【目錄】

第一章(Git)

1-1 版本控制 —— 004

1-2 Git安裝 —— 007

1-3 Git基本觀念 —— 024

1-4 Git基礎指令 —— 029

1-5 Git分支 —— 040

1-6 遠端儲存庫 —— 043

第二章(GitHub)

2-1 GitHub介紹 —— 046

2-2 建立Repository —— 050

備註

3-1 VIM操作 —— 056

3-2 SourceTree安裝 —— 058

3-3 Markdown語法 —— 063

GIT & GITHUB

洪子敬

2

第一章
CIT

第一節
版本控制

- 【版本控管】網路節錄

如果你用 Microsoft Word 寫過東西，那你八成會有這樣的經歷：

想刪除一段文字，又怕將來這段文字有用，怎麼辦呢？有一個辦法，先把當前檔案"另存為"一個檔案，然後繼續改，改到某個程度，再"另存為"一個檔案。就這樣改著、存著.....最後你的 Word 文件變成了下圖：

過了幾天，你想找回被刪除的文字，但是已經記不清儲存在哪個檔案了，只能挨個去找。真麻煩，眼睛都花了。看著一堆亂七八糟的檔案，真想把不用的都刪除，只保留一個終極版，可是又怕哪天會用上，還不敢刪，真愁人。

這時候你想，要是有一個軟體能幫你記錄每一個版本，並且能顯示出各個版本之間的差異，那多方便啊。幸運的是，這樣的軟體還真有，git 就是其中一個。

git 不僅具備記錄版本、比較差異等最基本的功能，還支援分支、離線開發、多點開發..... 總之 git 有非常豐富而強大的功能等著你去學習與發現。如果說你學習 git 是為了收穫一縷春風，那麼最後你得到的將是整個春天。

何謂GIT?

GIT是一種分散式版本控管工具。

原設計目的是為了更好地管理Linux核心開發，後發展成業界主流的版本控管工具。

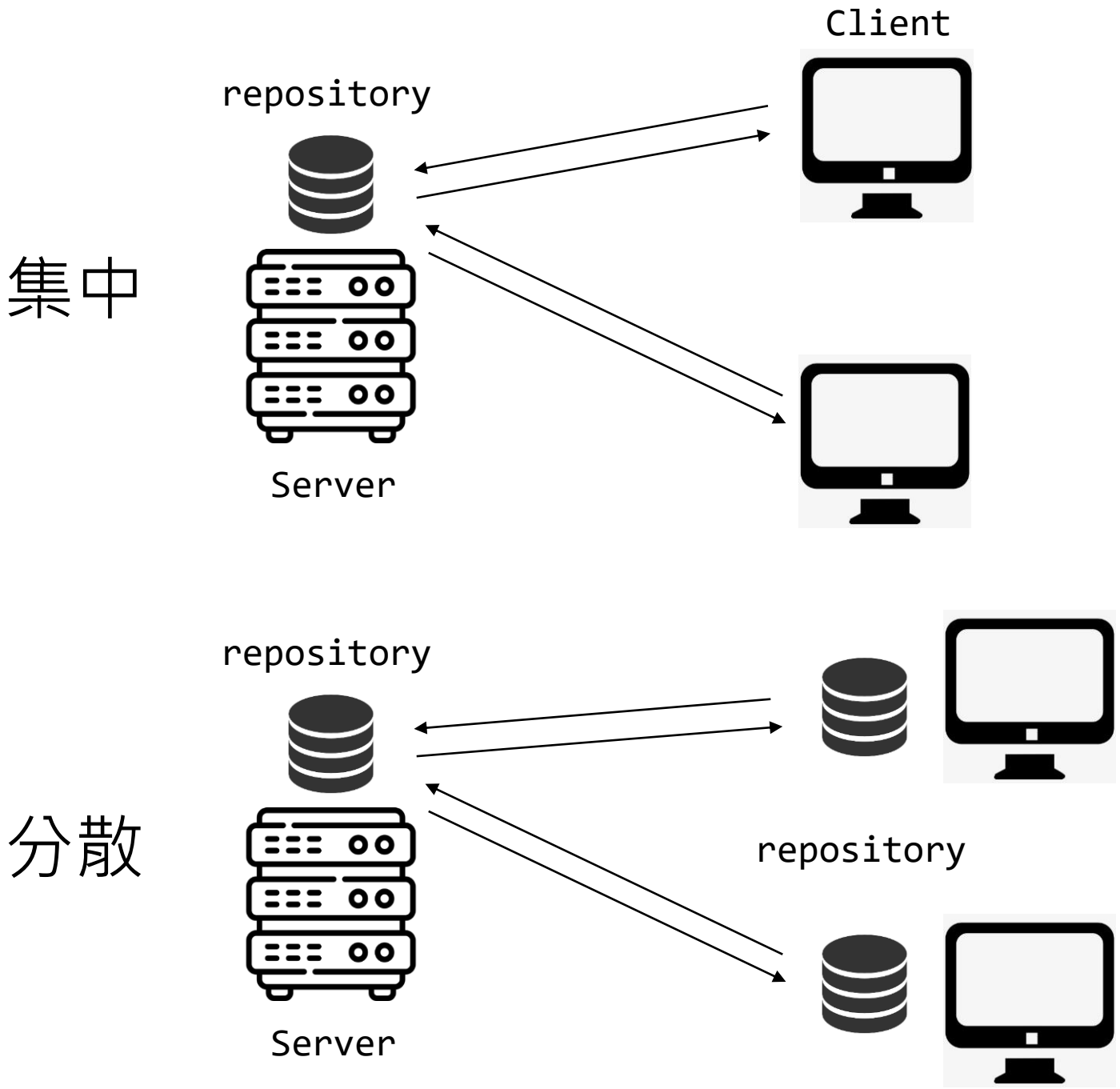
最初由林納斯·托瓦茲創作，於2005年以GPL授權條款釋出。

【林納斯·托瓦茲】
Linux核心的首要架構師與專案協調者、GIT專案發起人&主要開發者。

【Linux】
開源的作業系統，安全性高、速度快，適合安裝在Sever上；一般使用者多數使用發行版，如Ubuntu、Mint等等。

【GNU通用公眾授權條款】
也稱作GNU GPL或GPL，相對於版權Copyright，授權模式為Copyleft，此類授權允許他人自由使用、散佈、修改，惟修改後的衍生作品也得繼承相同授權條款。
Copyleft的精神不是反著作權。

集中 vs 分散



版本控制軟體分成兩種類型：

【集中】

優點：

1. 設計簡單
2. 每個使用者修改的都是最新版本
3. 利於檔案備份

缺點：

1. 檔案鎖定問題、開發效率低落
2. 所有操作均需連網(本機只保留最新版本)
3. 上傳未開發完成的程式，可能會造成預期外的錯誤
4. 有Server當機、資料損毀風險

【分散】

優點：

1. 每個使用者有獨立的repository，可在離線時作業
2. 可以容易的從現有專案分叉(fork)出新的專案

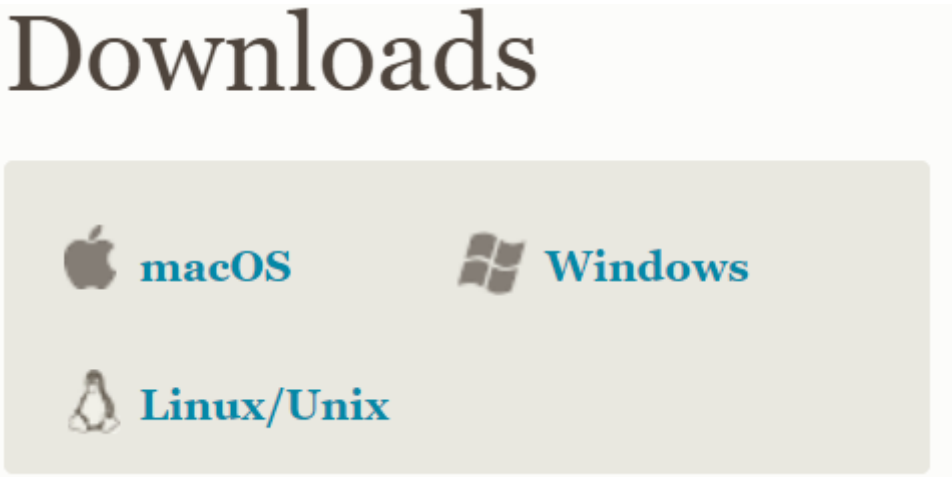
缺點：

1. 一開始複製專案速度較慢(因完整複製)
2. 需要更多的儲存空間

第二節
GIT安裝

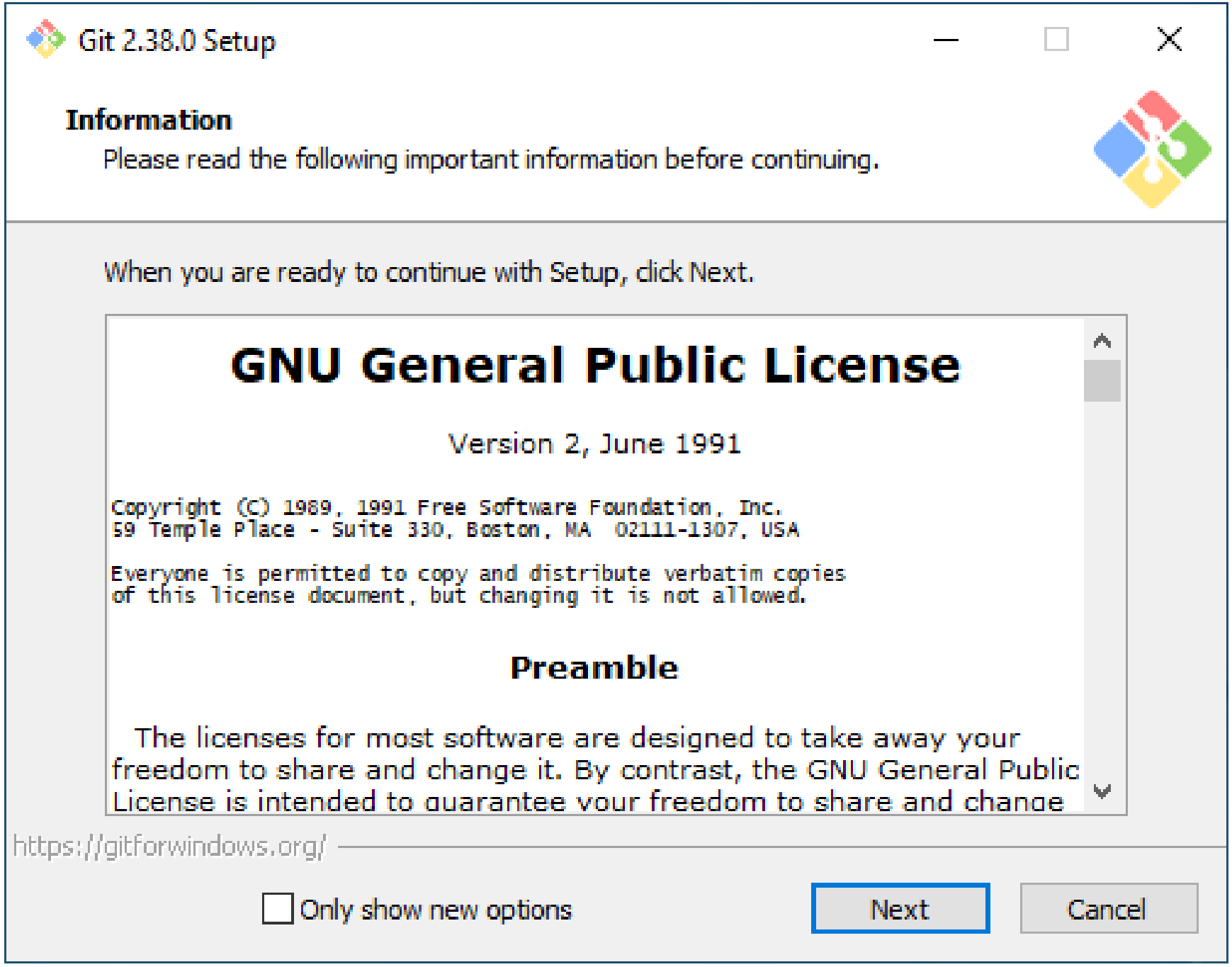
安裝步驟(以Windows為例)

1. 在搜尋引擎輸入關鍵字「git」
2. 連到git官方網站，注意不是GitHub(兩者不一樣!)
3. 點選Downloads按鈕，進入Download頁面
4. 根據對應的作業系統下載軟體



5. 下載完成後執行安裝檔
6. 按「下一步」到底，完成安裝

安裝步驟介紹(01)

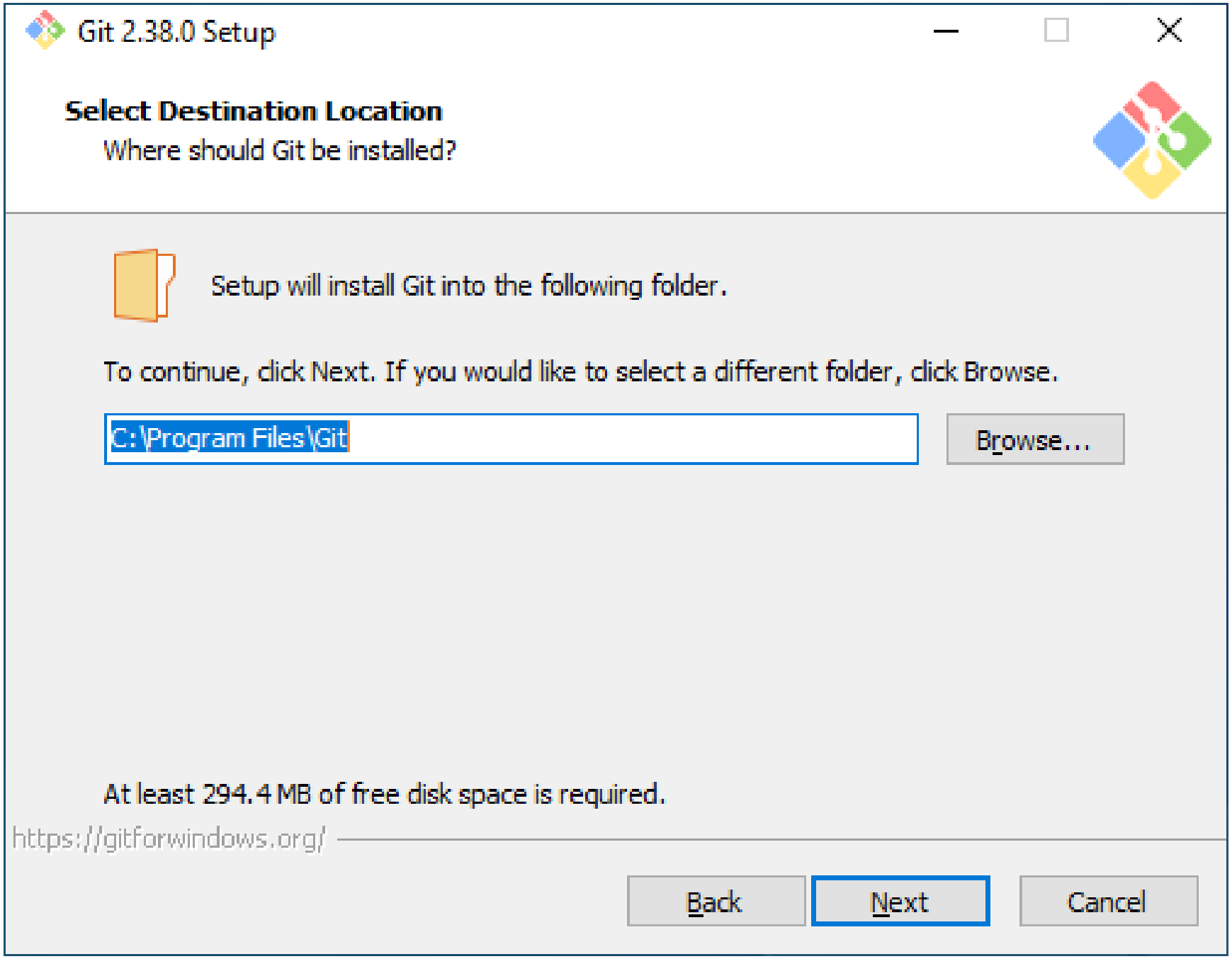


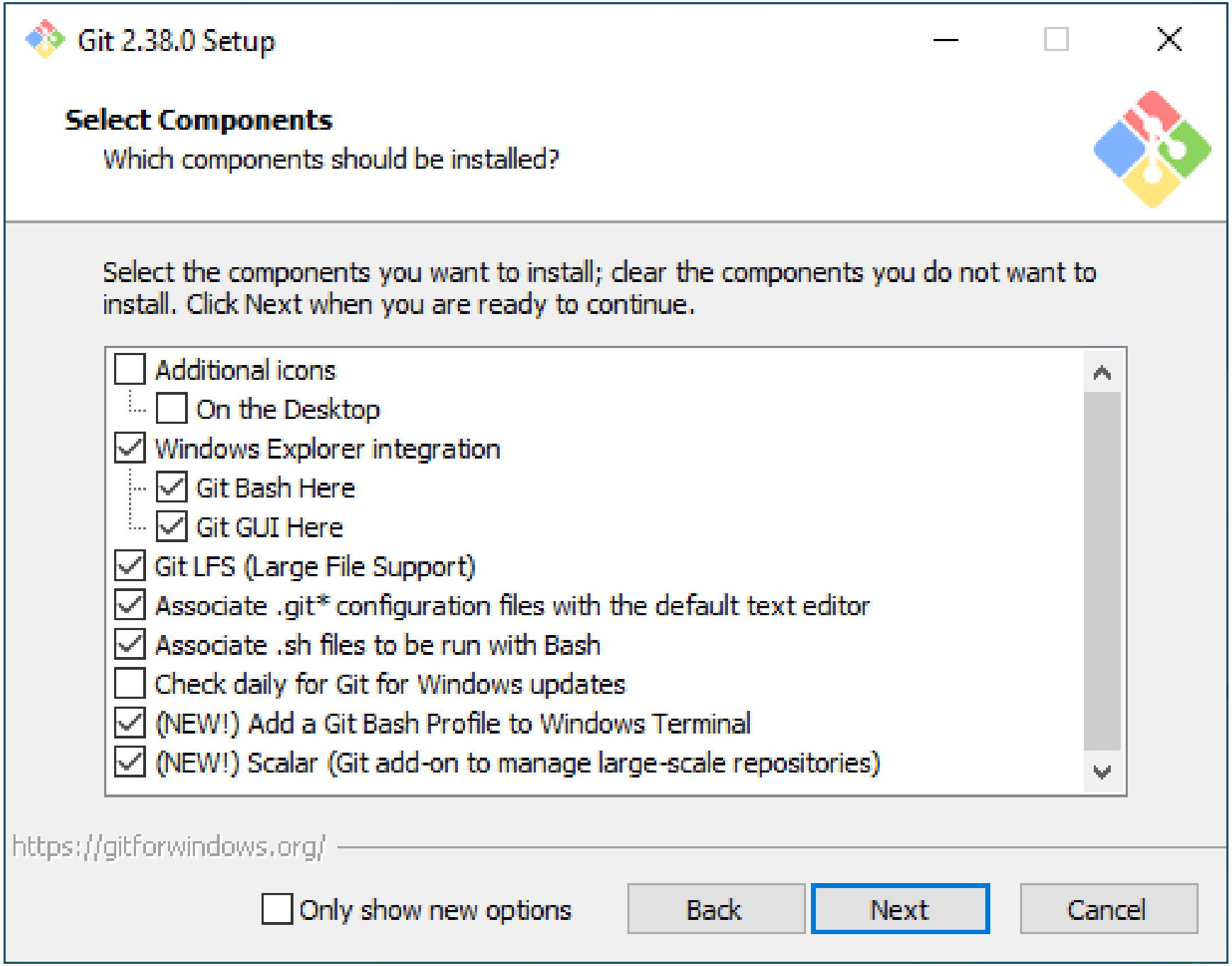
GNU授權條款

1. 此授權條款允許使用者自由使用、散佈、修改。
2. 對有此授權的軟體進行原始碼修改後，也須繼承此授權條款。
3. 點選下一步代表接受此授權條款。

安裝步驟介紹(02)

1. 選擇程式安裝路徑





安裝步驟介紹(03)

Additional icons
- 在桌面新增Git的圖示

Windows Explorer integration
- 在右鍵選單中加入Git選項

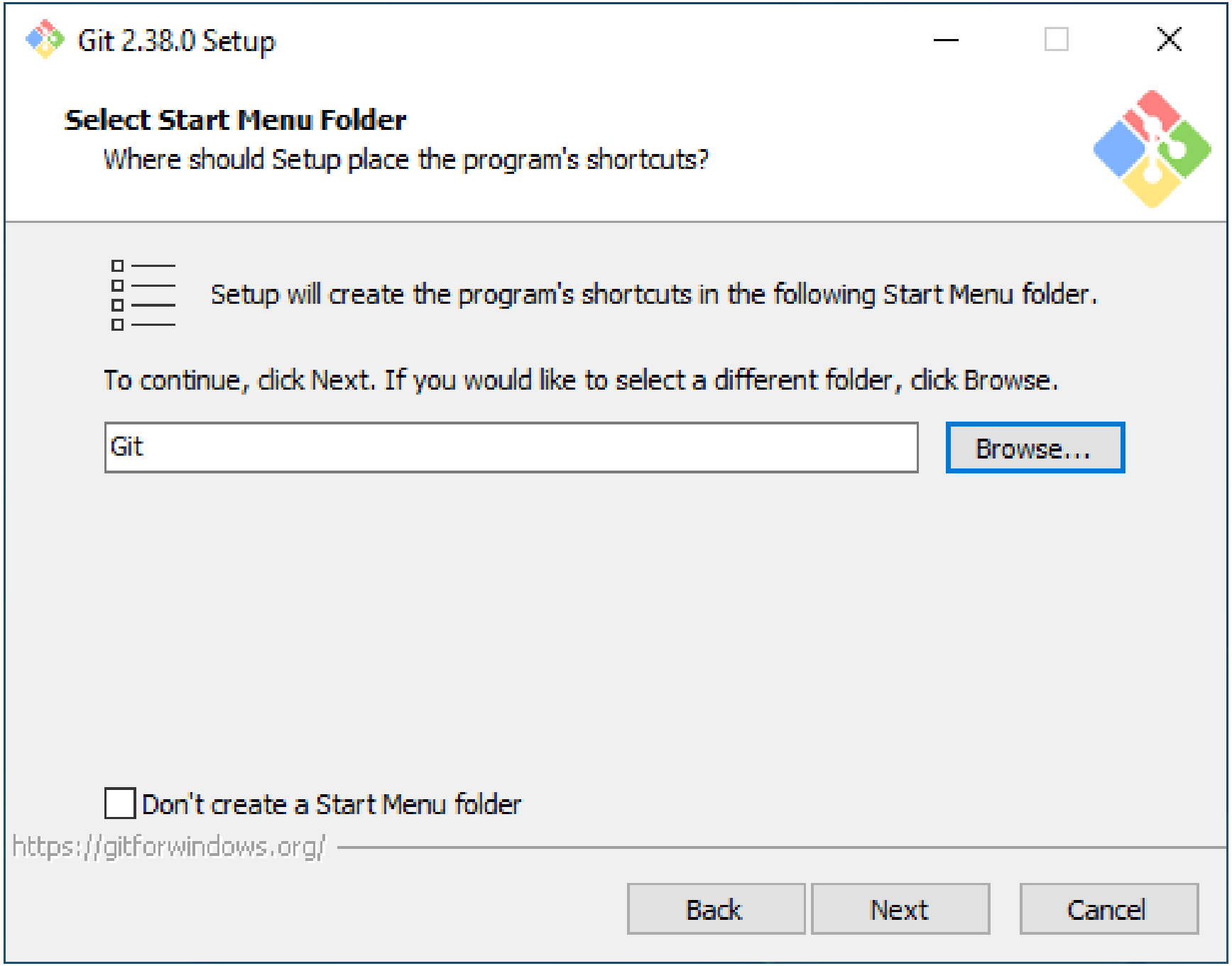
Git LFS(Large File Support)
- 支援控管大檔案(超過5GB)

Associate .git* …
Associate .sh files …
- 副檔名關聯

Check daily for Git …
- 每天檢查是否有針對windows的Git更新

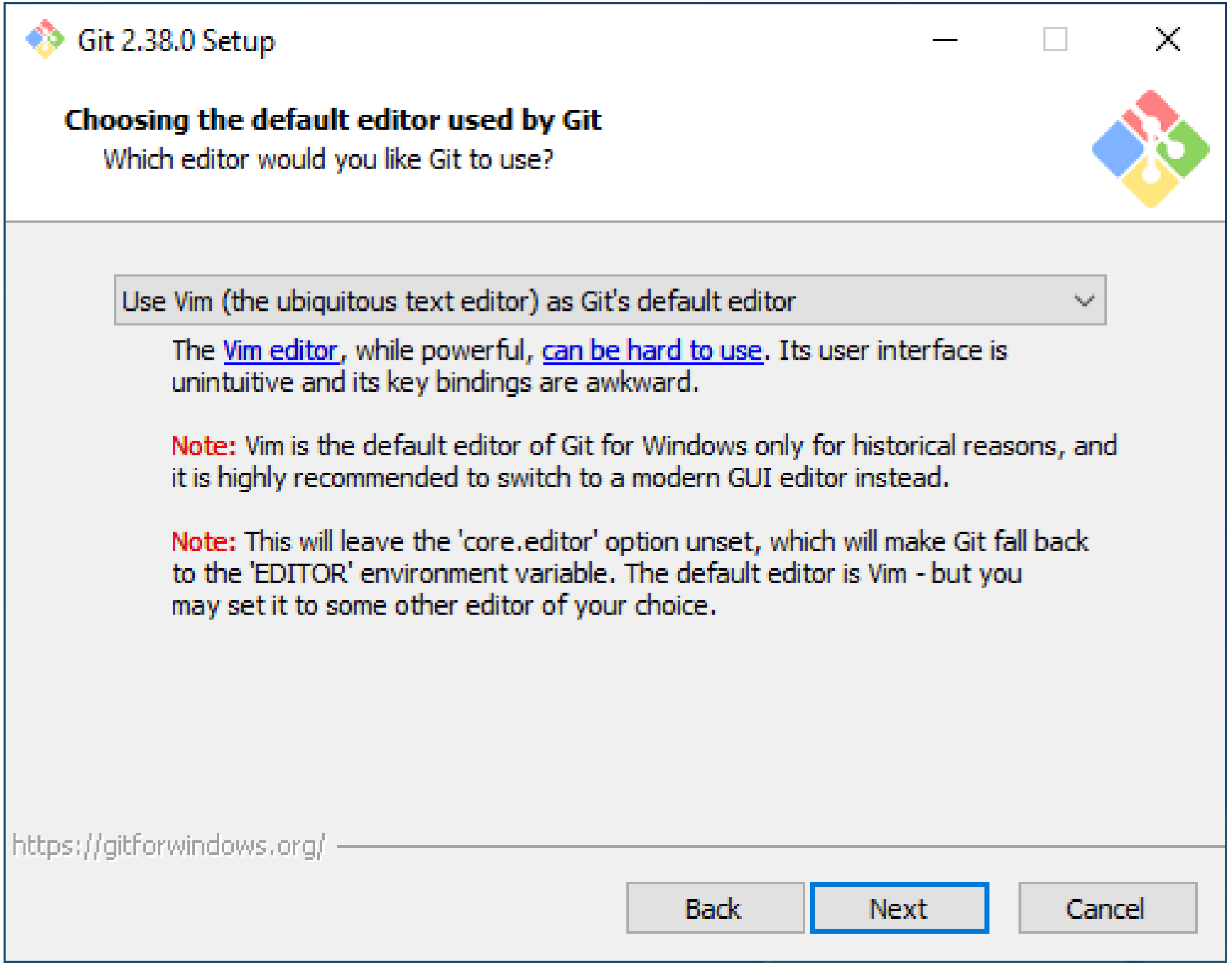
Add a Git Bash Profile …
- 將 Git Bash 的設定加入 Windows Terminal

Scalar(Git add-on to …)
- 提升git執行效率
- Windows Scalar



安裝步驟介紹(04)

1. 是否在windows程式集(開始選單)建立目錄

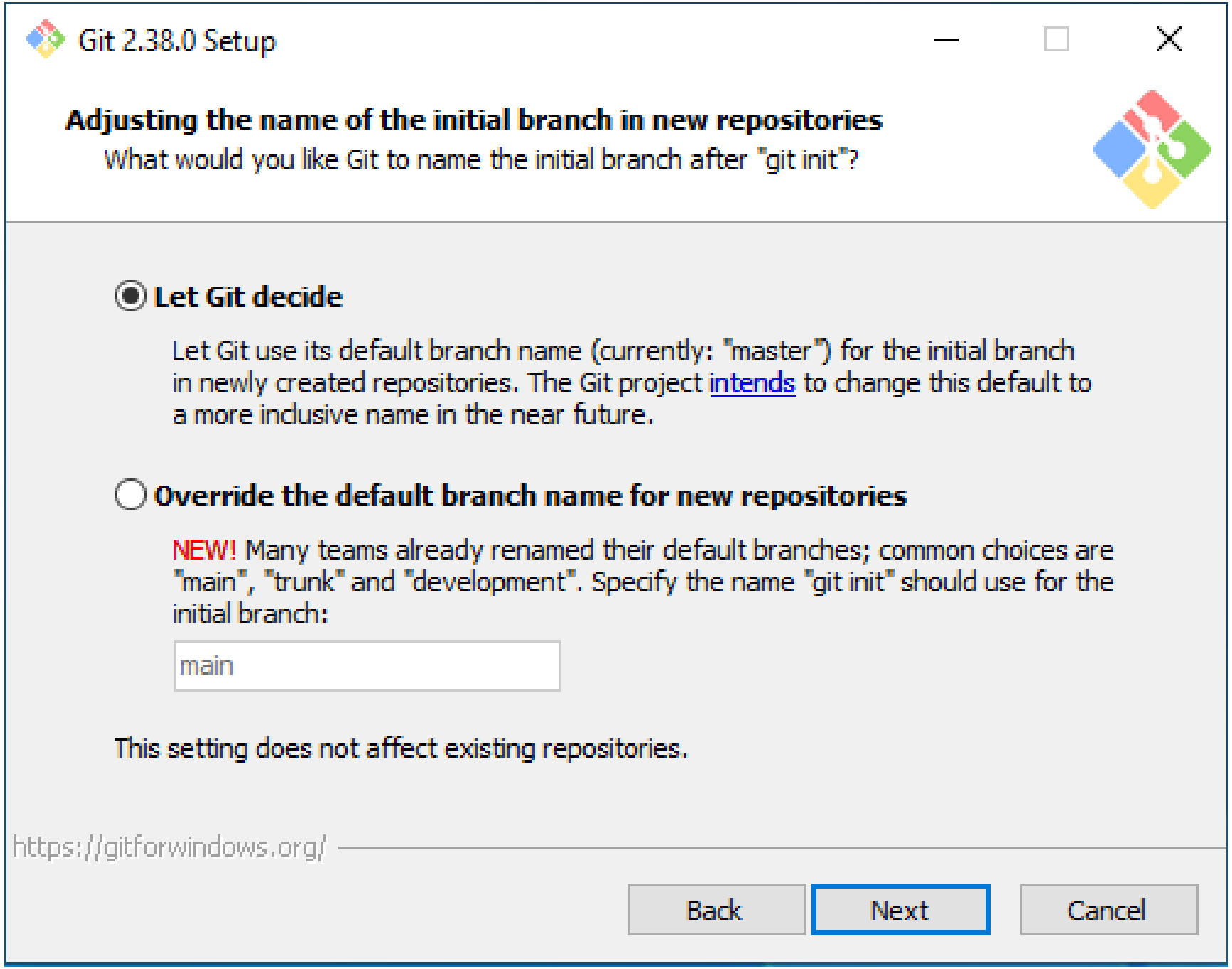


安裝步驟介紹(05)

選擇Bash預設的文字編輯器

1. Vim是上古時代的文字編輯器(但現在還是有部分支持者)，作為Git for Windows的預設編輯器是因為歷史原因。
2. Vim難以上手，但上手後效率很高；在以前，與之打擂臺的是emacs(詳見編輯器戰爭)
3. Vim操作請見附件，
4. 替換預設文字編輯器可使用以下指令：

```
git config --global core.editor
"'C:\Windows\System32\notepad.exe'"
```



安裝步驟介紹(06)

選擇主要分支的預設名稱

1. 主要分支通常為穩定的程式、正在運行中的專案等等。
2. 預設為「master」
3. 受2020年喬治·弗洛伊德事件引發的後續示威活動影響，部分專案將主要分支改成「main」。(因master有奴隸主的意思，被部分人認為是對黑人的不尊重)

master

noun [C]

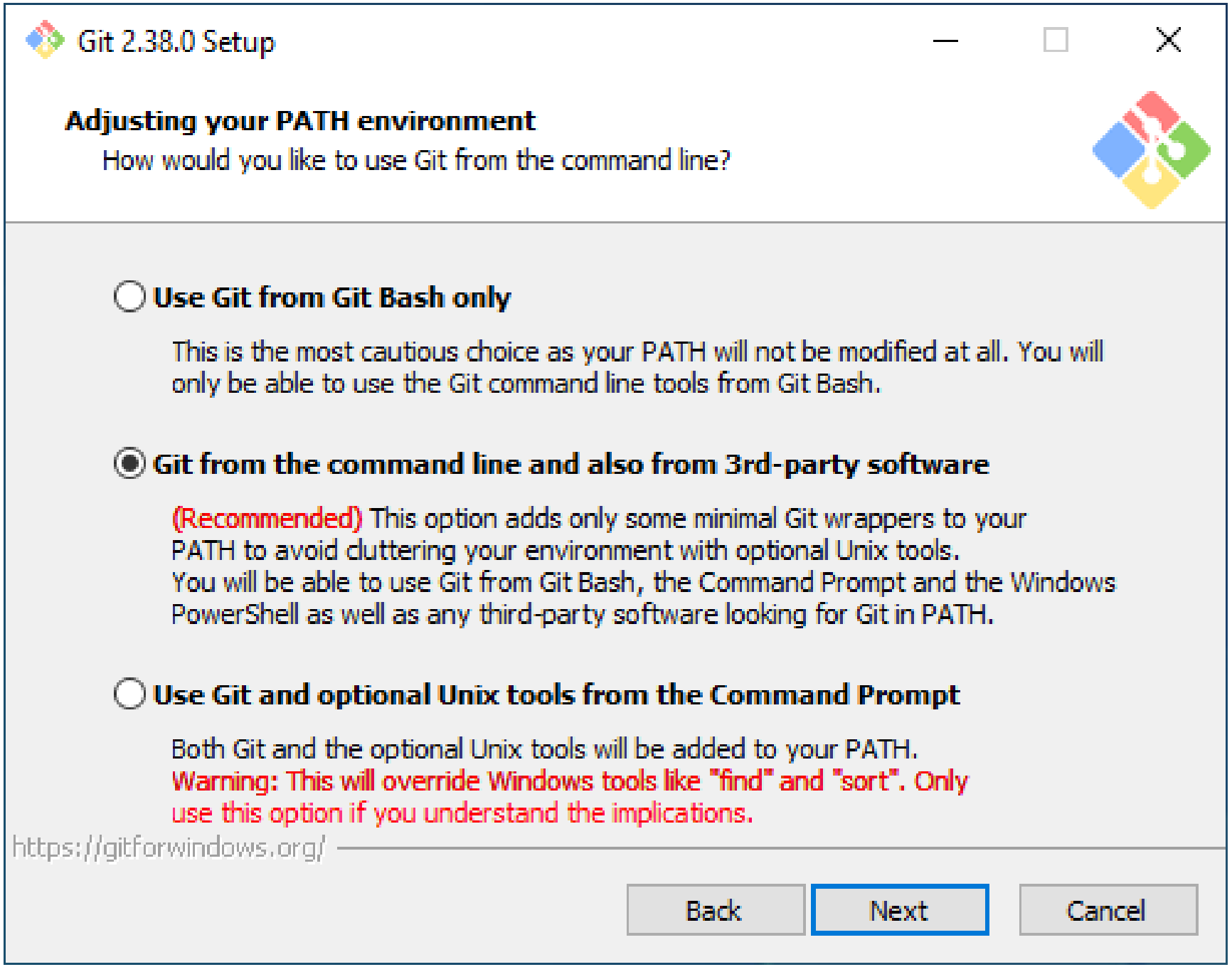
UK /ˈmɑː.stəʳ/ US /ˈmæs.tɚ/

master *noun* [C] (CONTROLLER)

a person who employs a servant or owns a slave

（僱傭人或擁有奴隸的）主人

- *Servants had to obey their masters.*
  傭人必須服從主人。



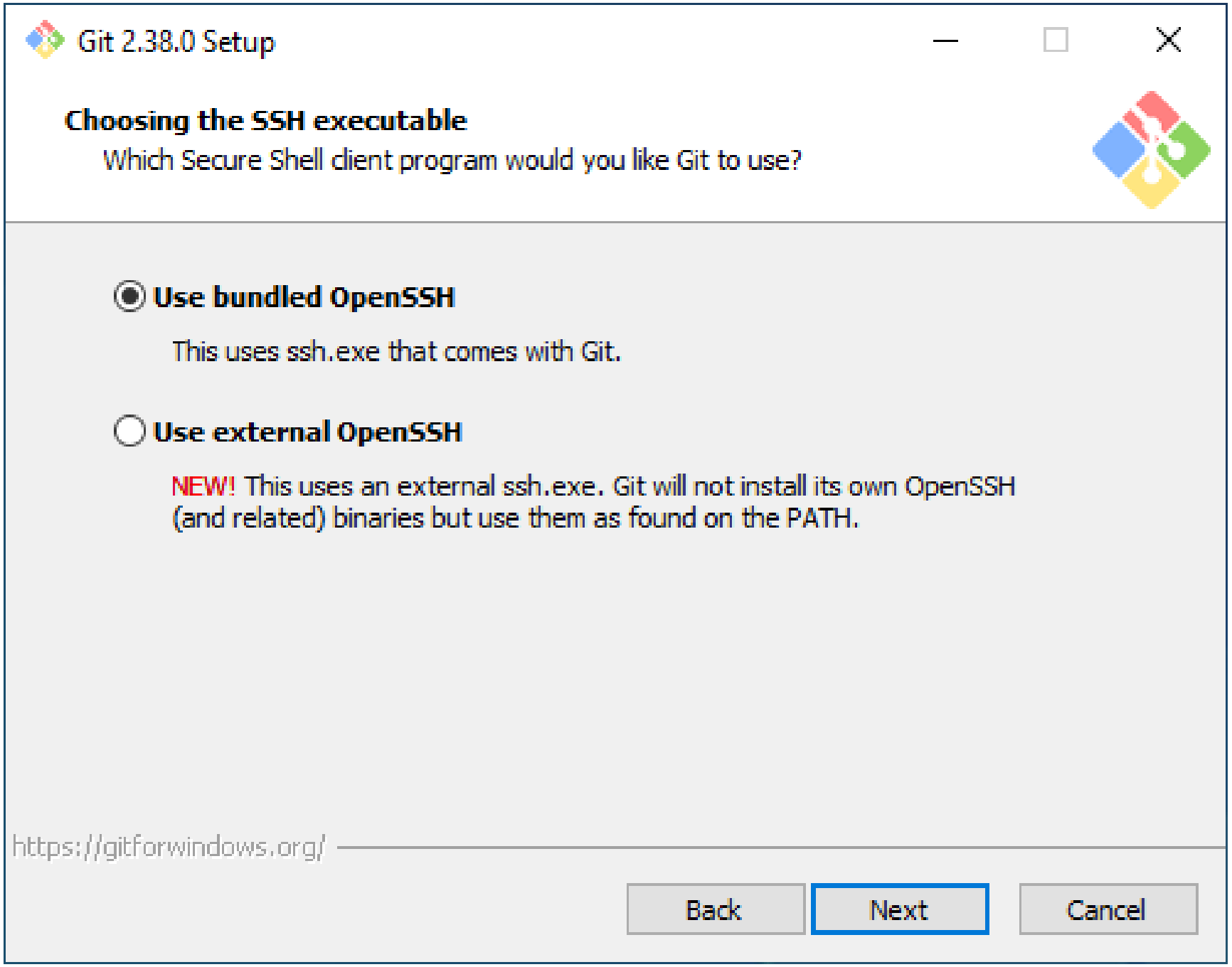
安裝步驟介紹(07)

調整環境變數

1. 不修改Windows環境變數；選擇此選項後只能用Git提供的終端(Terminal)軟體Git Bash，執行Git指令。

2. 將Git的相關參數加入Windows環境變數中；選擇此選項代表你能從Windows Command Prompt、Windows PowerShell、或其他第三方終端中執行Git指令，建議選項。

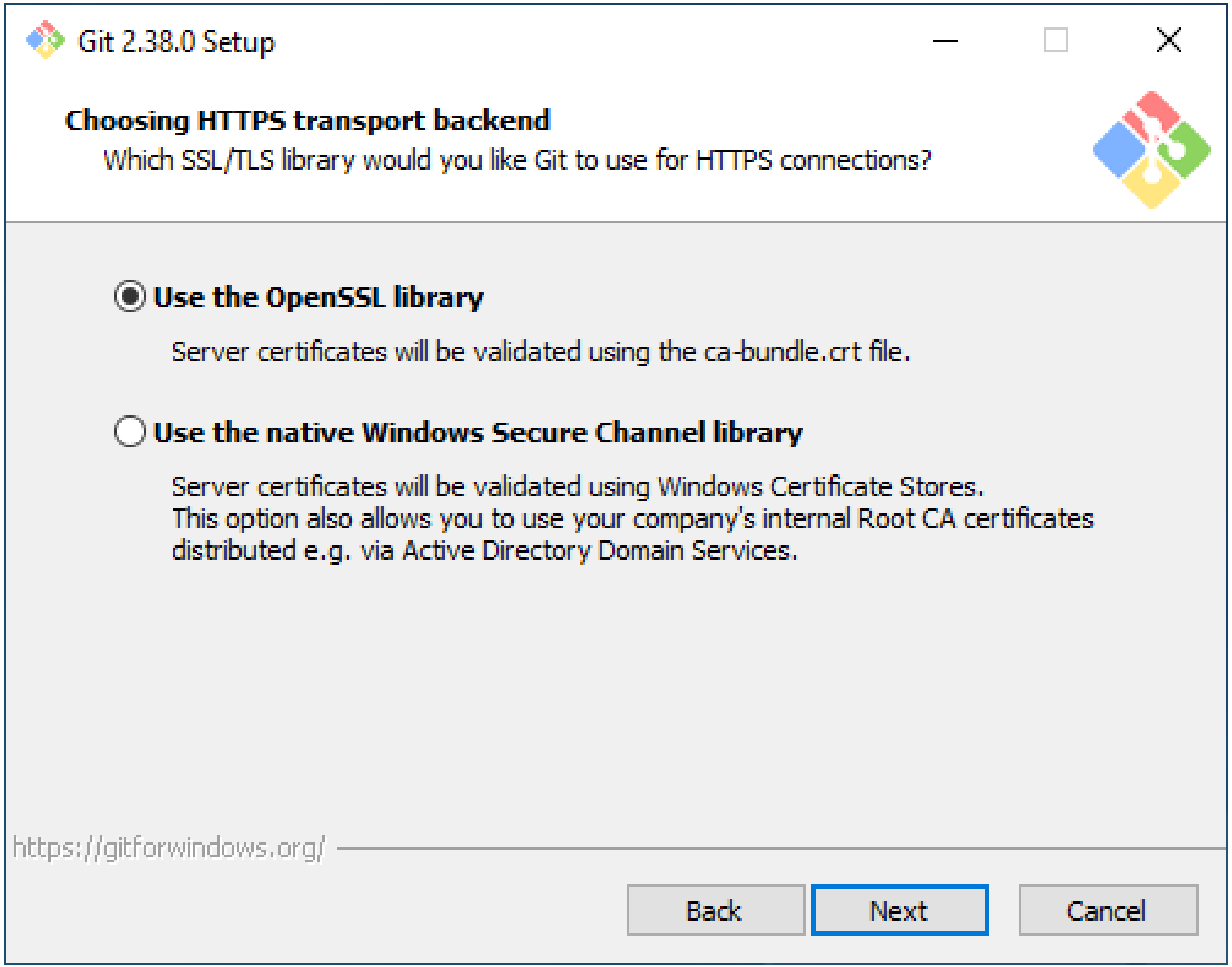
3. 覆蓋Windows部分的Command指令；選擇此選項會使Windows內建的find、sort等等指令失效，變成以Git指令為主。



安裝步驟介紹(08)

OpenSSH

1. 使用Git自帶的OpenSSH程式ssh.exe，建議選項。
2. 使用外部的OpenSSH，需自行處理OpenSSH的相關設定。
3. 沒特別需求的話選第一個。



安裝步驟介紹(09)

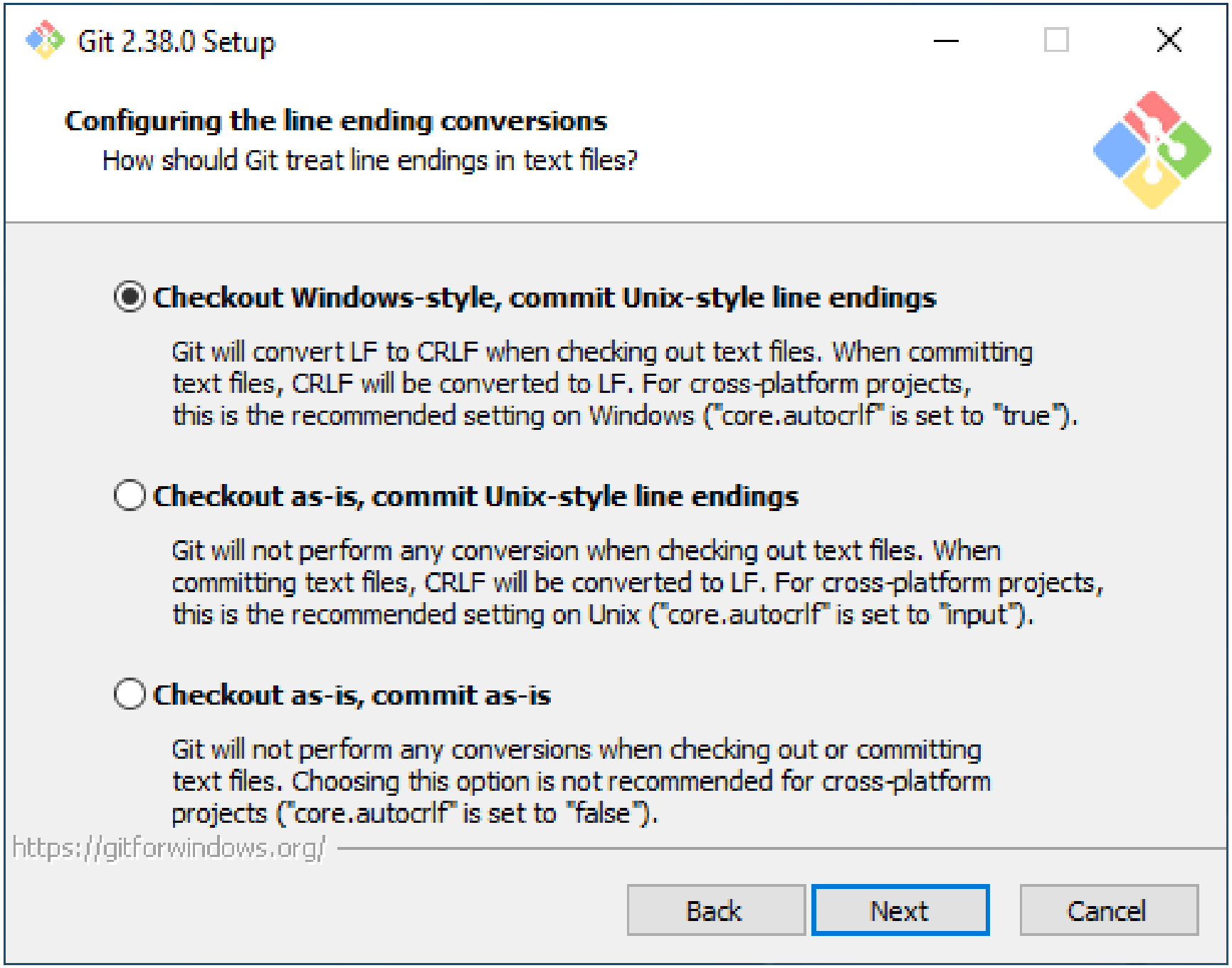
選擇Https傳輸設定

1. 使用OpenSSL函式庫。

2. 使用Windows本機安全通道函式庫，此選項也允許使用公司內部的CA證書。

3. 如果在具有企業管理憑證的組織中使用 Git，則需要使用安全通道。如果你僅使用 Git 來訪問公共存儲空間(例如 GitHub)，或者你的組織不管理自己的憑證，那麼使用 SSL 後端就可以了(它們只是同一協議的不同實現)。

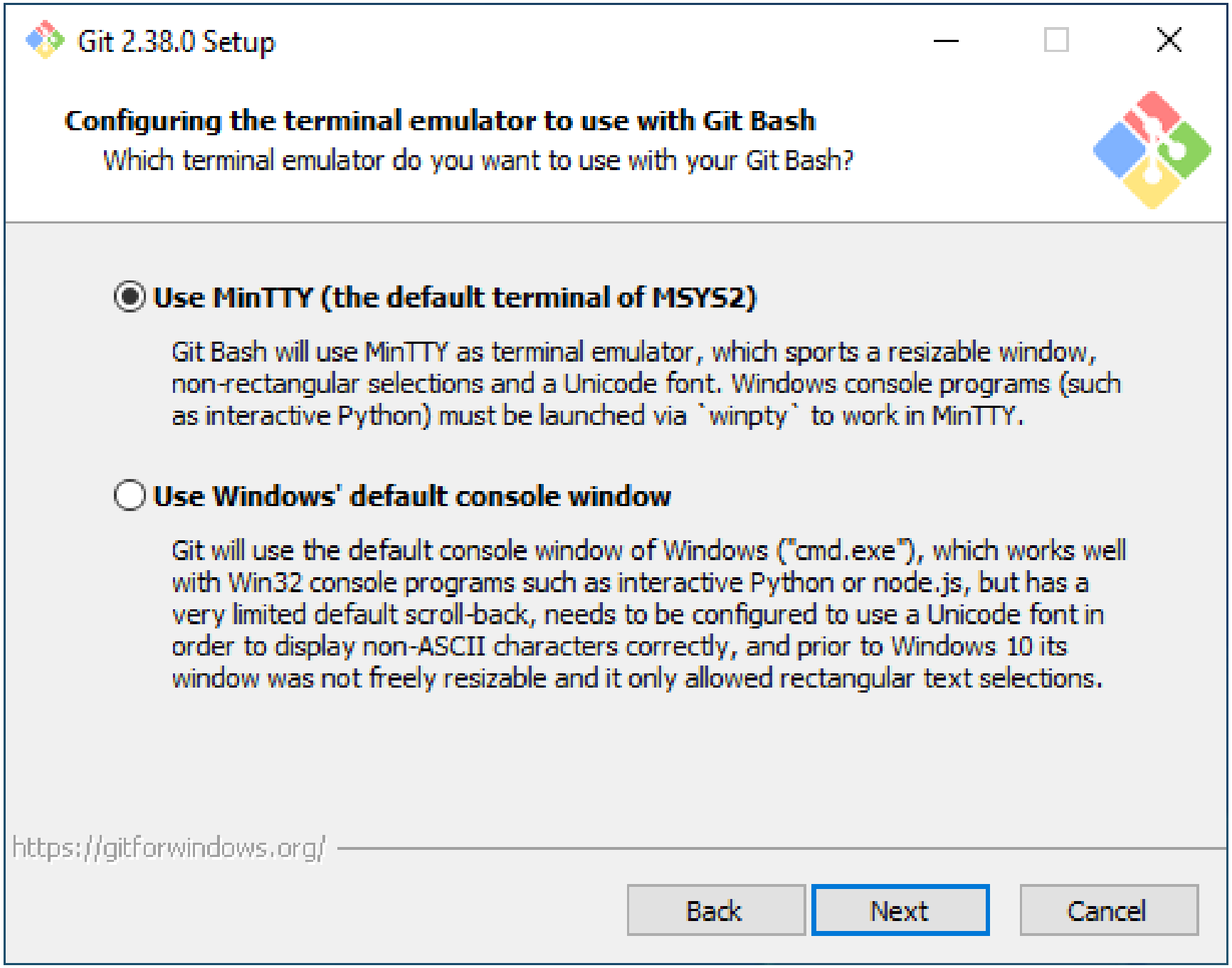
4. 沒特別需求的話選第一個。



安裝步驟介紹(10)

換行字元轉換

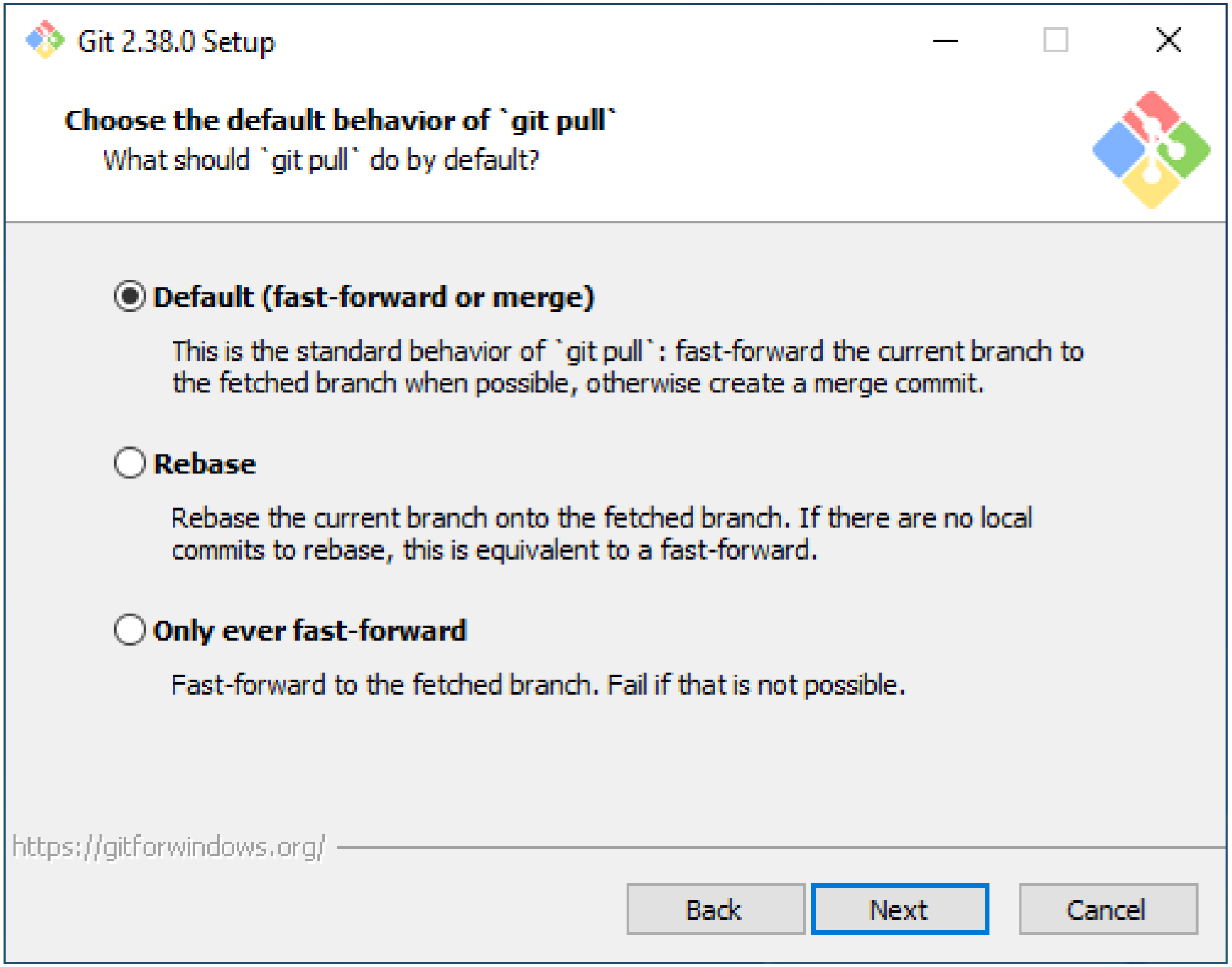
1. 執行Git checkout指令時，使用Windows格式結尾；提交(Commit)修改後，轉換成Unix格式結尾。此為Windows建議設定。
2. 執行Git checkout指令時，不做任何轉換動作；提交(Commit)修改後，轉換成Unix格式結尾。此為Unix(Linux)建議設定。
3. 執行Git checkout指令時與提交(Commit)修改後，皆不做任何動作。跨平台專案不建議使用。
4. Windows結尾格式為CRLF，用"\r\n"表示。
5. Unix(Linux)結尾格式為LF，用"\n"表示。
6. GitHub上是以LF換行。



安裝步驟介紹(11)

Git Bash預設終端

1. 使用MinTTY為Git Bash的預設終端；MinTTY是一種Windows的終端模擬，主要為模擬Unix(Linux)的終端。Git Bash在MinTTY上排版較好、功能也較多。
2. 使用Windows預設的終端(cmd)；歷史紀錄(中鍵回滾)非常有限，預設字元編碼也非UTF-8，須額外進行調整。Windows10之前，cmd甚至無法隨意調整視窗大小。



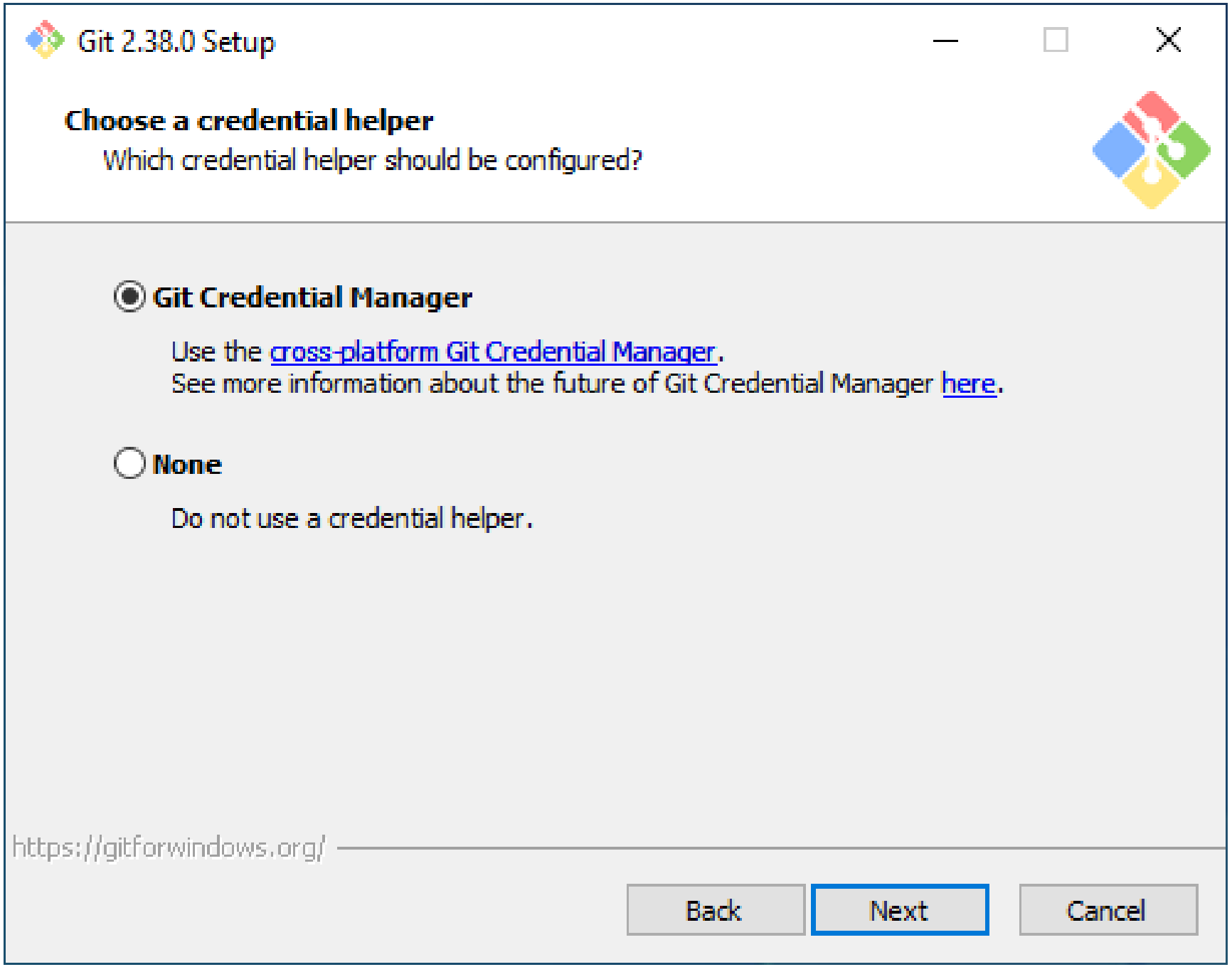
安裝步驟介紹(12)

Git pull預設方法

1. 使用merge做為拉下repository之後的預設動作。可以理解成：git pull = fetch + merge
2. 使用rebase做為拉下repository之後的預設動作。可以理解成：git pull = fetch + rebase
3. 取得repository後不進行動作。推測是git pull = fetch

Merge vs Rebase

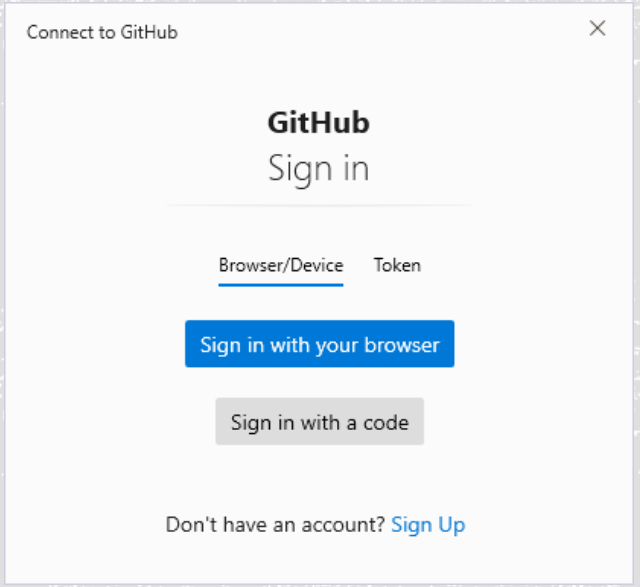
1. 此兩種合併方法主要差在「版本歷史紀錄」上。
2. Merge會保留主分支修改前的歷史記錄，比較完整，但會讓歷史記錄變得比較複雜。
3. Rebase會用新分支的紀錄取代主分支的紀錄，紀錄會有所簡化，但使用不當會造成嚴重災難。

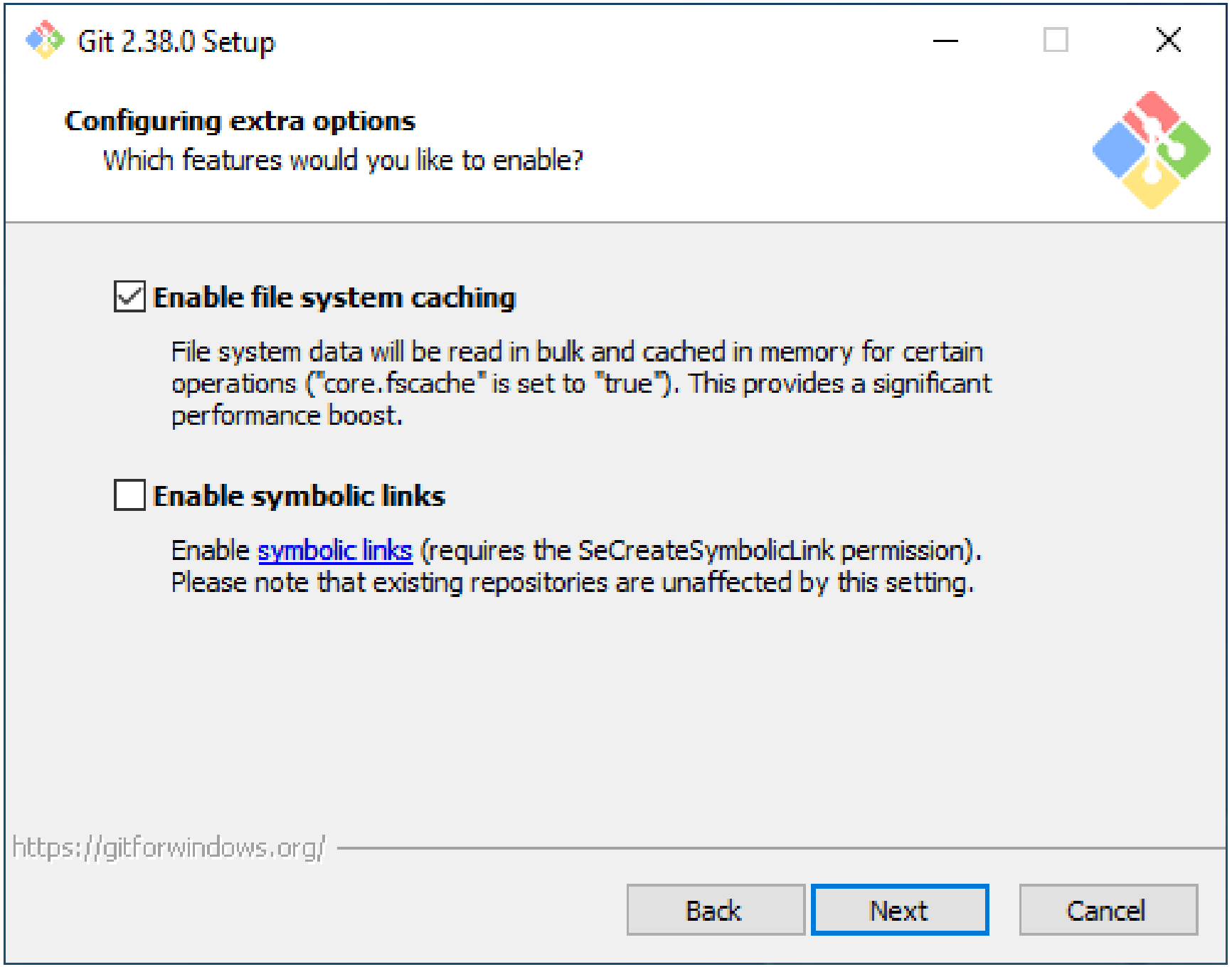


安裝步驟介紹(13)

憑證助手

1. 使用跨平台的Git憑證助手；當有登入需求(如：推送Repository到GitHub時，需輸入帳號密碼)，會跳出跨平台的登入視窗讓你輸入相關資訊。(如下圖)
2. 不使用憑證助手，登入資訊需額外設定。

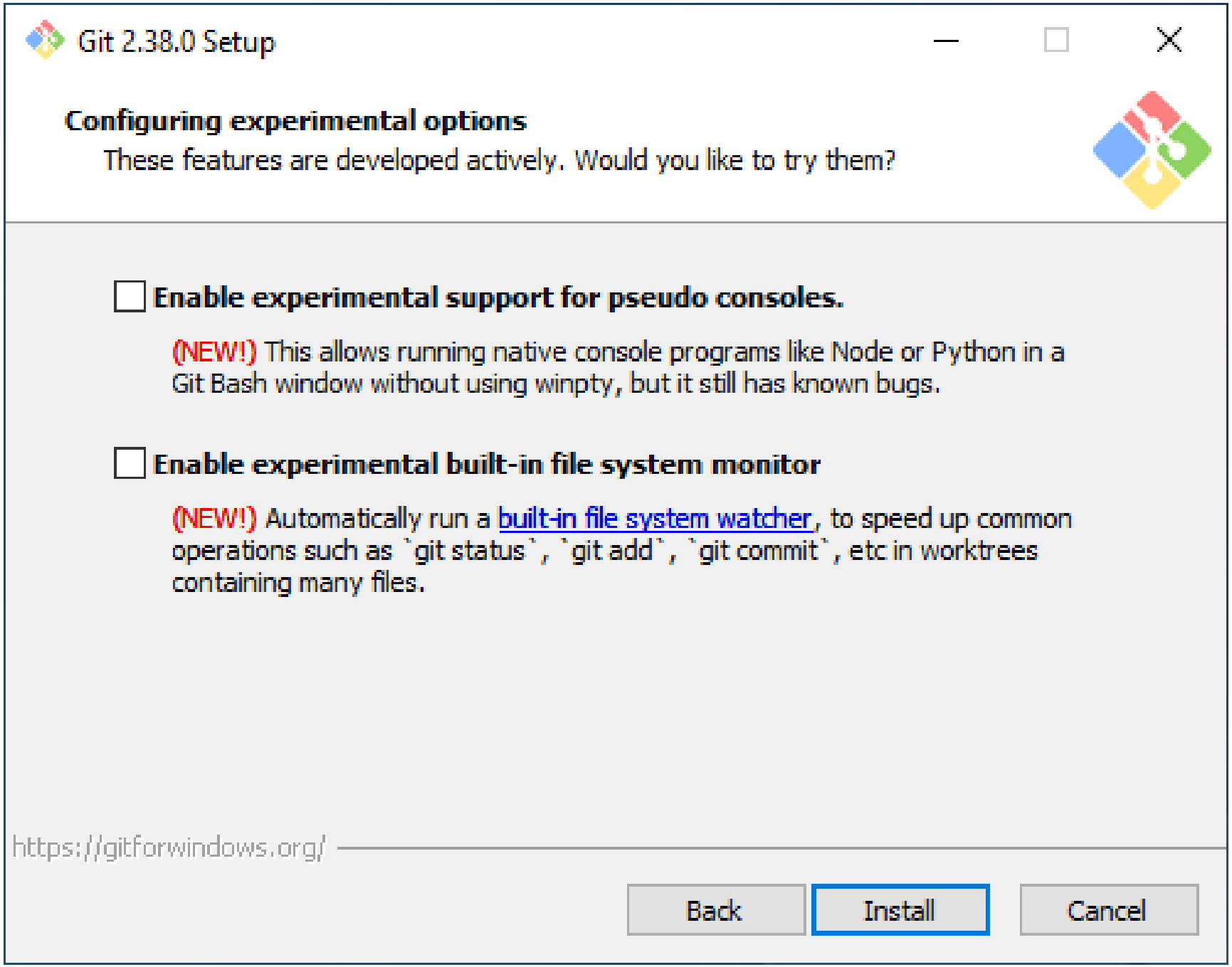




安裝步驟介紹(14)

額外設定選項

1. 啟用檔案系統快取，啟用後可顯著提升效能，預設啟用。
2. 啟用符號連結，類似於Windows的創建捷徑，啟用此功能需要額外的權限，預設不啟用。



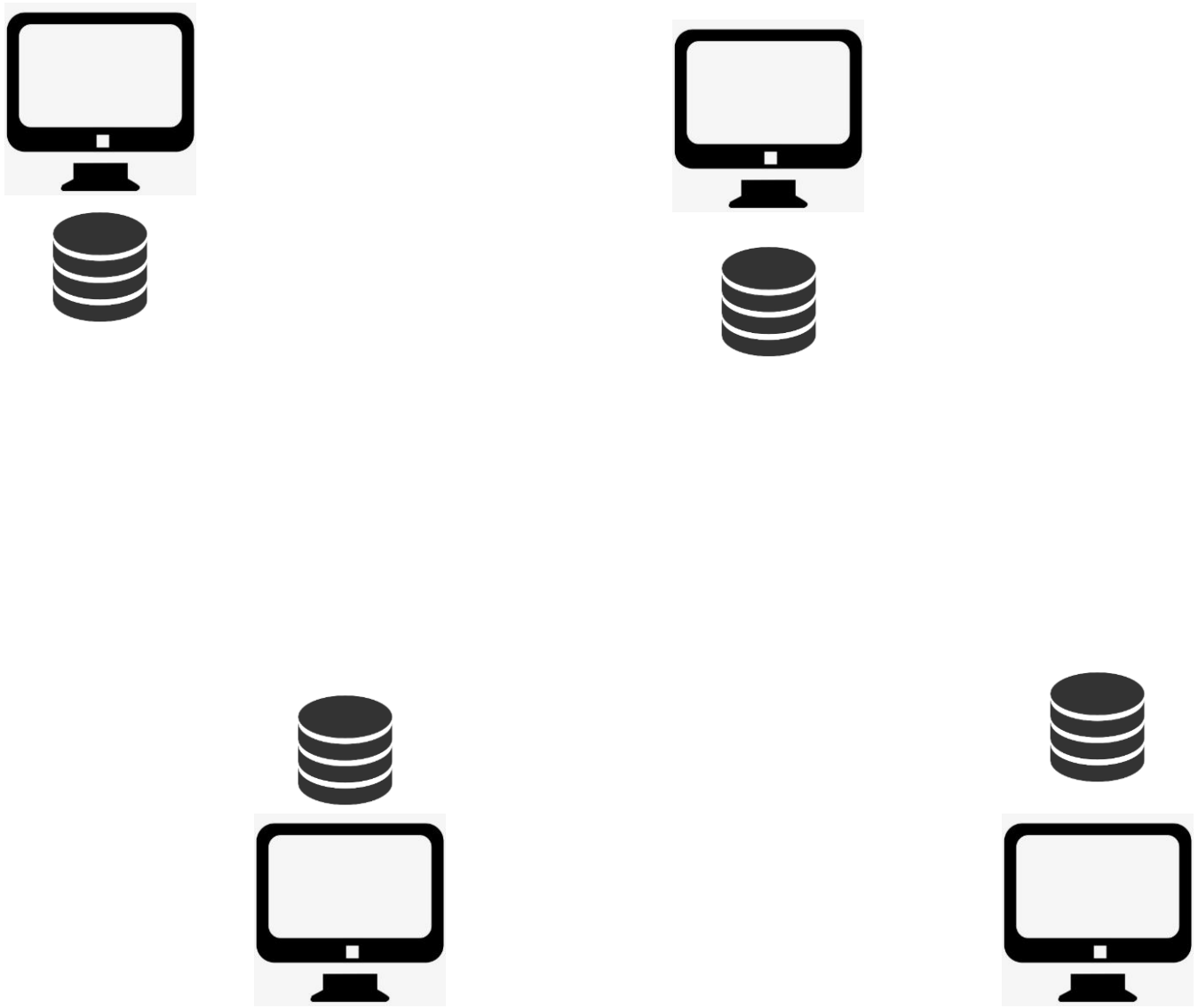
安裝步驟介紹(15)

實驗功能選項

1. 啟用對偽控制台的實驗性支持。允許在不使用winpty的情況下，在Git Bash視窗中進行Node或Python等的終端程式。但此功能有存在的已知Bugs。
2. 啟用實驗性的內部文件系統監視器，此選項可加快常規指令如：git status、git add等。
3. 以上都是實驗性功能，可能會有部分bug與錯誤，建議皆不開啟。

第三節
GIT基本觀念

分散式架構



REPOSITORY(儲存庫)

1. 儲存控管資料的地方(所有版本)。
2. 在集中式版控軟體的架構下，只有Server會擁有一個Repository，所有Client都是連入同一個儲存庫做修改。
3. 在分散式版控軟體的架構下，每一個Client都會有一個Local Repository，此外會從所有的Local Repository選擇一個作為Remote Repository(都常會架設在Server上)。
4. 對於分散式的架構而言，所有的Repository都所儲存完整的版本資訊。
5. 一個專案使用一個Repository。

(Modified)A.txt

(New)B.txt

commit

Repository

COMMIT(提交)

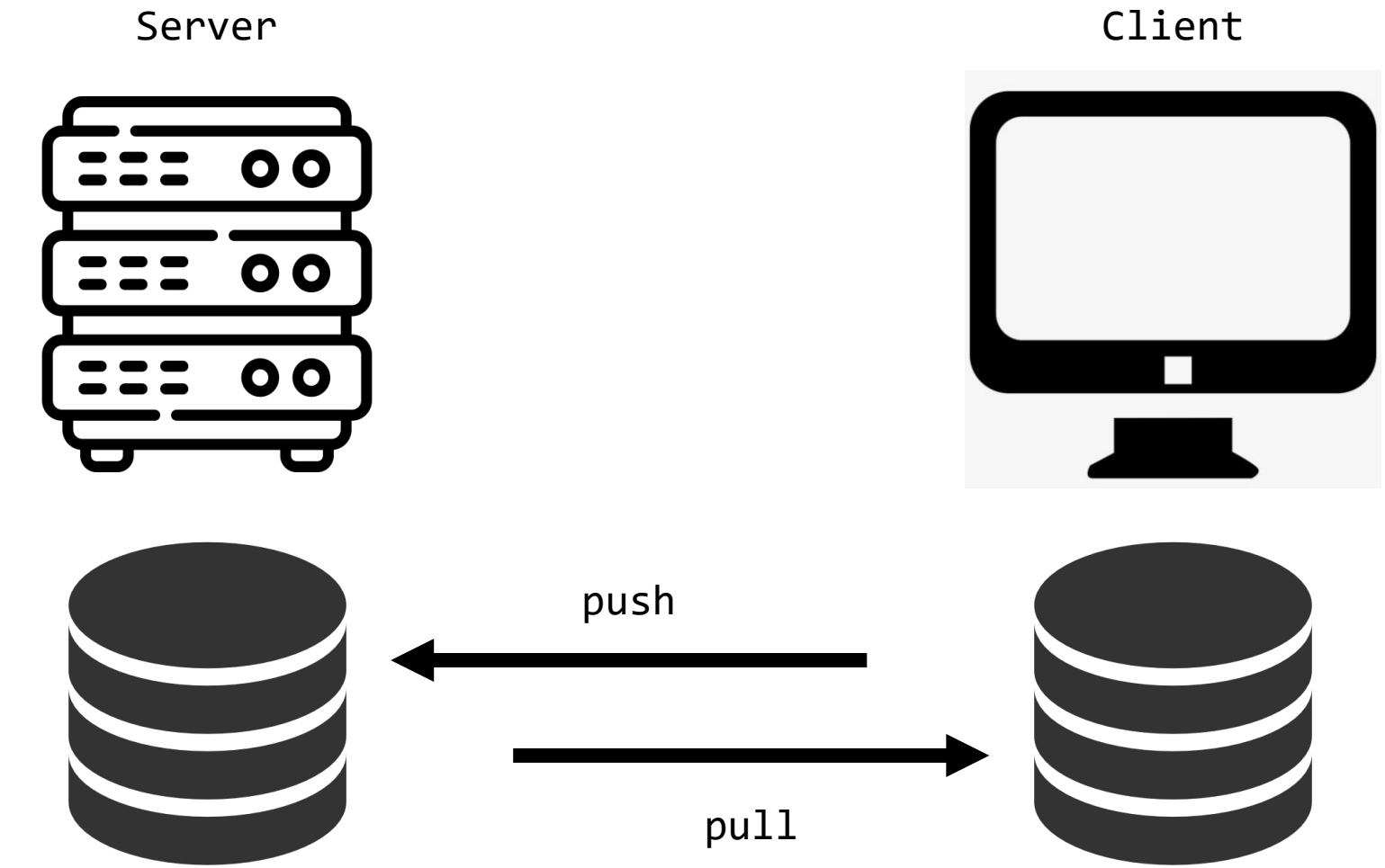
1. 將新建或修改的文件，儲存到Repository的動作，稱之為Commit(提交)。
2. Commit時，需要完成兩件事情：告訴Git你是誰、撰寫提交訊息
3. 提交訊息沒有強迫規定格式，隨意撰寫不會有任何影響，但建議撰寫有意義的訊息，以利之後的維護工作(可參考此連結)。
4. 建議格式

主旨

詳細資訊

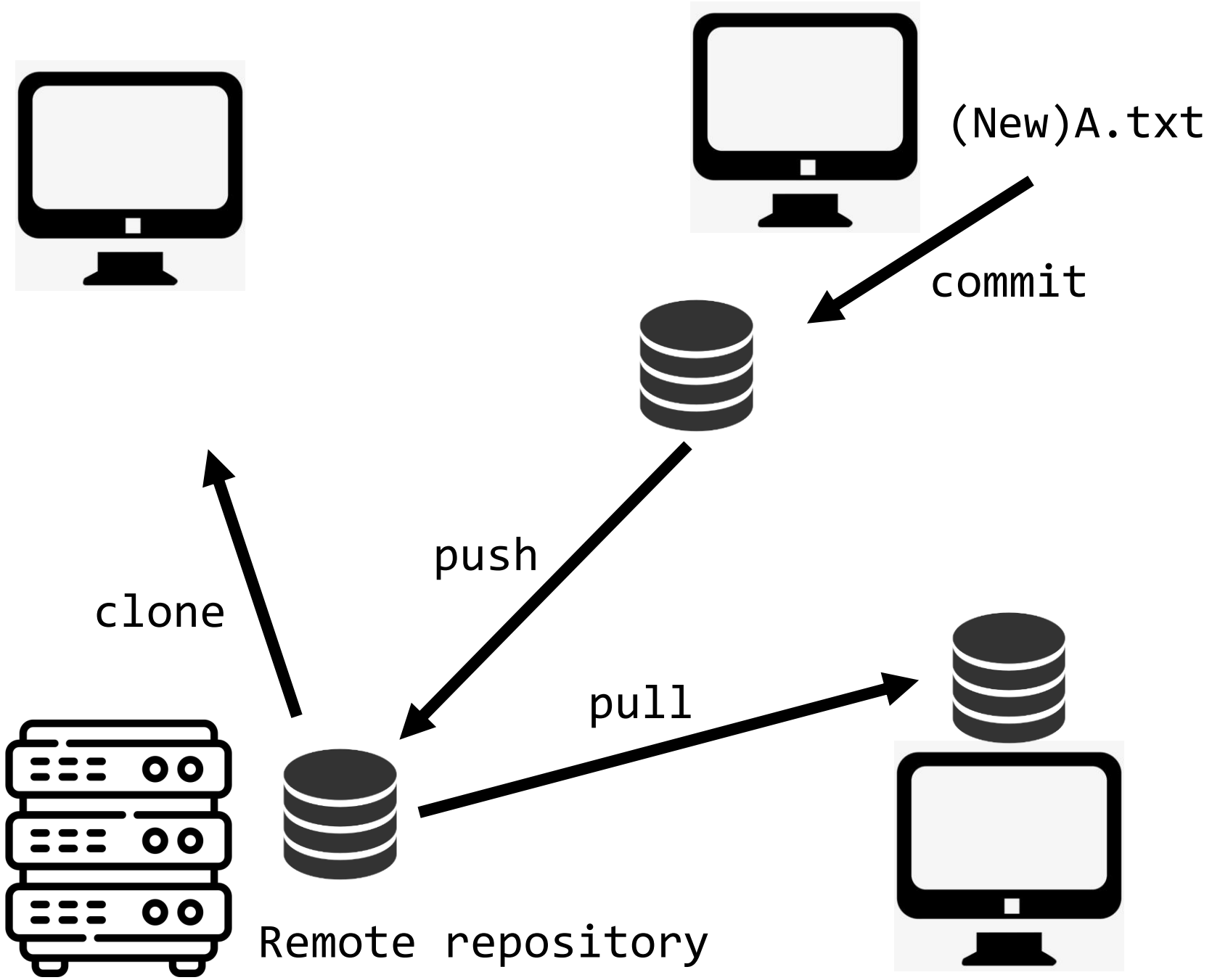
(事件編號)

PUSH、PULL



1. 將修改好的Repository上傳到 Remote Repository(遠端儲存庫) 的動作，稱之為Push。
2. 反之，將Repository下載的動作稱為Pull。

基礎流程



第四節
GIT基礎指令

```
git config --global user.email "Test@gmail.com"
```
設定提交者的信箱。

```
git config --global user.name "Allen"
```
設定提交者的名稱。

```
git config --list
```
檢查所有設定檔的參數。

```
git config --global core.editor "'C:\Windows\System32\notepad.exe'"
```
將預設文字編輯器換成成windows notepad 。

```
git config --global alias.st status
```
將指令「git status」簡寫為「git st」

```
git config --global alias.ls 'log --pretty=oneline'
```
將指令「git log --pretty=oneline」簡寫為「git ls」

設定後，global的資訊會記錄在
【Windows】
C:\Users\(你的使用者名稱)\.gitconfig

【Linux】
~/.config/git/config 或者 ~/.gitconfig

git config

1. 修改Git的相關設定。
2. 附加參數不同的參數會修改不同的設定：

【--system】

修改此作業系統內的所有使用者和使用者儲存庫的預設設定。

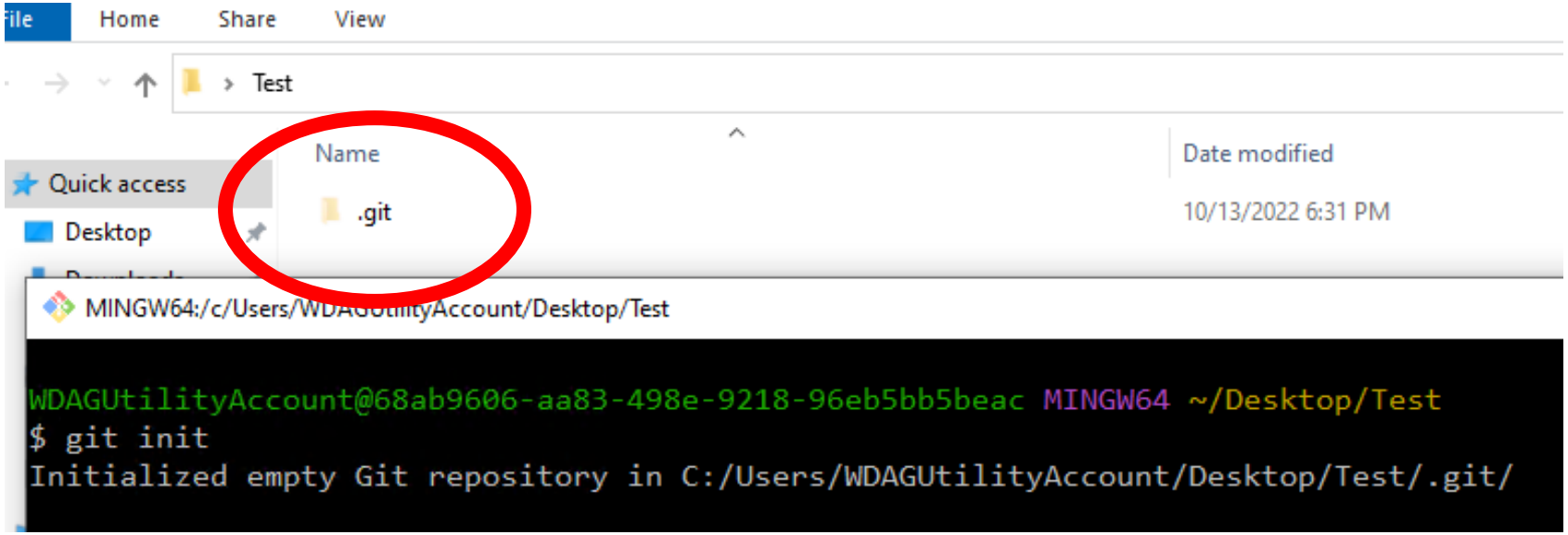
【--global】

修改登入此作業系統的使用者的相關設定。

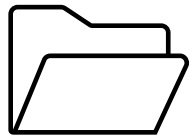
git init

1. 在當前的路徑建立起儲存庫。
2. 此指令會產生一個.git的隱藏資料夾，裡面會記錄此專案的所有版本紀錄與其他設定，此資料夾即為一個Repository。

git init
創建一個空的Git儲存庫或是重新初始化已存在的儲存庫。



追蹤的檔案
(Tracked files)

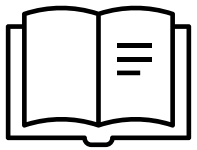


Directory(目錄)

未追蹤的檔案
(Untracked files)

git add

暫存檔案的不同狀態
modified
new file
deleted

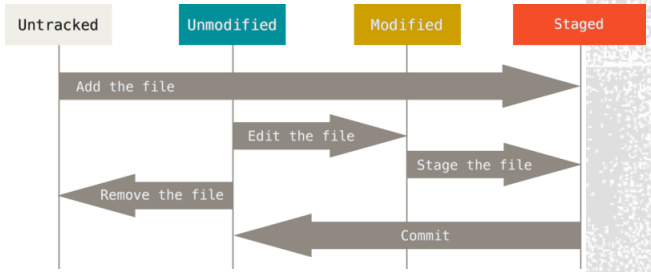


Staged(暫存區)

追蹤的檔案
(Tracked files)

git commit

Repository(儲存庫 .git)



git status

1. 顯示當前工作目錄的狀態。
2. 檔案分成兩種狀態：

【追蹤】

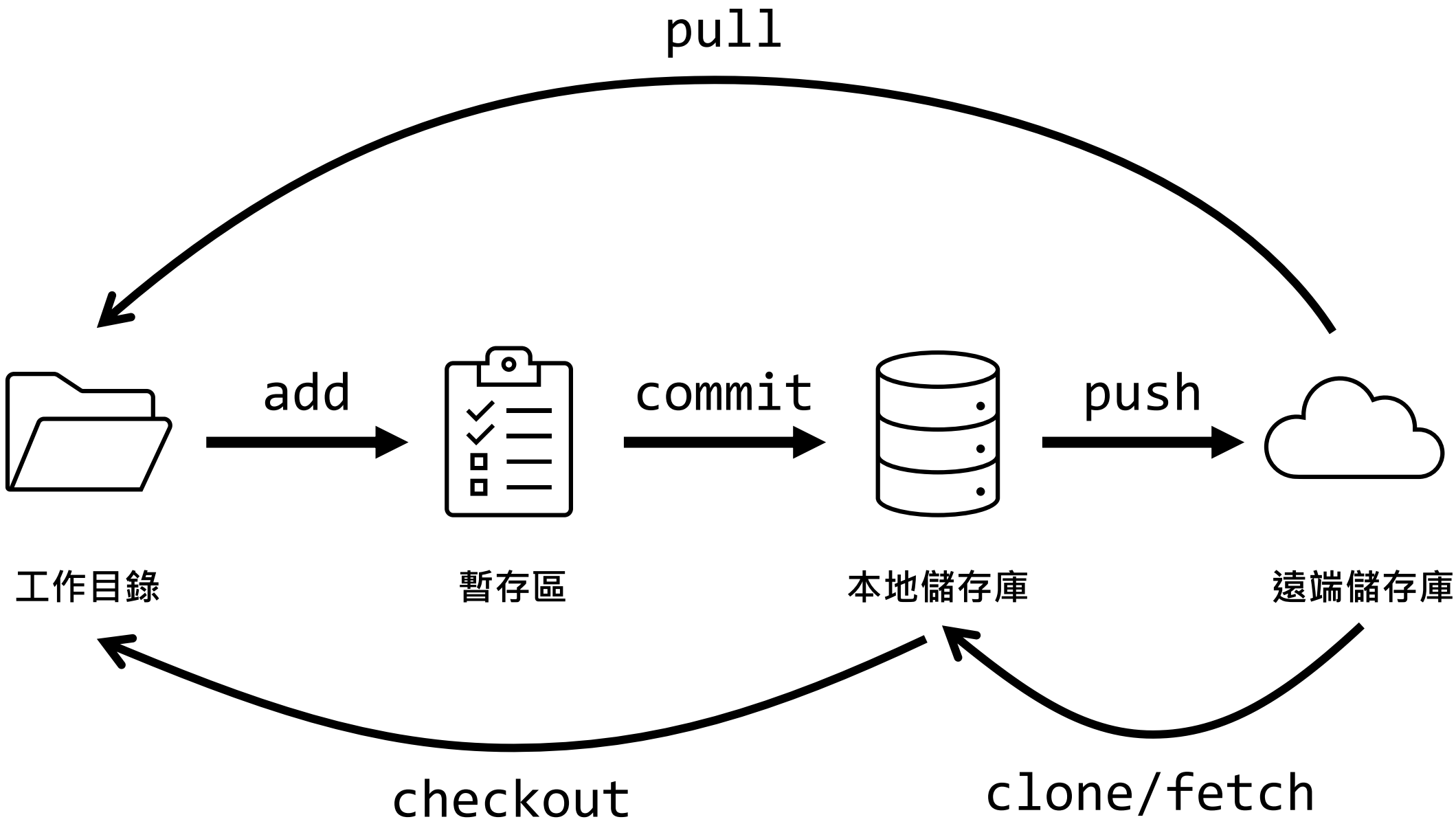
Modified

New file

Deleted

【未追蹤】

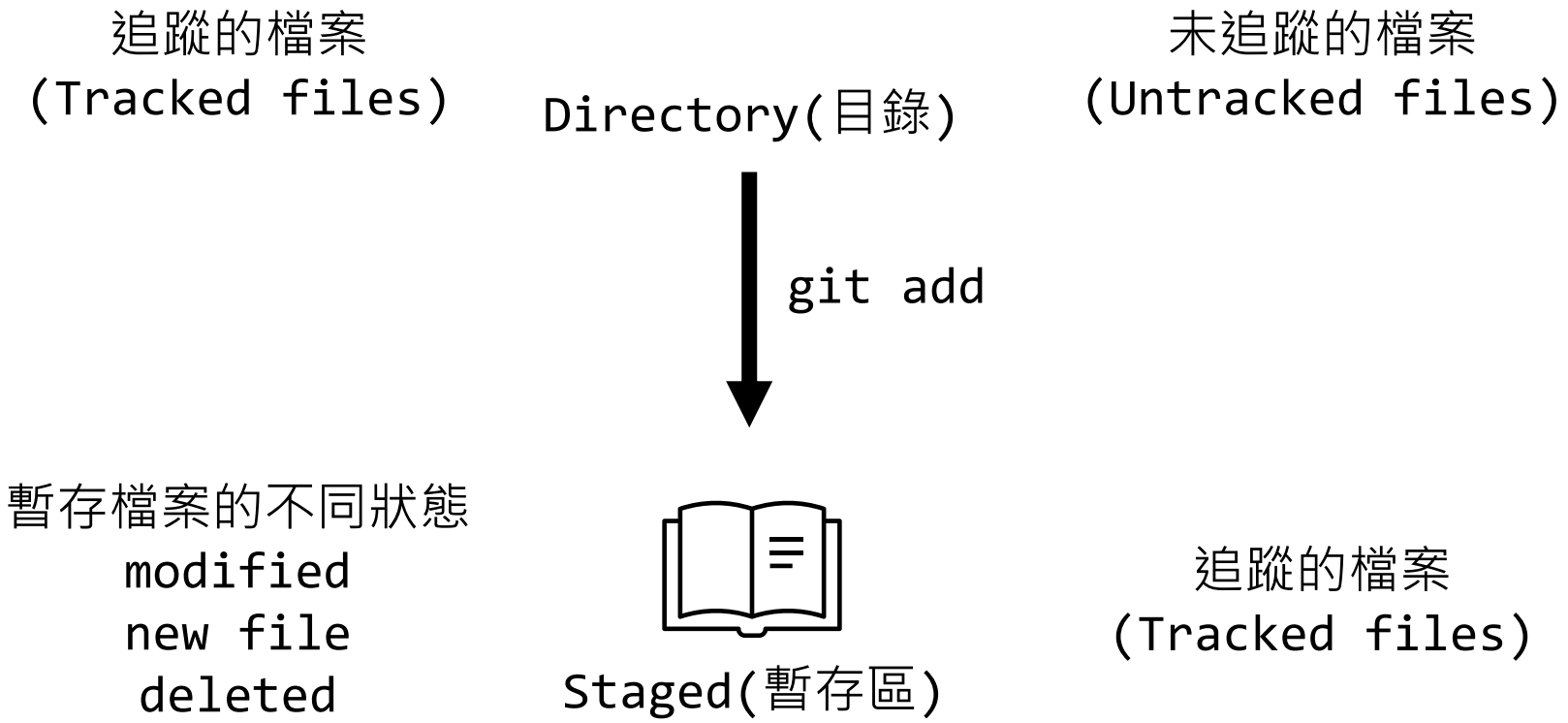
3. 可輸入git status -s 取得簡化的狀態。


pull
add
commit
push
工作目錄
暫存區
本地儲存庫
遠端儲存庫
checkout
clone/fetch

`git add "檔案名稱" / git add -- "檔案名稱"`
將單一檔案加入暫存區。

`git add *`
將此目錄下的所有檔案加入暫存區。

`git add .`
將此目錄下的所有檔案加入暫存區。



git add

1. 將檔案加入暫存區(Staged)
2. 對於【未追蹤】的檔案，會將其改成【已追蹤】，並加入暫存區。
3. 對於【已追蹤】的檔案，則將檔案變化加入暫存區。

`git commit -m "提交標題"`
使用輸入的標題，提交此次修改。

`git commit -am "提交標題"`
自動將所有「已追蹤的檔案」加入(add)暫存區，並且提交。

`git commit`
打開設定好的文字編輯器，待編寫提交訊息結束後，進行提交(推薦方法)。

commit應當有一定的規範(或上網google:「git commit規範」)，請記住大原則

「所有的軟體專案，都是由至少兩個以上的開發者所共同合作開發的」

以最小的專案來說，只有你，跟幾個月後的你自己。
推薦撰寫格式：

主旨

詳細資訊

(事件編號)

```
commit fb70d3129528a321d017b49052f9452181043e98 (HEAD -> master)
Author: Allen <Test@gmail.com>
Date:   Mon Oct 17 20:30:37 2022 +0800

    fix:點擊星星後，評分顯示錯誤

    問題：
      使用者點擊評分用的星星後，下方的分數會比星星數少1。
    原因：
      星星陣列的index起始值為0，顯示評分直接抓取了index。
      是故當有五顆星星時，顯示的評分會為4。
    調整：
      修改顯示評分的計算公式，改為星星陣列index+1。

    StarProject #003
```

git commit

1. 將當前修改的狀態提交到儲存庫。
2. 會自動產生一組SHA-1校驗碼。
3. SHA-1 是「Secure Hash Algorithm 1」的縮寫，它是一種雜湊演算法，計算之後的結果通常會以 40 個十六進位的數值呈現。輸入相同的物件會產生相同的雜湊值，常用來檢視檔案有沒有被修改。(輸入不同物件產生相同雜湊值稱之為碰撞。)
4. 不推薦移除任何commit記錄，若檔案已push到遠端，移除記錄會造成團隊版本不一致。移除方法可參考連結一、連結二。

可使用的Type(不同公司可能會有不同分類)，參考來源

名稱	意義
feat	新增/修改功能，feature
fix	修補bug，fix bug
docs	文件，documentation
style	格式，不影響程式碼運行的變動 white-space, formatting, missing semi colons, etc
refactor	重構，既不是新增功能，也不是修補 bug 的程式碼變動
perf	改善效能，A code change that improves performance
test	增加測試，when adding missing tests
chore	建構程序或輔助工具的變動，maintain
revert	撤銷回復先前的 commit 例如：revert: type(scope): subject (回復版本：xxxx)

Type 是用來告訴進行 Code Review 的人應該以什麼態度來檢視 Commit 內容。
如看到 Type 為 fix，進行 Code Review 的人就可以用「觀察 Commit 如何解決錯誤」的角度來閱讀程式碼。
若是 refactor，則可以放輕鬆閱讀程式碼如何被重構，因為重構的本質是不會影響既有的功能。
利用不同的 Type 來決定進行 Code Review 檢視的角度，可以提升 Code Review 的速度。
因此開發團隊應該要對這些 Type 的使用時機有一致的認同。

```
git log
```
顯示commit的歷史紀錄。

```
git log --oneline
```
將顯示的記錄縮短成一行(簡化雜湊值)，只留標題。

```
git log --pretty=oneline
```
將顯示的記錄縮短成一行(顯示40碼雜湊值)，只留標題。

```
git log --pretty=format:"%h - %an, %ar : %s"
```
將顯示的記錄以指定的格式呈現。

```
git log --graph
```
將顯示的記錄以圖形化的方式呈現。

```
git log master
```
顯示主分支(master branch)的commit記錄。

```
git reflag
```
顯示完整的操作紀錄

git log

1. 顯示commit的歷史記錄。
2. 詳細說明可參考官網。
3. git log是一個功能多樣、強大的指令，但往後換成圖形化介面後，此指令就相對變得較為不重要。
4. 當然，如果你是指令派的話，請務必熟練此指令。

git checkout SHA-1校驗碼

將HEAD移動到指定SHA-1的版本(至少需輸入前4碼)。

git checkout master

將HEAD移動到master分支的最新一次commit。

git checkout HEAD^^

將HEAD移動到上兩個版本(一個^代表一個版本)。

git checkout

1. 將HEAD移動到指定的節點(commit完成後)，並將版本狀態還原成指向的版本。
2. HEAD即是你當前指向的版本。
3. checkout是個通用(且危險)的指令，還有很多其他的不同功能，使用時要多加小心。

【未追蹤】
```
git clean -n
```
顯示即將刪除的檔案

```
git clean -f
```
刪除所有未追蹤的檔案

【已追蹤，但未放入暫存區】：修改了已追蹤的檔案，即會變成此狀態
```
git checkout -- '檔案名稱'
```
還原修改

```
git checkout .
```
還原所有修改

【已追蹤，且放入暫存區】：修改已追蹤檔案，並使用add加入暫存區
```
git reset / git rest HEAD
```
將暫存區的檔案取出(commit時不會包含)

還原檔案

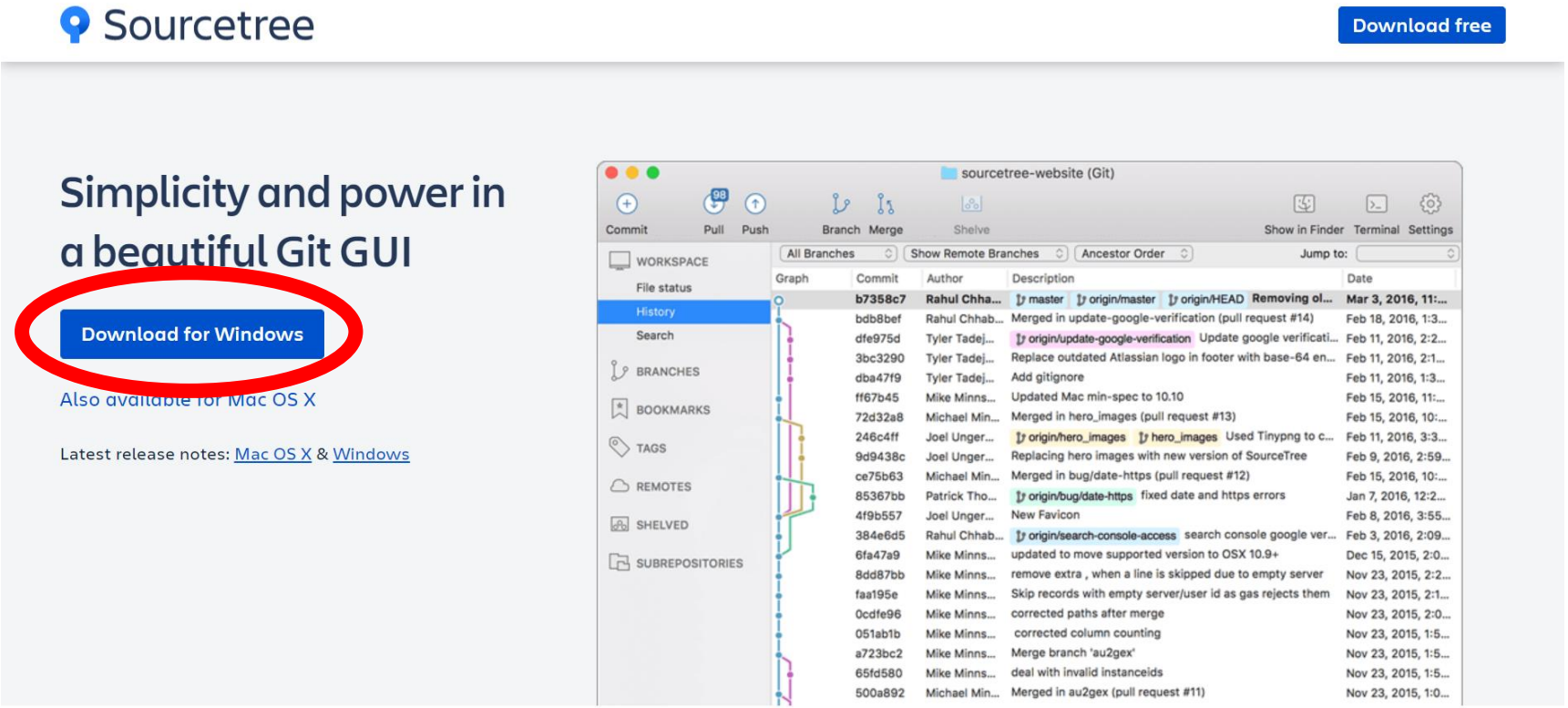
檔案修改有三種狀態

1. **未追蹤(新建檔案)**
2. **已追蹤，但未放入暫存區**
3. **已追蹤，也放入暫存區**

以上三種狀態，分別有不同的還原方式。

第五節
GIT分支

到Source Tree官網，根據作業系統選擇版本，下載後執行安裝檔，其餘安裝步驟請參考備註。



Source Tree

1. Source Tree是一套免費的GIT GUI軟體，可將各版本以圖形化的方式呈現。
2. 相同功能的替代品還有：
   - GitHub Desktop
   - Fork
   - TortoiseGit

分支

`git branch <分支名稱>`
在當前分支下，依據指定的名稱建立起分支。

`git branch`
顯示所有分支名稱。

`git checkout <分支名稱>`
切換到指定的分支。

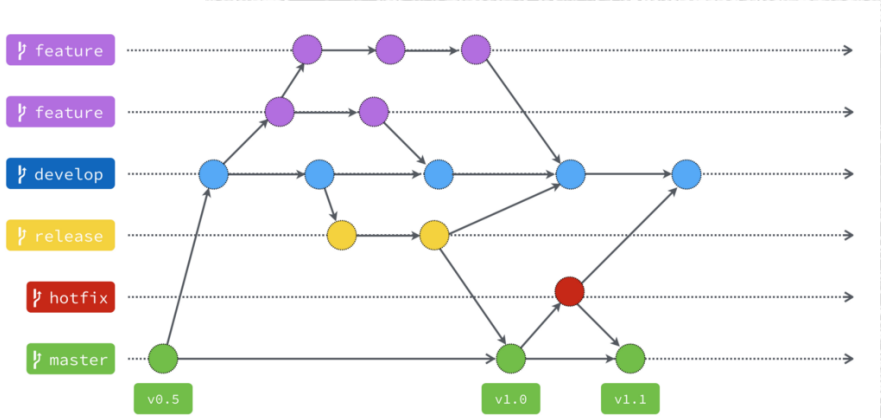
`git checkout -b <分支名稱>`
在當前分支下，創建並切換到指定的分支。

`git branch -m <舊分支名稱> <新分支名稱>`
重新命名分支。

`git branch -d <分支名稱>`
刪除指定分支。

`git merge <比較新的分支名稱>`
在當前分支下，合併指定分支(需比當前分支還新)。

`git merge <比較新的分支名稱> --no-ff`
在當前分支下，合併指定分支，且保留分支紀錄。



第六節
遠端儲存庫

遠端儲存庫

`git clone <url>`
從指定URL複製一份儲存庫到本地，並自動設定簡稱為origin。

`git remote`
顯示所有遠端儲存庫的資訊。

`git remote add <遠端簡稱> <url>`
手動新增遠端儲存庫。

`git remote show <遠端簡稱>`
顯示指定遠端儲存庫的詳細資料。

`git fetch`
取得預設遠端的最新資料，並在本地存成FETCH_HEAD分支。

`git pull`
取得預設遠端的最新資料，並直接合併到本地。(pull = fetch + merge)

`git push <遠端簡稱> <分支名稱>`
將本地的指定分支推送到遠端儲存庫，並執行合併。

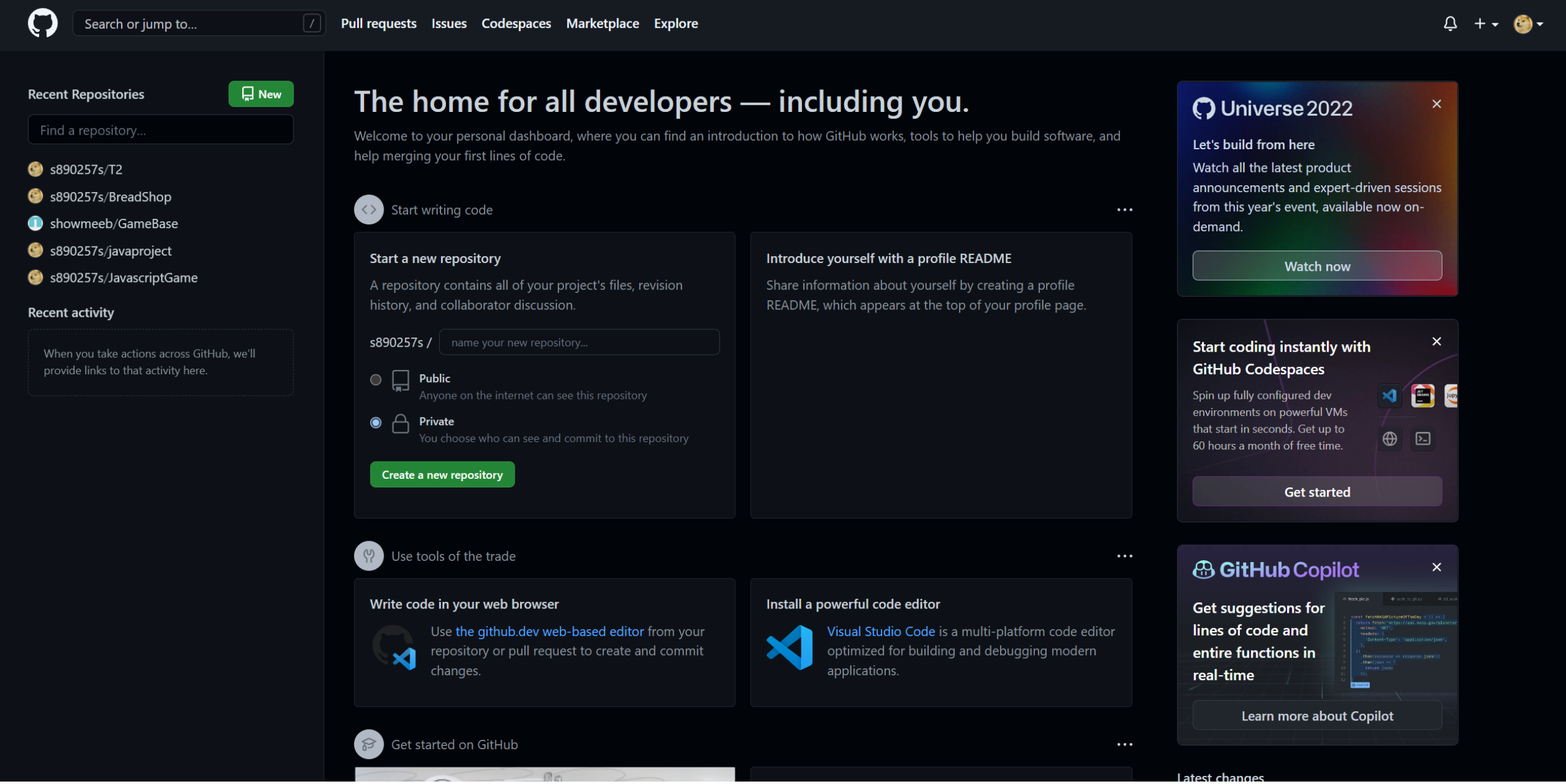
第二章
使用GIT HUB

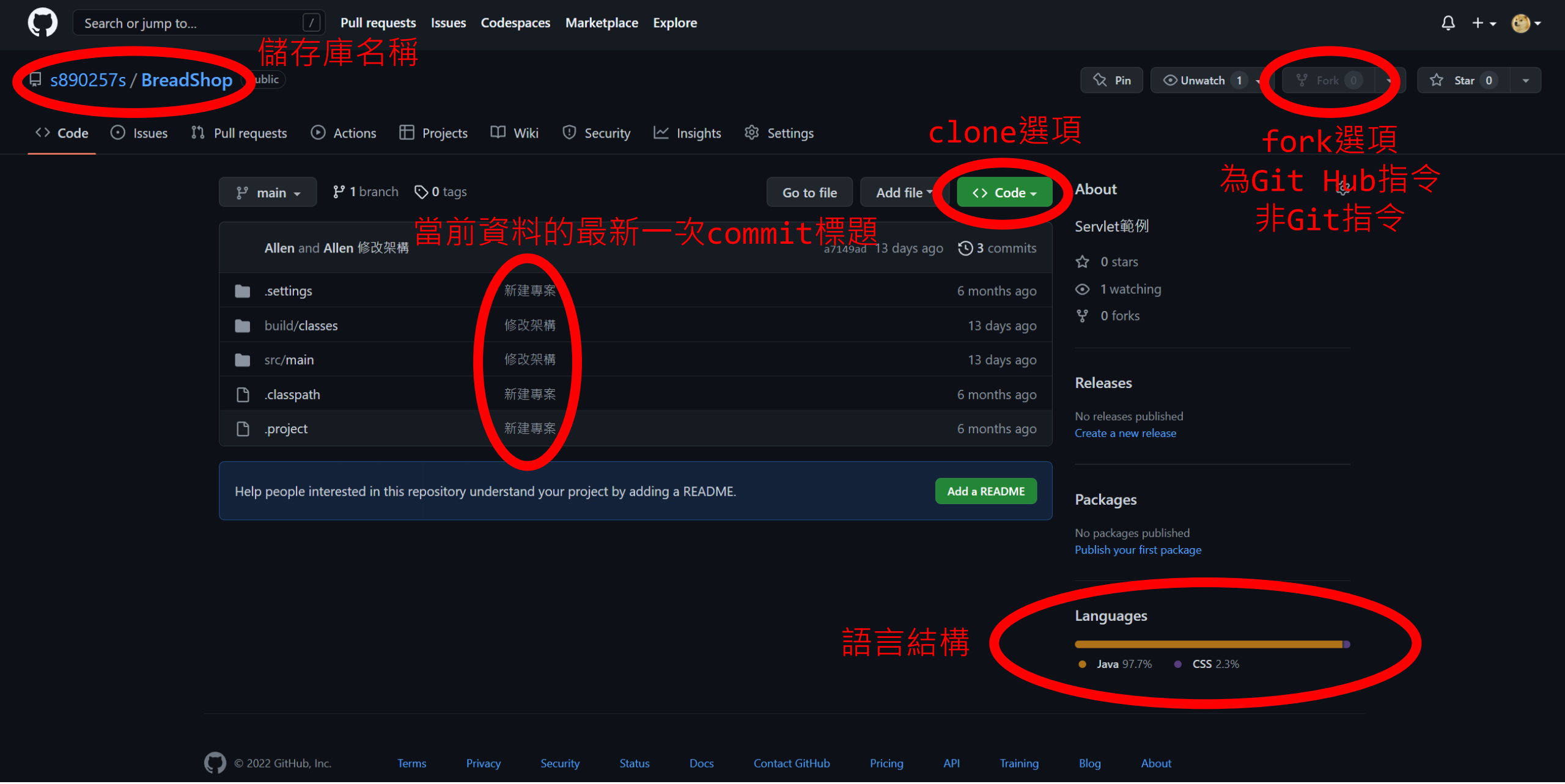
第一節
GIT HUB介紹

Git Hub

1. Git是一種分散式版本控制軟體。
2. Git Hub是一種線上的程式碼託管平台，以Git技術為核心，提供使用者建立自己的線上儲存庫。
3. 在2018年被微軟收購。
4. 截至2022年6月，GitHub已經有超過5700萬註冊使用者和1.9億代碼庫（包括至少2800萬開原始碼庫），事實上已經成為了世界上最大的代碼代管網站和開源社群。

--維基百科


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Recent Repositories
New
Find a repository...
s890257s/T2
s890257s/BreadShop
showmeeb/GameBase
s890257s/javaproject
s890257s/JavascriptGame
Recent activity
When you take actions across GitHub, we'll provide links to that activity here.
The home for all developers — including you.
Welcome to your personal dashboard, where you can find an introduction to how GitHub works, tools to help you build software, and help merging your first lines of code.
Start writing code
Start a new repository
A repository contains all of your project's files, revision history, and collaborator discussion.
s890257s /
name your new repository...
Public
Anyone on the internet can see this repository
Private
You choose who can see and commit to this repository
Create a new repository
Introduce yourself with a profile README
Share information about yourself by creating a profile README, which appears at the top of your profile page.
Use tools of the trade
Write code in your web browser
Use the github.dev web-based editor from your repository or pull request to create and commit changes.
Install a powerful code editor
Visual Studio Code is a multi-platform code editor optimized for building and debugging modern applications.
Get started on GitHub
Universe 2022
Let's build from here
Watch all the latest product announcements and expert-driven sessions from this year's event, available now on-demand.
Watch now
Start coding instantly with GitHub Codespaces
Spin up fully configured dev environments on powerful VMs that start in seconds. Get up to 60 hours a month of free time.
Get started
GitHub Copilot
Get suggestions for lines of code and entire functions in real-time
Learn more about Copilot
Latest changes


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
儲存庫名稱
s890257s / BreadShop
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
clone選項
fork選項
為Git Hub指令
非Git指令
main
1 branch
0 tags
Go to file
Add file
Code
About
Servlet範例
當前資料的最新一次commit標題
Allen and Allen 修改架構
a7149ad 13 days ago
3 commits
.settings 新建專案 6 months ago
build/classes 修改架構 13 days ago
src/main 修改架構 13 days ago
.classpath 新建專案 6 months ago
.project 新建專案 6 months ago
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Help people interested in this repository understand your project by adding a README.
Add a README
語言結構
Languages
Java 97.7%
CSS 2.3%
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

第二節
建立REPOSITORY

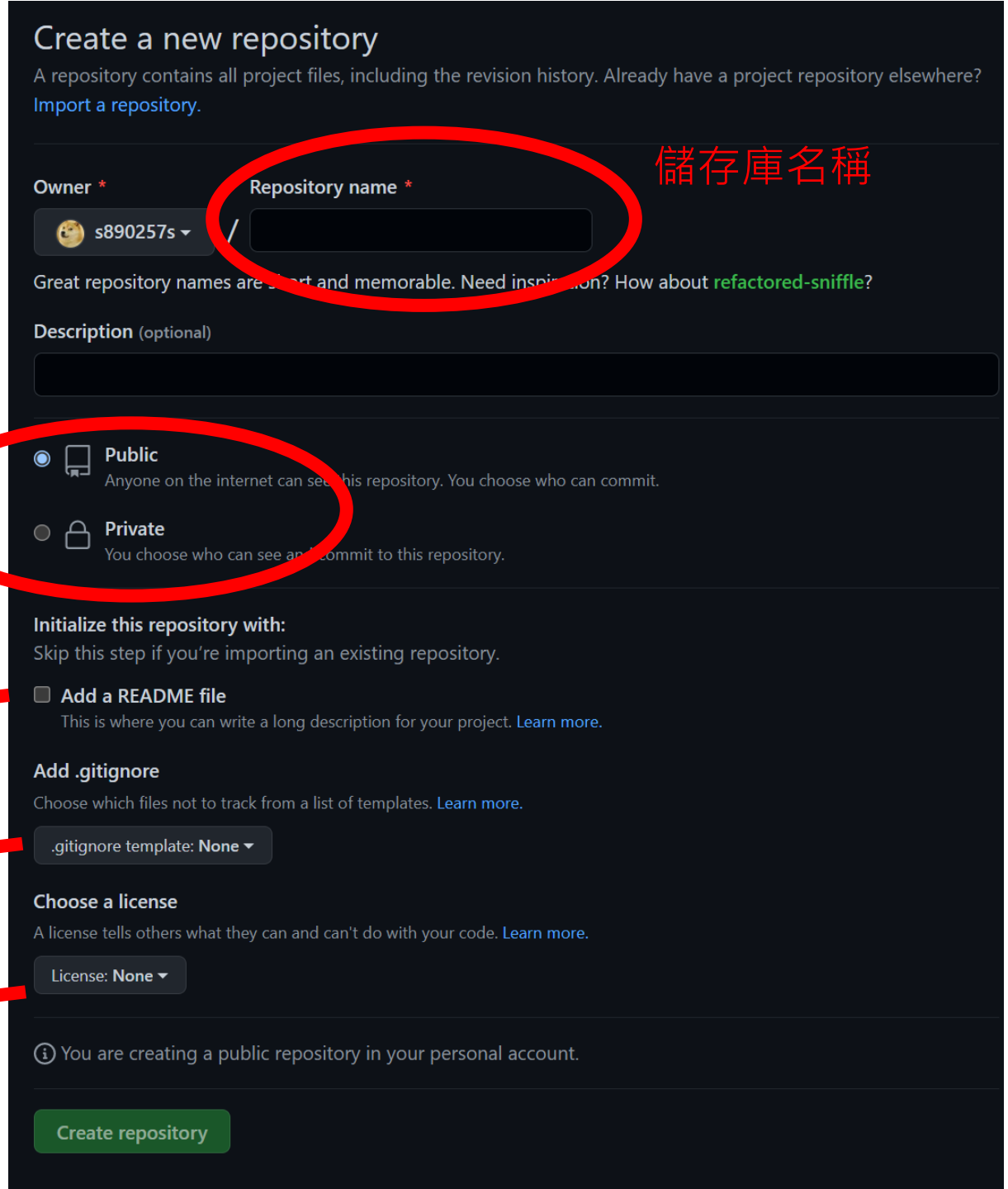
建立遠端儲存庫(1)

設定公開或私有
公開的話，所有人都
可以自由clone此專
案；私有的話，無法
使用GitHub page

加入說明文件，使用
Markdown語法(.md)

預設忽略檔案

使用的授權

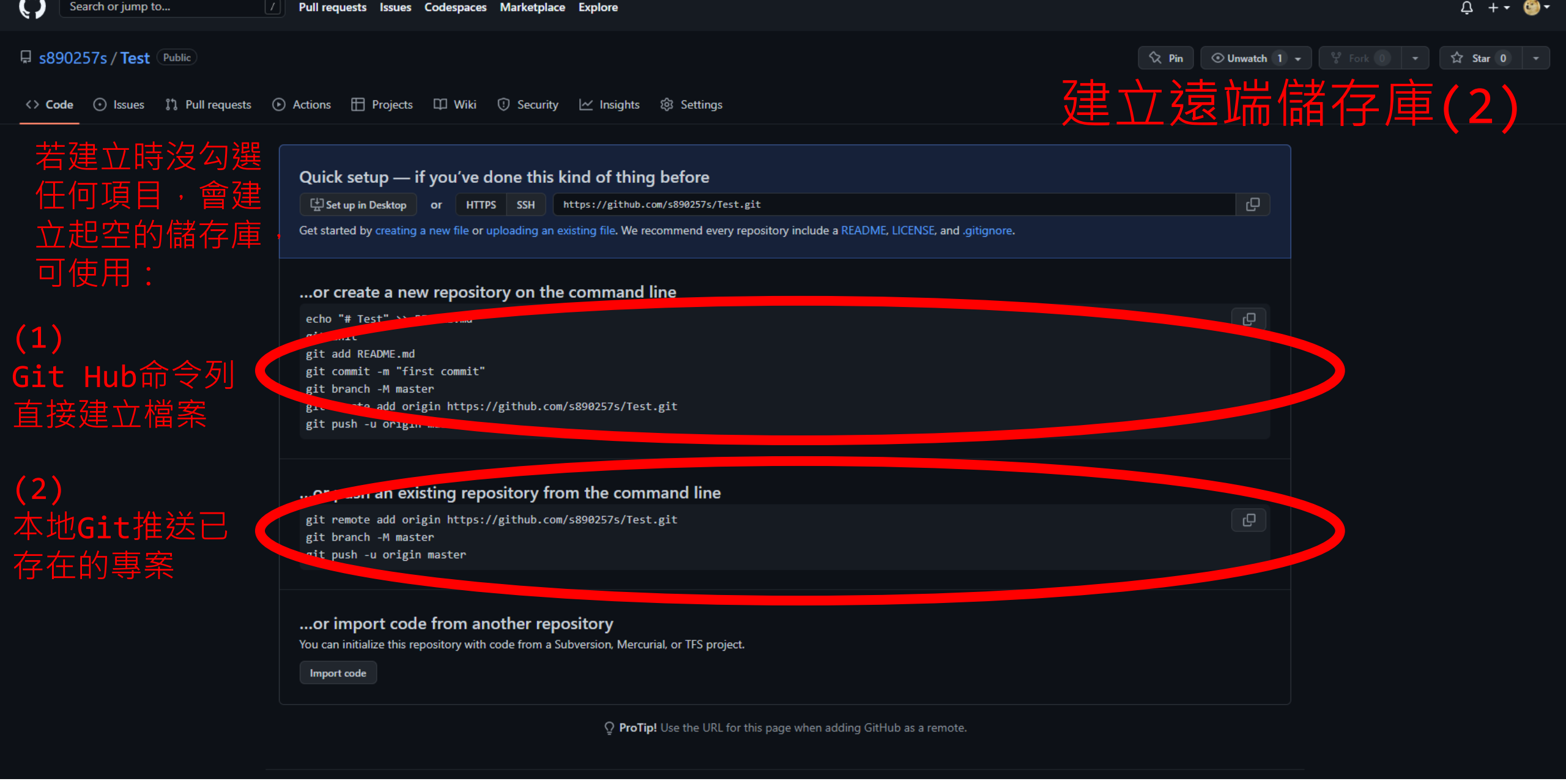


建立遠端儲存庫(2)

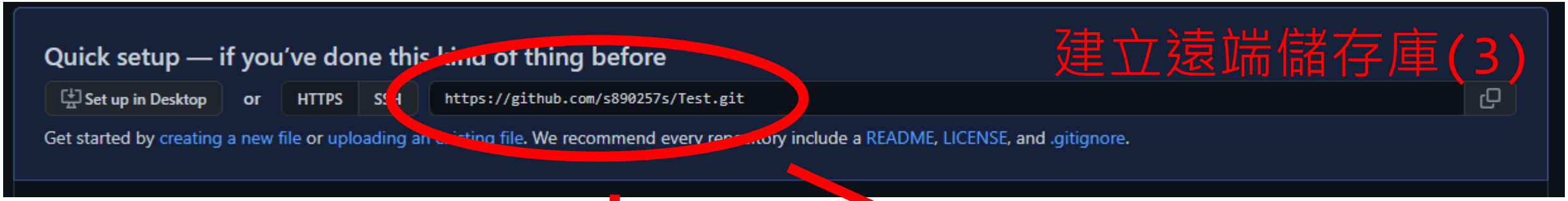
若建立時沒勾選任何項目，會建立起空的儲存庫，可使用：

(1)
Git Hub命令列直接建立檔案

(2)
本地Git推送已存在的專案



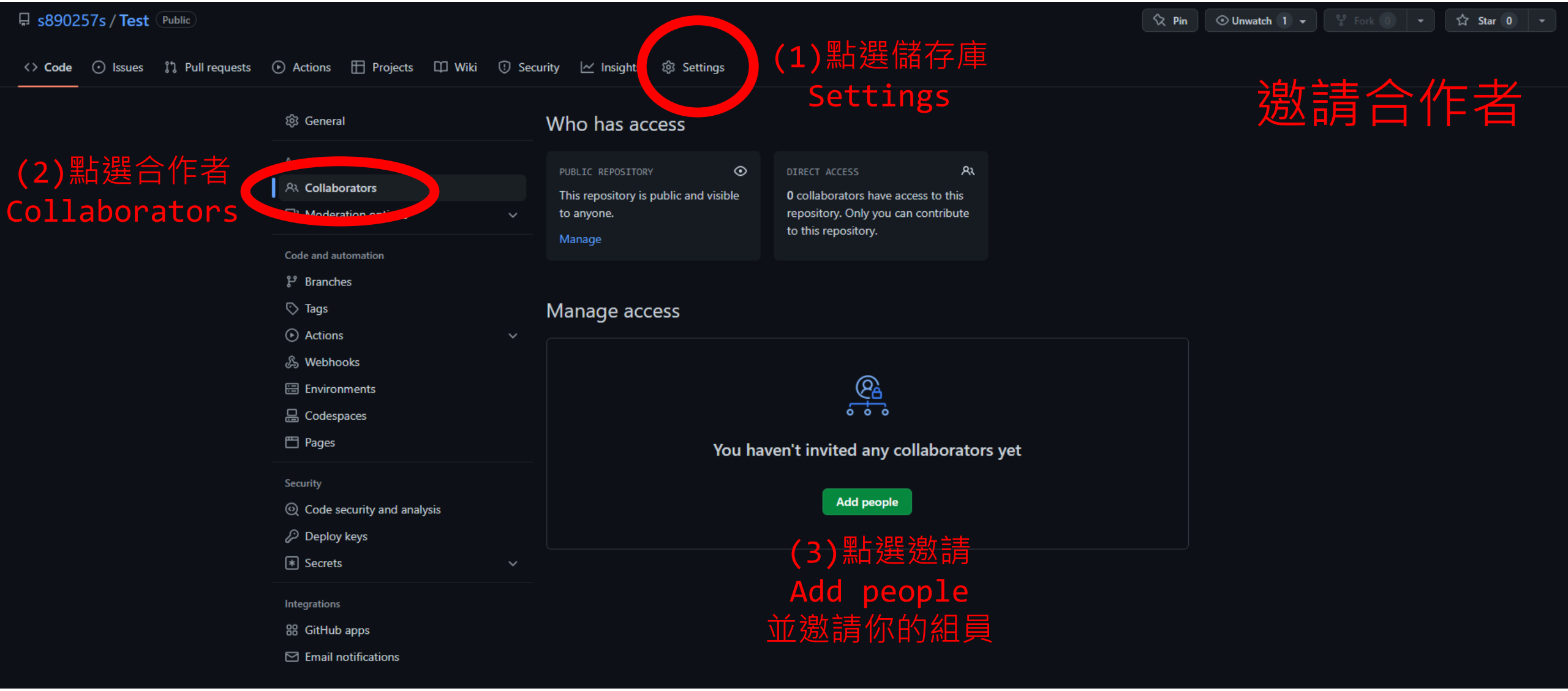
建立遠端儲存庫(3)



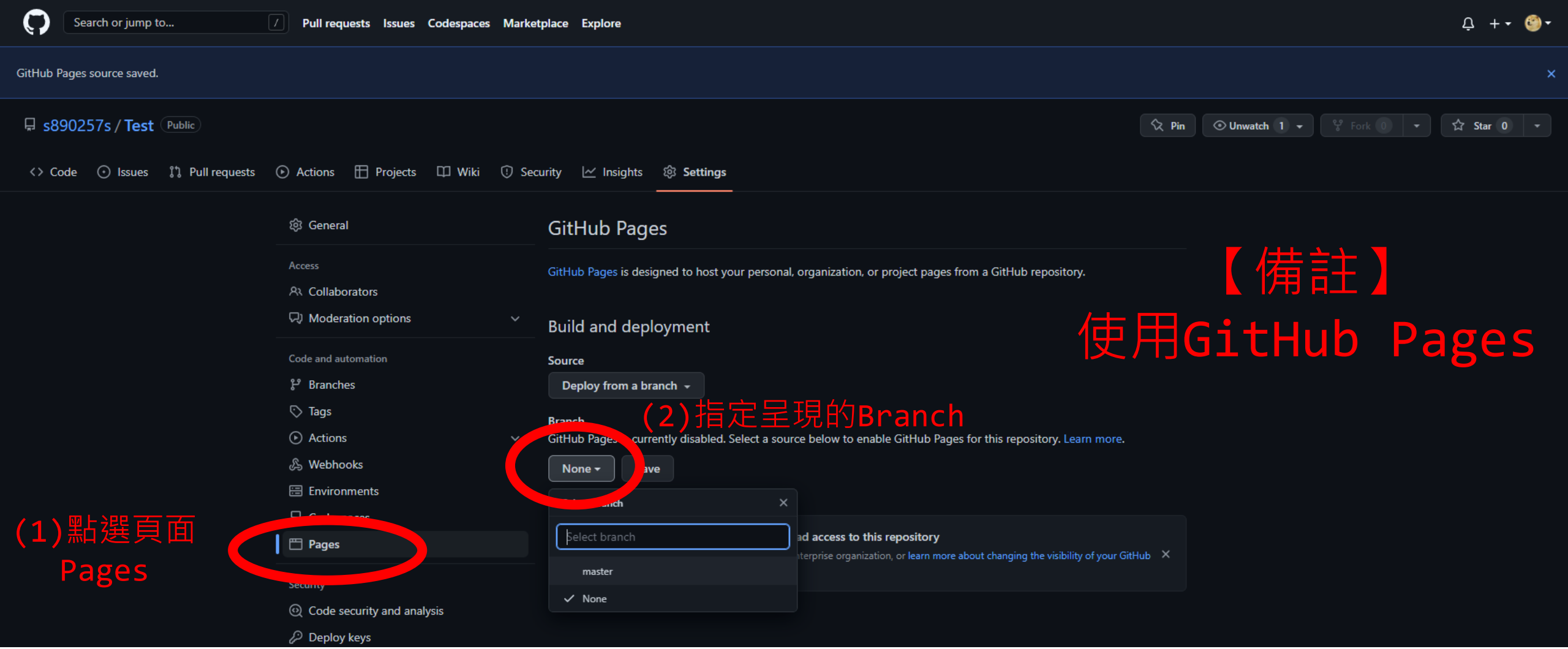

```
User@DESKTOP-N15SLMQ MINGW64 ~/Desktop/local/Test
$ git init
Initialized empty Git repository in C:/Users/User/Desktop/local/Test/.git/

User@DESKTOP-N15SLMQ MINGW64 ~/Desktop/local/Test (master)
$ git add .

User@DESKTOP-N15SLMQ MINGW64 ~/Desktop/local/Test (master)
$ git commit -m 'init'
[master (root-commit) 1400681] init
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

User@DESKTOP-N15SLMQ MINGW64 ~/Desktop/local/Test (master)
$ git remote add origin https://github.com/s890257s/Test.git

User@DESKTOP-N15SLMQ MINGW64 ~/Desktop/local/Test (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 216 bytes | 216.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/s890257s/Test.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

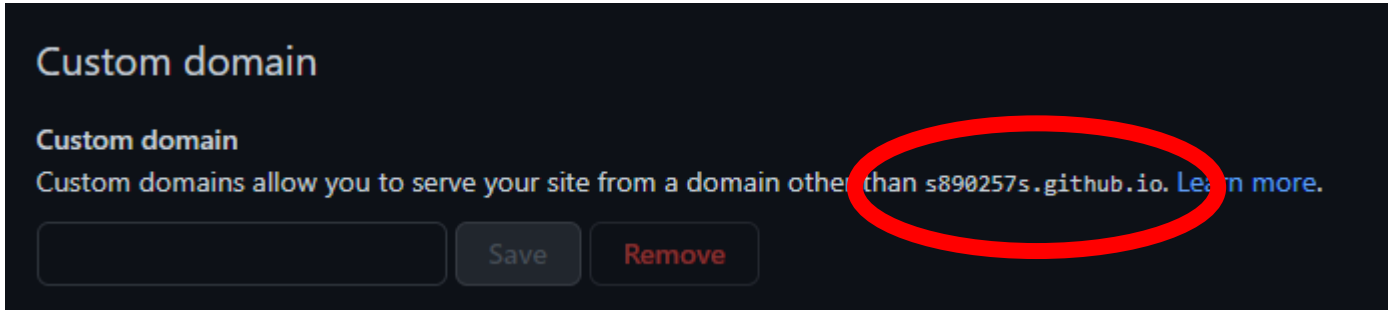
將本地的遠端儲存庫設定為GitHub產生的連結，並推送過去。

邀請合作者


s890257s / Test
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
(1)點選儲存庫
Settings
(2)點選合作者
Collaborators
General
Collaborators
Code and automation
Branches
Tags
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security and analysis
Deploy keys
Secrets
Integrations
GitHub apps
Email notifications
Who has access
PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage
DIRECT ACCESS
0 collaborators have access to this repository. Only you can contribute to this repository.
Manage access
You haven't invited any collaborators yet
Add people
(3)點選邀請
Add people
並邀請你的組員

【備註】
使用GitHub Pages

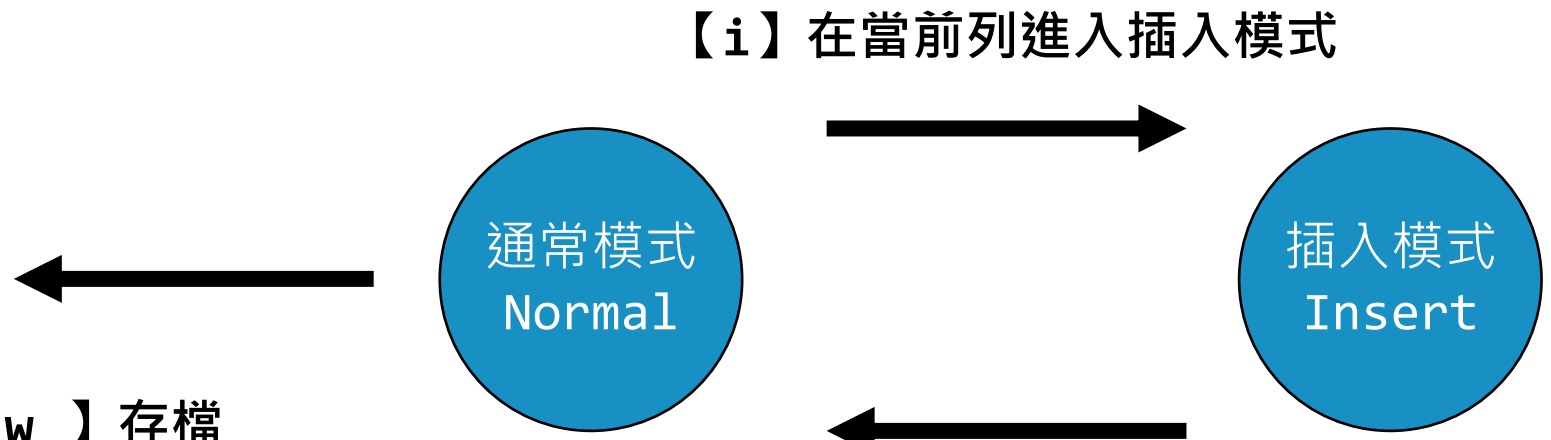




(3)
稍等一會後，可使用預設domain存取專案(限靜態頁面)

備註
VIM操作

我已經使用了vim 10 年了，因為我還不知道如何退出 - 程式笑話



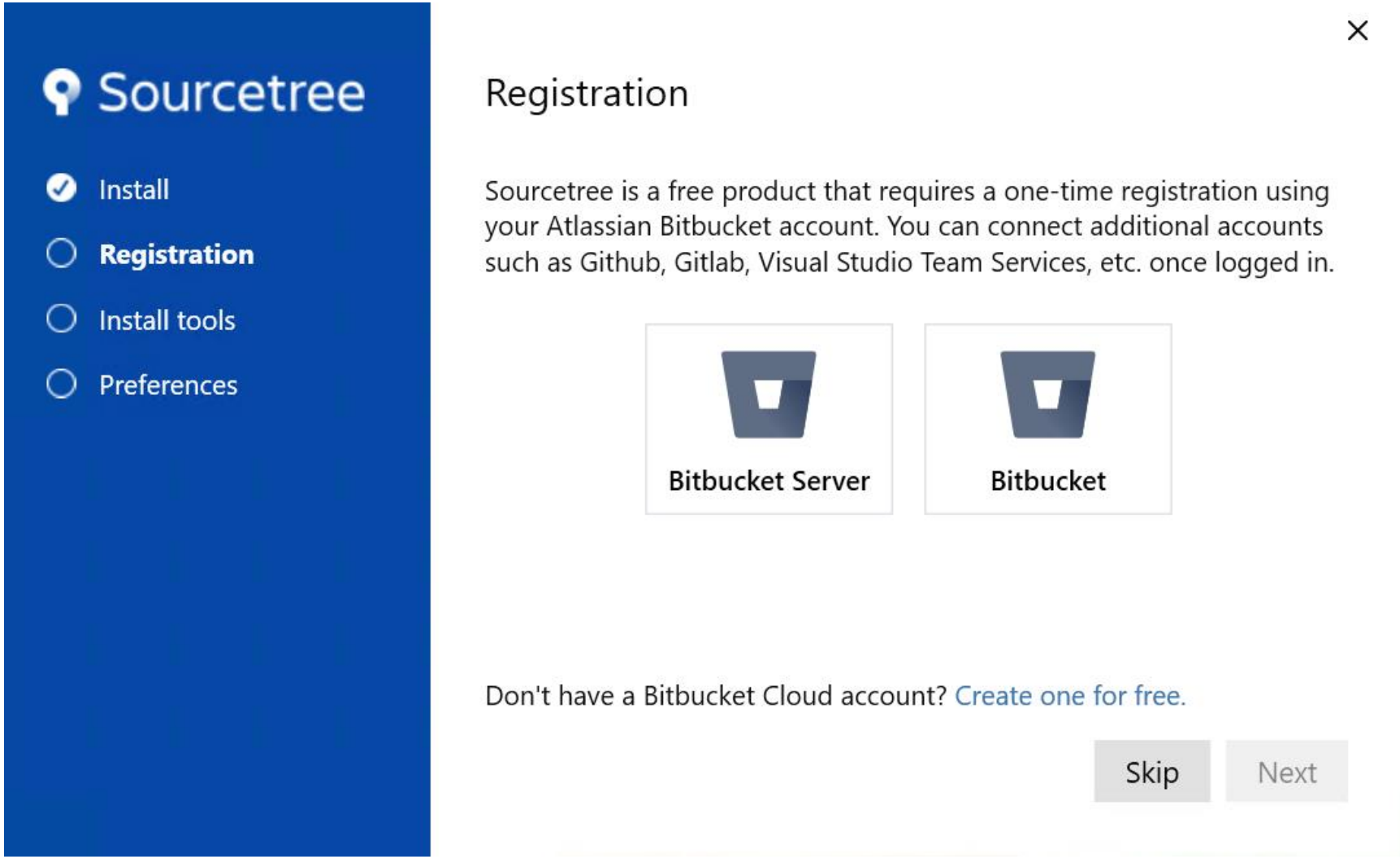
【:w 】存檔
【:q 】離開(未存檔會警告)
【:q!】強制離開(捨棄變更)
【:wq】存檔並離開

VIM基礎操作

1. Vim的前身是Vi，兩者都是文字編輯器；Vi於1976年發布(45年up)、Vim於1991年發布(30年up)。
2. Vim是Git for Windows的預設文字編輯器。(因歷史原因)
3. 對現代的使用者而言，它是不直觀且難用的。
4. Vim有6種基本模式與5種衍生模式，詳見維基百科；基本上只須關注：通常(Normal)、插入(insert)兩種模式。
5. 不知道自己在哪種模式的話，就多敲幾次ESC。

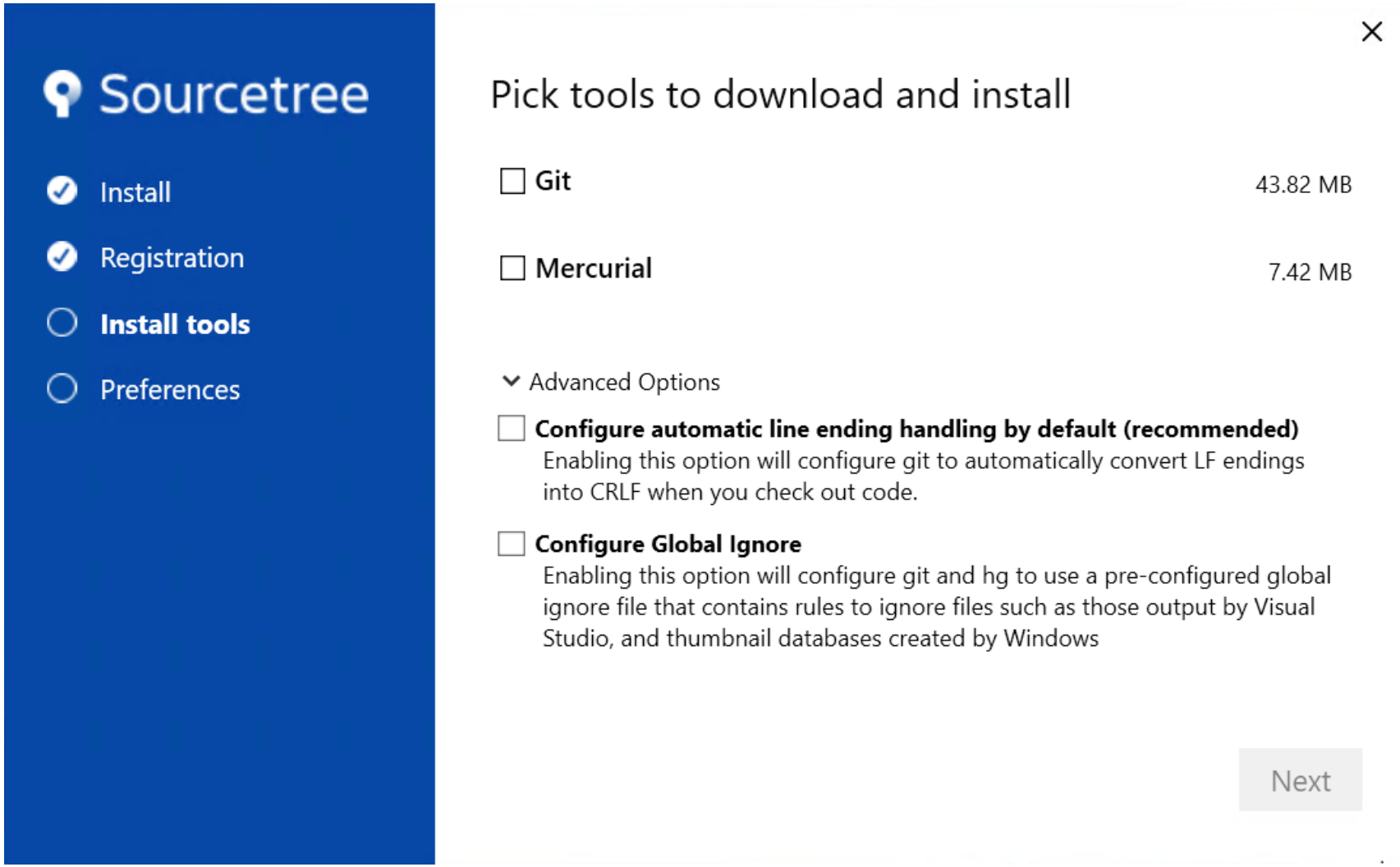
備註
SOURCETREE
安裝

安裝步驟介紹(1)



1. 選擇是否登入Bitbucket。
2. Bitbucket是類似GitHub的、雲端託管程式碼的空間，有免費方案與商用付費方案。
3. 本課程會使用GitHub作為遠端儲存庫(Remote Repository)，故此請按Skip跳過。

安裝步驟介紹(2)

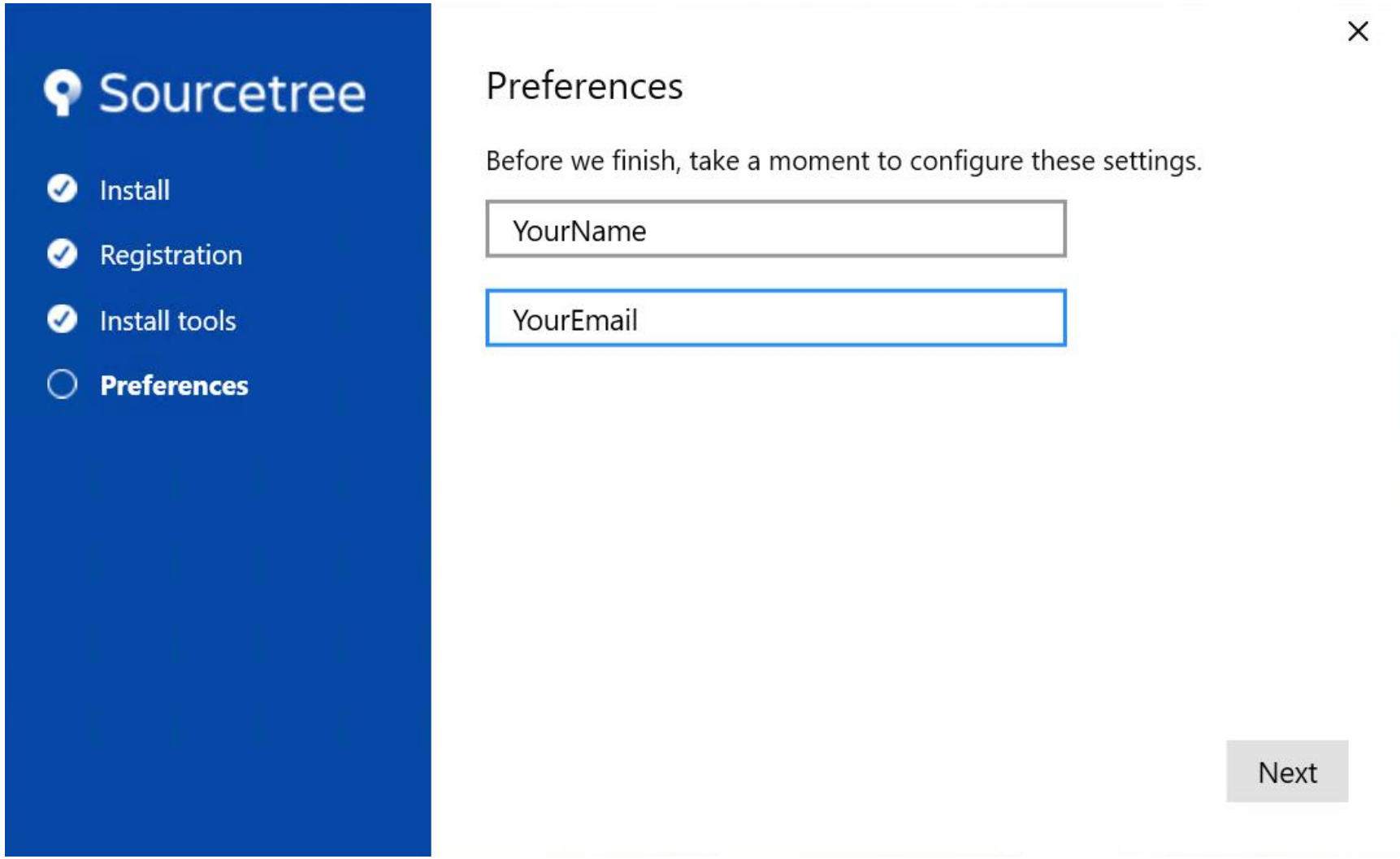


1. 若無安裝Git，在此則需勾選Git；SourceTree是一種Git的圖形化軟體，但沒有Git本體的話是無法執行的。
2. Mercurial是類似Git的分散式版本控制軟體，版本儲存方式有很大的差異，各有優劣。

【進階選項】

1. 自動轉換分段符號，請參考Git安裝步驟(10)。
2. 啟動預設Git ignore選項，不用勾選。

安裝步驟介紹(3)



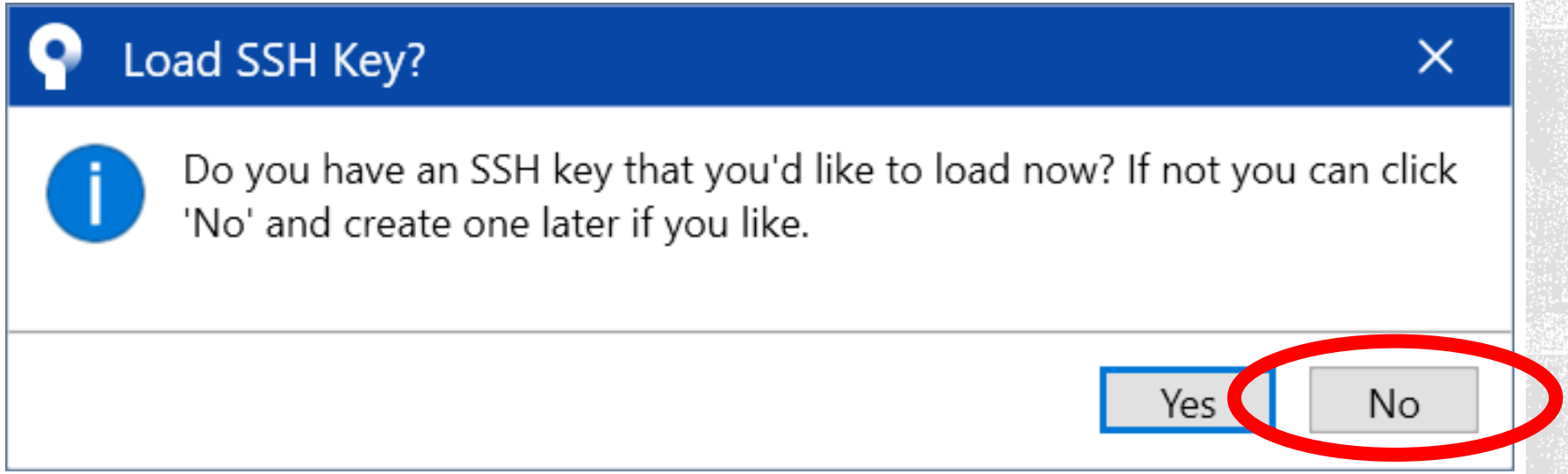
1. 輸入Git commit時的使用者與使用者Email。
2. 若有偵測到.gitconfig的話，則會自動帶入。

【.gitconfig預存路徑】

C:\Users\使用者\.gitconfig

安裝步驟介紹(3)

1. 若要使用自己的SSH認證，則選Yes；在此選擇No



備註 MARKDOWN 語法

語法	意義
# 標題	最大標題
###### 標題	最小標題
> 文字	引用文字
- 清單項目	也可使用 * +，皆為清單項目，空兩個或縮排即為子項目
1. 數字清單	子項目與上相同
``	三個反引號圍成的區塊，內部會顯示原始程式碼
---	分隔線(***亦同)
| -	使用左符號畫出表格
斜體	_斜體_ 亦可
粗體	__粗體__亦可
[名稱]連結	[google](https://www.google.com.tw) 超連結
![提示圖片]連結	![這是圖片](http://MyProject/A.jpg)

Thank you