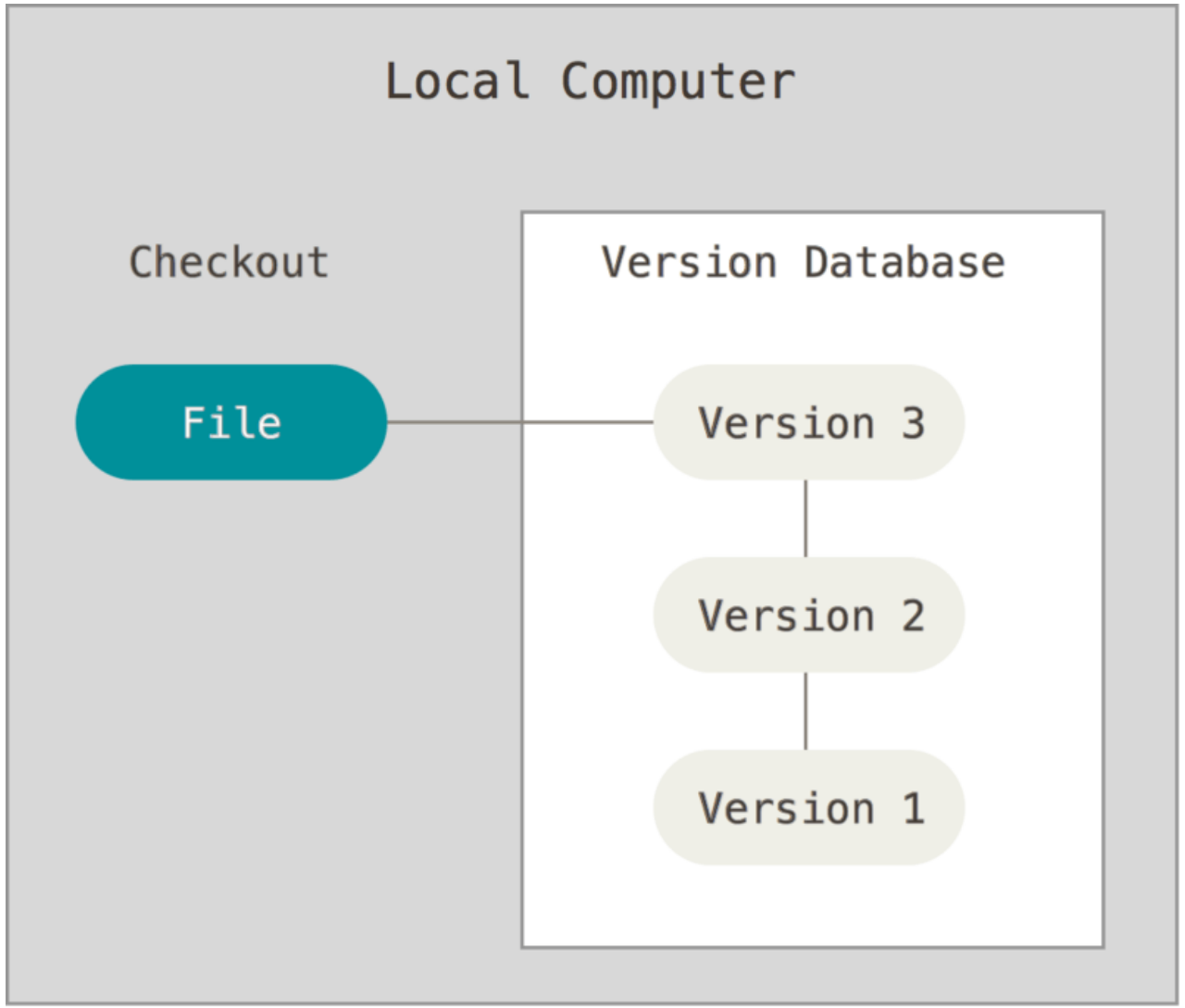
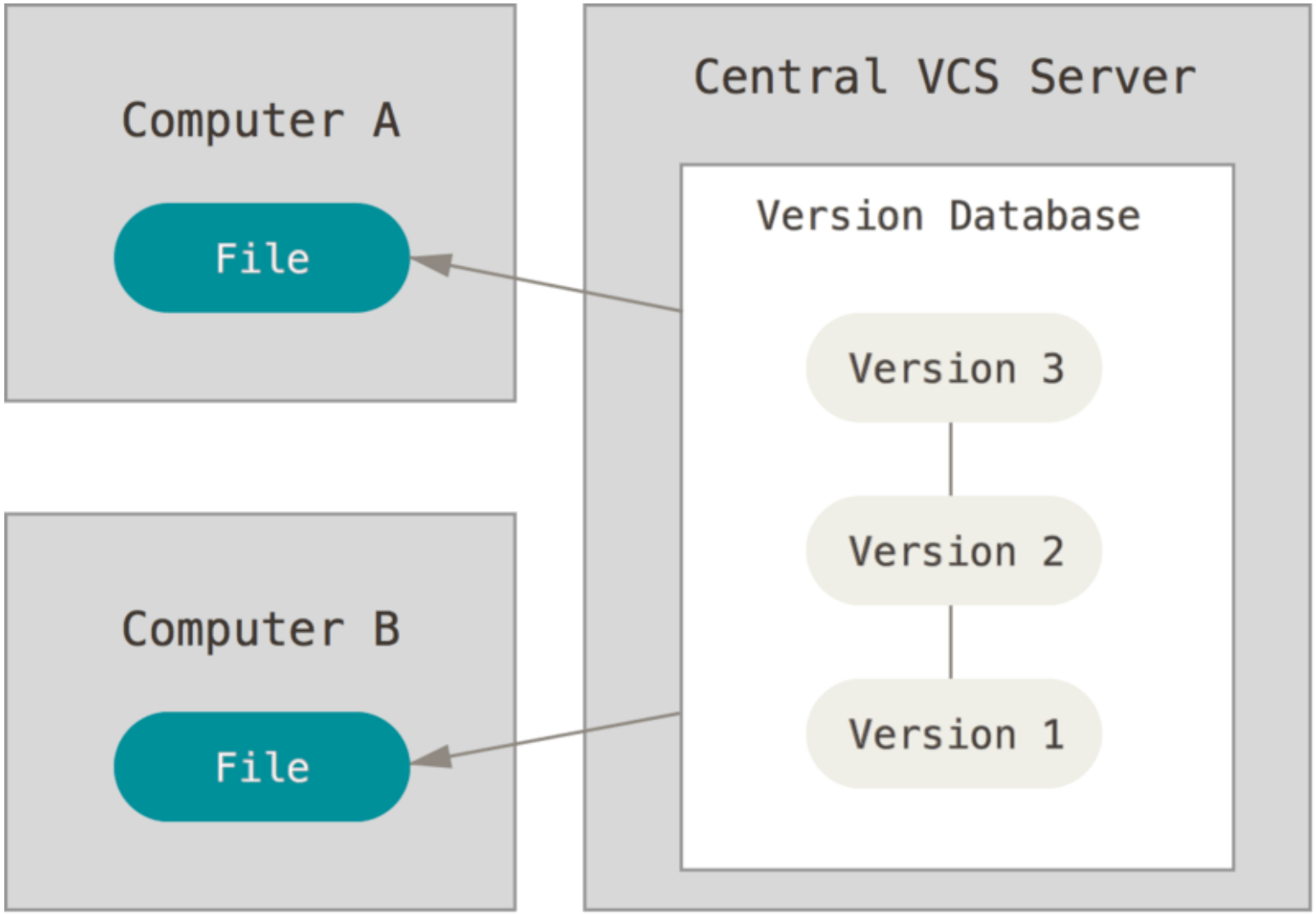
王裕縉 yubin@ispan.com.tw
資展國際股份有限公司

本地端版本控制



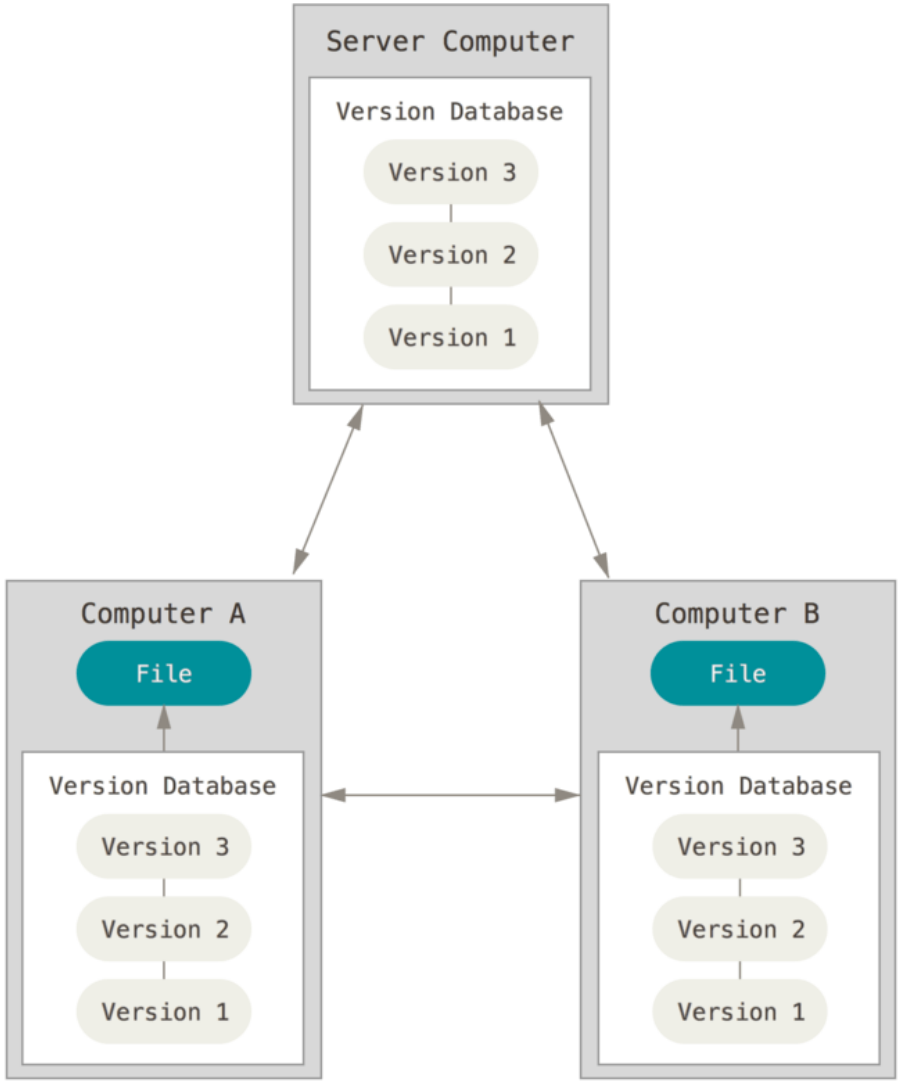
圖片來自：https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

集中化的版本控制系統



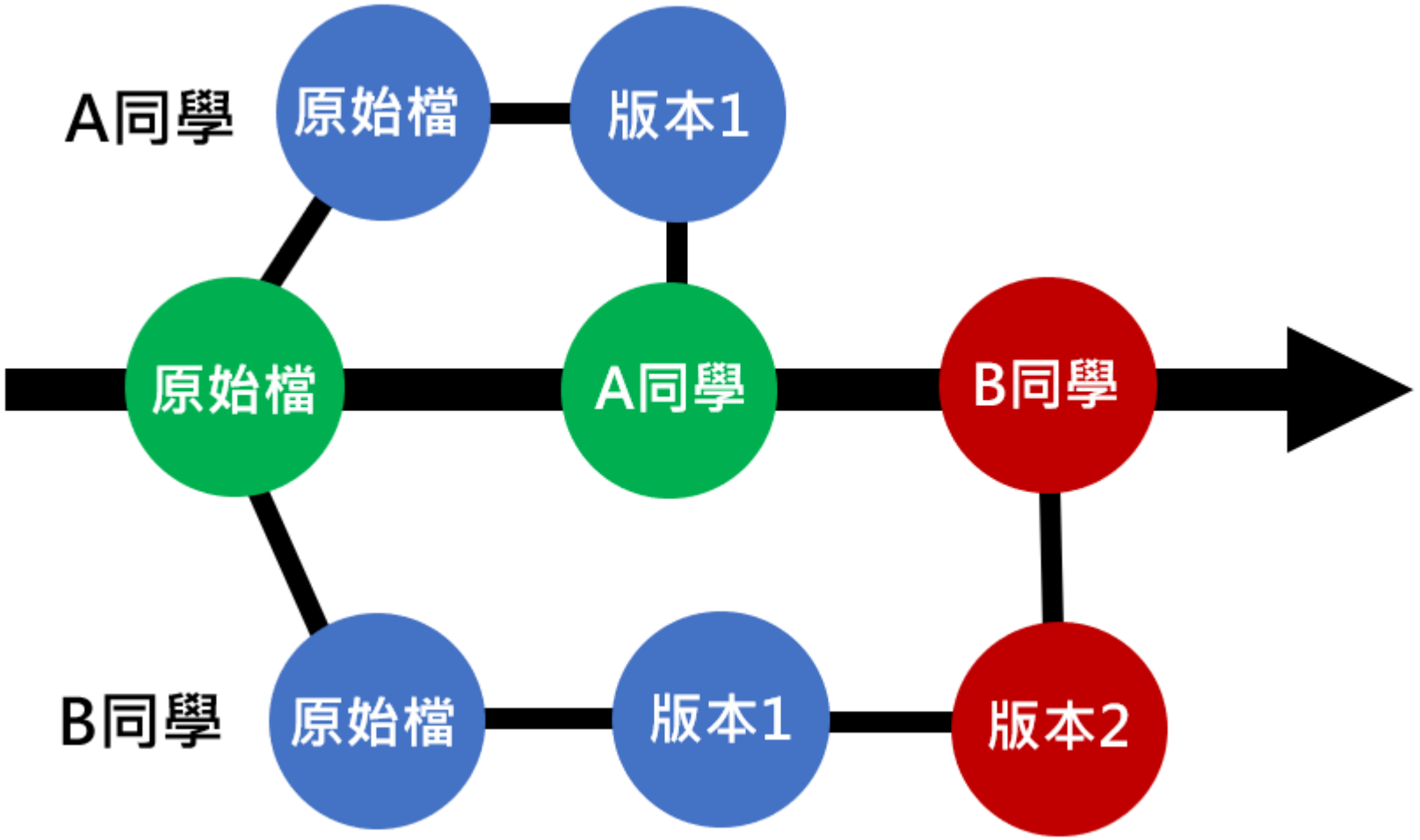
圖片來自：https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

分散式版本控制系統

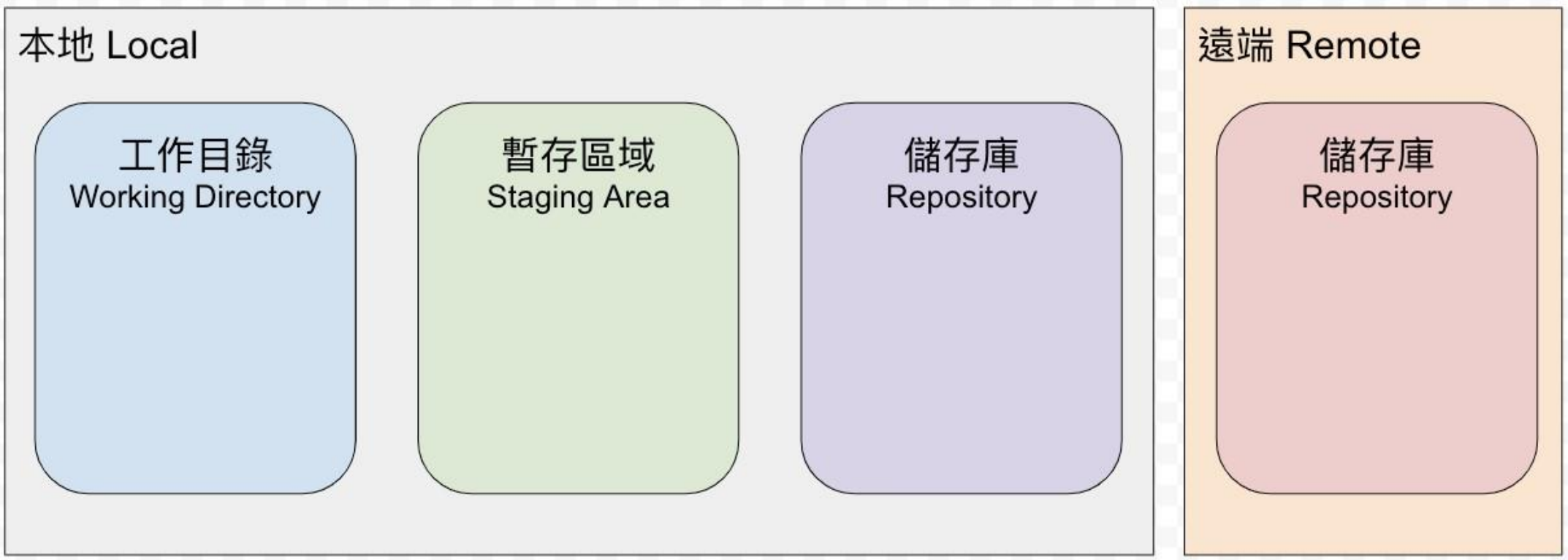


圖片來自：https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

分散式版本控制系統



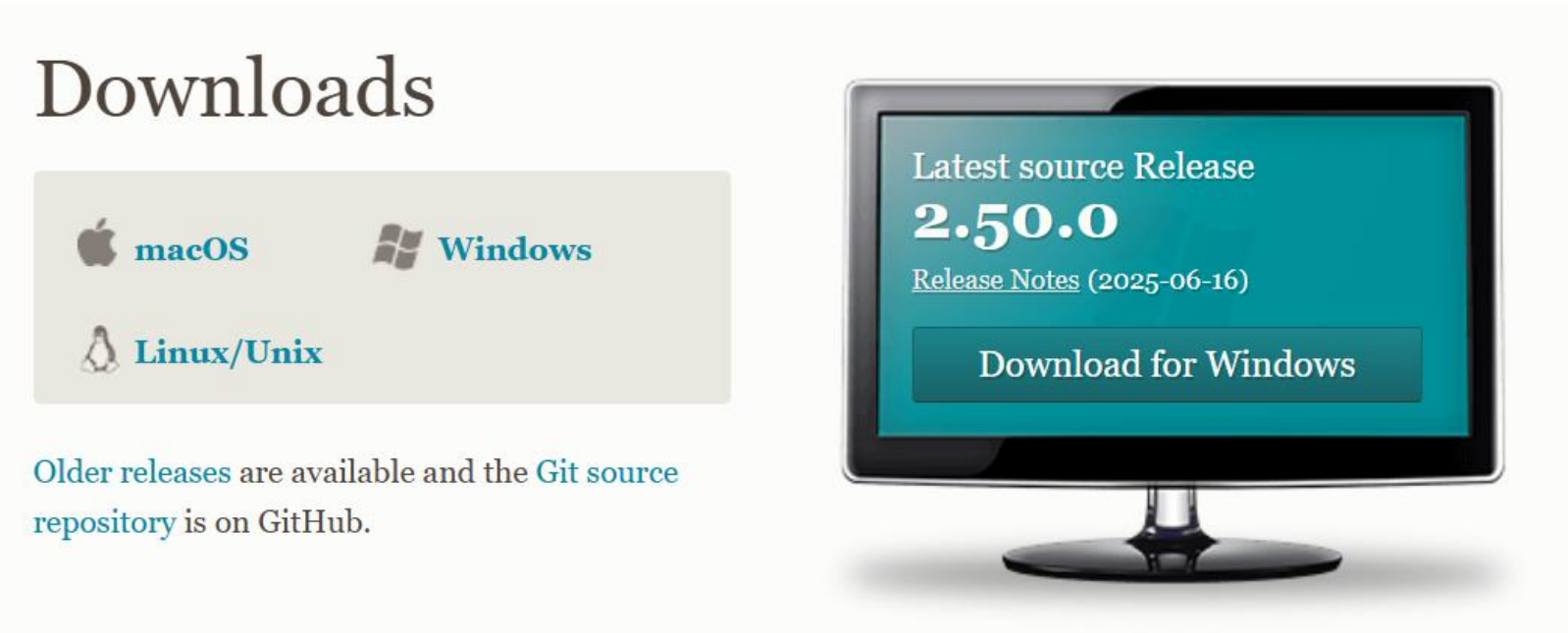
Git分散式版本控制



圖片來自：https://i.imgur.com/7WTvjCN.png

安裝 Git

- 下載 https://git-scm.com/downloads 及安裝



- 開啟命令提示字元(終端機)，輸入 **git --version**查詢安 裝的版本

使用者設定

- 開啟終端機設定使用者資訊

```
git config --global user.name “yubin”
git config --global user.email “yubin@ispan.com.tw”
```

- 檢視目前的設定

```
git config -- list
```

- 檢視單一設定

```
git config user.name
```

- 設定的資料儲存在
  - ✓ C:\Users\<使用者帳號>\.gitconfig

開始使用 – 建立 Git儲存庫(Repository)

```
git init
```

```
E:\myapp>git init
Initialized empty Git repository in E:/myapp/.git/
```

- 這個動作會建立一個叫.git隱藏目錄
  - .git目錄中紀錄了儲存庫中的所有更動的紀錄
- 查看儲存庫的狀態

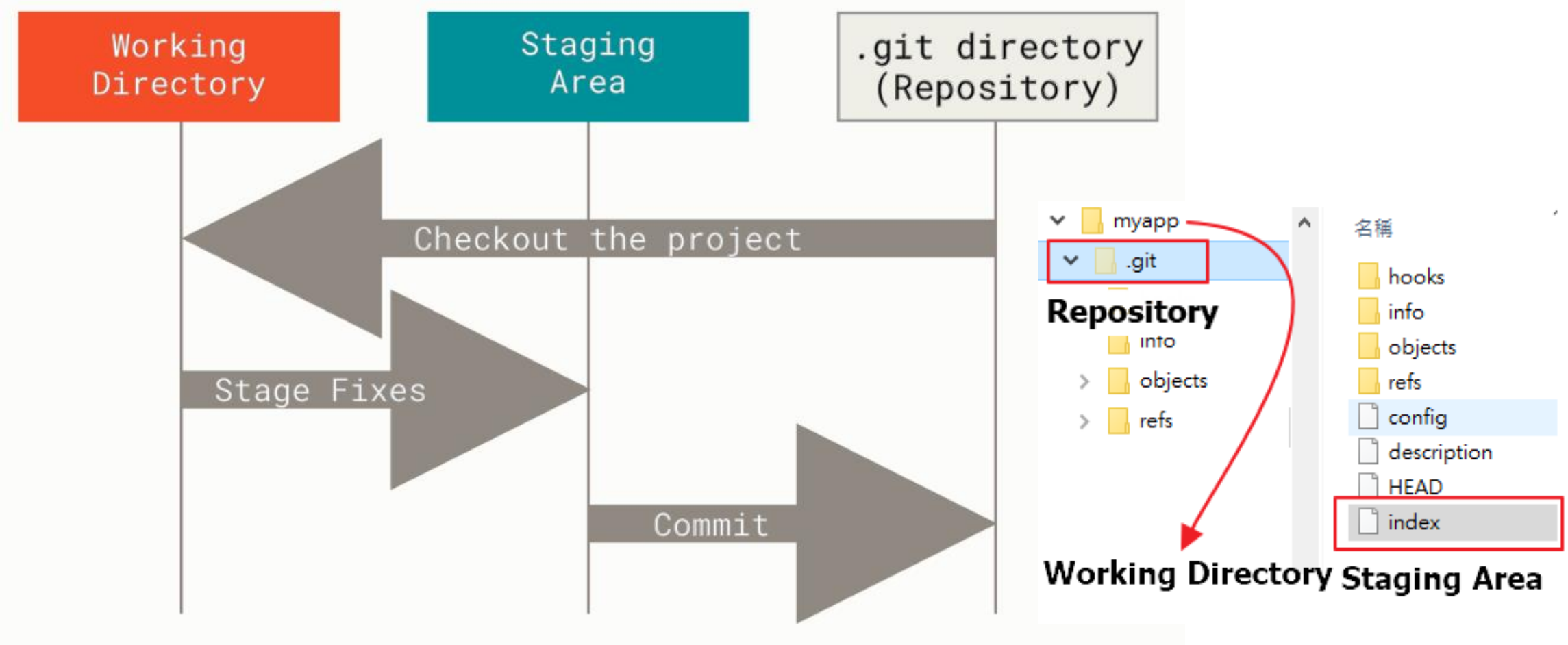
```
git status
```

```
E:\myapp>git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Working Directory工作目錄(區)

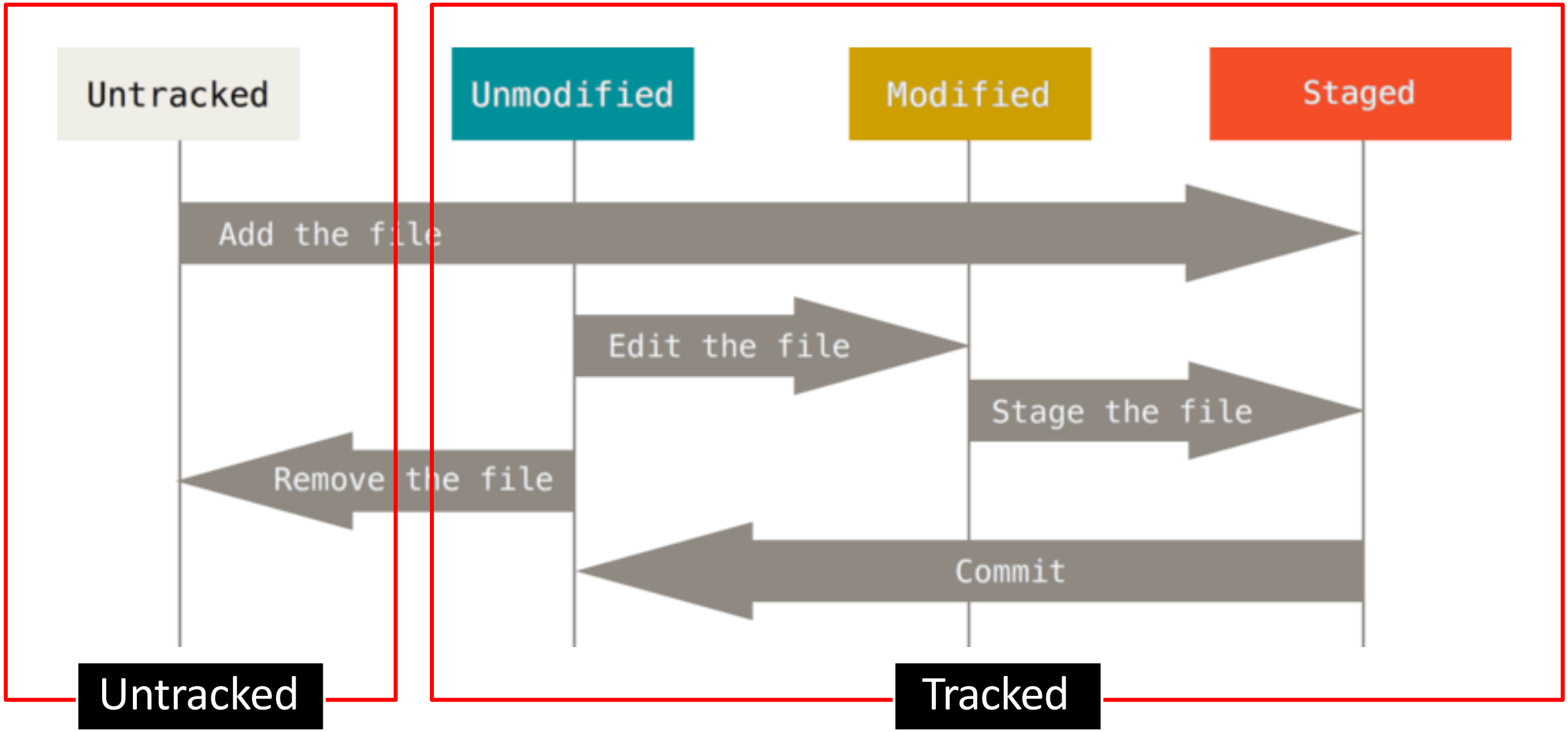


圖片來自： https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

工作目錄(Working Directory)中的檔案

- 兩種基本狀態
  - tracked：已經記錄在git repository中的檔案，其中又可以分成三種紀錄狀態
    - Unmodified：commit之後，還沒修改過的檔案
    - modified：commit之後，又修改過的檔案
    - staged：加到staging area的檔案
  - untracked：還沒有納入git控管的檔案

檔案狀態的生命週期



圖片來自：https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

顯示檔案狀態

- 加入一個檔案(page1.txt)到工作目錄中，然後顯示工作目錄中檔案的狀態

```
git status
```

```
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page1.txt

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

將檔案交給Git管理，進入到暫存區 (Staging Area)

- untracked的檔案，可以透過git add指令，將檔案加到staging area，讓Git來管理
- 單一檔案

```
git add page1.txt
```

- 副檔名是.txt的所有檔案

```
git add *.txt
```

- 所有檔案

```
git add –all
git add .
```

git add

```
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page1.txt

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

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git add page1.txt

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   page1.txt
```

從暫存區(Staging Area)進入到儲存庫(Repository)

- staging area中的檔案，可以透過commit指令，將檔案確認存到透過Repository

```
git commit -m "init commit"
```

  - -m "init commit" 是說明這次的commit做了甚麼事

```
$ git commit -m "init commit"
[main (root-commit) 137750e] init commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 page1.txt

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git status
On branch main
nothing to commit, working tree clean
```

修改檔案

- 將剛剛備份完成的檔案拿來修改，再透過git status查詢檔案的狀態

```
$ git status
On branch 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:   page1.txt

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

- 將修改的檔案備份，就重新再做一次 git add、git commit，這兩個動作可以合併成一次
  - git commit -am “update content”

將修改追加到最近一次commit

- 修改最近一次commit的message

```
git commit --amend –m  "修改的message "
```

- 將要備份的檔案加到最近一次的commit

```
git commit --amend --no-edit
```

```
$ git commit --amend --no-edit
[main 0432ca1] add some text
 Date: Tue Mar 15 23:02:08 2022 +0800
 1 file changed, 3 insertions(+), 1 deletion(-)
```

檢視commit紀錄

- 使用log指令，會由新到舊的顯示，是誰、什麼時候做了什麼事

```
git log
git log --oneline [--graph]
```

```
$ git log
commit 96916b70ac7fcf342aaa4db52f7f142c3a96e364 (HEAD -> main)
Author: hhwang <wang@ispan.com.tw>
Date:   Thu Feb 17 14:49:11 2022 +0800

    add aaaaaaa

commit 137750e86bf6c0087681b465bbc55499b1976d98
Author: hhwang <wang@ispan.com.tw>
Date:   Wed Feb 16 23:11:14 2022 +0800

    init commit
```

```
$ git log --oneline
96916b7 (HEAD -> main) add aaaaaaa
137750e init commit
```

- 退出 log 狀態，輸入 :q

git restore 還原修改 -1

- 檔案修改後，還沒有新增到暫存區(Staging Area)，要復原檔案中的修改

```
git restore <file>
```

```
$ git status
On branch 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:   page1.txt

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

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git restore page1.txt

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git status
On branch main
nothing to commit, working tree clean
```

git restore還原修改 -2

- 檔案修改後，已經新增到暫存區(Staging Area)，要退回到工作目錄(Working Directory)

```
git restore --staged <file>
```

```
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   page1.txt


wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git restore --staged page1.txt

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git status
On branch 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:   page1.txt

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

把檔案設定不被git控管

git rm --cached page1.txt

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   page1.txt


wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git rm --cached page1.txt
rm 'page1.txt'

wang0804@980517-NB3-DEI MINGW64 ~/Documents/workspace/temp (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page1.txt

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

檔案差異比較 git diff – 情境1

- 比對工作目錄(Working Directory) 修改檔案之間的差異

```
git diff
```

- 將比較的結果匯出

```
git diff > page.diff
```

```
$ git diff
diff --git a/page2.txt b/page2.txt
index 0362c55..682f232 100644
--- a/page2.txt
+++ b/page2.txt
@@ -1,3 +1,4 @@
 abcdefg
-xyz
-zzzzzzz
\ No newline at end of file
+aaaaaaa
+zzzzzzz
+ccccccc
\ No newline at end of file
```

--- 表示舊的檔案
+++ 表示新的檔案
-1 ,3表示舊的檔案中內容的1-3行
+1,` 表示新的檔案中內容的1-4行

空白 字元開頭表示這行兩個檔案都有
- 字元表示內容只有在舊檔案中
+ 字元表示內容只有在新檔案中

檔案差異比較 git diff – 情境2

- 比對暫存區(staging area)與儲存區(Repository)檔案之間的差異

```
git diff --staged
```

```
User@P215-2203-NB01 MINGW64 /d/workspace/git-works
$ git diff --staged
diff --git a/demo1.html b/demo1.html
index b7cd915..aed0330 100644
--- a/demo1.html
+++ b/demo1.html
@@ -11,5 +11,6 @@
      <p>aaa</p>
      <p>bbb</p>
      <div>1</div>
+     <div>2</div>
 </body>
 </html>
```

檔案差異比較 git diff – 情境3

- 儲存區(Repository)中任意兩個版本檔案之間的差異

git diff commit1 commit2

```
$ git diff 96916b7 58b9995
diff --git a/page1.txt b/page1.txt
index 2bd8960..633a71a 100644
--- a/page1.txt
+++ b/page1.txt
@@ -1 +1,3 @@
-aaaaaaaaaaaaaaaaaaaaaaaaa
\ No newline at end of file
+aaaaaaaaaaaaaaaaaaaaaaaaa
+bbbbbbbbbb
+ccccccccccccccc
\ No newline at end of file
diff --git a/page2.txt b/page2.txt
new file mode 100644
index 0000000..682f232
--- /dev/null
+++ b/page2.txt
@@ -0,0 +1,4 @@
+abcdefg
+aaaaaaa
+zzzzzzz
+ccccccc
\ No newline at end of file
```

git reset 切換版本 - 1

- 保留工作目錄(Working Directory)中的內容，暫存區(Staging Area)以及儲存庫(Repository)的內容回到指定的版本
- 指定版本的幾種寫法

```
git reset HEAD~~~~~
```

```
git reset HEAD~5
```

```
git reset HEAD^^
```

```
git reset <commit>
```

切換到之前的版本 - 情境2

- 保留工作目錄(Working Directory)及暫存區(Staging Area)的內容，儲存庫(Repository)的內容回到指定的版本

```
git reset --soft HEAD~~
```

切換到之前的版本 - 情境3

- 工作目錄(Working Directory)、暫存區(Staging Area)，儲存庫(Repository)的內容都回到指定的版本

```
git reset --hard HEAD~~
```

Reset 後的還原

- 列出所有的歷史紀錄

```
git reflog
```

```
6179f5f HEAD@{15}: reset: moving to HEAD~
4509c3e (HEAD, main) HEAD@{16}: reset: moving to 4509c3e
6179f5f HEAD@{17}: reset: moving to HEAD~
4509c3e (HEAD, main) HEAD@{18}: commit: add bbbbb
6179f5f HEAD@{19}: commit: add Page4.txt
ab1602c HEAD@{20}: commit: add some content
2c37cfb (origin/main, feature-login) HEAD@{21}: checkout:
```

- 回歸原始版本

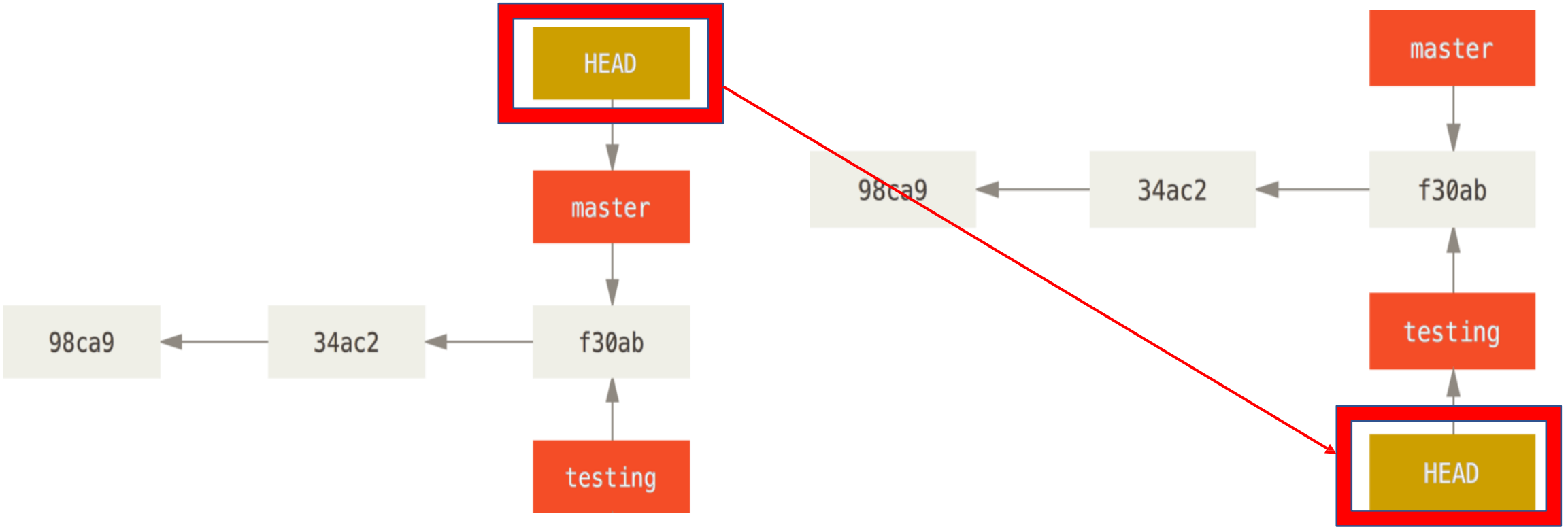
```
git reset --hard HEAD@{8}
```

branch 分支

- 將程式開發從開發主線上分離開來
  - 不同的版本就使用不同的分支，ex：1.0、1.0.1、1.1、2.0、....
  - 不同的軟體版本週期使用不同的分支，ex：Alpha、Beta、RC、RTM、.....
  - 單一功能的開發使用不同的分支
  - 不同的開發人員使用不同的分支
  - 為了修復問題也可以使用不同的分支
- 預設的分支名稱叫做?以前是master，現在是main
- 透過HEAD指標，指定目前使用中的branch

指標 HEAD

- 被 HEAD 指向的分支是目前分支



圖片來自：https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

新增分支

- 查詢目前專案有哪些分支

```
git branch
```

- 新增分支

```
git branch <branch分支名稱>
```

- 切換分支

```
git checkout <branch分支名稱>
```

- 同時建立及切換分支

```
git checkout -b <branch分支名稱>
```

```
$ git branch
* main

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git branch
* main

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git branch dev

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git branch
  dev
* main

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git checkout dev
Switched to branch 'dev'

wang0804@980517-NB3-DEI MINGW64 ~/Docu
$ git branch
* dev
  main
```

切換分支的另一種寫法(新)

- 切換分支 `git switch <branch分支名稱>`

```
wang0804@980517-NB3-DEI MINGW64 ~/documents/workspace/gitwork (relase)
$ git switch main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```

- 同時建立及切換分支 `git switch -c <branch分支名稱>`

```
wang0804@980517-NB3-DEI MINGW64 ~/documents/workspace/gitwork (main)
$ git switch -c feature1
Switched to a new branch 'feature1'
```

修改分支名稱

git branch –m <舊的分支名稱> <新的分支名稱>

```
$ git branch
* dev
  main

wang0804@980517-NB3-DEI MINGW64
$ git branch -m dev develop

wang0804@980517-NB3-DEI MINGW64
$ git branch
* develop
  main
```

刪除分支

- 不能刪除現在使用的分支，因此要先切換到別的分支

```
git branch –d(-D) <分支名稱>
```

```
$ git branch
  develop
* main

wang0804@980517-NB3-DEI MINGW64 ~/Docume
$ git branch -d develop
Deleted branch develop (was a6a3b72).

wang0804@980517-NB3-DEI MINGW64 ~/Docume
$ git branch
* main
```

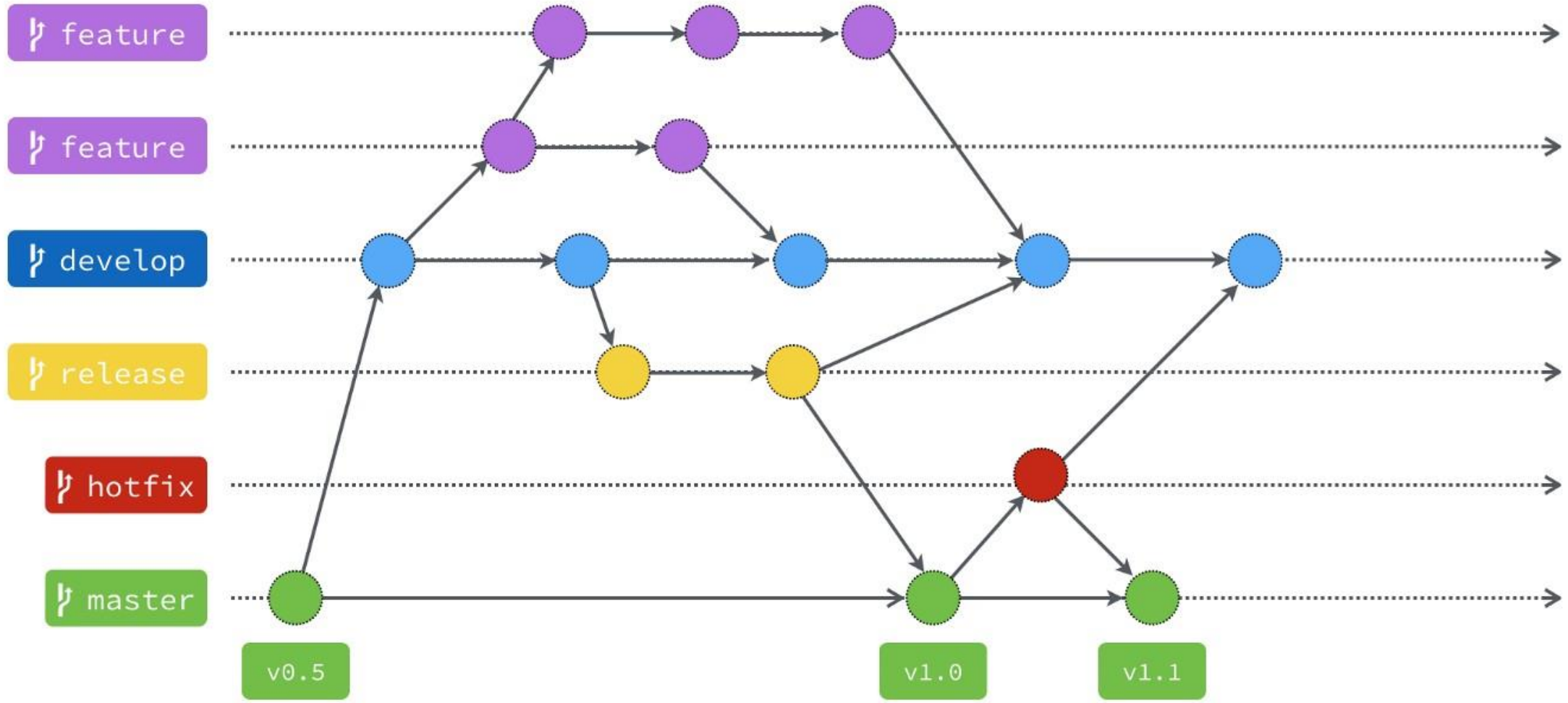
合併分支

- 如果要用main分支合併其它分支，就要先切換到main分支

```
git merge <分支名稱> -m "message"
```

```
$ git merge feature -m "daily merge"
Merge made by the 'ort' strategy.
 feature2.txt | 2 ++
 1 file changed, 2 insertions(+)
```

Git Flow



Git remote repository

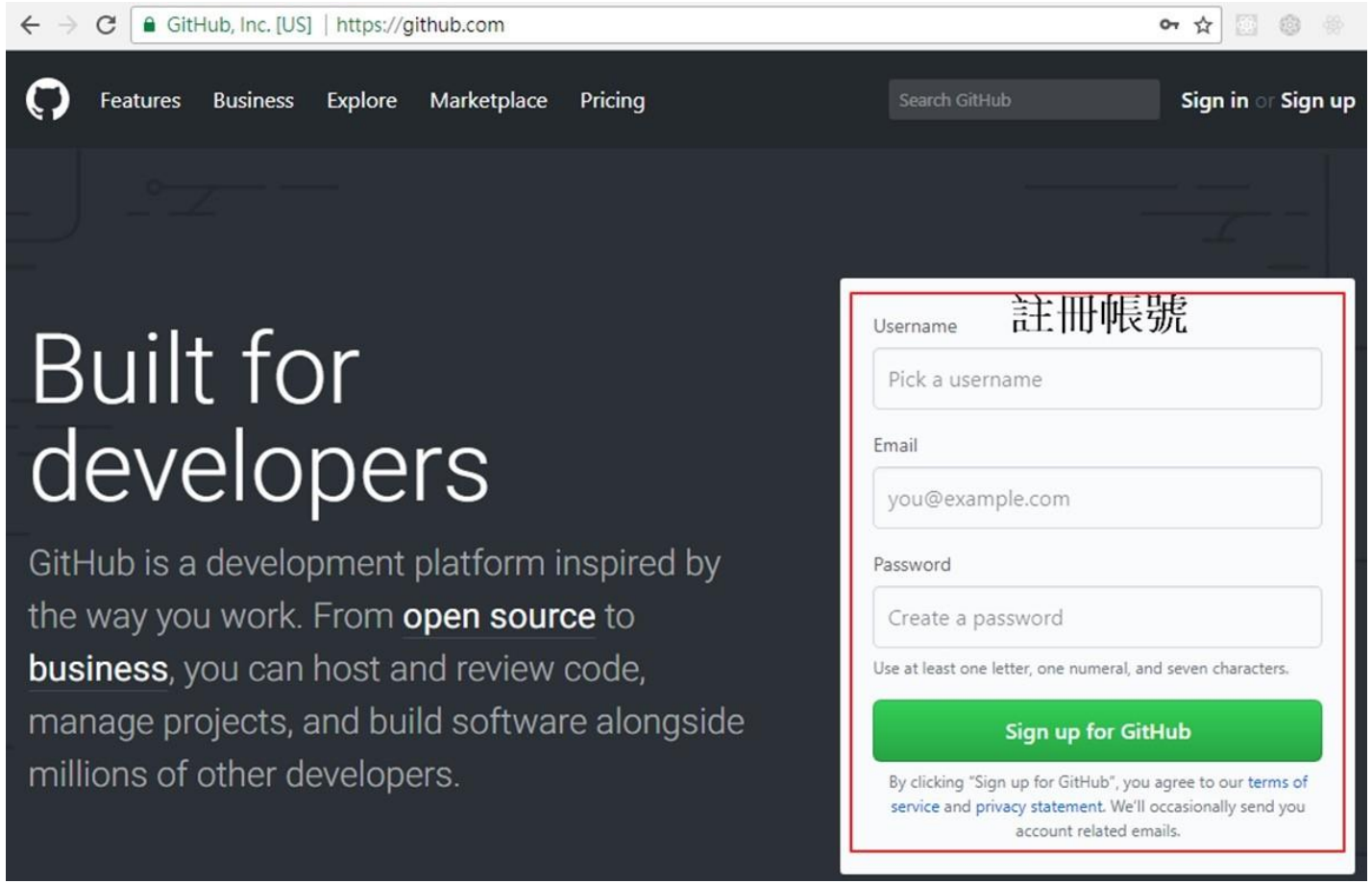
Git Remote Repository

- Git Repository
  - Local Repository：之前的用法就是local repository
  - Remote Repository：讓開發團隊成員分享各自的local repository資料而建立

- 我們可以自行建置 Git Server或使用坊間的Git Repository託管服務
  - GitHub、GitLab、Bitbucket、Gitorious

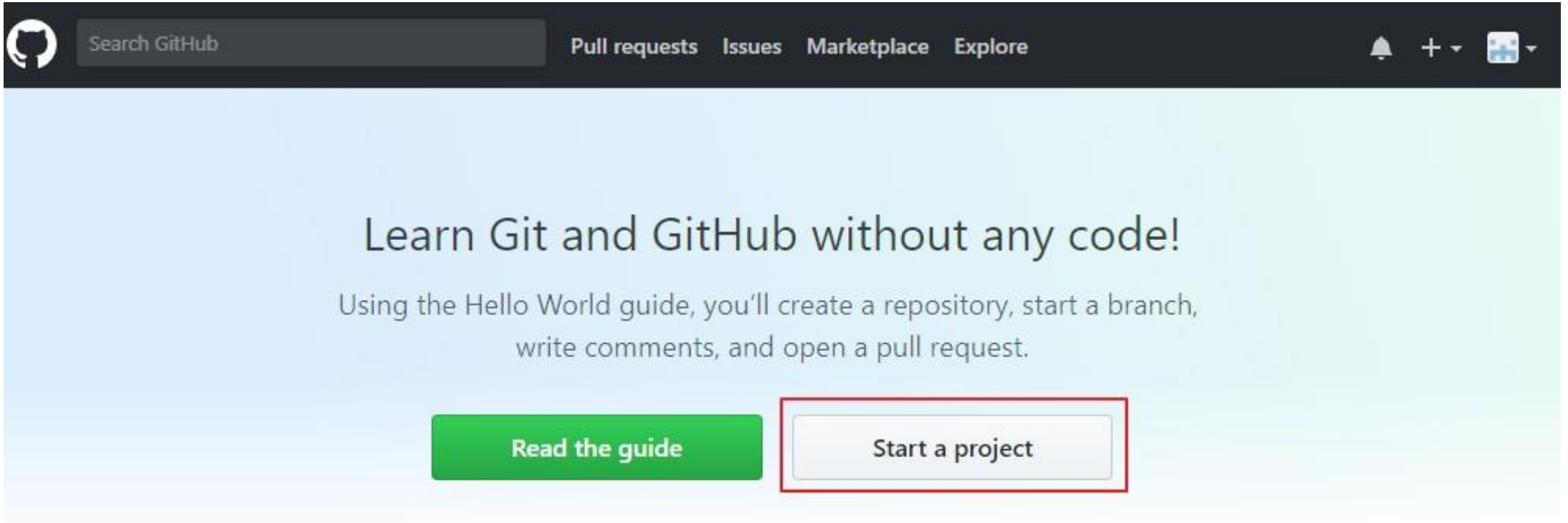
GitHub

- 申請GitHub帳號　　https://github.com/

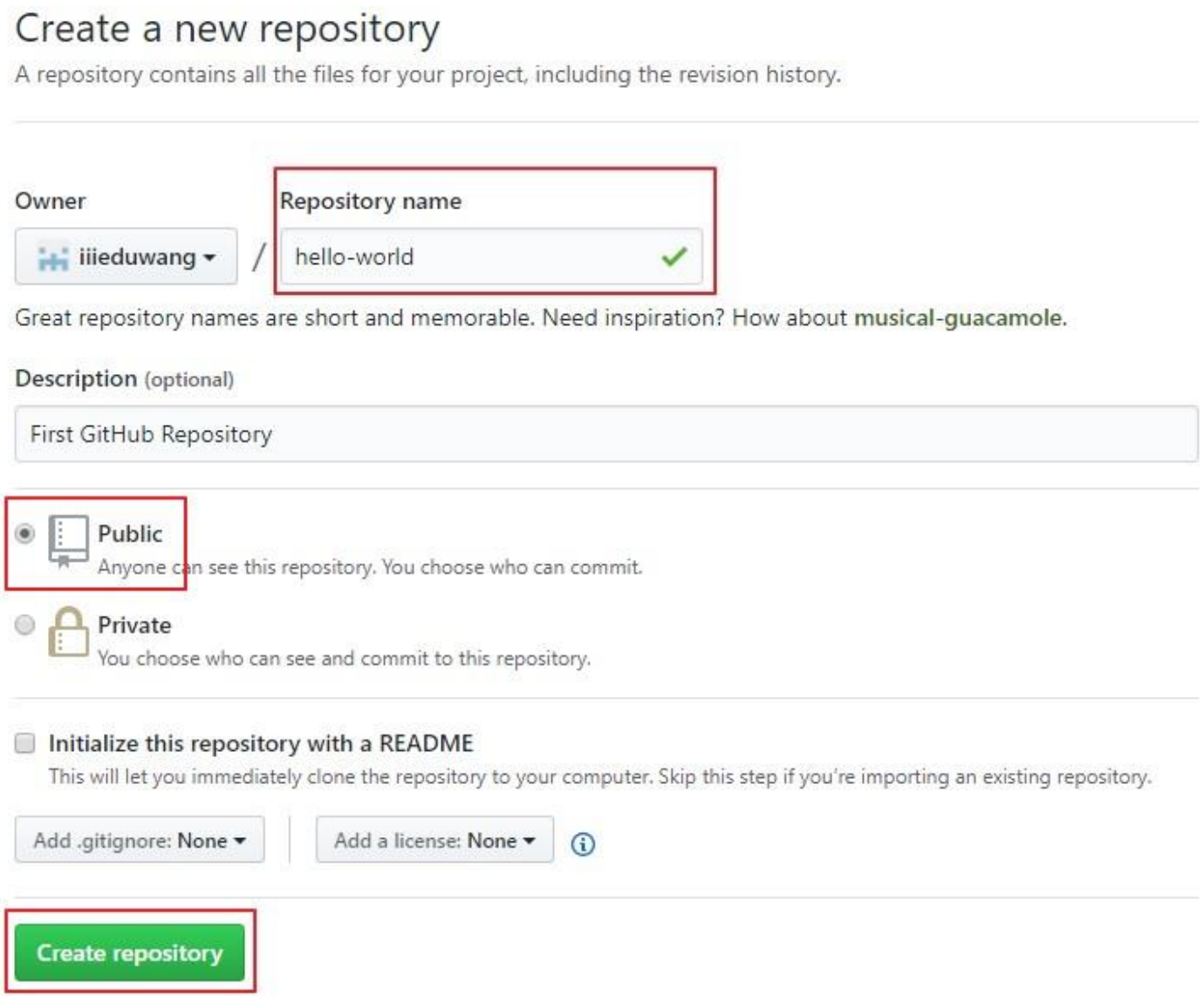


註冊完後

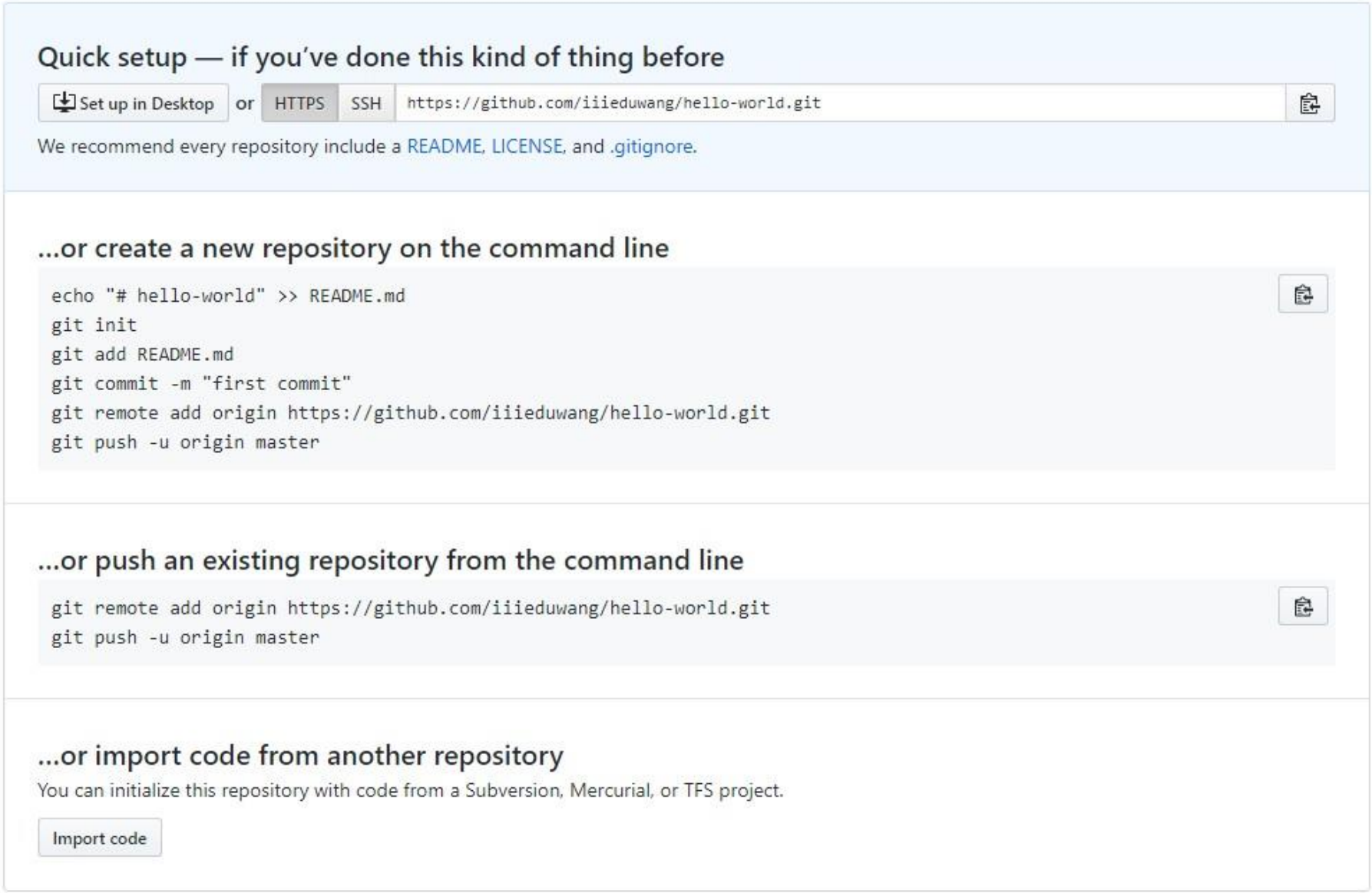
- 按下Start a project後，還會有一個驗證email的步驟，接下來才能建立專案[Start a project]



建立新專案


Create a new repository
A repository contains all the files for your project, including the revision history.
Owner
iiieduwang
/
Repository name
hello-world
Great repository names are short and memorable. Need inspiration? How about musical-guacamole.
Description (optional)
First GitHub Repository
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

完整的使用方式說明



從local連到remote 1

- 在Git環境加入remote repository的設定

```
git remote add <自取的名稱> <remote repository 的URL>
```

- remote repository url 格式
  - https://github.com/<GitHub帳號>/<GitHub上的Repository名稱>.git
  - 檢視所有remote repository的設定

```
git remote -v
```

- 檢視某個remote repository的設定

```
git remote show <名稱>
```

範例參考

```
E:\myapp>git remote add origin https://github.com/iiieduwang/hello-world.git

E:\myapp>git remote -v
origin  https://github.com/iiieduwang/hello-world.git (fetch)
origin  https://github.com/iiieduwang/hello-world.git (push)

E:\myapp>git remote show origin
* remote origin
  Fetch URL: https://github.com/iiieduwang/hello-world.git
  Push  URL: https://github.com/iiieduwang/hello-world.git
  HEAD branch: master
  Remote branch:
    master new (next fetch will store in remotes/origin)
  Local ref configured for 'git push':
    master pushes to master (local out of date)
```

從local連到remote 2

- 改變remote repository設定的名稱
- 檢視所有remote repository的設定

```
git remote rename <舊的名稱> <新的名稱>
```

- 刪除remote repository的設定

```
git remote remove <設定的名稱>
```

上傳 下載資料

- local repository的資料**上傳**到remote repository

```
git push <設定的名稱> <branch的名稱>
git push --all origin
```

- Remote repository的資料**下載**到local repository

```
git pull <設定的名稱> <branch的名稱>
```

git push 範例參考

```
E:\fromgit>git remote add origin https://github.com/iiieduwang/fromlocal.git

E:\fromgit>git push origin master
Fatal: HttpRequestException encountered.
Username for 'https://github.com': iiieduwang
Password for 'https://iiieduwang@github.com':
Counting objects: 14, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (14/14), 1.21 KiB | 0 bytes/s, done.
Total 14 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/iiieduwang/fromlocal.git
 * [new branch]      master -> master
```

git pull 範例參考

```
E:\fromgit>git remote add origin https://github.com/iiieduwang/hello-world.git

E:\fromgit>git remote -v
origin  https://github.com/iiieduwang/hello-world.git (fetch)
origin  https://github.com/iiieduwang/hello-world.git (push)

E:\fromgit>git pull origin master
remote: Counting objects: 14, done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 14 (delta 1), reused 14 (delta 1), pack-reused 0
Unpacking objects: 100% (14/14), done.
From https://github.com/iiieduwang/hello-world
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master
```

透過 README.md 說明Repo的內容

README.md

Ajax

上課日期 ： 2022/3/1 - 2022/3/3

> 這是一個上課練習 Ajax 的專案，課程內容包含了：

- XMLHttpRequst API
- XMLHttpRequest advanced features
  - FormData
- 甚麼是 Callback Hell
- 透過 Promise 解決 Callback Hell 的問題
- fetch()
- async await
- 什麼是 CORS 以及它的設定方法
- jQuery Ajax 相關方法
- Restful api 的開發使用 ASP.NET Web API

ignore file

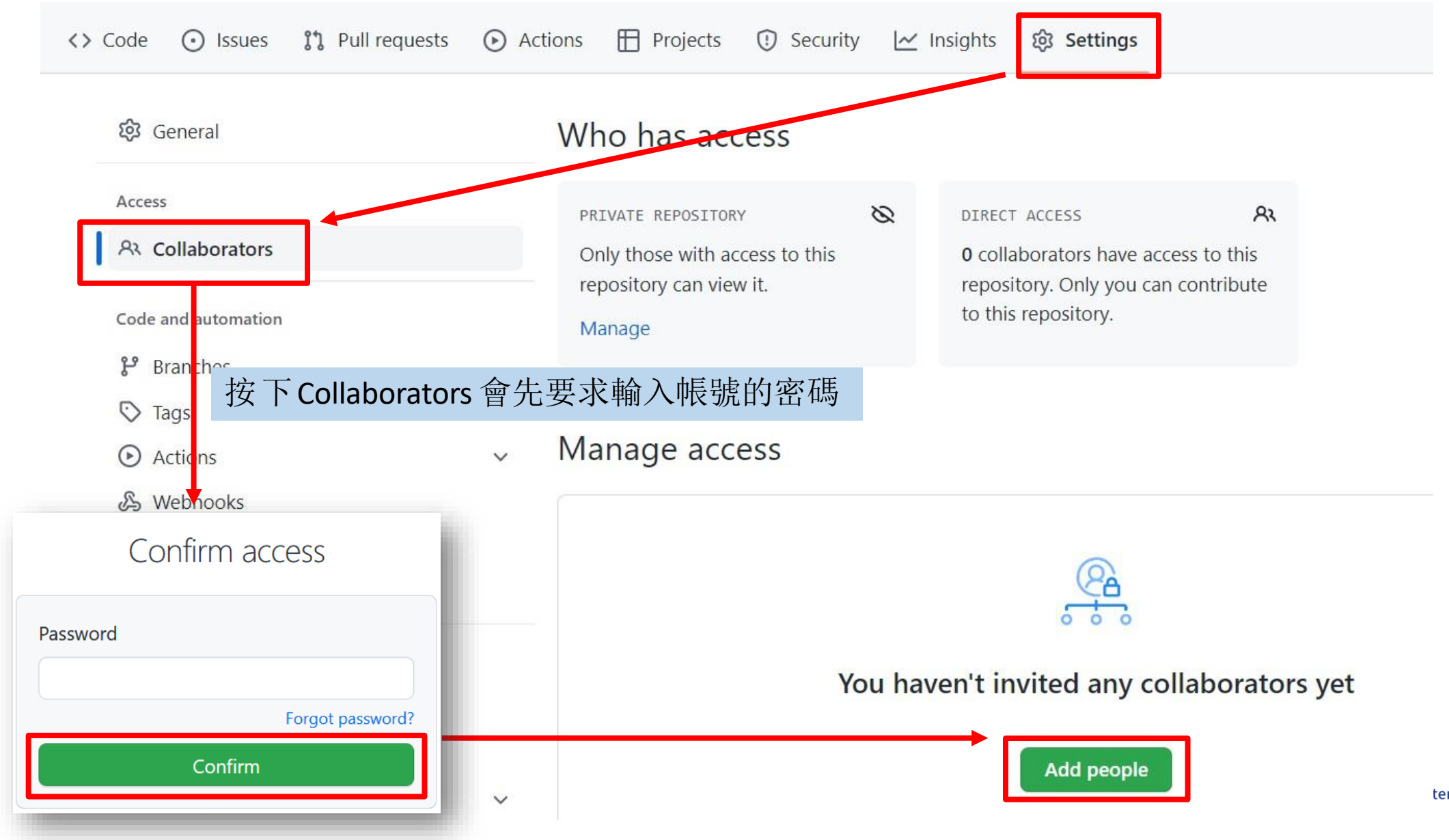
- 紀錄不想被Git管理的檔案
- 在工作目錄中放一個.gitignore檔案

.gitignore

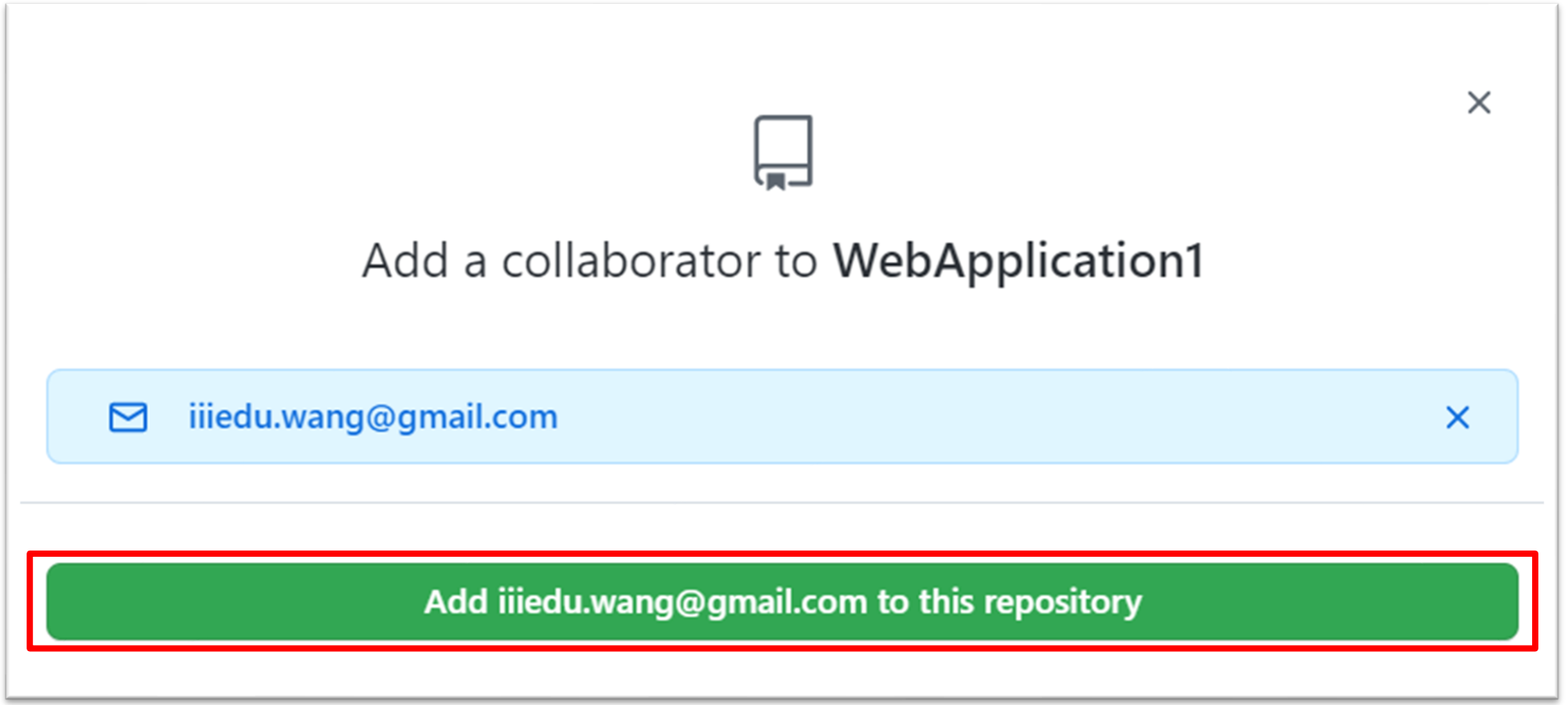
```
1  #忽略log目錄下的所有檔案
2  log/
3
4  #忽略config.json檔案
5  config.json
6
7  #忽略所有附檔名是.tmp的檔案
8  *.tmp
```

- Github提供了.gitignore檔案的範例
  - https://github.com/github/gitignore
  - https://www.toptal.com/developers/gitignore

新增團隊成員

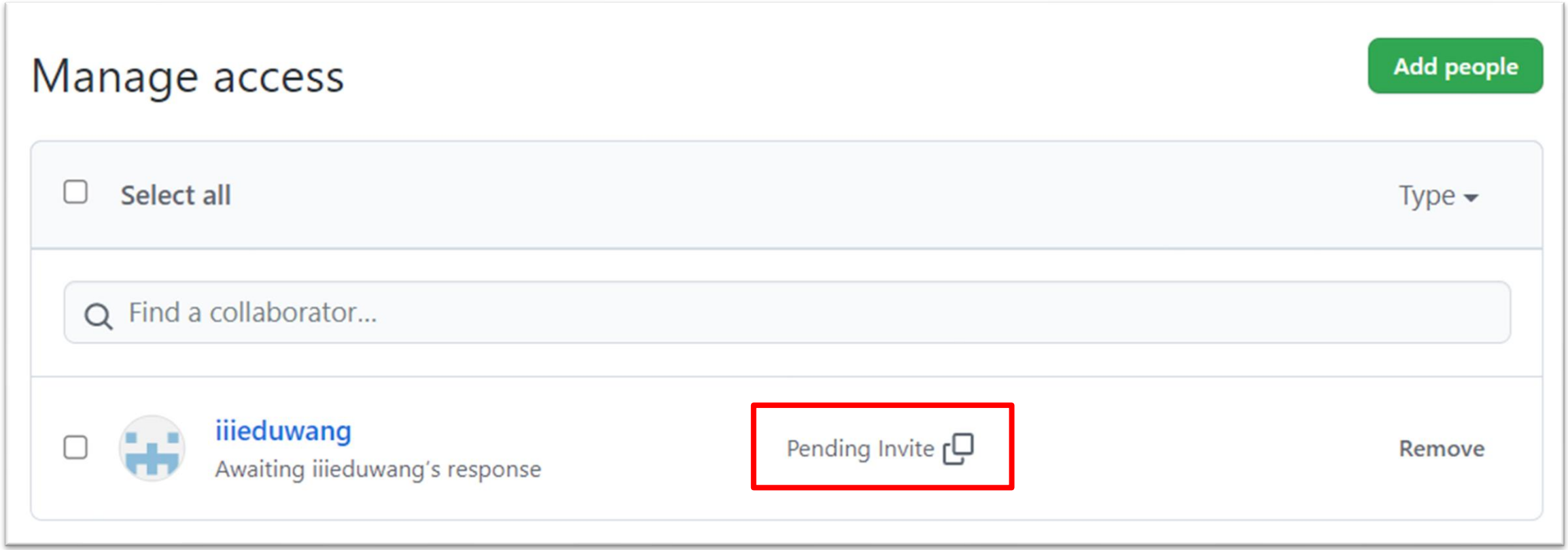


透過帳號或電子郵件找到團隊成員

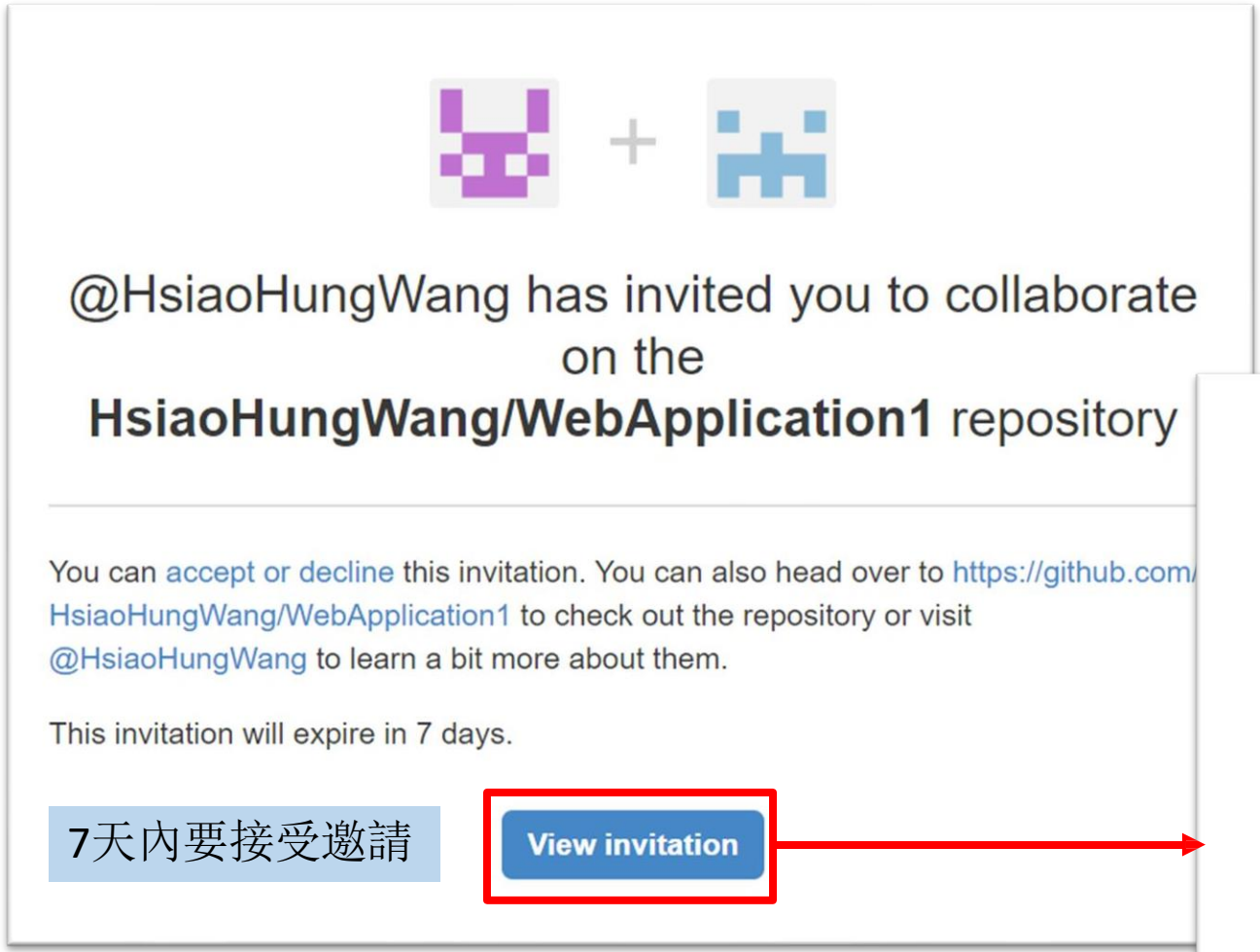


按下按鈕後會發一封請團隊成員確認的電子郵件

等候團隊成員的同意

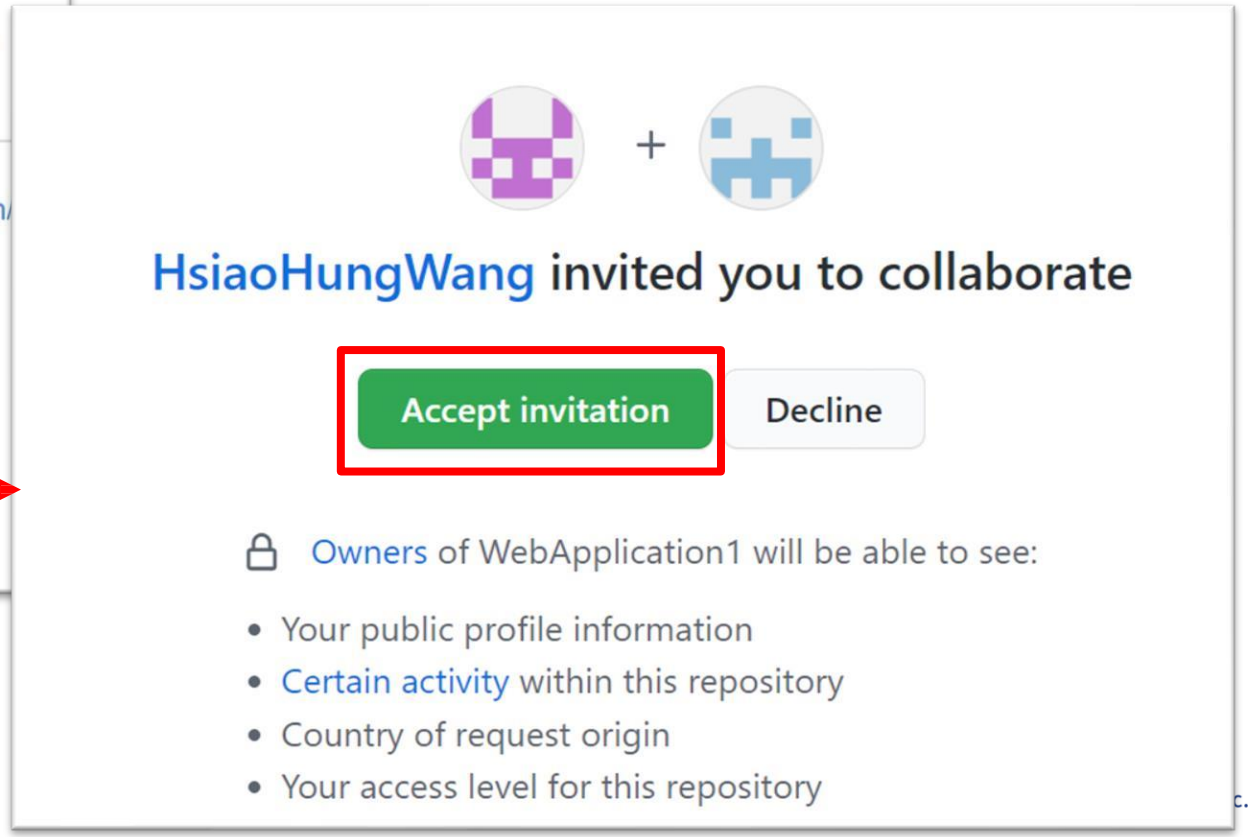

Manage access
Add people
Select all
Type
Find a collaborator...
iiieduwang
Awaiting iiieduwang's response
Pending Invite
Remove

團隊成員收到邀請的電子郵件

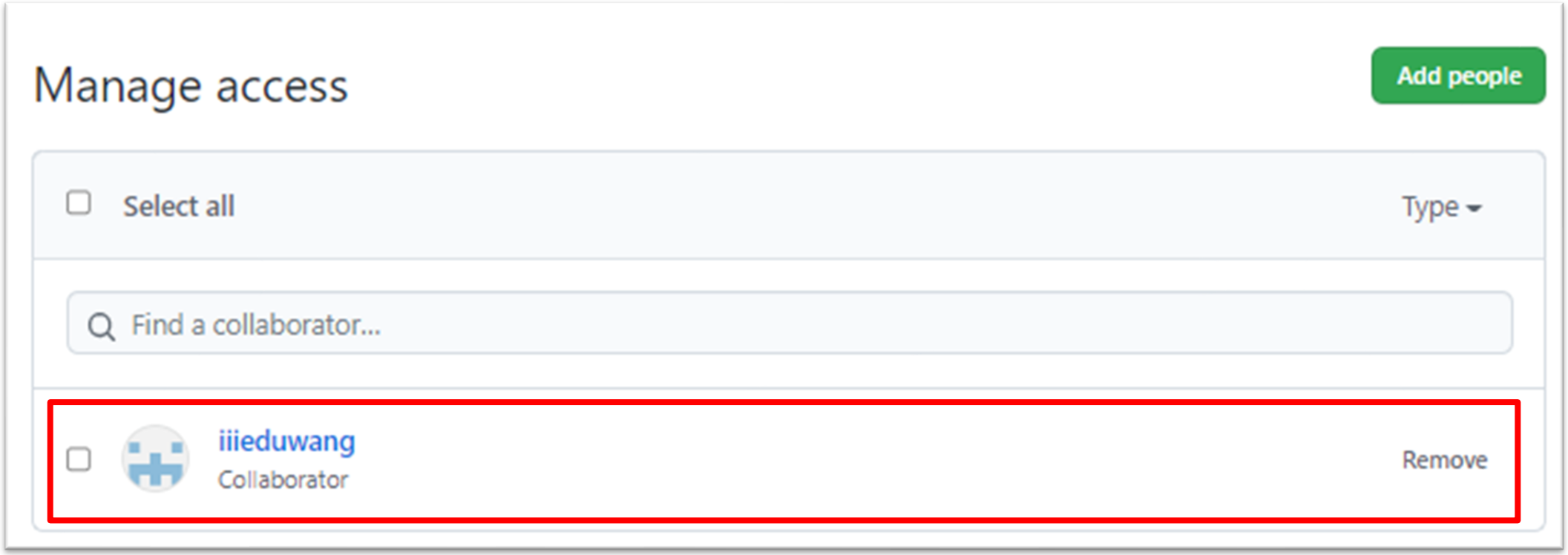


7天內要接受邀請

按下 [Accept Invitation] 按鈕，接收邀請



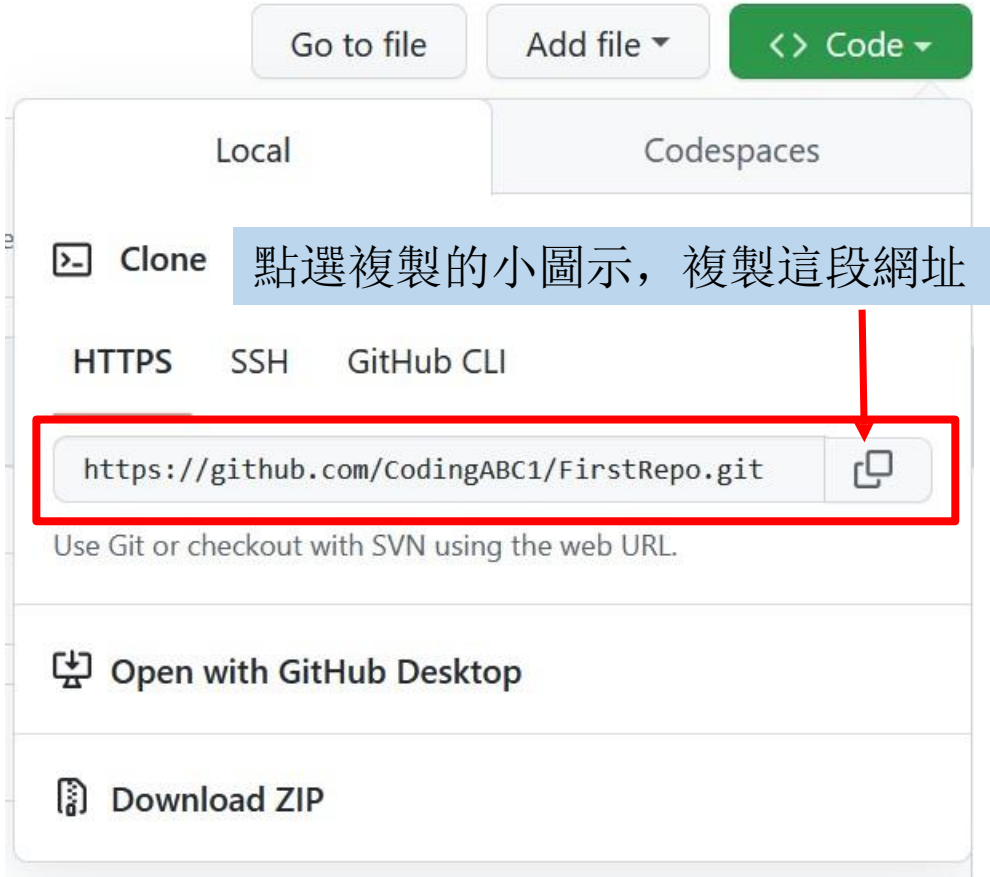
完成邀請團隊成員的流程



團隊成員下載 Github上的Repo

git clone https://github.com/CodingABC1/.....

將複製的網址貼到clone 的指令之後



之後團隊成員修改檔案要做版本控管就是:

- 在本機上使用
  - git add <file>
  - git commit –m '....'
- 將本機上的版本推到Github
  - git push

iSpan
LEARNING UNLIMITED

Thank you