



# GIT/GITHUB

## 入門教學

范凱禎

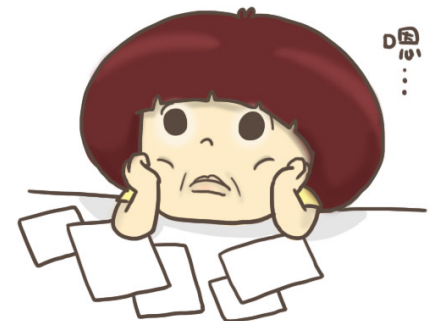
資策會

2022.04.21



# 寫程式做網站、專案碰到的問題

- 反反覆覆...來來回回... 改回來又改回去、到底動了哪些程式碼已經分不清楚了...
  - 此時工程師小智靈機一動——不然我把每次的修改都存成一個檔案呢？一再再的修改，從此出現了各種重新命名檔案名的腦力激盪：
    - Project\_1、Project\_1\_Revised、Project\_1\_Small Revised、Proejct\_1\_Final、Project\_1\_New Final、Project\_1\_Final2 ...
- (Final後還有Final、不斷增生的Final...)





# 什麼是版本控制系統

- 版本控制系統 (Version Control System, VCS) 是一套能記錄所有修改版本、讓使用者能隨時取回特定版本的系統。
- 大家都有玩過RPG嗎？





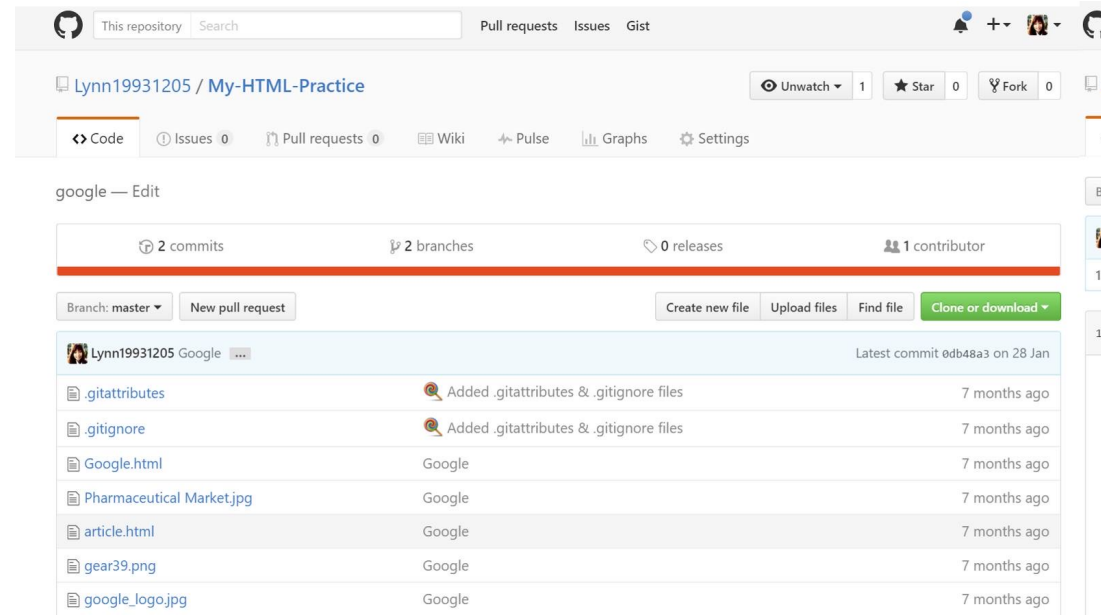
# 最受歡迎的版本控制系統：GIT

- Git是由Linus Torvalds所發明。Linus同時也是Linux作業系統之父。Linus至今仍在進行Linux作業系統的維護工作。
- Linux作業系統開發是一個龐大的計劃，包含了1500萬行左右的程式碼、每天有3500多行新程式碼被加進去，而每一次釋出新版本的Linux、至少有1000位以上的開發者。
- 此時勢必要有協同開發工具，讓開發以最快且最無痛的方式進行，Git於是應運而生。



# GIT有三大優點

1. Git是分散式版本控制系統
2. Git的設計架構讓開發者能碰到底層、進行更彈性的開發
3. GitHub是全世界最受歡迎的程式碼管理/分享網站





# 受歡迎的資源庫

- 機器學習
  - <https://github.com/instillai/deep-learning-roadmap>
- Java 、 python
  - <https://github.com/jobbole>
- 前台開發
  - <https://www.gushiciku.cn/pl/a4UW/zh-hk>
- 中文資源
  - <https://github.com/GrowingGit/GitHub-Chinese-Top-Charts>



# 安裝GIT

- 請先下載並安裝Git(<https://git-scm.com>)，並建立一個GitHub帳號。安裝完Git後請打開命令介面，輸入git --version確認是否安裝成功。



```
> Windows PowerShell  
PS C:\project_1> git --version  
git version 2.35.1.windows.2  
PS C:\project_1> █
```





# 下載與安裝

## Download for Windows

[Click here to download](#) the latest (2.35.1) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **2 months ago**, on 2022-02-01.

### Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)





# 告訴Git使用者資訊

- 打開Git介面(開始→GIT CMD)，輸入git config --global user.name和git config --global user.email的使用者名稱和Email (就是你的GitHub使用者名稱和信箱)。
- 可以使用 git config --list 這個指令來看你的Git 設定內容

```
$ git config --global user.name " Kenfan"
```

```
$ git config --global user.email "kenfan@iii.org.tw"
```

```
$ git config --list
```



# 開始使用Git

- Git中的檔案有三種狀態：已修改、已暫存和已提交，三者分別位於Working Directory (本地端)、Staging Area (暫存端)和Repository (版本庫or專案庫)。





# 建立專案，進駐Git

- 在桌面創建一個叫做gitProject1的資料夾(專案)，並在裡面創建一個名稱為hello.txt的(專案)文件。
- 建立完我們第一個專案後，打上git init指令、讓Git遊戲存檔專家進駐吧！



```
PS C:\gitProject1> git init
Initialized empty Git repository in C:/gitProject1/.git/
```



# 查看目前的Git狀態：git status

- 輸入git status指令，意思是查看目前的Git狀態

```
PS C:\gitProject1> git status
On branch master

No commits yet

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

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

1. 由於我們還沒有開起任何branch(分支)，**Git**設定我們目前的主幹道就是**master**。
2. 看到**Git**大神告訴你：「hello.txt」這個檔案還沒有被追蹤嗎？別擔心，只有當我們對這個檔案進行修改、**add**和**commit**三個動作後，這個檔案才正式進入被追蹤版本的狀態。



The diagram illustrates a sequence of events over time, represented by a horizontal axis labeled "time". The sequence starts with a series of nodes (red circles) connected by arrows. A branch labeled "nice\_feature" leads to a node that then merges back into the main sequence. Another branch labeled "very\_nice\_feature" leads to a node that also merges back into the main sequence. The final node is labeled "master".

# 進入暫存區(git add)開始追蹤檔案

- 打開hello.txt這個檔案，隨便打上一些指令並存檔



- 回到Git命令介面，輸入git add 主檔名.副檔名將修改推到 Staging Area(暫存區)。如果只打「git add .」不輸入檔案名稱，就會將整個資料夾的檔案都一起推送上去到Staging Area。

```
PS C:\gitProject1> git add hello.txt
PS C:\gitProject1> git status
On branch master

No commits yet

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





## 進入正式區(git commit -m)

- 最後用commit將Staging Area內的東西推到Repository成為正式的版本時，方式是輸入git commit -m並在一行之內寫上這次的修改動了哪些東西 (記得加上引號 ' ' )，方便其他協作者瞭解

```
PS C:\gitProject1> git commit -m "add a line"
[master (root-commit) 951a971] add a line
1 file changed, 1 insertion(+)
create mode 100644 hello.txt
PS C:\gitProject1> git status
On branch master
nothing to commit, working tree clean
```



# git log查看所有的commit紀錄

```
PS C:\gitProject1> git log
commit 951a971f09fc74df98ca72b3b23256ca32013c23 (HEAD -> master)
Author: Kenfan <kenfan@iii.org.tw>
Date: Thu Apr 7 16:32:00 2022 +0800

    add a line
```



# git add之後反悔

- `git rm --cached` 檔案名稱
- 若檔案不在Repository內：
  - `git rm --cached`幫我們從stage刪除，且檔案本來就是untracked，執行完還是untracked。
- 若檔案已經在repository內：
  - `git rm --cached`會幫我們從repository刪除，並且從stage刪除，因為已經從repository刪除檔案，檔案會從tracked變成untracked。



# git diff、git log、git show查詢修改紀錄

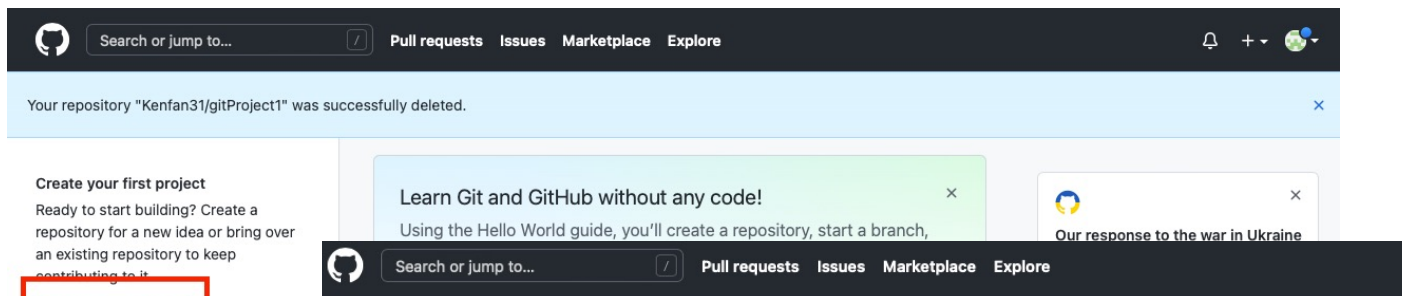
- 任何的修改後，再從Git命令列鍵入git diff就可以看到修改過後的紀錄。
- commit後，鍵入git log來查看所有commit的紀錄
- **git show** + commit代碼 (複製六碼或以上) 顯示該次詳細的修改內容

```
Kenfan@DESKTOP-3IPP46H MINGW64 /c/gitProject1 ((224f591...))
$ git diff
diff --git a/hello.txt b/hello.txt
index 4d890e1..3f7752f 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,3 @@
 print "hello world"
-print "hello world2"
\ No newline at end of file
```



# 推送程式碼至GitHub

- 在本機端寫了這麼久的程式，來試試看將本機端的程式碼推到GitHub網站上吧！
- 打開<https://github.com/>網站，登入後找到首頁
- 建立新的專案(repository)




## Create a new repository


A repository contains all project files, including the revisions elsewhere? [Import a repository.](#)

Owner <sup>\*</sup> Repository name <sup>\*</sup>  
 Kenfan31 / gitProject1



Great repository names are short and memorable. Need i

Description (optional)

☒  **Public**  
Anyone on the internet can see this repository. You cho

☐  **Private**  
You choose who can see and commit to this repository.

## Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** <https://github.com/Kenfan31/gitProject1.git> 

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

## ...or create a new repository on the command line

```
echo "# gitProject1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Kenfan31/gitProject1.git
git push -u origin main
```

## ...or push an existing repository from the command line

```
git remote add origin https://github.com/Kenfan31/gitProject1.git
git branch -M main
git push -u origin main
```

## ...or import code from another repository





# Git push 推送更新至github

- git add、git commit後，最後用git push將目前更新的版本「推送」到github端

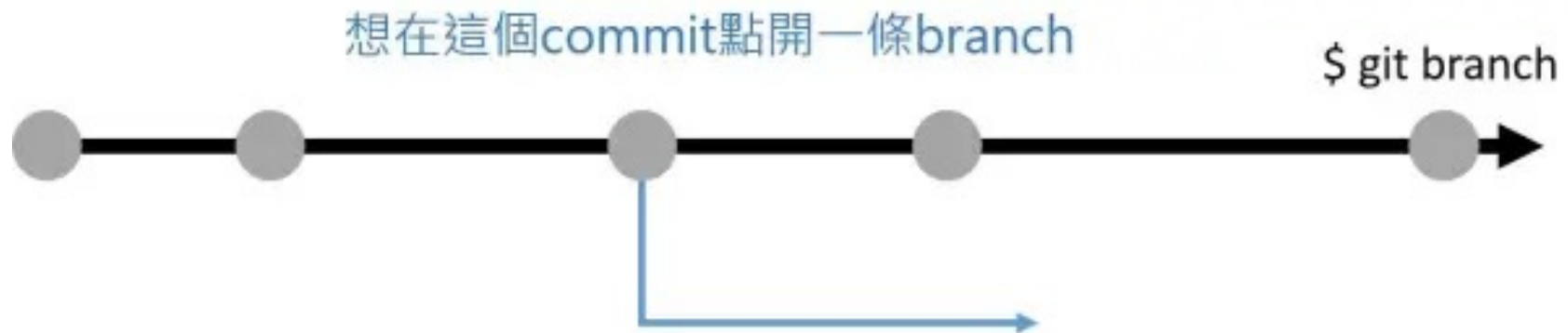
```
PS C:\gitProject1> git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (11/11), 845 bytes | 211.00 KiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Kenfan31/gitProject1.git
 * [new branch]      master -> master
```





# Branch(分支)

- 主幹(master)與分支(branch)是稱呼專案的主要版本和分支版本。在Git第一個建立的專案版本會被稱為master版本。
- master也僅是其中一條branch，所有branch間的關係都是平等的、彼此間無主從關係。一般習慣將穩定版本稱主幹，其餘的變動、開發中版本則都稱作分支。
- 開branch的方式非常簡單，直接輸入**git branch + 名稱**即可。





# git checkout回到過去

- 決定commit的版本之後，選好回溯的時間點後，發動時光機git checkout的威力(checkout後面輸入commit代碼至少6碼)
- 完成修改後推到 github

```
$git push -u origin branch_a
```



# 分支整合到主幹master

- 利用github整合分支到主幹上

The screenshot shows the GitHub interface for a repository named 'branch\_a'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. A yellow notification bar highlights that 'branch\_a' has recent pushes 2 minutes ago, with a 'Compare & pull request' button. Below this, the repository status shows 'branch\_a' with 2 branches and 0 tags. A message indicates 'This branch is 1 commit ahead of master.' with a 'Contribute' link. The commit history shows a recent commit by 'Kenfan31' titled 'update hello' with hash '84816ca' and 5 commits. Below the commit history, there is a table of files:

hello.txt	update hello	5 minutes ago
hello2.txt	update datas	2 hours ago
hello3.txt	update datas	2 hours ago

At the bottom, there is a light blue box with the text 'Add a README with an overview of your project.' and an 'Add a README' button.



# 常用指令集

Git 常用指令	說明
git init	將目前的目錄初始化為 <b>Git</b> 目錄, 建立本地儲存庫
git config	設定或檢視 <b>Git</b> 設定檔資訊
git add	將檔案加入 <b>Git</b> 暫存區
git rm	將檔案移出 <b>Git</b> 暫存區
git status	顯示 <b>Git</b> 狀態
git commit	將暫存區的檔案提交至儲存庫納入版本控制
git log	顯示過去歷次的版本異動
git reflog	顯示完整的版本異動歷史紀錄
git show	顯示指定版本的異動內容
git branch	建立一個新分支 (branch)
git checkout	取出分支內容還原為工作目錄
git merge	合併分支
git reset	重設某一版本
git clone	從遠端儲存庫 ( <b>GitHub</b> 或 <b>Bitbucket</b> ) 複製副本至本地儲存庫
git push	將本地儲存庫內容推送到遠端儲存庫
git pull	將遠端儲存庫拉回合併更新到本地儲存庫



# “You are not currently on a branch” 錯誤解決方法

- `$git branch temp` (暫存代碼)
- `$git checkout master`
- `$git merge temp`
- `$git branch -d temp`

[https://blog.csdn.net/zhai\\_865327/article/details/105031756](https://blog.csdn.net/zhai_865327/article/details/105031756)



# “failed to push some refs to” 錯誤解決方法

- `$git pull origin [your-branch]`
- `$git push origin [your-branch]`

<https://komodor.com/learn/how-to-fix-failed-to-push-some-refs-to-git-errors/>