

Github操作教學

關於版本控制

- 即為Version Control System (VCS) (版本控制)
- 紀錄修改歷程
- 簡單地徹修對檔案的編輯
- 無法更改他人的變更
- 為什麼要版本控制

GIT簡史

- Linux kernel 的開發者，Linus Torvalds 在2005年以十天的時間開發了Git的第一個版本
- Git是用C語言開發的用於Linux核心開發的版本控制工具，它採用了分散式版本庫的作法
- 它採用了分散式版本庫的作法，不需要伺服器端軟體，就可以運作版本控制



Git開發者: Linus Torvalds

Git的優點

- 不需要網路
可以在本地端完成作業，稍後再連網協作
- 是屬於分散式的版本控制系統
不用擔心檔案丟失，可以透過任一戶用端做鏡像
- 開放原始碼
Git沒有對版本庫的瀏覽和修改做限制
- 快速及簡潔
對於大型專案來說非常好用且效能非常高
- 支線非線性的開發
可以自由的合併或開發任一分枝，不必受到主要專案線受限制
- 線上學習手冊(1)： <https://git-scm.com/book/zh-tw/v2>
- 線上學習手冊(2)： <https://zlargon.gitbooks.io/git-tutorial>

Git狀態

- Git的四大區塊
- Git的三個狀態
- Git的工作流程

Git的四大區塊

- 工作目錄 (Working directory)

專案中其中一個版本，這些檔案將會被提出任使用者修改

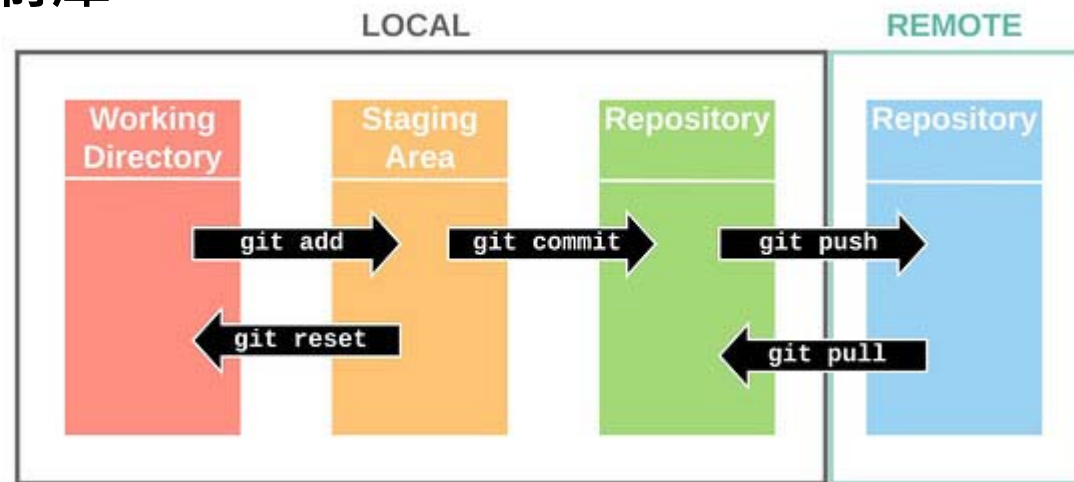
- 預存區 (Staging area)

通常在 Git 目錄下，儲存關於下次提交的資訊

- Git 倉庫 (Git directory)

用來存放專案的版本資料庫，也就是版本控制庫

- 遠端倉庫 (Remote Repository)



Git的三個狀態

■ 已修改(modified)

- 代表這檔案已被修改但尚未提交到本地端資料庫

■ 已預存(staged)

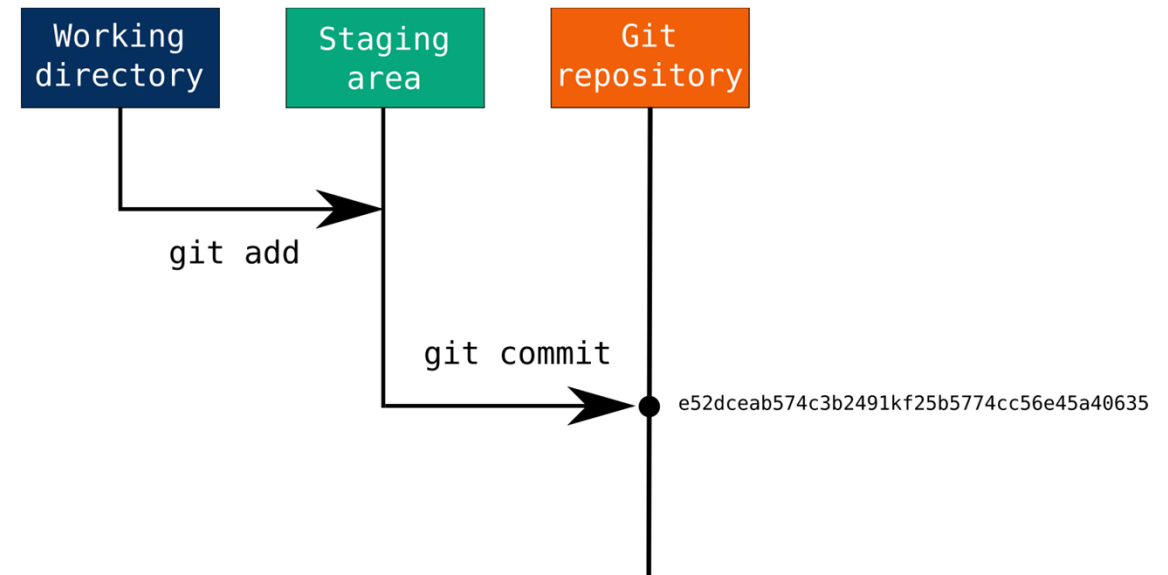
- 代表這檔案將會被存到下次你提交的快照中

■ 已提交(commited)

- 代表這檔案已安全地存在你的本地端資料庫

Git的工作流程

1. 在資料庫中檢視
2. 在工作目錄修改檔案
3. 預存檔案 (`git add .`)，將有更動的檔案快照新增到預存區
4. 提交(`git commit -m "message"`)，這會讓存在預存區的檔案快照永久地儲存在 Git 目錄中。



不同區域的檔案狀態

■ 未追蹤(Untracked):

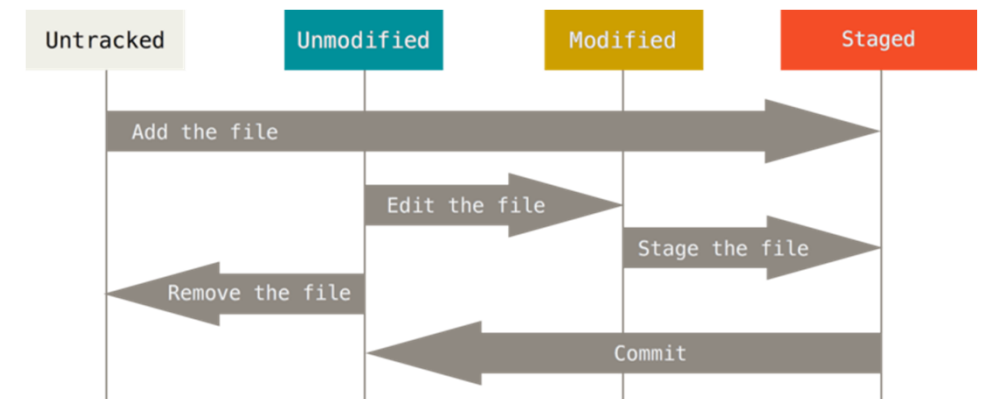
- 沒有被追蹤的檔案(Untracked files)
- 有修改(Modified)、還沒準備要被遞交 (Changes not staged for commit)
- 在工作目錄下，不包含上次的更動，也不在預存中的任何檔案

■ 已追蹤(Tracked):

- 有修改、準備要被遞交的檔案(Changes to be committed)

■ 已經被遞交的檔案 (Committed)

- 已被加入資料庫的檔案



安裝 Git 軟體

- 在Linux下安裝 Git
- 在mac/windows下安裝git
- 初次設定Git
- 介紹不同版本的git

在Linux安裝Git

■開啟終端機(Terminal)

□點擊左下角選單(Show Applications) ，輸入「Terminal」，開啟終端機

■依序輸入以下指令：

■sudo apt update

■sudo apt upgrade

■更新後或許需要重啟系統：sudo reboot

■輸入指令碼 sudo apt install git

□此指令碼將會自動安裝git，稍等片刻即可安裝完成

在其他系統下安裝Git

■在Windows下安裝git

- 到官方網站下 <https://git-scm.com/download/win> 選擇適合你版本的Git for Windows

■在Macos下安裝git

- 在lanunchapd中找到終端機，(在 Mavericks (10.9) 或更新版的系統中)，開啟終端機
- 直接打入「git」指令。若還沒安裝，系統會自動一步步安裝

初次設定Git

■確認Git版本

- 輸入git --version 來確認git版本(2.32.1)

■使用者設定 請輸入你個人的姓名及Email

- 輸入git config --global user.name [xxx自己的暱稱，與Github ID無關]
- 輸入git config --global user.email [xxx@domain.com自己的email地址]
- 這樣別人才知道是誰修改的檔案內容 (畫押的概念 🐼)



■識別自己的設定

- 輸入git config --list 檢視你的資料有無正確

建立一個Git資料庫(使用終端機)

- 在本機中建立本地的儲存庫(local repository)
 - 找到資料庫的存放的位置，再來建立專案資料庫
- 初始化
 - 輸入指令git init，初始化這個目錄，自動生成一個.git隱藏目錄
- 如何將檔案交給Git做版本控制
 - 輸入指令git status查詢此目錄的「狀態」
- 新增/修改/刪除檔案，這裡我們建立一個readme.md的檔案，並加入內容
- 使用操作索引的指令
 - 輸入指令git add . 將新的檔案的更動加入索引檔
- 使用操作索引的指令
 - 輸入指令：git commit --m "first-commit" 提交第一個版本

建立一個Git資料庫(使用Visual Studio Code)

- 先開啟Visual Studio Code，選擇「檔案 > 開啟資料夾」
- 點選左側「原始檔控制」，選擇「將資料庫初始化」(即git init指令)
- 左側切回「檔案總管」，新增readme.md檔案，並輸入內容
- 再切到「原始檔控制」，將readme.md「儲存變更」，移至「暫存區」
- 輸入commit的備忘訊息，按下Ctrl+Enter即可提交；或按下此鈕，選擇「認可>認可」，同樣可以完成commit
- 安裝套件「Git History」，在VS Code按下F1或Ctrl+Shift+P，輸入git log，選擇「Git: View History (git log)」，即可觀察版本變化

關於GitHub

■ GitHub

- 是全球目前最大的Git Server

■ GitHub官方網站

- <https://github.com/>

■ 協作的第一步

- 建立你的github帳號



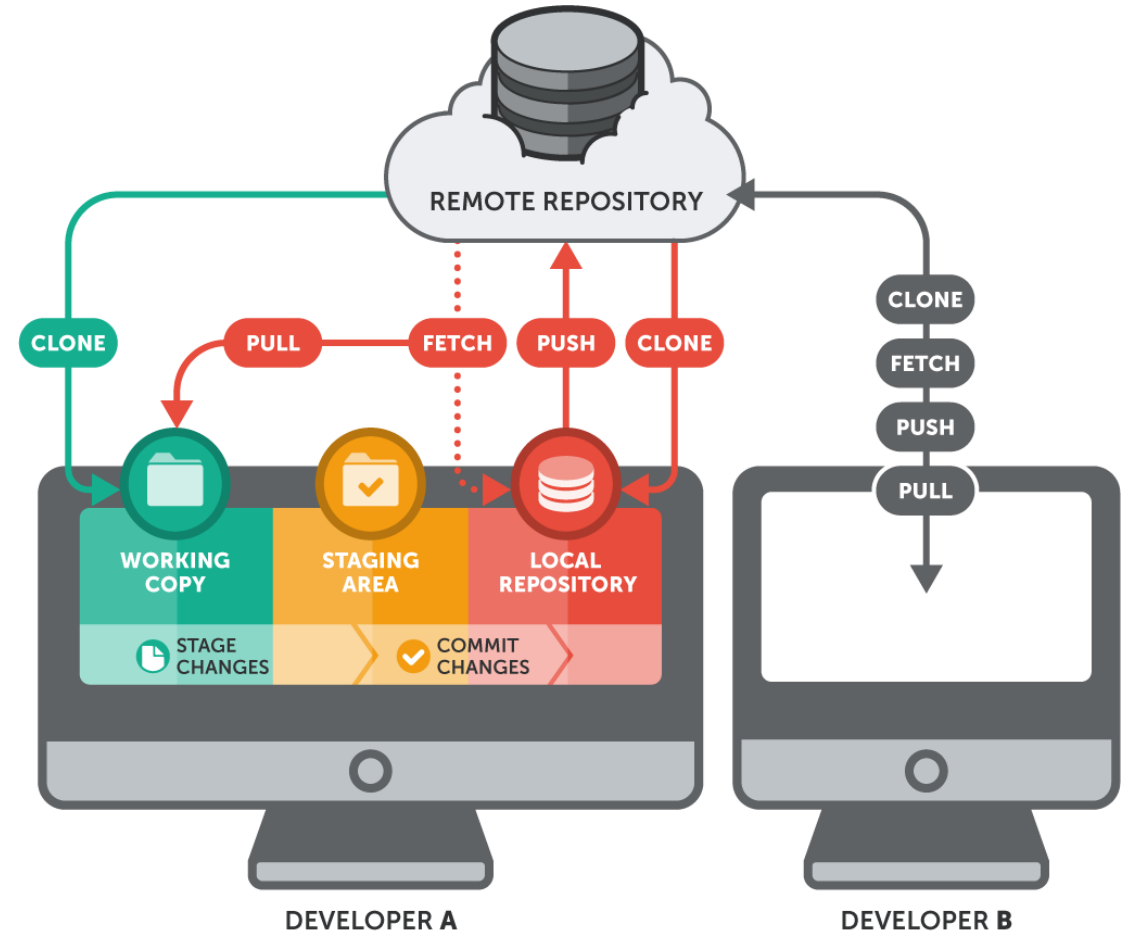
協作的第一步

■ GitHub專案

- 建立你的協作平台專案資料庫



■ 開始協作專案

- 選擇公開此專案，並獲取專案網址



<https://hellojs-tw.github.io/git-101/git-remote/>

建立GitHub專案

- 註冊github.com會員，登入後，連到<https://github.com/>
- 點擊這個按鈕  New，新增一個儲存庫(Repository)
- 輸入儲存庫名稱(Repository name)、簡短說明(選填)、公開或私人，設定儲存庫的初始內容(README.md檔案、.gitignore檔案、license說明)
- 按下  確定新增儲存庫，首次進入空的儲存庫後，github網頁有提供三種新增檔案的方式教學：
 - 使用Github Desktop軟體
 - 使用SSH或HTTP網址搭配IDE工具
 - 使用命令列

使用命令列上傳檔案 - 1

- `git init` 初始化，只有資料夾尚未初始化才須執行此命令
- 新增一個README.md文字檔，並在其中加入一些內容
- `git status` 檢查狀態
- `git add .`
- `git status` 檢查狀態
- `git commit -m "create README.md"`

使用命令列上傳檔案 -2

- 接著把本機的專案送到github雲端
- `git remote -v` 檢查目前資料夾關聯的遠端空間
- `git remote add [name, 習慣使用origin] [github URL]`
- 輸入Github的帳號密碼，進行身分驗證
- `git remote -v`
- `git push [遠端名稱：origin] [分支名稱：master]`

常用的Git GUI工具

■ Visual Studio Code

由微軟開發且跨平台的免費原始碼編輯器。該軟體支援語法突顯、代碼自動補全、代碼重構、檢視定義功能，並且內建了命令列工具和 Git 版本控制系統。

可用來做為Python, Javascript, Node.js, Flask等開發環境

■ PyCharm

主要用於Python語言開發，由捷克公司JetBrains開發，提供代碼分析、圖形化除錯器，整合測試器、整合版本控制系統，並支援使用Django進行網頁開發

■ Github Desktop

開發者可以透過合併提交，或是重新排序提交，來整理提交歷史紀錄



第一次下載雲端儲存庫

■Clone遠端協作的專案

■Clone指令

- 使用指令：`git clone [儲存庫網址] [本地新資料夾名稱]`
- 會將整個專案連同目錄分支一同複製到本機資料端

從Github雲端儲存庫下載合併更新

■ Clone 「克隆」

- git clone 指令通常只會使用第一次

■ Fetch 「獲取資訊」

- git fetch：從遠端專案中取得資料，執行完成後，你應該會有那個遠端版本庫中所有分支的參照 (reference)

■ Pull 「拉取檔案並合併」

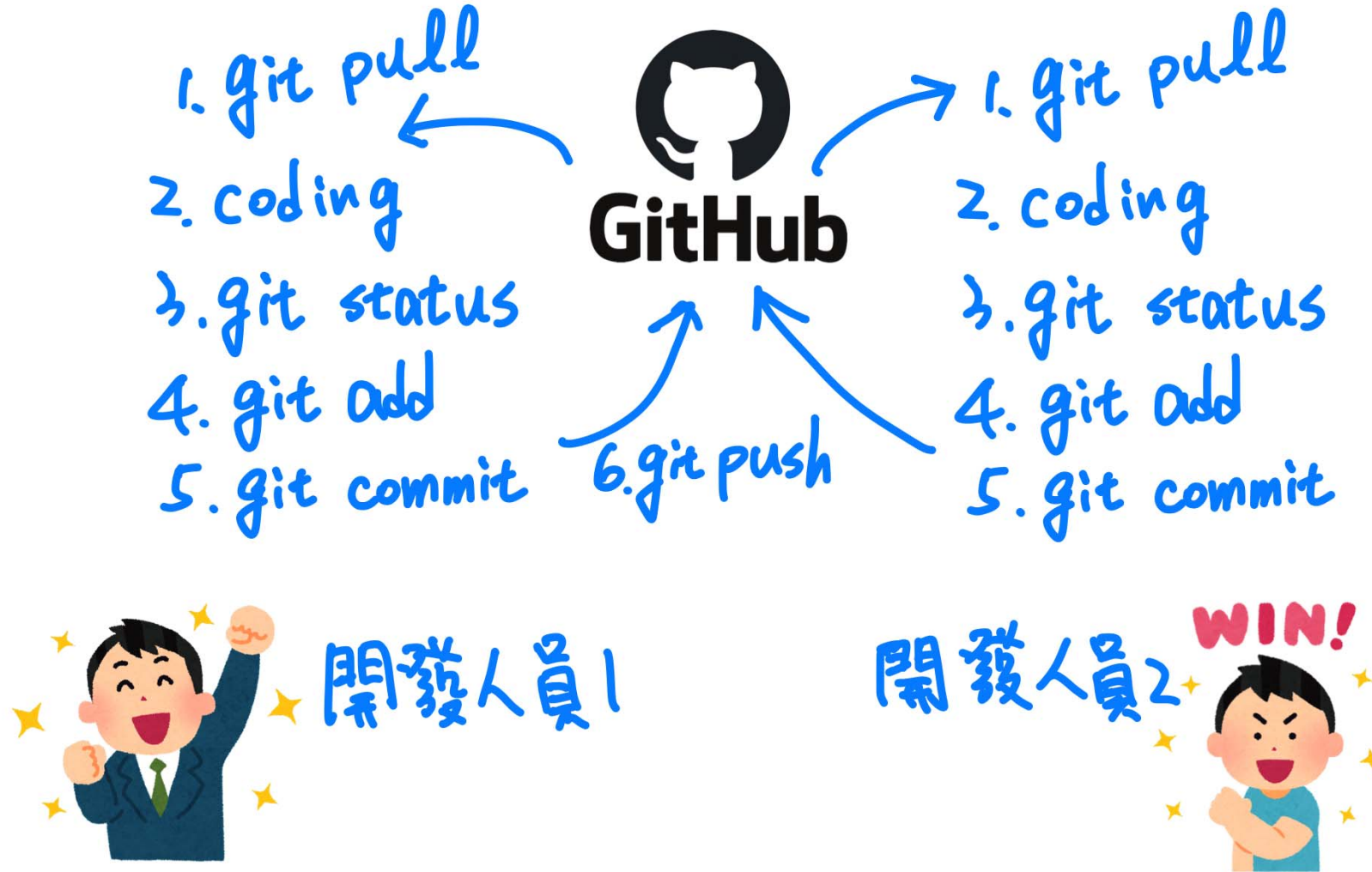
- git pull [遠端空間的名稱(預設origin)] [遠端空間的分支(預設master)]：從最初克隆的伺服器上獲取資料，然後試著自動合併到目前的分支

■ Push 「推送」

- git push [遠端空間的名稱(預設origin)] [遠端空間的分支(預設master)]：將 master 分支推送到 origin 伺服器上時(克隆時通常會自動地幫你設定好 master 和 origin 這二個名稱)

■ 指令 git remote show origin：檢視遠端儲存庫資訊

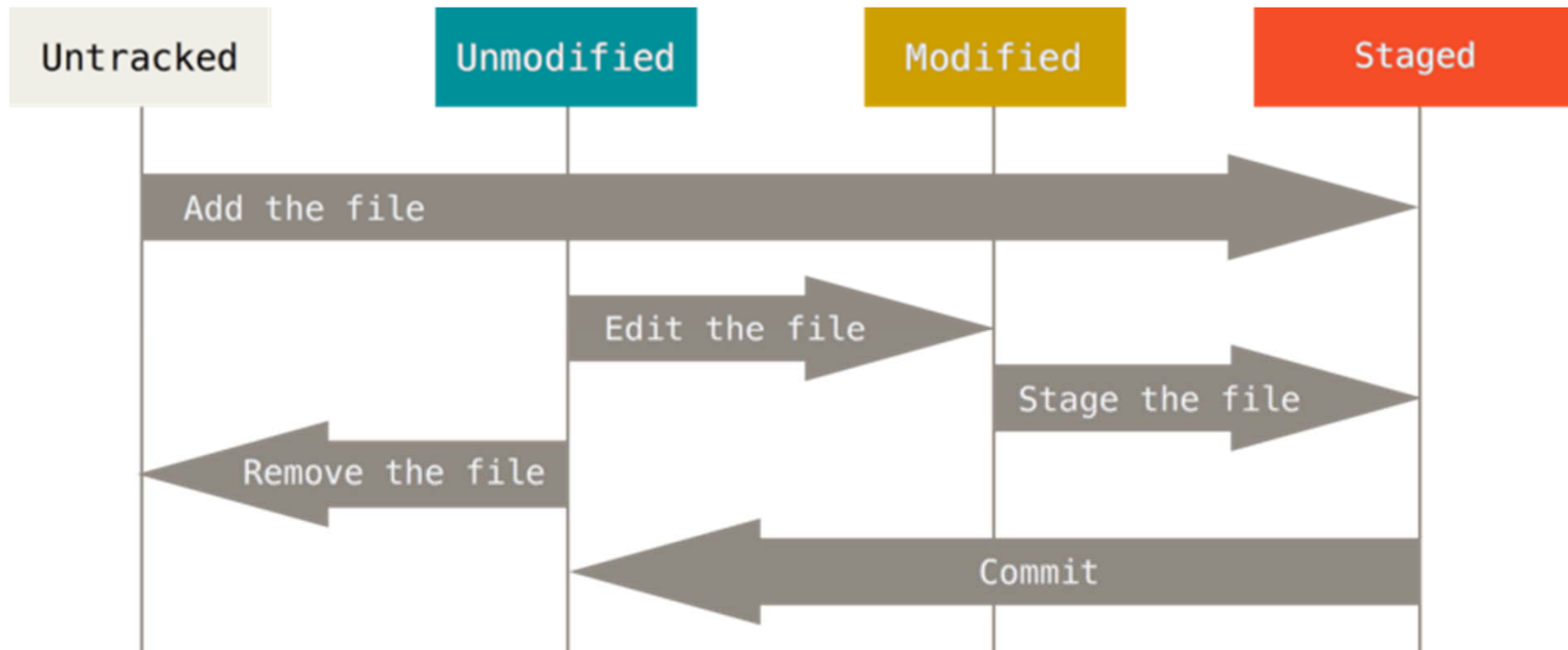
現實場景



狀態指令

■ 狀態指令 git status

□ 檢視檔案在什麼狀態(如圖)



追蹤檔案

- `git add .` : 會將所有工作的更動加入到暫存區內
 - (包括文件內容修改以及新文件，以及刪除的文件)
- `git add --u` : 只會加入將上一個版本已經被加入的檔案
 - (不包含新文件)
 - 也就是說這次有修改過任何檔案將會被加入暫存區
- `git add --A` : 為上面兩個功能的綜合版
 - 即為(`git add --all`)的縮寫

Git的提交流程



<https://kopu.chat/2017/01/18/git%E6%96%B0%E6%89%8B%E5%85%A5%E9%96%80%E6%95%99%E5%AD%B8-part-1/>

Commit常用的指令

■ 省略git add步驟直接提交

- 輸入指令：`git commit --a`

■ 編輯上一次提交的訊息

- 輸入指令：`git commit --amend`

■ 不進入文字編輯模式

- 輸入指令：`git commit -m [訊息內容]`

■ 在版本訊息內簽名

- 輸入指令：`git commit -s`

ignore files 忽略檔案的時機與用處

- 當使用 `git add .` 忽略掉特定文件
 - 不需要逐個文件 `add`
- 當上傳到 Git 與他人協作時
 - 避免與他人的配置有衝突
- 臨時文件或日誌文件或編譯中間文件
 - 一些不需要提交到資料庫中的文件

常用的忽略指令

■ 一次忽略多個檔案

- 在文字編輯模式內加入米字號 例如 (*.exe)

■ 忽略某幾種特定檔案

- 在編輯器內輸入*.[] 例如: *.*[oa] 此時 *.a 及 *.o會同時被會忽略

■ 使用 `git add -f <file>` 強制 add 被忽略的檔案

- 某一些情況，必須要提交已被忽略的檔案，就可以使用這個 -f 參數，強制加入檔案

檢查文件是否被忽略

- `git status --ignored` 利用此指令檢視並除錯
 - `git status --ignored` 將列出被忽略的文件名
- `git status -u`
 - 使用 `-u` 參數完整顯示所有資訊
- `--untracked-files=all` 使用此參數
 - 檢視目錄底下的被忽略的文件

忽略的規則寫法

■ 將已忽略檔案類型加入單一檔案的例外

- *.txt
! 1.txt

■ 將已忽略資料夾中不忽略指定資料夾

- /pack/*
! /pack/icon/*

忽略檔案的原則

- 忽略作業系統自動生成的檔案

- 像是縮圖或日誌和IDE的配置文件

- 忽略編譯生成的中間檔案、執行檔

- 通常這種檔案會由另外一個檔案生成

- 忽略使用者個人帶有敏感資訊的配置檔案

- 例如一些個人資料或密碼表單

編譯忽略範本

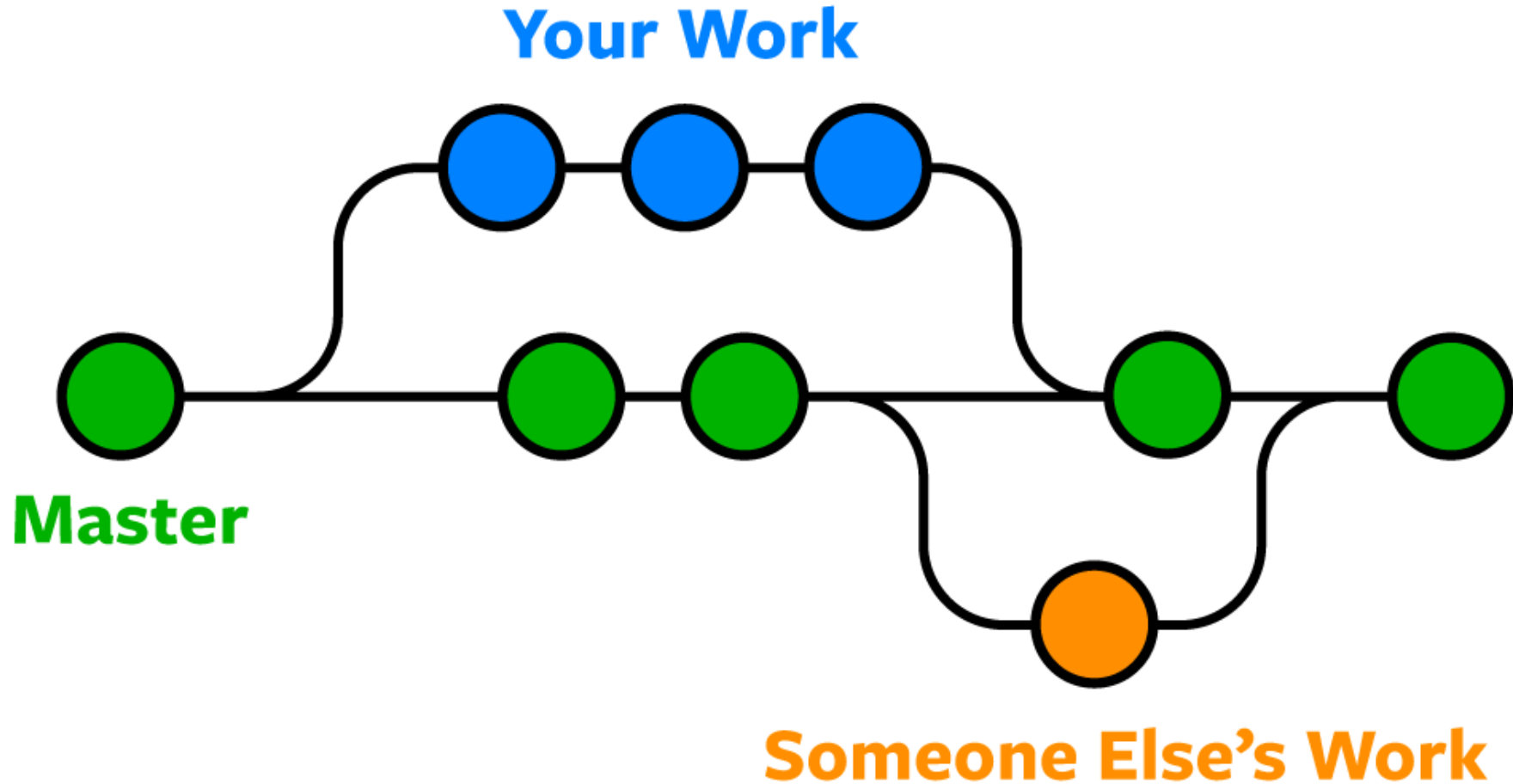
- GitHub官方網站提供的範本有很多版本可供下載

- <https://github.com/github/gitignore>

- 編譯自己的忽略列表

- 要注意忽略規則的優先順序

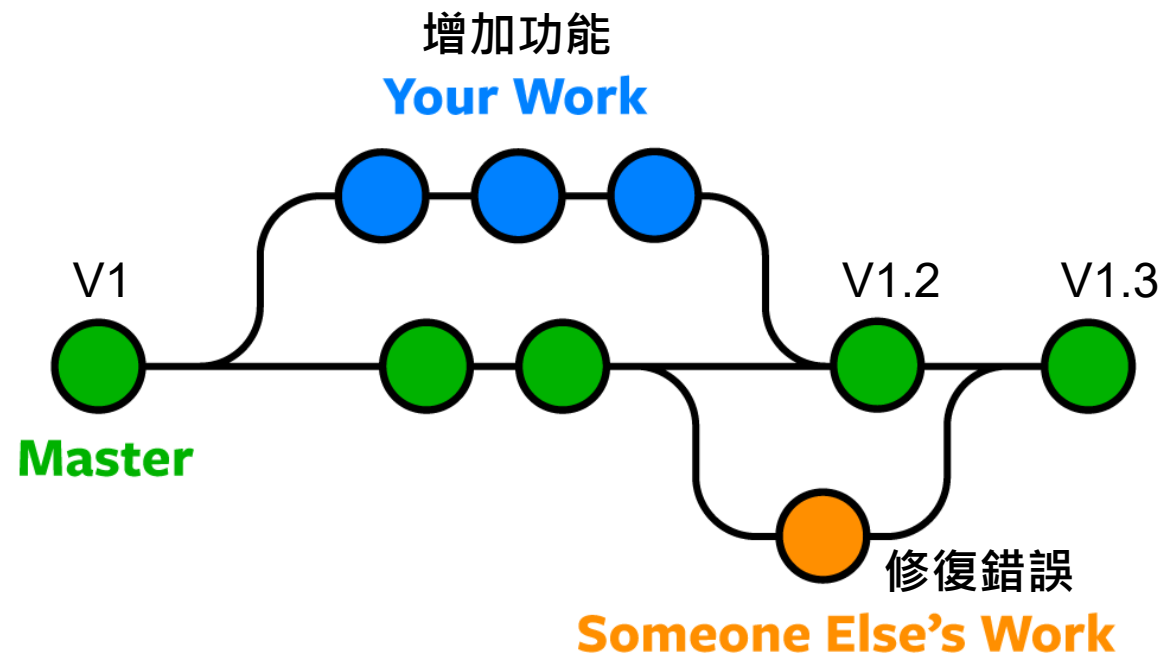
Branch 分支



<https://www.nobledesktop.com/learn/git/git-branches>

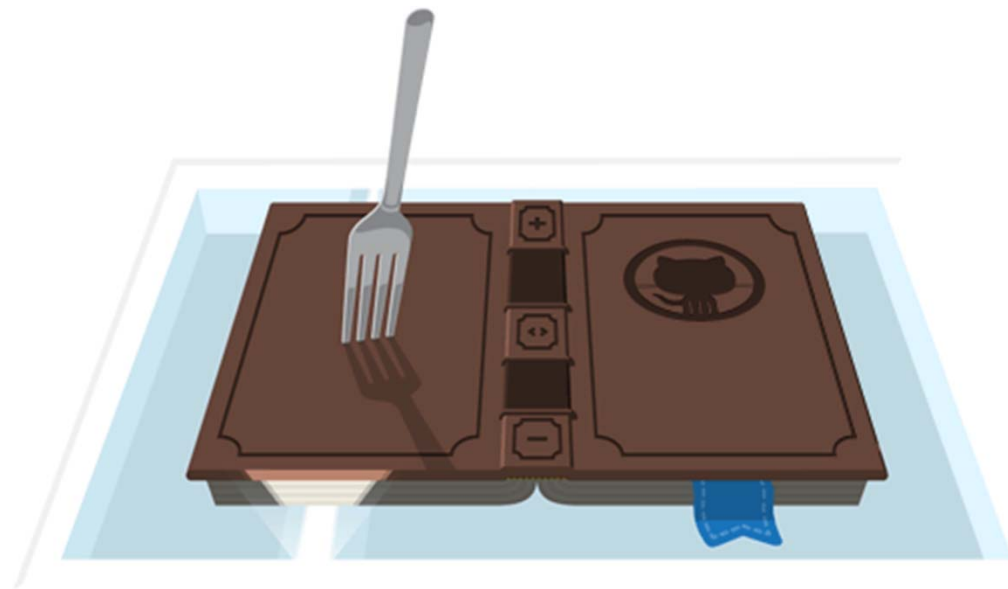
何謂分支

- 分支是為了將修改記錄的整體流程分開儲存
 - Git分支本質上是獨立的開發線。
- 只要不同的分支屬於同一個存儲庫，就可以將它們合併到任何一個分支中

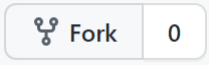


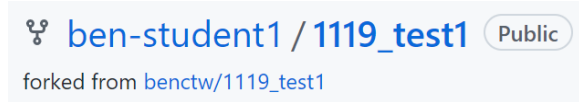
Fork

- Github的Fork功能就是拷貝一份別人的儲存庫到自己的帳號中
- 想對別人的開源專案儲存庫做貢獻(改bug、加功能)，但是沒有權限修改別人的儲存庫
- 可以先Fork到自己的帳號中，修改完再送 Pull Request給對方



實際操作Fork與Pull Request -1

- 點選別人專案的Fork按鈕 ，選擇要目的地做Fork
- git clone [剛剛Fork回自己帳號的專案網址] [新建資料夾名稱]
- git remote -v：檢視遠端資訊
- git remote add [原儲存庫，預設upstream] [原儲存庫的網址]
- git remote -v：檢視遠端資訊
- Git fetch upstream
- Git pull upstream master



實際操作Fork與Pull Request -2

- 修改程式
- Git add .
- Git commit -m "update README.md"
- Git push : push到fork的儲存庫
- 再回到Fork的儲存庫，由於比原始儲存庫多出一次更新，詢問是否發PR

This branch is 1 commit ahead of benctw:master.

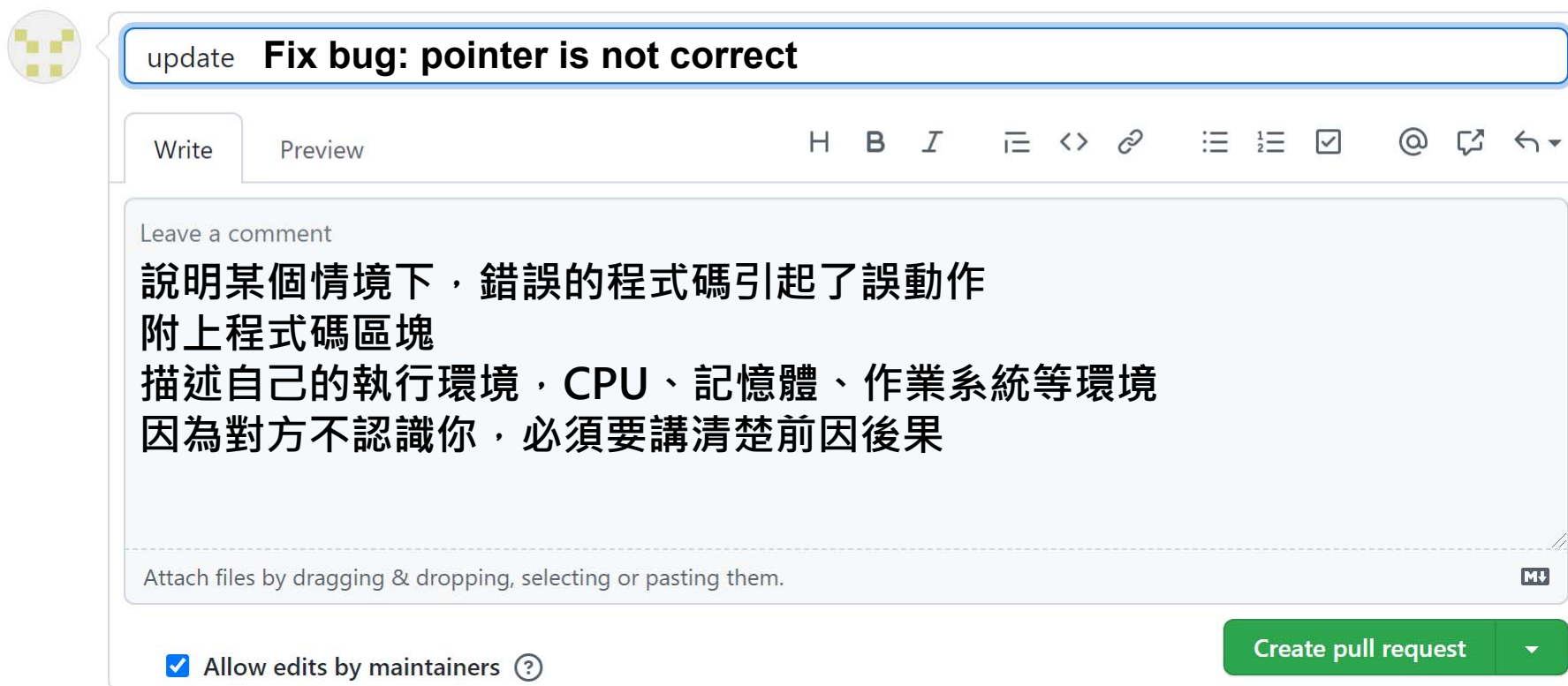
 Contribute ▾  Fetch upstream ▾

- 按下「Contribute」>「Open pull request」啟動PR
- 確認來源與目的的分支，確認修改內容後，按下

Create pull request









實際操作Fork與Pull Request -3

■說明發起PR的原因




update **Fix bug: pointer is not correct**


Write Preview

H B I       @  

Leave a comment

說明某個情境下，錯誤的程式碼引起了誤動作
附上程式碼區塊
描述自己的執行環境，CPU、記憶體、作業系統等環境
因為對方不認識你，必須要講清楚前因後果



Attach files by dragging & dropping, selecting or pasting them. 

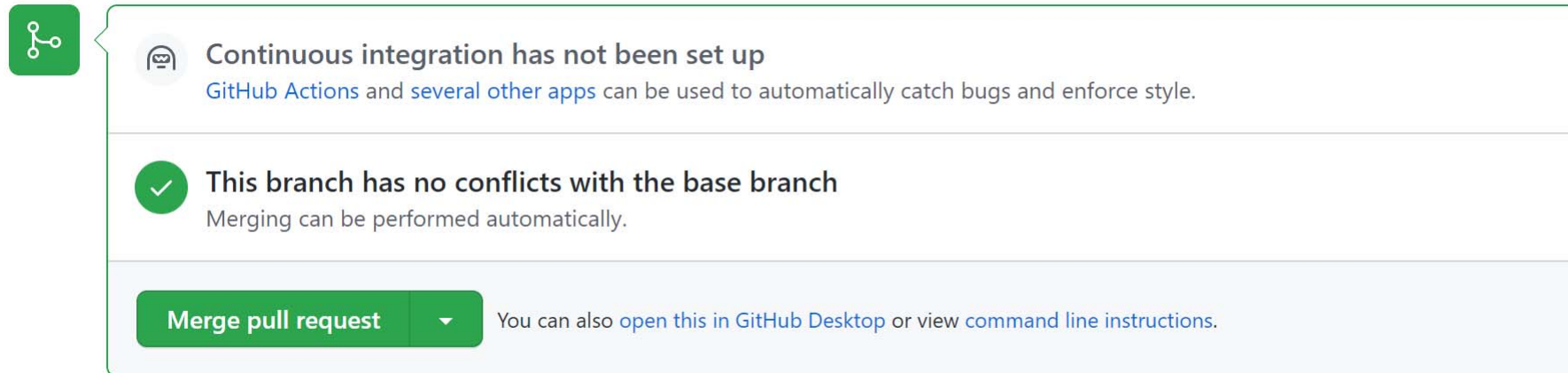
☒ Allow edits by maintainers 

Create pull request

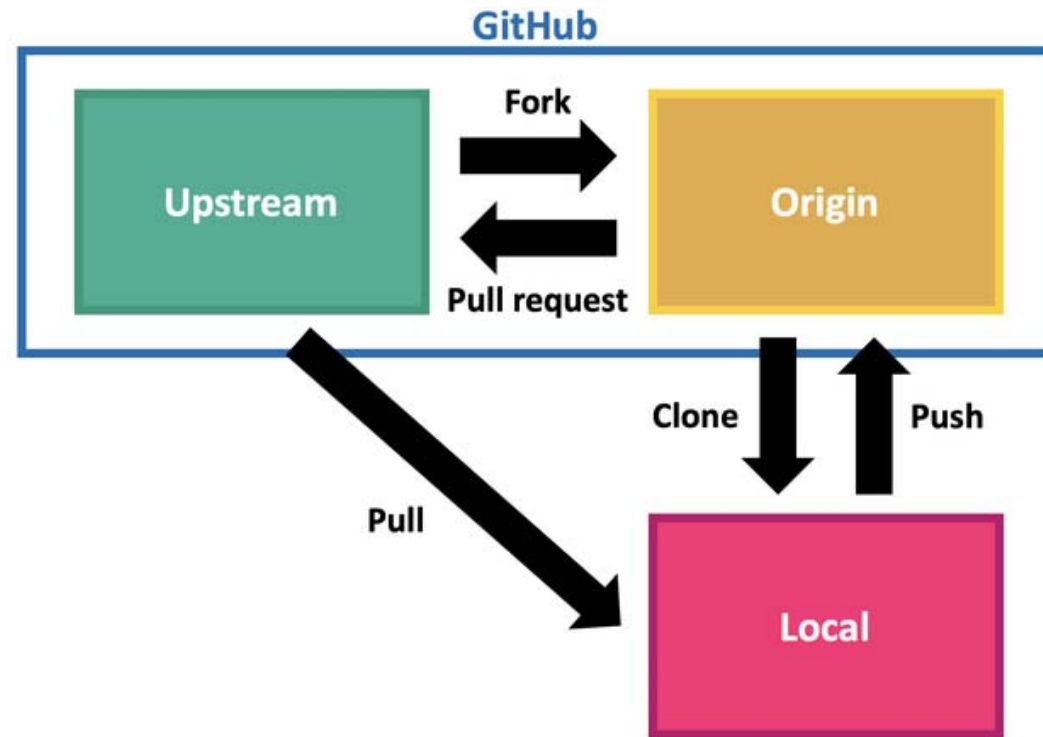
■按下「Create pull request」

實際操作Fork與Pull Request -4

- 原作者在原儲存庫看到PR訊息： ben-student1 wants to merge 1 commit into `benctw:master` from `ben-student1:master` 
- 原作者閱讀標題內文，檢查程式碼，可能會有討論交流
- 經過一番討論後，原作者可能會將程式merge進專案



Fork與Pull Request資料流程圖



<https://www.tomasbeuzen.com/post/git-fork-branch-pull/>

本課程使用Github的時機

1. 下載講義
2. 繳交個人作業
3. 繳交分組討論會議紀錄(使用PR)
4. 分組專題實作練習
5. 參與公開社群互動討論