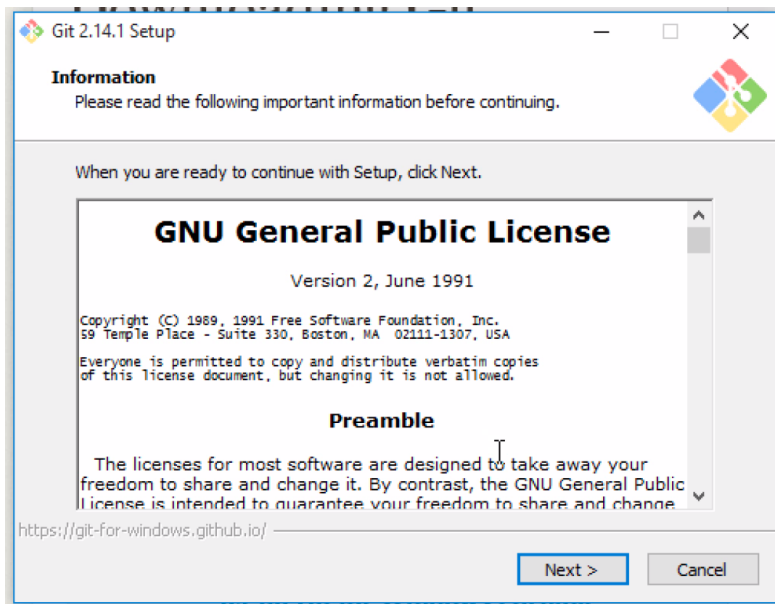
64-bit Git for Windows Portable.

The current source code release is version **2.21.0**. If you want the newer version, you can build it from [the source code](#).

Git Official Website

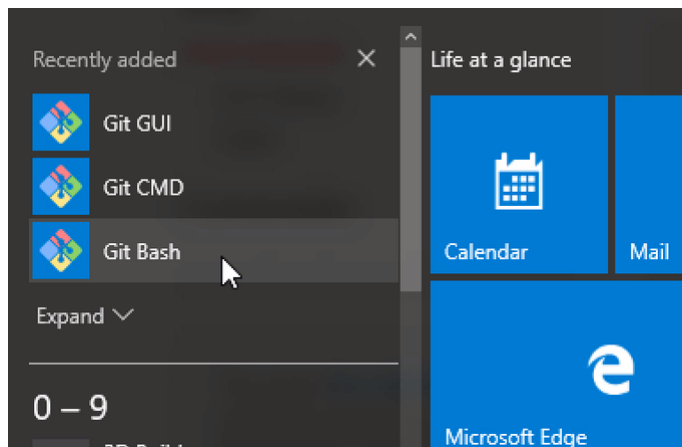
網址：<https://git-scm.com/download/win>

選擇好版本後，安裝也是相當無腦的可以一路按下一步按到底：



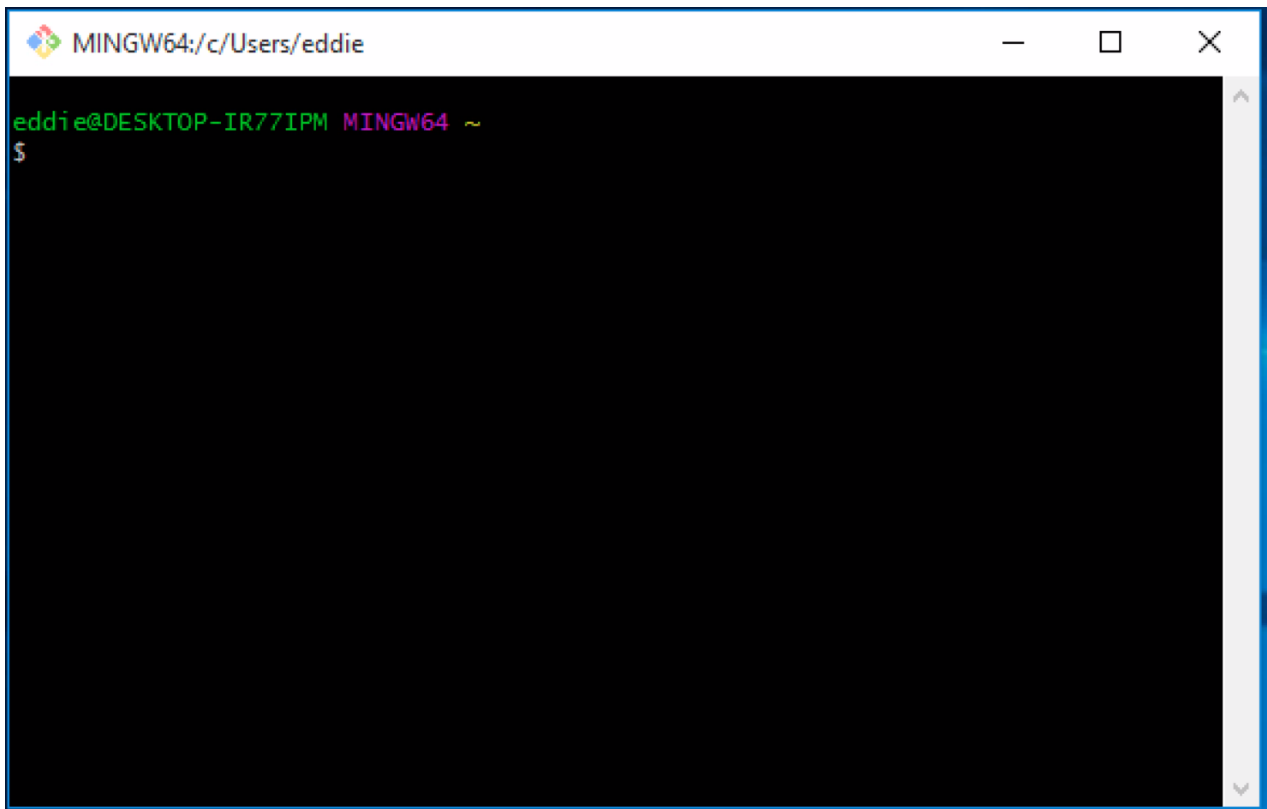
Install Git in Windows

安裝完成之後，請找到「Git Bash」這個應用程式：



Install Git in Windows

進入 Git Bash 後，雖然一樣都是黑黑的視窗，但這個跟 Windows 內建的「命令提示字元」不太一樣，它本身模擬了一個在 Linux 的世界還滿有知名度的軟體（其實不能算是一般的應用軟體）叫做 Bash：



Install Git in Windows

到這裡，你可以在那個黑黑的視窗試試輸入指令，驗證一下 Git 是不是有安裝起來，以及版本資訊對不對：

```
$ which git
/mingw64/bin/git

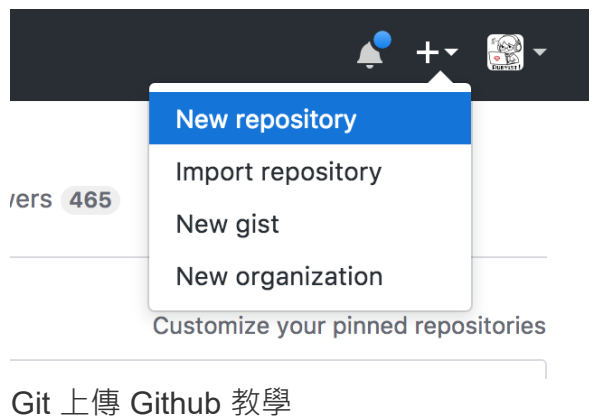
$ git --version
git version 2.21.0.windows.1
```

如果看到類似的訊息，就是安裝成功了。

Git教學：如何 Push 上傳到 GitHub？

在 GitHub 上開新專案

要上傳檔案到 GitHub，需要先在上面開一個新的專案。請先在 GitHub 網站的右上角點選「+」號，並選擇「New repository」：



接著填寫專案名稱：

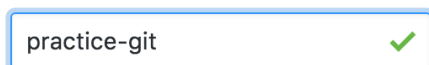
Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name



Great repository names are short and memorable. Need inspiration? How about **supreme-broccoli**.

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▼

Add a license: **None** ▼



Create repository

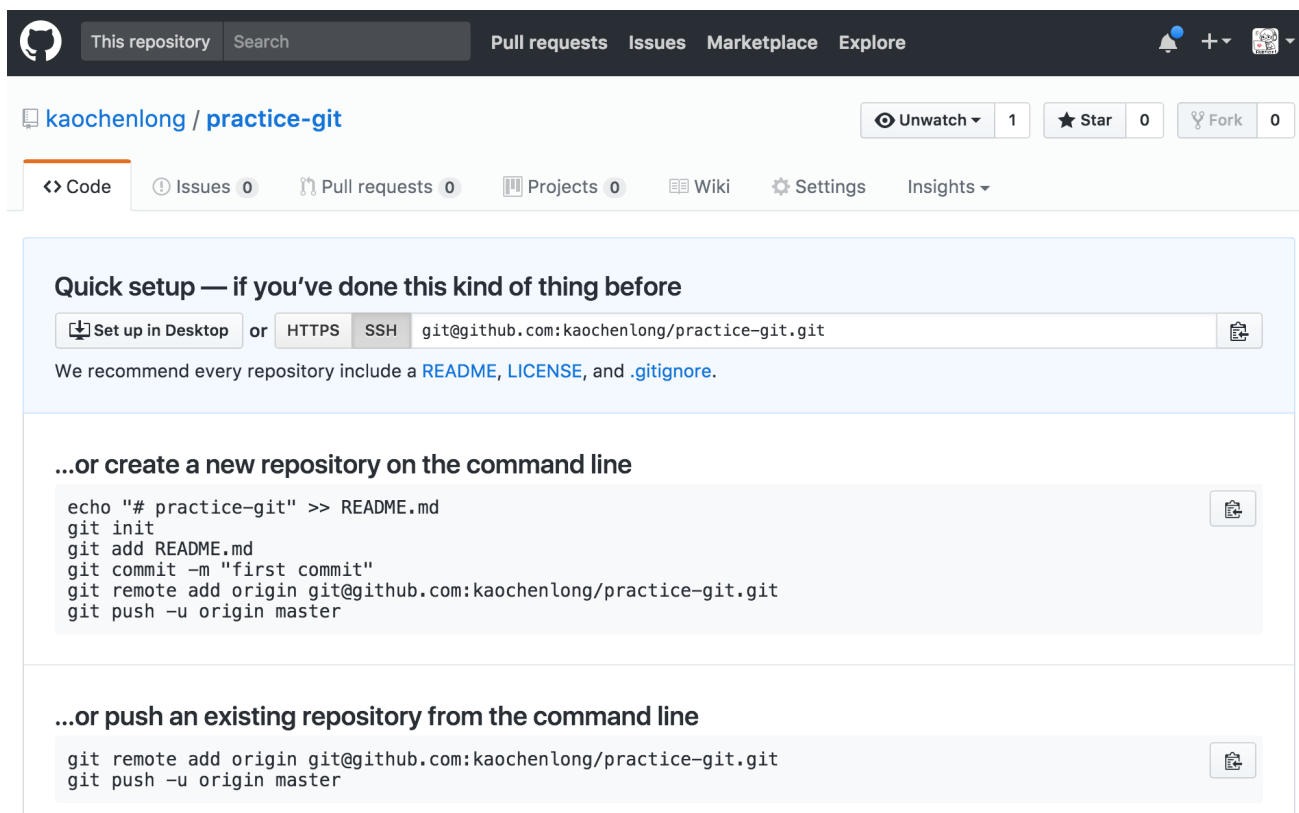
Git 上傳 Github 教學-填寫專案名稱

1. Repository name 可填寫任意名稱，只要不重複即可。
2. 存取權限選擇 Public 可免費使用，選擇 Private 則需收每個月 7 元美金的月費。

補充：在 GitHub 被 Microsoft 併購後，現在即使是免費帳號也都可以開立 Private 專案喔。

按下「Create repository」即可新增一新的 Repository。

接下來，會看到的這個引導畫面：



Git 上傳 Github 教學-Github 說明

說明：

1. 如果是全新開始，請依「create a new repository on the command line」的內容指示進行；如果是要上傳現存專案，則依照「push an existing repository from the command line」選項進行。
2. 畫面中間上面，有兩個按鈕可以切換，分別是「HTTPS」以及「SSH」。至於要選擇哪一個可看個人喜好，但選擇 SSH 的話需要設定 SSH Key，SSH Key 的設定可參閱「[GitHub 是什麼？](#)」章節介紹。因為我已經有設定好 SSH Key，所以這裡我選擇「SSH」。

如果仔細觀察，便會發現選擇全新開始跟上傳現有專案這兩個選項的最後兩個步驟是一樣的。

假設我們現在什麼都沒有，要重新開始一個專案，找一個空的目錄，就照著它的說明進行。首先，先建立一個 README.md 檔案：

```
$ echo "# Practicing Git" > README.md
```

這個 README.md 是 GitHub 專案的預設說明頁面，附檔名 .md 表示它是一個 Markdown 格式，Markdown 語法可以很輕鬆的把純文字格式轉換成 HTML 的網頁格式，如果這是你第一次接觸這個語法，非常推薦花一點時間學習它。

接下來這段，就是我們熟悉的 Git 了：

```
$ git init
Initialized empty Git repository in /private/tmp/practice-git/.git/

$ git add README.md

$ git commit -m "first commit"
[master (root-commit) adc1a5a] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

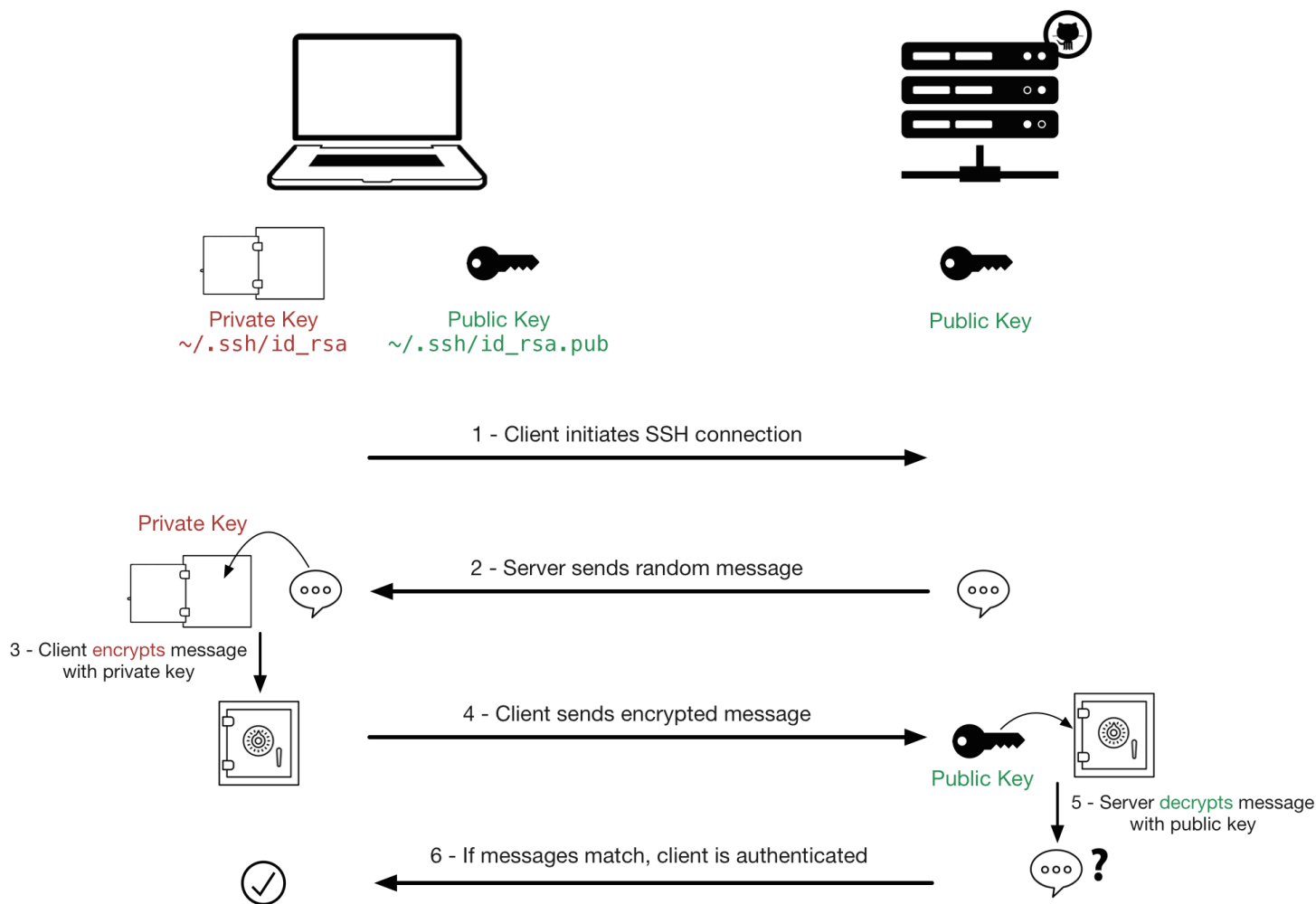
使用 SSH 金鑰與 GitHub 連線

前言

在 GitHub 中有兩種存取資料的連線方式一種是 `https` 另一種是 `ssh`，兩者的差別在於 `https` 在每次 `git push` 前都要輸入密碼，而 `ssh` 相對的就不用輸入密碼即可立即上傳，若不想每次都輸入密碼可以參考 `ssh` 連線方式。

SSH 連線運作方式

SSH(安全殼層協定，Secure Shell)，SSH 公開金鑰預設保存在帳戶的主目錄下的 `~/.ssh` 目錄，包含一個公鑰與私鑰，運作方式可以看下面這張圖(參考於)，這張圖的 Client 為我們使用者端而 Server 為 GitHub 伺服器，當本機端用戶 `push` 資料到遠端時，就會先發送 SSH 連線請求，然後 GitHub 就會隨機傳送一串訊息給本機端用戶電腦使用私密加密，加密後 GitHub 再利用公鑰解密在驗證解密後的訊息是否跟原先的吻合，若是的話則完成 SSH 驗證。



圖片來源: <http://sebastien.saunier.me/blog/2015/05/10/github-public-key-authentication.html>

- Client 端發送 SSH 連線請求
- Server 傳送隨機訊息
- Client 端使用私鑰加密，並回傳加密訊息回 Server 端
- Server 端使用公鑰解密，若訊息吻合代表用戶驗證成功

教學

1. 產生金鑰 {#1-產生金鑰 }

用 `ssh-keygen` 來建立。該程式在 Linux/Mac 系統上由 SSH 包提供，而在 Windows 上則包含在 MSysGit 包裡，輸入以下指令，來產生新的 SSH Key。

```
$ ssh-keygen -t rsa -b 4096 -C "你的信箱"
```

它先要求你確認保存公開金鑰的位置 (`.ssh/id_rsa`)，然後它會讓你重複一個密碼兩次。


```
1. caiyilin@caiyilins-MacBook-Pro: ~ (zsh)
Last login: Thu Mar 22 09:29:20 on ttys003
~ ssh-keygen -t rsa -b 4096 -C 'andy6804tw@yahoo.com.tw' ✓ 09:30:14
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/caiyilin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/caiyilin/.ssh/id_rsa.
Your public key has been saved in /Users/caiyilin/.ssh/id_rsa.pub.
The key fingerprint is:
The key's randomart image is:
```

2. 設定金鑰代理 {#2-設定金鑰代理 }

如果沒設定代理每次上傳時還是會要求輸入密碼。


```
## 啟用 SSH 代理伺服器
$ eval "$(ssh-agent -s)"
## 將私鑰加入到 SSH 代理上
$ ssh-add -K ~/.ssh/id_rsa
```

```
~ eval "$(ssh-agent -s)" ✓ 09:32:51
Agent pid 41301
~ ssh-add -K ~/.ssh/id_rsa ✓ 09:36:38
Enter passphrase for /Users/caiyilin/.ssh/id_rsa:
Identity added: /Users/caiyilin/.ssh/id_rsa (/Users/caiyilin/.ssh/id_rsa)
```


3. 上傳公鑰至 GitHub {#3-上傳公鑰至-github }

開啟終端機進入 `.ssh` 資料夾，此資料夾為隱藏的在 Mac 內的個人目錄下可以找到，進入後可以看到 `id_rsa`、`id_rsa.pub` 兩個檔案分別為私鑰和公鑰，打開公鑰(`id_rsa.pub`)把全部複製起來，開啟瀏覽器進入 GitHub 頁面，Github > Settings > SSH and GPG keys 的設定頁面，選擇 New SSH Key，填入 Title 與 Key 後即可送出。

```
$ cd ../ssh
$ cat id_rsa.pub
```

 Search GitHub

Pull requestsIssuesMarketplaceExplore



Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

Blocked users

Repositories


Organizations

Saved replies

Applications

Developer settings

Organization settings

 1010code

SSH keys / Add new

Title

SSH Key

Key

ssh-rsa

Add SSH key

Signed in as andy6804tw

Your profile

Your stars

Your gists

Help

Settings


Sign out

完成後就可以測試上傳囉！此外後台也會看到最後一次連線存取的時間


SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 SSH

SSH Key

Fingerprint: 

Added on 22 Mar 2018

Last used within the last week — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

到這裡，都還是 Git 的基本招式。如果忘記怎麼操作可參閱 [把檔案交給 Git 控管](#) 章節。

在這之前，我們所有的操作都是在自己電腦上，接下來就是要準備把東西推上遠端的 Git 伺服器了。首先，需要設定一個端節的節點，例如：

```
$ git remote add origin [email protected]:kaochenlong/practice-git.git
```

說明：

1. `git remote` 指令，顧名思義，主要是跟遠端有關的操作。
2. `add` 指令是指要加入一個遠端的節點。
3. 在這裡的 `origin` 是一個「代名詞」，指的是後面那串 GitHub 伺服器的位置。

在慣例上，遠端的節點預設會使用 `origin` 這個名字。如果是從 Server 上 clone 下來的話，它的預設的遠端節點就會叫 `origin`。（關於 Clone 的使用，請參閱[從伺服器上取得 Repository](#)」章節說明）

不過別擔心，這只是個慣例，不用這名字或是之後想要改也都可以，如果想改叫七龍珠 `dragonball`：

```
$ git remote add dragonball [email protected]:kaochenlong/practice-git.git
```

總之，它就是只是指向某個位置的代名詞罷了。設定好遠端節點後，接下來，就是要把東西推上去了：

```
$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 228 bytes | 228.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:kaochenlong/practice-git.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

這個簡單的 Push 指令其實做了幾件事：

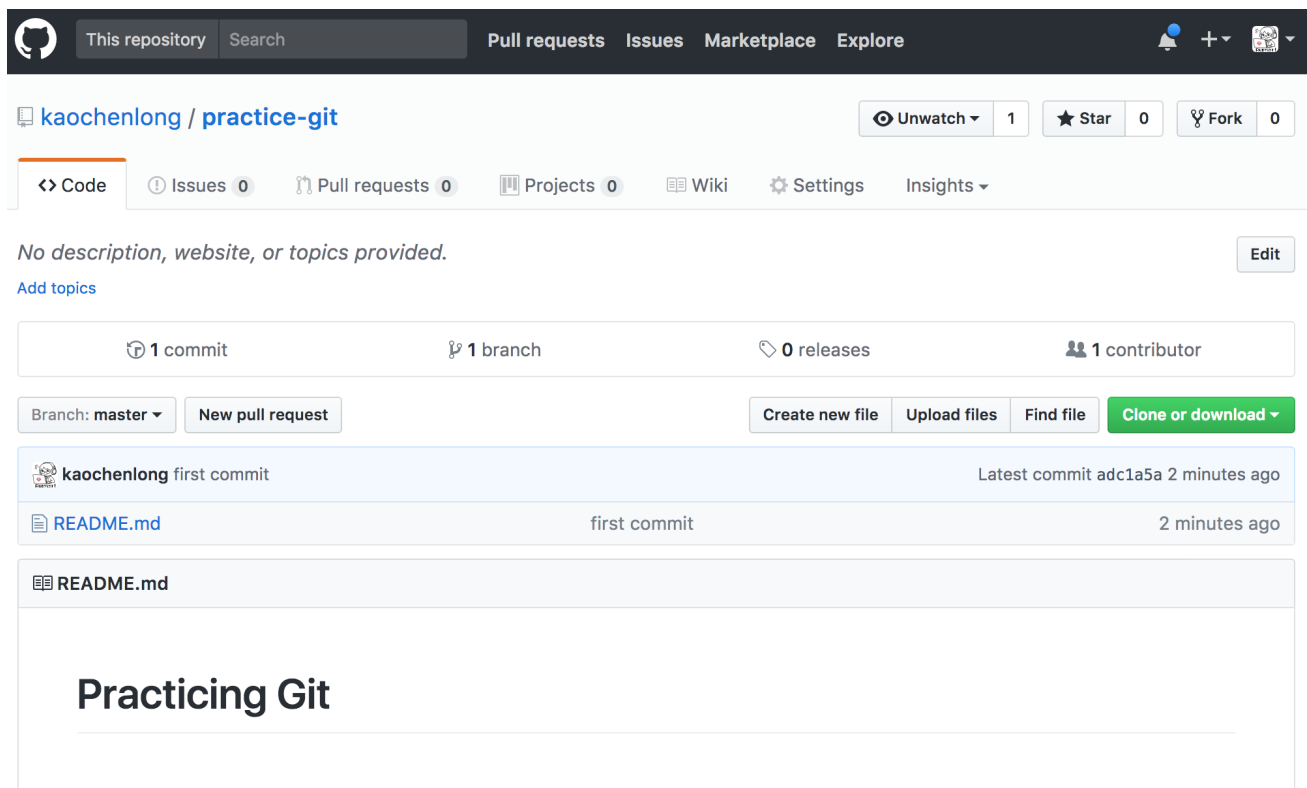
1. 把 `master` 這個分支的內容，推向 `origin` 這個位置。
2. 在 `origin` 那個遠端 Server 上，如果 `master` 不存在，就建立一個叫做 `master` 的同名分支。
3. 但如果本來 Server 上就存在 `master` 分支，便會移動 Server 上 `master` 分支的位置，使它指到目前最新的進度上。
4. 設定 `upstream`，就是那個 `-u` 參數做的好事，這個稍候說明。

如果你能理解上面這個指令的意思，你就可以再做一些變化。例如你的遠端節點叫做 `dragonball`，而且你想把 `cat` 分支推上去：

```
$ git push dragonball cat
```

這樣就會把 `cat` 分支推上 `dragonball` 這個遠端節點所代表的位置，並且在上面建立一個名為 `cat` 同名分支（或更新進度）。

回到 GitHub 網站，重新整理一下頁面，剛才那個引導指令的畫面，現在應該會變這樣：

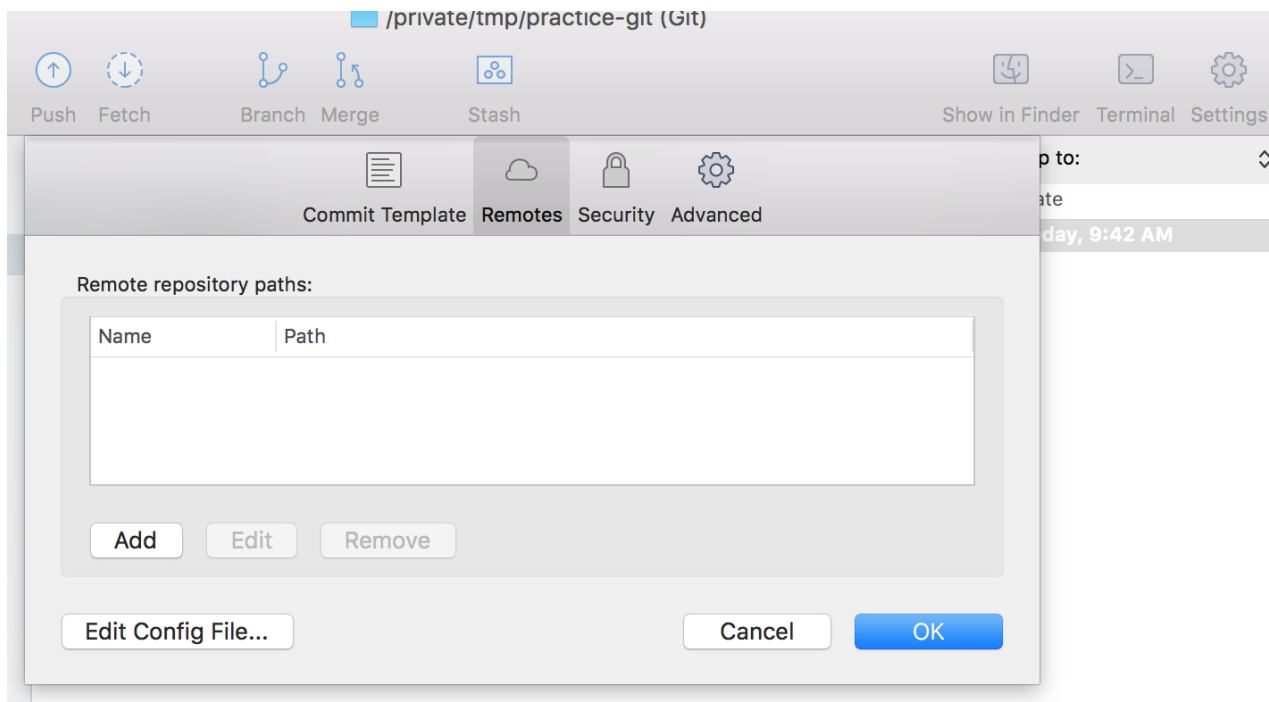


Git 上傳 Github 教學-Github 畫面

恭喜你，看到這畫面表示你已經順利把你本地專案的東西推到這個遠端的專案裡了。

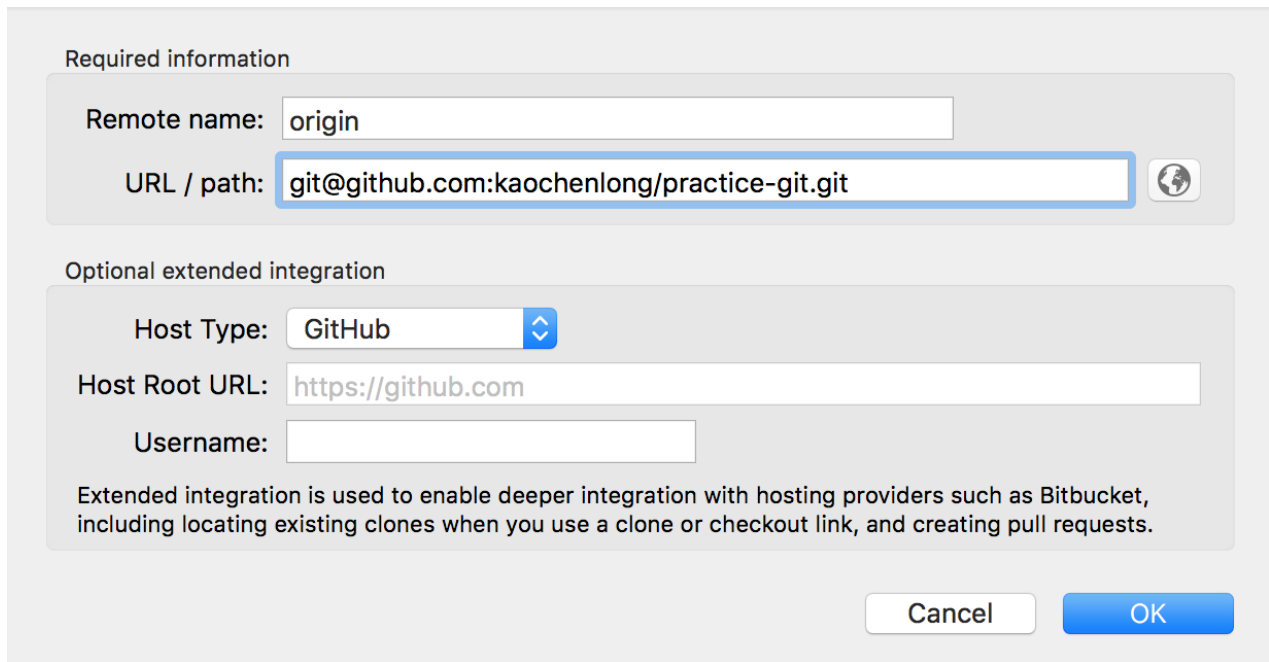
Git 教學：使用 SourceTree

如果是使用 SourceTree，請按下右上角的「Settings」按鈕，跳出對話框後，選擇「Remote」頁籤：



Git 上傳 Github 教學-SourceTree教學畫面

點選「Add」按鈕：



Required information

Remote name:

URL / path:

Optional extended integration

Host Type:

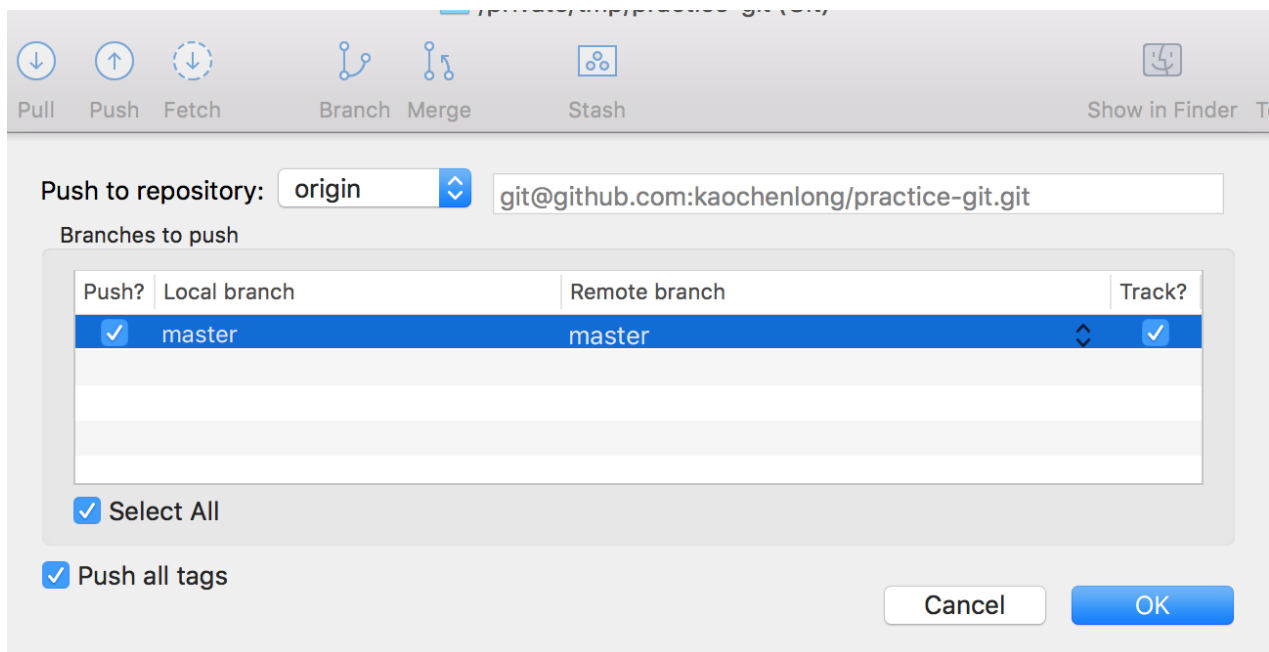
Host Root URL:

Username:

Extended integration is used to enable deeper integration with hosting providers such as Bitbucket, including locating existing clones when you use a clone or checkout link, and creating pull requests.

Git 上傳 Github 教學-SourceTree教學畫面

填寫「Remote name」以及「URL」欄位後，按下 OK 按鈕後，遠端節點即設定完成。接下來要把東西往上推，請點選上面工具列的「Push」按鈕：



Push to repository:

Branches to push

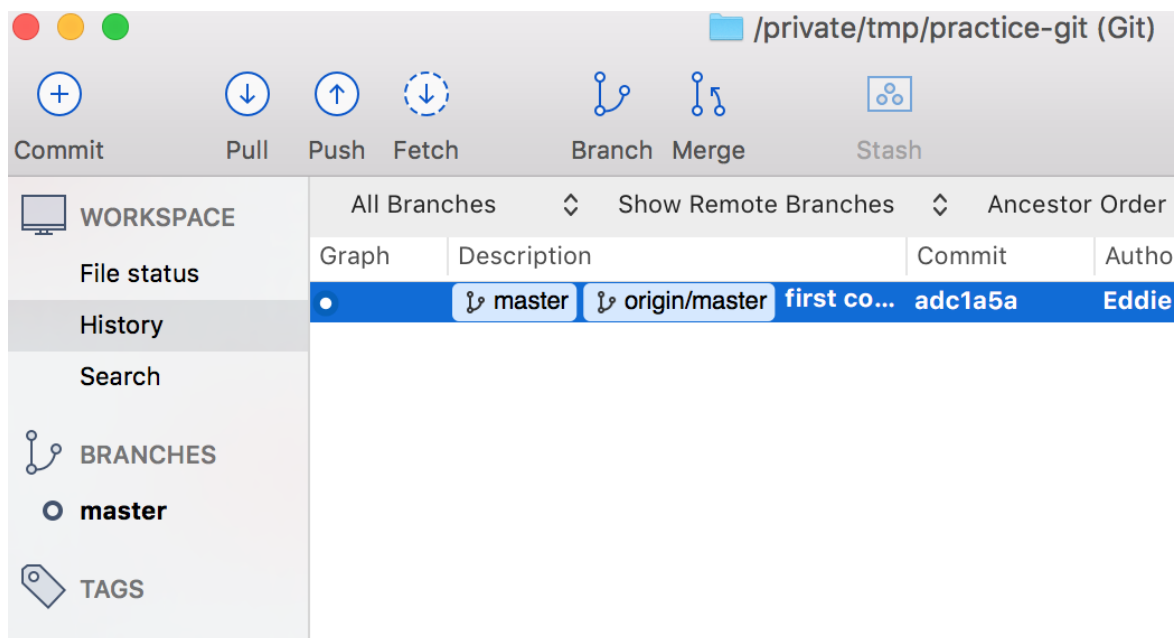
Push?	Local branch	Remote branch	Track?
<input checked="" type="checkbox"/>	master	master	<input checked="" type="checkbox"/>
<input type="checkbox"/>			<input type="checkbox"/>
<input type="checkbox"/>			<input type="checkbox"/>
<input type="checkbox"/>			<input type="checkbox"/>

☒ Select All

☒ Push all tags

Git 上傳 Github 教學-SourceTree教學畫面

勾選你想推的分支，按下 OK 鈕之後就會開始把指定的分支往指定的遠端節點推。成功之後再回來看，現在在 master 分支旁邊就多了一個 origin/master 的分支了：



Git 上傳 Github 教學-SourceTree教學畫面

【冷知識】設定 upstream 是什麼意思

在前面進行 Push 的時候，有加入了一個 `-u` 參數，表示要設定 upstream 的意思。嗯...問題是，什麼是「upstream」？

upstream，中文翻譯成「上游」。不要被名詞嚇到了，這 upstream 的概念其實就只是另一個分支的名字而已。在 Git 裡，每個分支可以設定一個「上游」（但每個分支最多只能設定一個 upstream），它會指向並追蹤（track）某個分支。通常 upstream 會是遠端 Server 上的某個分支，但其實要設定在本地端的其它分支也可以。

如果有設定，當下次執行 `git push` 指令的時候，它就會用來當預設值。舉例來說：

```
$ git push -u origin master
```

就會把 `origin/master` 設定為本地 `master` 分支的 upstream，當下回執行 `git push` 指令而不加任何參數的時候，它就會猜你是要推往 `origin` 這個遠端節點，並且把 `master` 這個分支推上去。

反之，沒設定 upstream 的話，就必須在每次 Push 的時候都跟 Git 講清楚、說明白：

```
$ git push origin master
```

否則，光就執行 `git push` 指令而不帶其它參數的話，Git 會跟你埋怨說不知道該 Push 什麼分支以及要 Push 去哪裡：

```
$ git push
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master
```

【冷知識】如果不想要同名的分支名字該如何更改？

前面提到 Push 的指令是這樣：

```
$ git push origin master
```

其實上面這個指令跟下面這個指令是一樣的效果：

```
$ git push origin master:master
```

意思是「把本地的 `master` 分支推上去後，在 `Server` 上更新 `master` 分支的進度，或是如果不存在該分支的話，就建立一個 `master` 分支」。但如果你想推上去之後不要叫這個名字的話，可以把後面的那個名字改掉：

```
$ git push origin master:cat
```

這樣當把本地端的 `master` 分支推上去之後，就不會在線上建立 `master` 分支，而是建立（或更新進度）一個叫做 `cat` 的分支了。

資料來源

- <https://andy6804tw.github.io/2018/03/22/github-ssh/>
- <https://gitbook.tw/chapters/github/push-to-github.html>