

# Git 與 Github

TYIC 桃高資訊社

# Git

Git 是個版本控制系統(Version Control System，簡稱 VCS)

Git 是由 Linux 之父林納斯·托瓦茲(Linus Torvalds)開發  
開發的目的就是用來管理 Linux 的程式碼

版本控制系統，顧名思義，就是能控制版本  
每筆提交(commit)都會被紀錄下來，隨時可檢視和復原(revert)  
也可以建立分支(branch)，進行獨立修改  
並在必要時將分支合併(merge)

# Git

Git 的第一版 **README(讀我)** 中寫道：

 Git 的第一版 README

*GIT - the stupid content tracker*

*"git" can mean anything, depending on your mood.*

- random three-letter combination that is pronounceable, and not actually used by any common UNIX command.*

*The fact that it is a mispronunciation of "get" may or may not be relevant.*

- stupid. contemptible and despicable. simple. Take your pick from the dictionary of slang.*
- "global information tracker": you're in a good mood, and it actually works for you.*

*Angels sing, and a light suddenly fills the room.*

- "goddamn idiotic truckload of sh\*t": when it breaks*

*This is a stupid (but extremely fast) directory content manager.*

*It doesn't do a whole lot, but what it does do is track directory contents efficiently.*

*- Linus Torvalds*

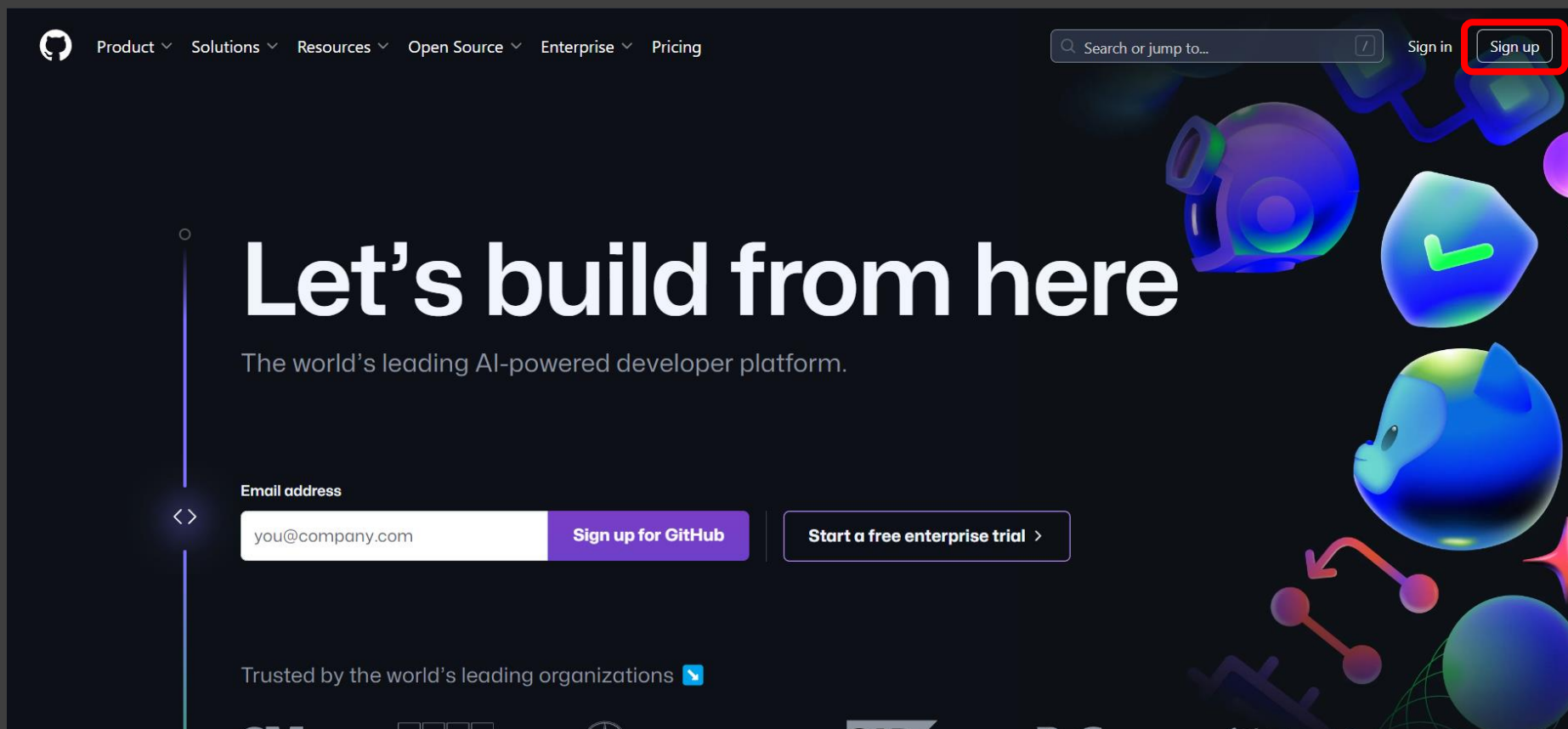
# Github

使用 **Git** 時，需要有地方可以存放檔案  
而當今最常見的就是 **Github**，網站為 <https://github.com>  
**Github** 除了提供倉庫(repository，簡稱 repo)供檔案存取外  
還提供了問題(issue)、動作(action)、維基(wiki)  
分叉(fork)、拉取請求(pull request，簡稱 pr)等服務  
使用 **Github** 相比直接使用 **Git**  
更著重在一個網路上的第三者參與或貢獻(contribute)該倉庫  
而倉庫擁有者(owner)則可以很方便的管理他人貢獻

# Github 帳號註冊

Github 帳號註冊非常簡單

只要在 <https://github.com> 首頁右上方選擇 "Sign up"  
接著依照指示即可完成註冊



# 倉庫

點選右上方頭像即會出現右方導覽欄

從右方導覽欄點選 **"Your repositories"**

即會顯示自己的所有倉庫

點擊右上方 **"New"** 即可開始創建新倉庫

The screenshot shows the GitHub profile of user 'Myster7494'. The header includes the username, a search bar, and navigation links for Overview, Repositories (12), Projects, Packages, Stars (67), and a 'New' button highlighted with a red box. The profile section on the left features a custom avatar of a purple hooded figure, the username 'Myster神秘陌生人', and a bio in both Chinese and English. Below the bio is an 'Edit profile' button. The main content area displays a list of repositories: 'PolynomialFactorization' (Python, updated Jun 29), 'tao-bus-app' (Dart, updated Jun 17), and 'cap-countdown' (Dart, updated May 30). Each repository entry includes its name, language, update date, and a 'Star' button. The 'New' button in the top right of the repository list is highlighted with a red box.

The screenshot shows the sidebar navigation menu for the user 'Myster7494'. The menu items are: 'Set status', 'Your profile', 'Your repositories' (highlighted with a red box), 'Your Copilot', 'Your projects', 'Your stars', 'Your gists', 'Your organizations', 'Your enterprises', 'Your sponsors', 'Try Enterprise' (with a 'Free' badge), 'Feature preview', 'Settings', 'GitHub Docs', 'GitHub Support', 'GitHub Community', and 'Sign out'.

# 創建倉庫

按照說明填寫即可  
星號(\*) 表示必填

這些欄位或選項  
在創建倉庫後也可以更改  
唯獨倉庫名稱須謹慎填寫  
創建後再更改倉庫名稱  
可能會造成一些問題

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

### Repository template

No template ▾

Start your repository with a template repository's contents.

Owner \*

 Myster7494 ▾

 / 

Repository name \*

倉庫名稱

Great repository names are short and memorable. Need inspiration? How about [probable-computing-machine](#) ?

Description (optional)

☒  **Public** 公開(大家都看得到)

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private** 私人(自己才看得到)

You choose who can see and commit to this repository.

倉庫說明

# 創建倉庫

同樣按照說明填寫即可

內容填寫完成後  
點擊右下角

"Create repository"  
即完成創建倉庫

The screenshot shows the GitHub 'Create repository' form with several annotations in Chinese:

- Initialize this repository with:**
  - ☐ **Add a README file** (新增 README). Below it: "This is where you can write a long description for your project. [Learn more about READMEs.](#)"
- Add .gitignore** (新增 .gitignore). Below it: ".gitignore template: None" and "Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)"
- Choose a license** (選擇開源協議). Below it: "License: None" and "A license tells others what they can and can't do with your code. [Learn more about licenses.](#)"

At the bottom, there is a note: "You are creating a public repository in your personal account." and a large green button labeled **Create repository** (創建倉庫).



# 創建倉庫

創建好後  
會跳轉到  
該頁面

The screenshot shows the GitHub interface for creating a new repository. At the top, the user 'TYSHIC' is logged in, and the repository name 'ExampleRepository' is visible. A search bar indicates 'Repository not found'. Navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings are present. The repository is marked as 'Public'. Action buttons for 'Edit Pins', 'Watch' (0), 'Fork' (0), and 'Star' (0) are shown. Two main cards offer options: 'Start coding with Codespaces' (with a 'Create a codespace' button) and 'Give access to the people you work with' (with a 'Manage access' button). A 'Quick setup' section, highlighted with a yellow border, provides instructions for users who have done this before, showing 'HTTPS' as the selected protocol and the URL 'https://github.com/TYSHIC/ExampleRepository.git'. Below this, two sections provide command-line instructions for creating a new repository or pushing an existing one.

ExampleRepository Public

Edit Pins Watch 0 Fork 0 Star 0

**Start coding with Codespaces**  
Add a README file and start coding in a secure, configurable, and dedicated development environment.  
Create a codespace

**Give access to the people you work with**  
Ensure the right people and teams have access to this repository.  
Manage access

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

Set up in Desktop or **HTTPS** SSH `https://github.com/TYSHIC/ExampleRepository.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 "# ExampleRepository" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/TYSHIC/ExampleRepository.git
git push -u origin main
```

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

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

# Git

有了倉庫之後，我們便可以開始上傳檔案  
但要使用 **Git** 上傳檔案之前  
電腦需要先與在 **Github** 上的遠端倉庫(remote repo)建立連線  
使用 **Git** 指令建立連線最簡單的方式就是：

1. 複製(clone)遠端倉庫到本地(local)
2. 進入複製下來的本地倉庫(local repo)
3. 直接連線遠端倉庫

# 複製遠端倉庫

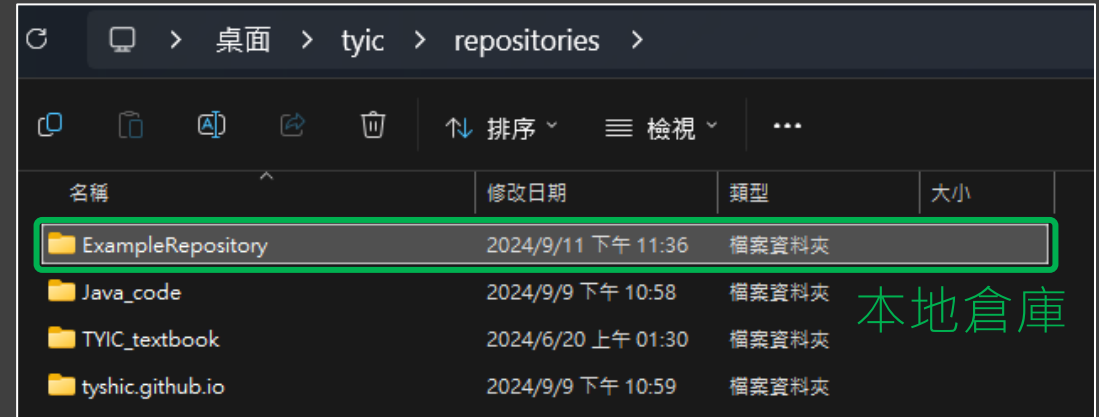
欲使用 **Git** 複製遠端倉庫，須使用該 **Git** 指令：

```
git clone 倉庫url
```

bash

```
Windows PowerShell x + 工作目錄
PS C:\Users\jacky\Desktop\tyic\repositories> git clone https://github.com/TYSHIC/ExampleRepository.git
Cloning into 'ExampleRepository'...
warning: You appear to have cloned an empty repository.
PS C:\Users\jacky\Desktop\tyic\repositories> |
```

使用該指令後會在  
工作目錄(**working directory**)  
出現一個新資料夾(**folder**)  
名稱與倉庫相同，即為本地倉庫

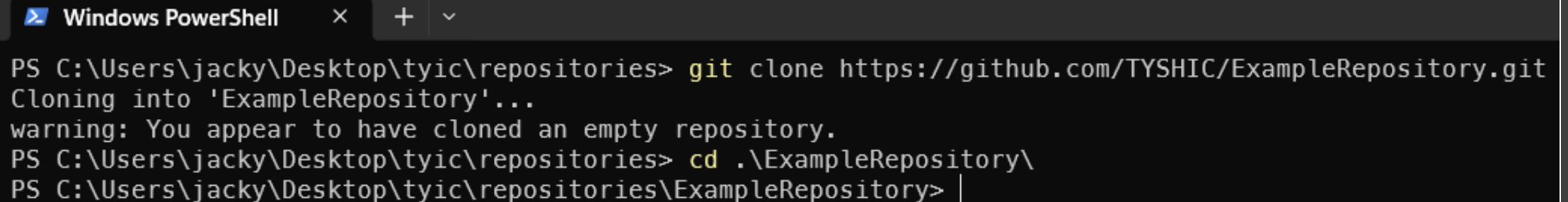


# 進入本地倉庫

欲進入本地倉庫，須在本地倉庫內開啟新的 `bash`  
或在剛剛的 `bash` 中使用該 `Windows / Unix` 指令  
該指令會切換當前 `bash` 的工作目錄至指定資料夾

`cd` 資料夾名稱

`bash`



```
Windows PowerShell  ×  +  v
PS C:\Users\jacky\Desktop\tyic\repositories> git clone https://github.com/TYSHIC/ExampleRepository.git
Cloning into 'ExampleRepository'...
warning: You appear to have cloned an empty repository.
PS C:\Users\jacky\Desktop\tyic\repositories> cd .\ExampleRepository\
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> |
```

# 連線遠端倉庫

在複製的本地倉庫中

輸入下方 **Git** 指令即可直接連線遠端倉庫：

```
git remote
```

bash

```
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git remote  
origin  
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> |
```

# Git 檔案狀態

Git 將檔案分為四種狀態：未追蹤(untrack)  
未修改(unmodified)、未暫存(modified)、已暫存(staged)  
可通過下方 Git 指令來查看當前的檔案狀態：

```
git status
```

bash

提交時只會提交已暫存的檔案，且已暫存檔案提交後會變回未修改  
而欲將未追蹤或未暫存的檔案變為已暫存，須使用下方 Git 指令：

特定資料夾或檔案：

```
git add 資料夾或檔案
```

bash

所有檔案：

```
git add -A
```

 bash 所有已修改檔案：

```
git add -u
```

 bash

# Git 檔案狀態

```
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> echo "# 範例倉庫" > README.md
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git status
On branch main

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git add .\README.md
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git status
On branch main

No commits yet

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

PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> █
```

# 提交與推送

提交就是將本地倉庫的變更加入到新版本

使用右方 **Git** 指令進行提交：`git commit -m "提交訊息"` bash

提交只會儲存在本地倉庫，不會自動推送(push)到遠端倉庫

要推送提交到遠端倉庫，須使用下方 **Git** 指令：

`git push` bash

```
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git commit -m "init"
[main (root-commit) d57cc20] init
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 232 bytes | 232.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/TYSHIC/ExampleRepository.git
 * [new branch]      main -> main
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> |
```



# 提交訊息

提交訊息最好要寫清楚該提交到底做了什麼  
但也必須保持精簡

常見的寫法是 "提交種類：簡單說明"

提交種類有：`feat(feature, 新功能)`、`fix(修復錯誤)`  
`docs(documentation, 更新說明)`、`style(格式變更)`  
`refactor(重構)`、`chore(雜項)`、`revert(撤回提交)` 等  
但在首次提交，常常使用 "`init`" 或 "`initial commit`"

# 提交與推送

The screenshot shows the GitHub interface for a repository named 'ExampleRepository'. The repository is public and has 1 star, 0 forks, and 0 watches. The 'main' branch is selected, showing 1 branch and 0 tags. The commit history shows a single commit by 'Myster7494' titled 'init' from 13 minutes ago. The file list shows 'README.md' and 'LICENSE' files. The 'About' section on the right provides repository details, including a link to 'Create a new release' and 'Publish your first package'.

**最近一筆提交**

**倉庫內容**

**README、LICENSE 等**

# 提交與推送

```
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git add -u
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git commit -m
"docs: changed README.md file encoding to UTF-8"
[main b4947d8] docs: changed README.md file encoding to UTF-8
 1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/TYSHIC/ExampleRepository.git
 d57cc20..b4947d8  main -> main
PS C:\Users\jacky\Desktop\tyic\repositories\ExampleRepository> |
```

