

BreakIt 專案設定

喬逸偉 (Yiwei Chiao)

1 簡介

[Node.js](#)¹ 是以 Google Chrome V8 engine 為核心打造的一個 Javascript ([ECMAScript](#)²) 執行環境。在 [Node.js](#) 的加持下，Javascript 脫離了 Web Browser 的羈絆，有了更廣闊的揮灑舞台。

[Breakout](#)³ (打磚塊) 是 1976 由 Atari Inc. 發行的一款電子遊戲。玩家操縱設於螢幕下方的一塊長方形檔板 (paddle)，將上方落下的球反彈，以擊破 (落) 列於螢幕上方的磚牆。將磚牆的全部磚塊擊落，則過關。

[Breakout](#) 遊戲推出後，因為規則簡單，容易上癮 (addictive)，成為電子遊戲經典之一；並衍生成一個獨特的遊戲類型 (genre)。

作為一個 [Node.js](#)/Javascript 的練習專案，[BreakIt](#)⁴ 嘗試撰寫以 [Node.js](#) 為基礎的 Web 版 [Breakout](#) 遊戲。

2 開發工具

[BreakIt](#) 使用 [Node.js](#) 撰寫。[Node.js](#) 本身是一個龐雜的生態系 (eco-system)，如果之前沒有開發過 [Node.js](#) 的專案，我們需要先安排好它的工作環境。基本上，我們會需要以下的工具 (tools)：

2.1 [Node.js](#):

Javascript/[ECMAScript](#) 的工作的運作引擎。

¹<https://nodejs.org>

²<https://en.wikipedia.org/wiki/ECMAScript>

³[https://en.wikipedia.org/wiki/Breakout_\(video_game\)](https://en.wikipedia.org/wiki/Breakout_(video_game))

⁴<https://github.com/ywchiao/breakit>

2.2 Babel⁵:

Javascript 的 transpiler；Javascript 在 2009~2011 年間迎來了期待已久的真正意義上的標準化改變，但是既存的程式需要維護，瀏覽器 (browser) 的支援需要時間趕上。於是我們需要一個能將以新版 Javascript (ECMAScript 6/7) 語言撰寫的程式轉譯 (transpile) 成瀏覽器/Node.js 能理解的 Javascript 的工具。

Babel 就是這樣的一個工具。

2.3 rollup.js⁶:

rollup.js 是一個能將多個.js 檔案打包 (pack) 成單一檔案，節省瀏覽器下載時間的工具；類似的工具早期有 browserify⁷，近期當紅有 webpack⁸ 等。

BreakIt 採用 rollup.js。

開始時提過了，Node.js 擁有一個龐雜的生態系，不同的問題常常都有多個不同的解決方案可供選擇，沒有對錯，只是要小心亂花迷眼。

2.4 git⁹:

git 是 Linus Torvalds¹⁰ (是，就是 Linux Kernel 的原作者) 給現代的程式設計師 (programmer) 的另一個禮物；一個功能強大而又易用的版本管理系統 (Version Control System¹¹)。

在 BreakIt 專案裡，我們將使用 git 來管控專案的發展。

2.5 GitHub¹²:

GitHub 不是一個工具，它是一個網站，一個雲服務。

顧名思義，GitHub 是以 git 為基礎架設的網路服務；無論如何，它是目前最熱的開源軟體集散地；包括臉書 (Facebook)，領英 (Linkedin)，亞馬遜 (Amazon)，谷歌 (Google)，所謂的 FLAG 就業首選，和蘋果 (Apple)，微軟 (Microsoft) 都將它們開源的軟體放在 GitHub 上，就可以知道它的熱門程度。

⁵<https://babeljs.io>

⁶<https://rollupjs.org>

⁷<http://browserify.org>

⁸<https://webpack.github.io>

⁹<https://git-scm.com>

¹⁰https://en.wikipedia.org/wiki/Linus_Torvalds

¹¹https://en.wikipedia.org/wiki/Version_control

¹²<https://github.com>

對程式設計師而言，因為 [GitHub](#) 承載了大量的開源專案，所以已成為學習，分享，交流，認識世界同時也被世界看到的場域。所以，儘早加入這個程式設計師的社群網絡，對程式設計師的職涯發展絕對是正向的影響。

[BreakIt](#) 的源碼當然也放在 [GitHub](#) 上。對開源軟體而言，[GitHub](#) 的服務是免費的；而我們的練習專案當然是開源的。所以實在沒有理由不去登錄一個 [GitHub](#) 的帳號。

隨著專案的進展，我們也將慢慢地熟悉 [git/GitHub](#) 的使用。

2.6 [Atom](#)¹³ (選擇性):

[Atom](#) 是 [GitHub](#) 推出的，以 [Node.js](#) 打造的開源文字編輯器 (editor)；v1.21 版之後，更和 Facebook 合作將它擴張成一個完整的 IDE¹⁴。

關於 [Node.js/JavaScript](#) 可以作些什麼，[Atom](#) 作了一個強而有力的見證；類似的，微軟 (Microsoft) 推出了以 [Node.js/TypeScript](#)¹⁵ (微軟版 JavaScript) 開發的 [VS Code](#)¹⁶。

2.7 [terser](#)¹⁷ (選擇性):

[terser](#) 是以 JavaScript 撰寫的 JavaScript 程式碼混淆工具，最小化工具，壓縮工具，最佳化工具。

因為瀏覽器端執行的 JavaScript 程式需要由伺服器端下載，所以程式的大小愈小愈好，如此可以減少網路流量的使用，加快下載速度。[terser](#) 就是為這個目的設計的工具，透過移除不會使用到的程式碼，變數改名，程式碼壓縮等動作，產生的結果和原始輸入可以有三到四倍的差異。

3 環境設定

3.1 [Node.js](#):

故事總是由 [Node.js](#) 的安裝開始。依據作業系統的不同，有不同的流程。

¹³<https://atom.io>

¹⁴https://en.wikipedia.org/wiki/Integrated_development_environment

¹⁵<https://www.typescriptlang.org>

¹⁶<https://github.com/Microsoft/vscode>

¹⁷<https://github.com/terser-js/terser>

3.1.1 Windows:

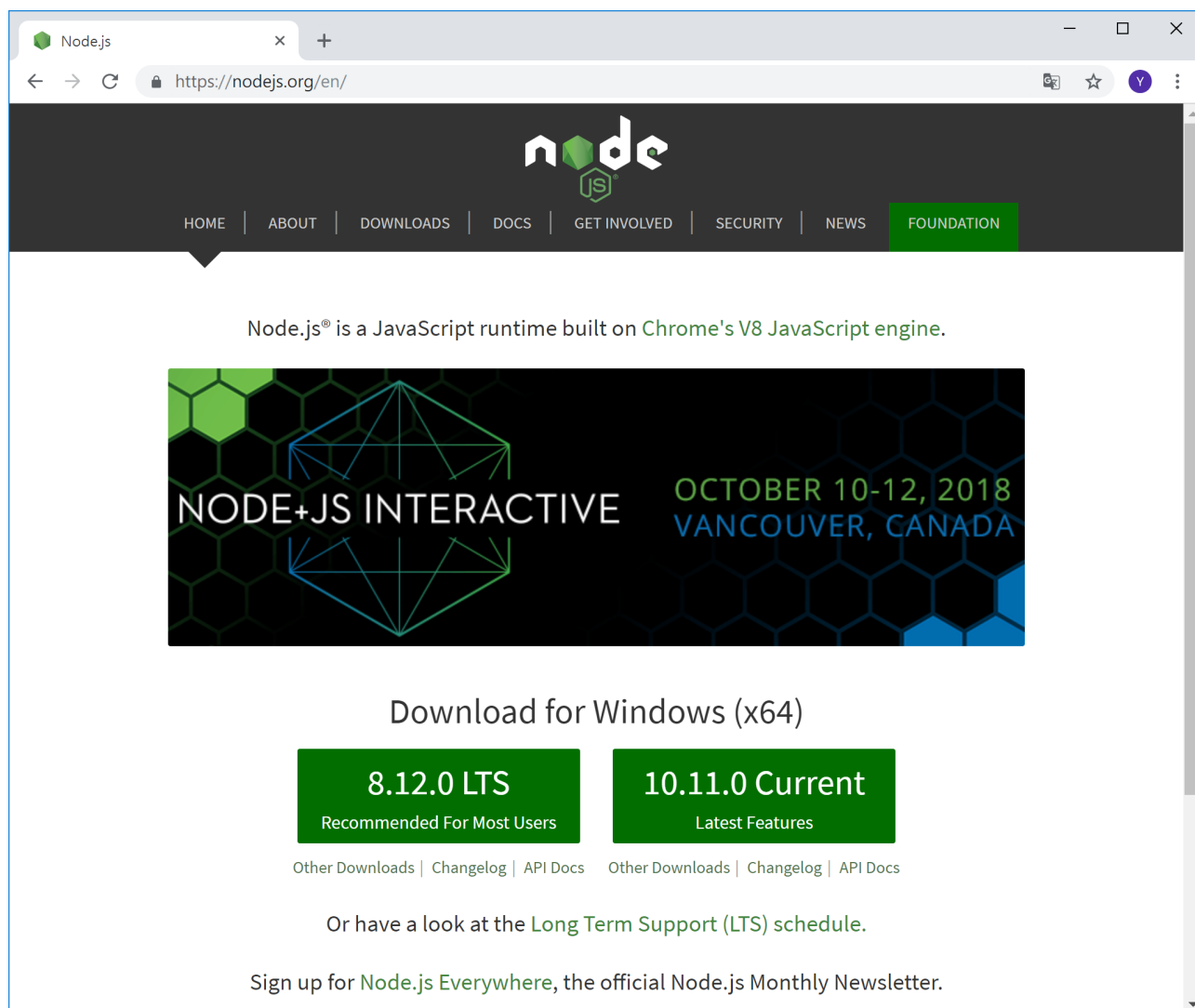


Figure 1: Nodejs 下載頁面

Figure 1 是 Node.js 的首頁；點擊 Figure 1 畫面右下的 [Current] 就可以取得當前版本的 Node.js。下載的檔案是 .msi 檔，直接選執行，接受授權條款，選擇安裝位置，就完成了。

Node.js 因為持續演進，所以一直保持著同時有兩 (2) 條版本線在發行：*LTS*: Long Term Support (長期維護) 版本和 *Current* (目前開發) 版本。

原則上，每個版本的開發周期 (current) 是六 (6) 個月；開發周期結束，如果是 *LTS* 版本，那麼會有額外的十八 (18) 個月的生命週期；如果，不是 *LTS* 版本，那麼直接結束，不會再有任何更新。*LTS* 版本生命週期結束後，還會有十二 (12) 個月的維護週期，也就是會收到 bug 修復更新；再之後才結束。

一般而言，偶數版號的版本會是 *LTS* 版本，而奇數版本的版本不是。

3.1.2 Linux/MacOS:

如果電腦系統是 Linux/MacOS，因為 Node.js 演化快速，建議安裝 `nvm`¹⁸ (Node Version Manager)，再透過 `nvm` 安裝管理不同版本的 Node.js。

3.1.3 MacOS/Homebrew¹⁹:

當然在 MacOS 上如果不需要在多個 Node.js 版本中切換，那麼直接用 Homebrew 來安裝 Node.js 也是一種方法。

注意，`nvm` 團隊有聲明**不**支援 Homebrew 安裝；所以，不要混用 `nvm` 和 Homebrew。任選一種方式就好。

3.2 git:

BreakIt 練習專案使用 `git`/GitHub，所以我們需要 `git`：

¹⁸<https://github.com/creationix/nvm>

¹⁹<https://github.com/Homebrew/brew>

3.2.1 Windows:

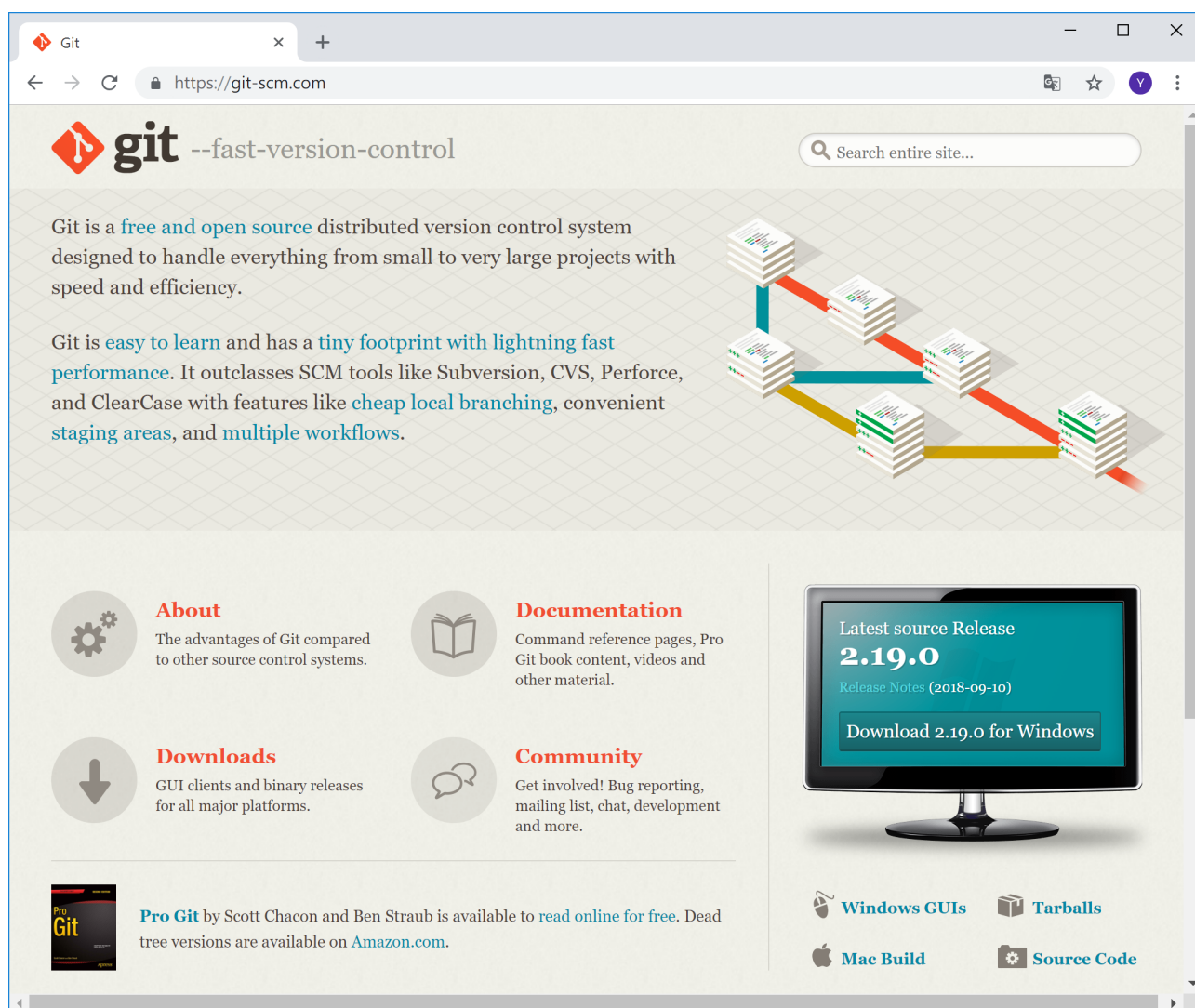


Figure 2: Git 下載頁面

Git 的首頁如 Figure 2；點擊 Figure 2 右下的 *Downloads for Windows* 就可以開始下載。

下載後執行，基本上如果你不知道它問的選項是什麼意思就按下一步就好；反之，如果你知道它選項的意思，那你也已經知道你要選什麼了。

- `cmdr`²⁰:

因為 Windows 的**命令提示字元**真的只能說是堪用而已，而 `git` 基本上又是個 *CLI*²¹ (Command-Line Interface，相對於 *GUI*²²: Graphical User Interface) 工具；所以，既然都已經要下載安裝 `git` 了，不如順便就下載安裝一個好用點的命令提示字元工具，`cmdr`。

²⁰<https://github.com/cmdrdev/cmdr>

²¹https://en.wikipedia.org/wiki/Command-line_interface

²²https://en.wikipedia.org/wiki/Graphical_user_interface

岔題，其實有不少 *GUI* 工具都整合/支援了 *git* 的功能。比如說，GitHub 推出的 *atom*。只是，如果不熟悉 *git* 操作的話，看到 *GUI* 上的一堆操作可能還是墜於五里霧中，不知道什麼是什麼。所以，還是建議由 *CLI* 入手，之後再由 *GUI* 來簡化操作。

這裡有兩 (2) 個選擇，

1. 完整版 *cmder* + *git*: *cmder* 的作者很貼心的將 *git* 整合到 *cmder* 的發行檔內，省去使用者另外下載安裝的繁瑣。
2. 精簡版 *cmder mini*: 如果已經安裝了 *git*，或者想自行安裝維護 *git*，可以選擇 *mini* 版，只有 6M 大小。

不管那個選擇，下載的都是壓縮檔。下載完，解壓縮到喜歡的資料夾，如 Figure 3，最後再將 Figure 3 中的 *cmder.exe* 建立一個捷徑，放到桌面上方便取用就行了。

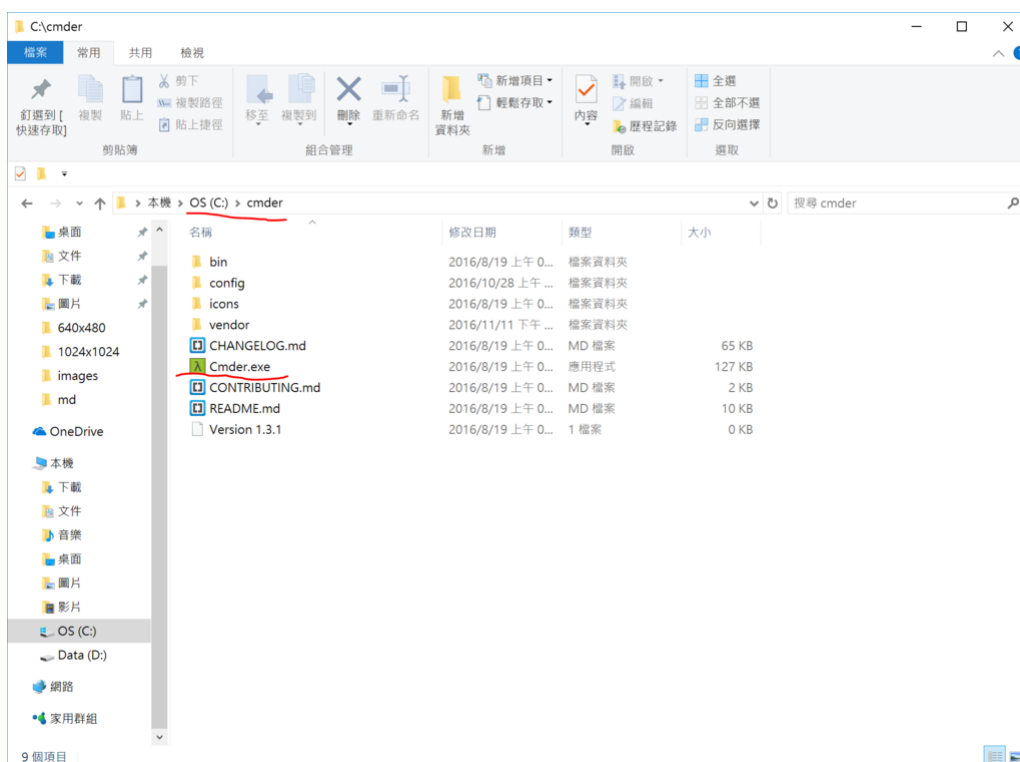


Figure 3: Cmder 資料夾畫面

3.2.2 Linux/MacOS:

如果是 Linux/MacOS 作業環境，其基本上系統都內建 *git*。如果在 MacOS 環境下找不到 *git*：

1. 請記得 *git* 是 *CLI* 工具，到 *terminal* 下去執行。
2. 先確認有沒有安裝 *Xcode*²³，Apple 的官方 IDE；其次，在 *Xcode* 的 *Preferences* 裡確認有沒有安裝 *Xcode Command-Line Tools*。

Linux 的話。各發行版都不同，基本上都整合在關於程式開發的套件內，並預設安裝。如果在 *terminal* 裡找不到，先到發行版的套件庫裡找找。

²³<https://developer.apple.com/xcode>

3.3 atom (選擇性):

atom 是 GitHub 推出的開源編輯器/IDE，內建支援 git/GitHub 是自然的事；

如果不習慣在命令列下工作，比如說使用 git，可以試試 atom 或其它的 ide。

breakit 專案本身並不仰賴在 atom。比如說，雖然這份文件是用 atom 寫作完成的，但我個人一般還是習慣用 Vim²⁴ 工作。

原則上，這樣環境設定就完成了。之前工具介紹提到的 Babel，rollup.js 等，都會在專案進展到那兒時，透過 Node.js 隨附的 npm²⁵ (Node Package Manager) 安裝。

4 GitHub 基本使用

4.1 登錄 (sign up) GitHub 帳號

1. GitHub 的登錄畫面很簡單，如 Figure 4，僅需要帳號名稱，電子信箱，和密碼。

²⁴<https://vim.sourceforge.io>

²⁵<https://www.npmjs.com>

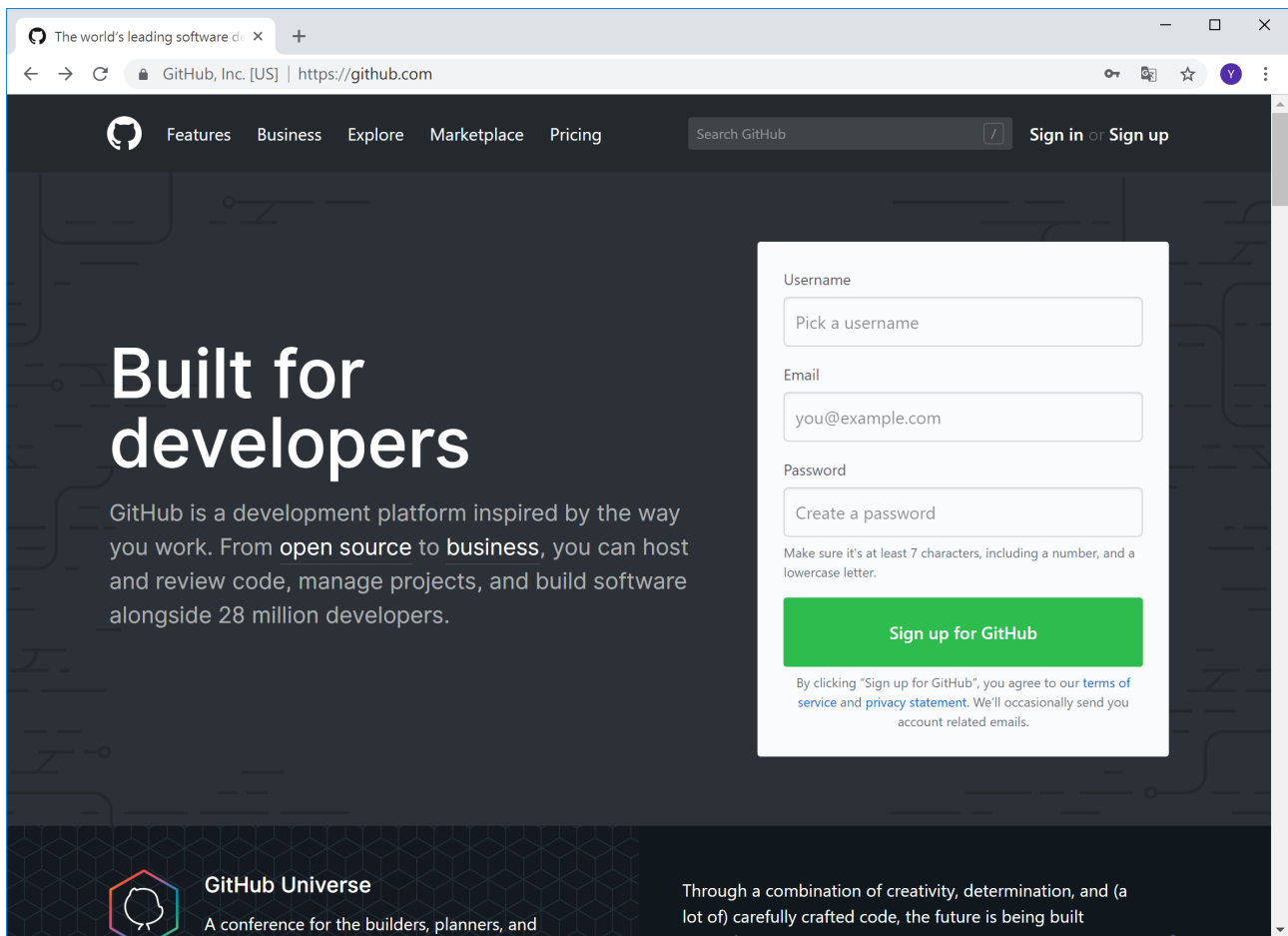


Figure 4: github 登錄畫面

登錄完記得回信箱收認證信。

2. 用新建的帳號登入後，因為是首次登入，GitHub 會讓你選擇服務類型：

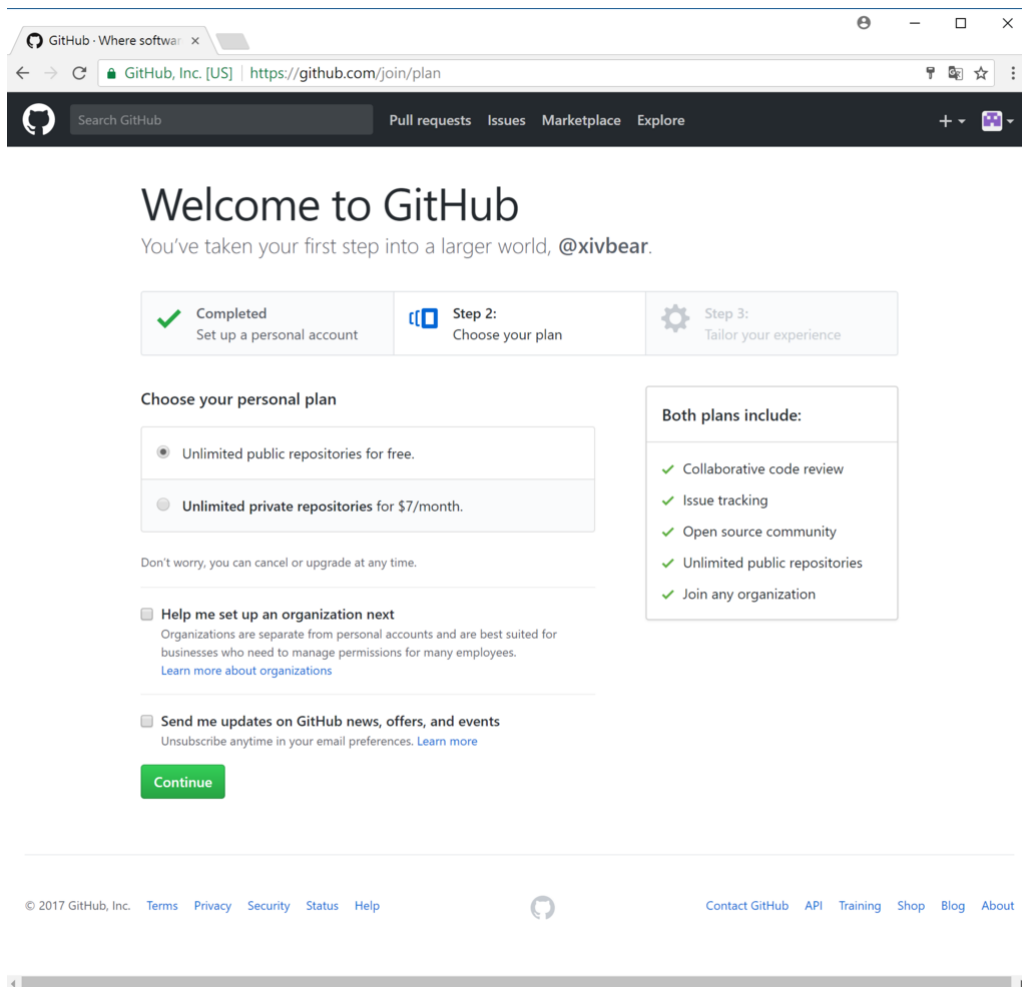


Figure 5: github 服務類型

GitHub 預設是免費，無限的開源專案；如果你想付點錢將專案藏起來，它也有選項給你。

按 Figure 5 左下方綠色的 [Continue] 繼續。

3. 第三個畫面 (Figure 6) 會問你一些簡單，不涉隱私的資料，可以選擇回答，也可以跳過。

GitHub - Where software starts

GitHub, Inc. [US] | <https://github.com/join/customize>

Search GitHub Pull requests Issues Marketplace Explore

Welcome to GitHub

You'll find endless opportunities to learn, code, and create, @xivbear.

✓ Completed
Set up a personal account

📁 Step 2:
Choose your plan

⚙️ Step 3:
Tailor your experience

How would you describe your level of programming experience?

☐ Totally new to programming ☒ Somewhat experienced ☐ Very experienced

What do you plan to use GitHub for? (check all that apply)

☒ Project Management ☒ School projects ☐ Design
☒ Development ☒ Research ☐ Other (please specify)

Which is closest to how you would describe yourself?

☐ I'm a hobbyist ☐ I'm a student ☒ I'm a professional
☐ Other (please specify)

What are you interested in?

game-development web-development open-source

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

[Submit](#) [skip this step](#)

Figure 6: github 問卷

4. 終於到最後一個畫面了，

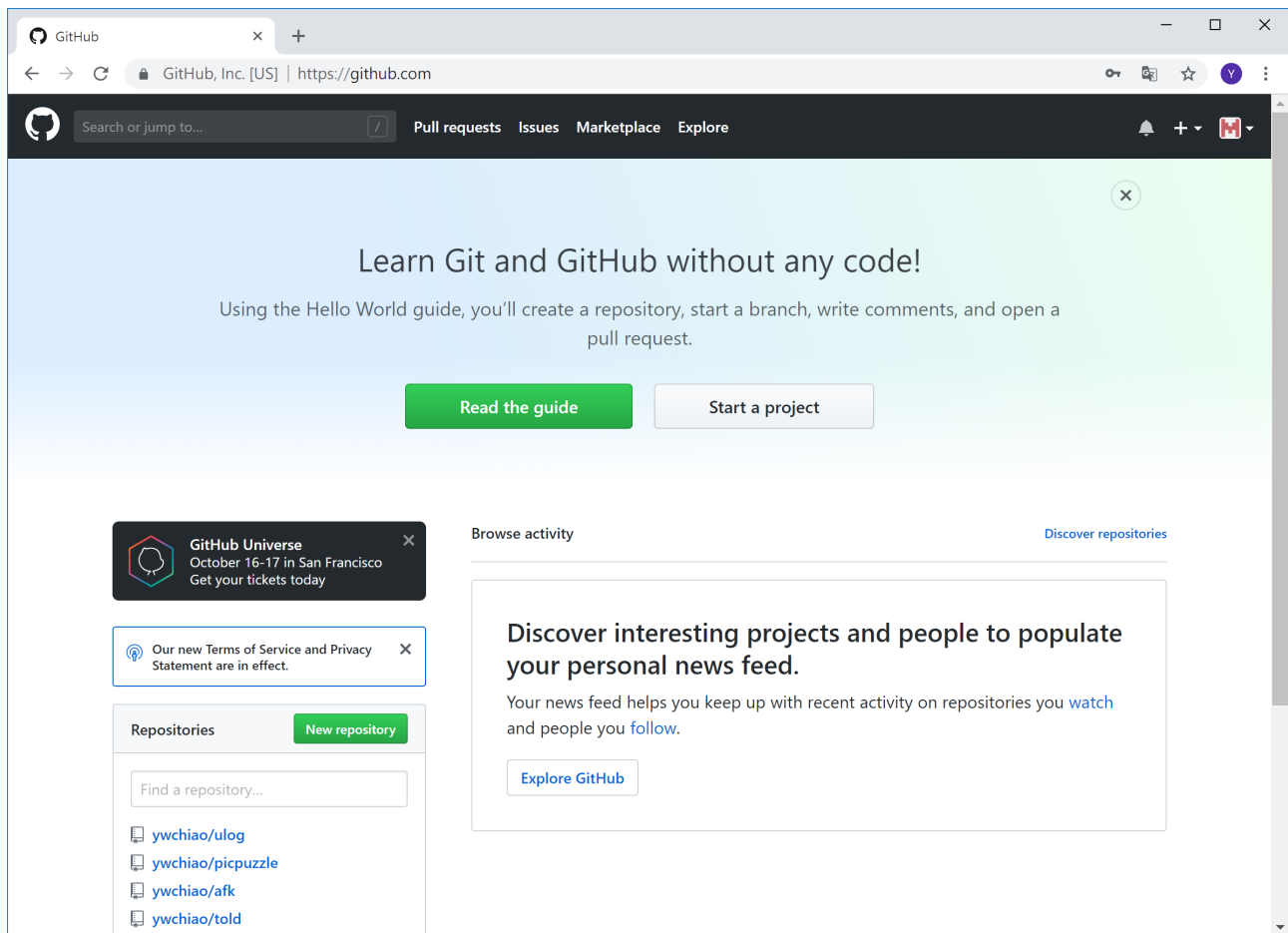


Figure 7: github 教學或建立專案

Figure 7 左邊的按鈕是 [GitHub](#) 教學，右邊則是 [建立專案]，選擇 [建立專案]。如果之前還沒有去登錄的信箱收取認證信，這時你會看到 Figure 8：

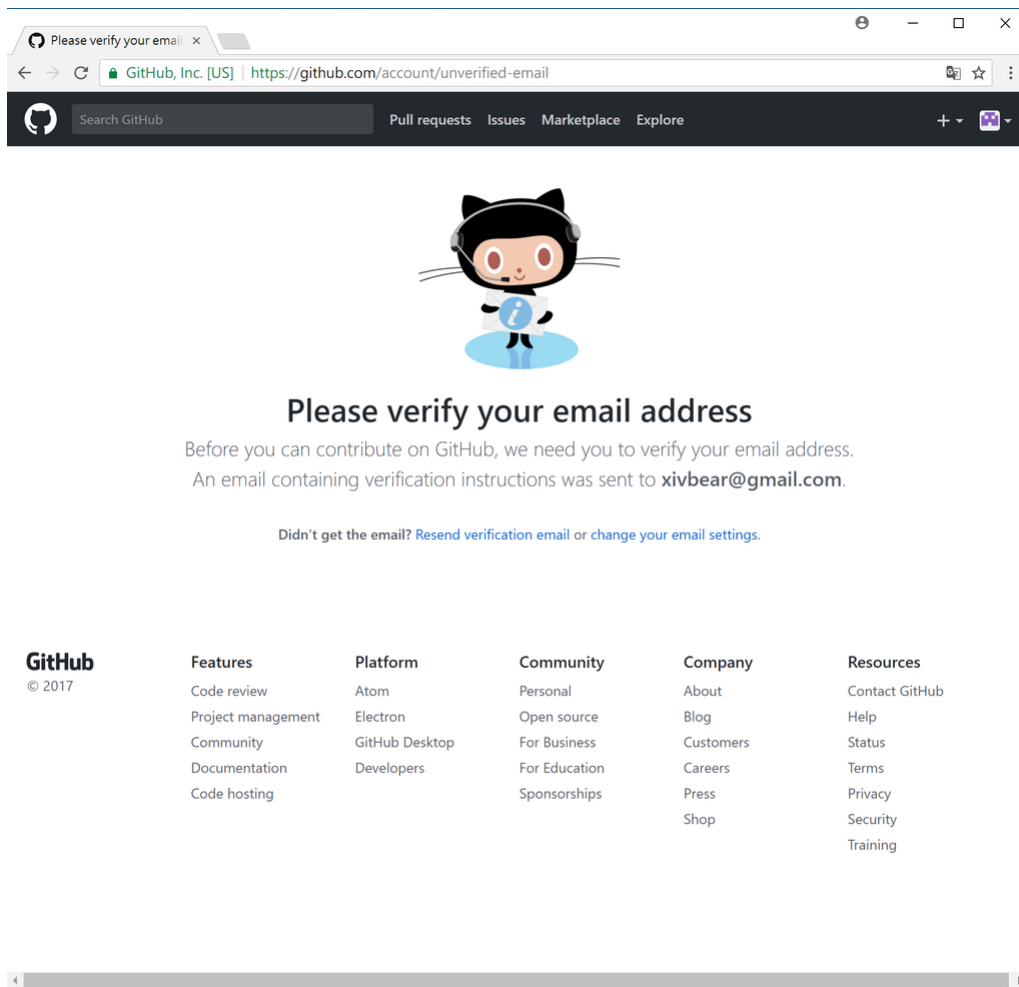


Figure 8: github 要求認證畫面

記得回登錄的信箱去收認證信。

4.2 建立專案 (project)

在 [GitHub](#) 建立新專案，也稱作建立一個 *repository*，其實就是在檔案系統裡建立個存放程式碼相關檔案的資料夾 (folder)。建立專案的畫面如 [Figure 9](#) 所示。

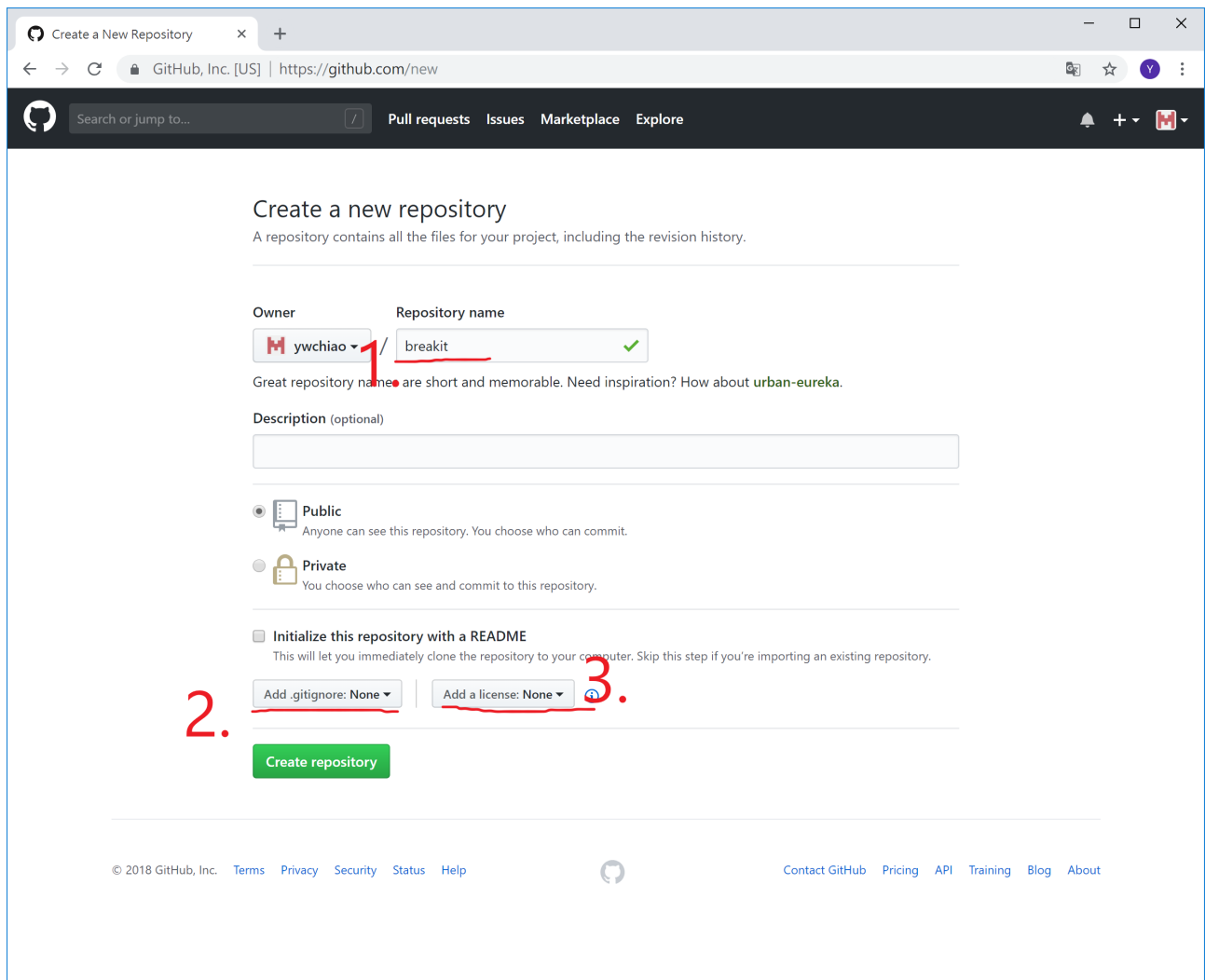


Figure 9: github 建立專案畫面

1. 首先輸入 [專案] 名稱
2. 選擇 `.gitignore`²⁶ 檔案；這裡 GitHub 貼心的幫忙準備了不同語言專案的通用 `.gitignore` 檔。下拉選擇 Node。 `.gitignore` 檔案的作用與目的後面說明。
因為 `breakit` 是 Node.js 的專案，所以這裡先選擇 Node
3. 授權方案 (License)。如果不知道該選那種授權或想多少理解一些不同授權的差別，旁邊的 *i* 按下去，有簡單的白話說明。個人一般選擇 MIT²⁷。

在 Figure 9 按下 *Create Repository* 之後，就會進入專案的主頁 (homepage) 畫面。

4.2.1 `.gitignore`

`.gitignore` 檔案一個簡單的文字檔，用來記錄不需要放在 *repository* 裡的檔案，檔案類型，與

²⁶<https://git-scm.com/docs/gitignore>

²⁷<https://opensource.org/licenses/MIT>

資料夾等資訊。

舉例而言，C/C++ 的.o/.obj 檔案，Java 的.class 檔案，一般意義的 tmp/ 資料夾，都是專案進行/程式編譯過程中，由工具產生的過渡產物，和我們的工作沒有直接關係，因此沒有必要放到 *repository* 內。

因為不同的程式語言，不同的工具會有不同的過渡產出，git 不可能事先知道，所以 git 將決定權放到使用者手裡，由使用者編輯 .gitignore 檔案，告訴它那些檔案，資料夾是不重要，不需要管理的。

.gitignore 檔名前面那個句點 (*period, dot*)，**不是**打錯字，它是檔名的一部份。

檔名由句點開始，在 Linux/MacOS 環境下代表**隱藏檔** (*hidden*)，意思是當使用者下 dir 或 ls 這類列出資料夾內容的指令時，系統**不會**顯示它的存在。

而在 Windows 環境下，這個句點**沒有**作用，就只是檔名的一部份而已。

4.3 專案主頁

GitHub 上每個專案的**主頁** (*homepage*)，都如 Figure 10 所示：Figure 10 因為是剛產生的專案，所以看起來空空的，可以開始動手添加些資訊。

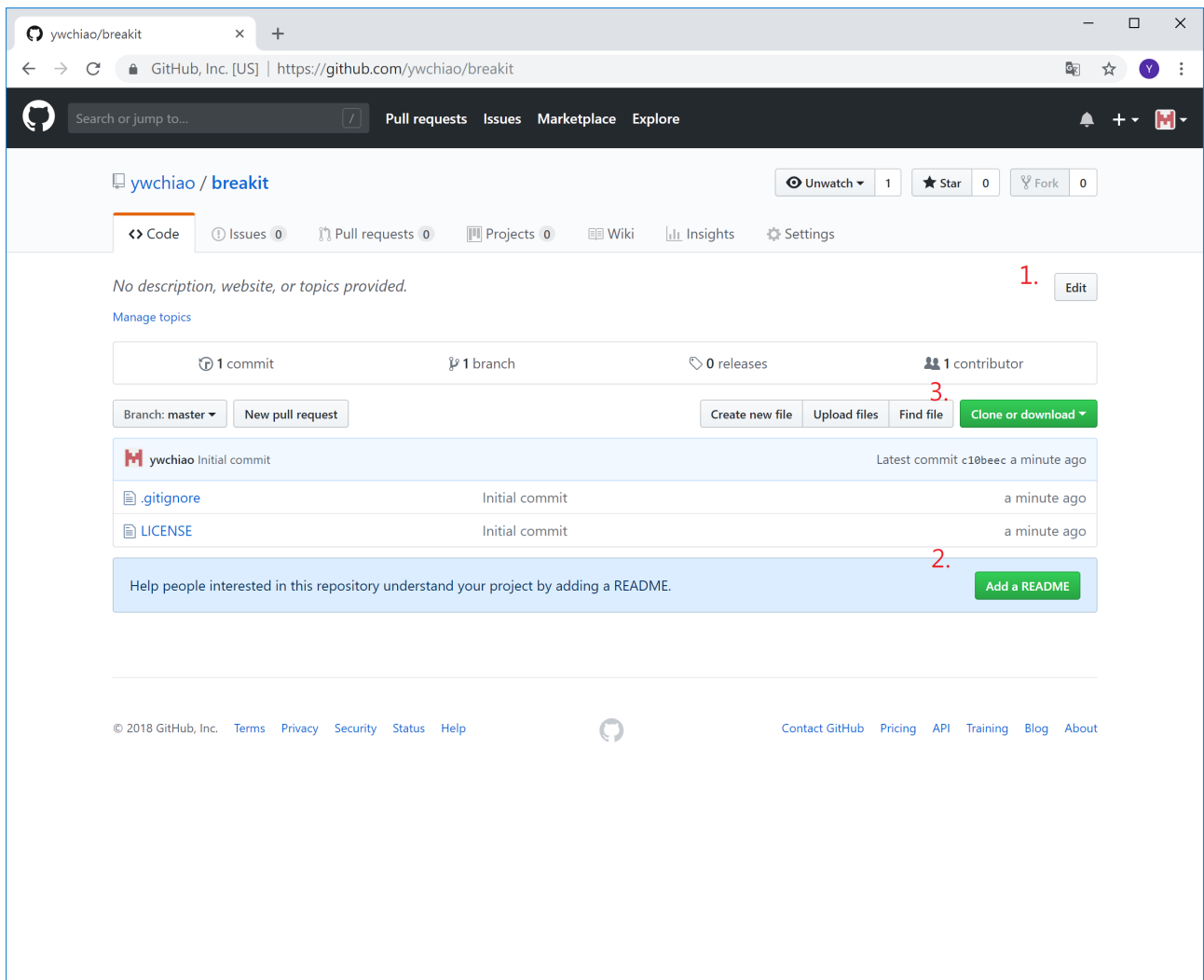


Figure 10: github 專案畫面

1. 按下 *Edit* 可以加上網頁的說明。
2. 按下 *Add a README* 會進入網頁編輯器的畫面，編輯 *README.md* 檔案。說明這個專案的細節。等等再回頭來談關於 *README.md* 的事，現在先略過。
3. 我們現在關心的是這個 *Clone or download*。展開後是這樣，

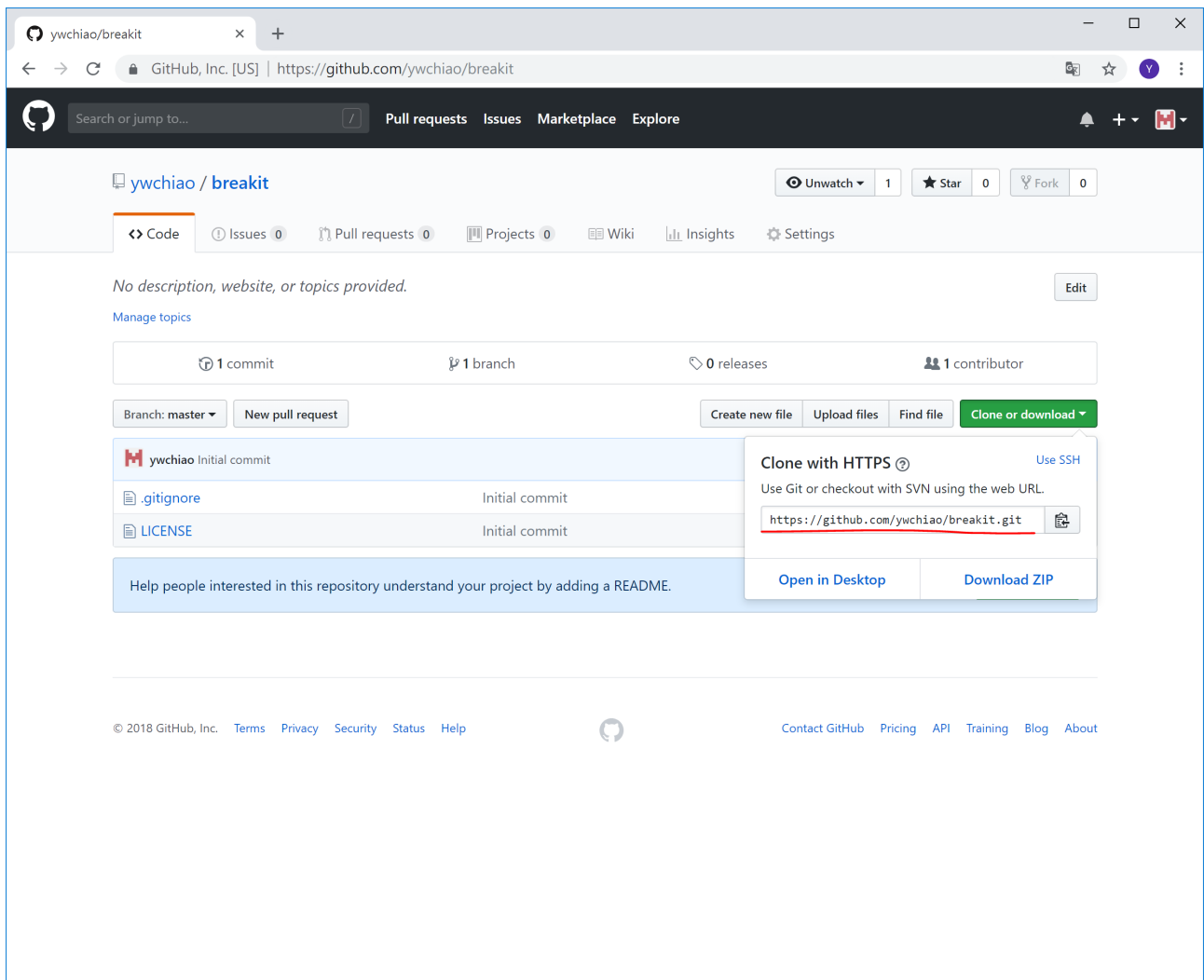


Figure 11: github repository url

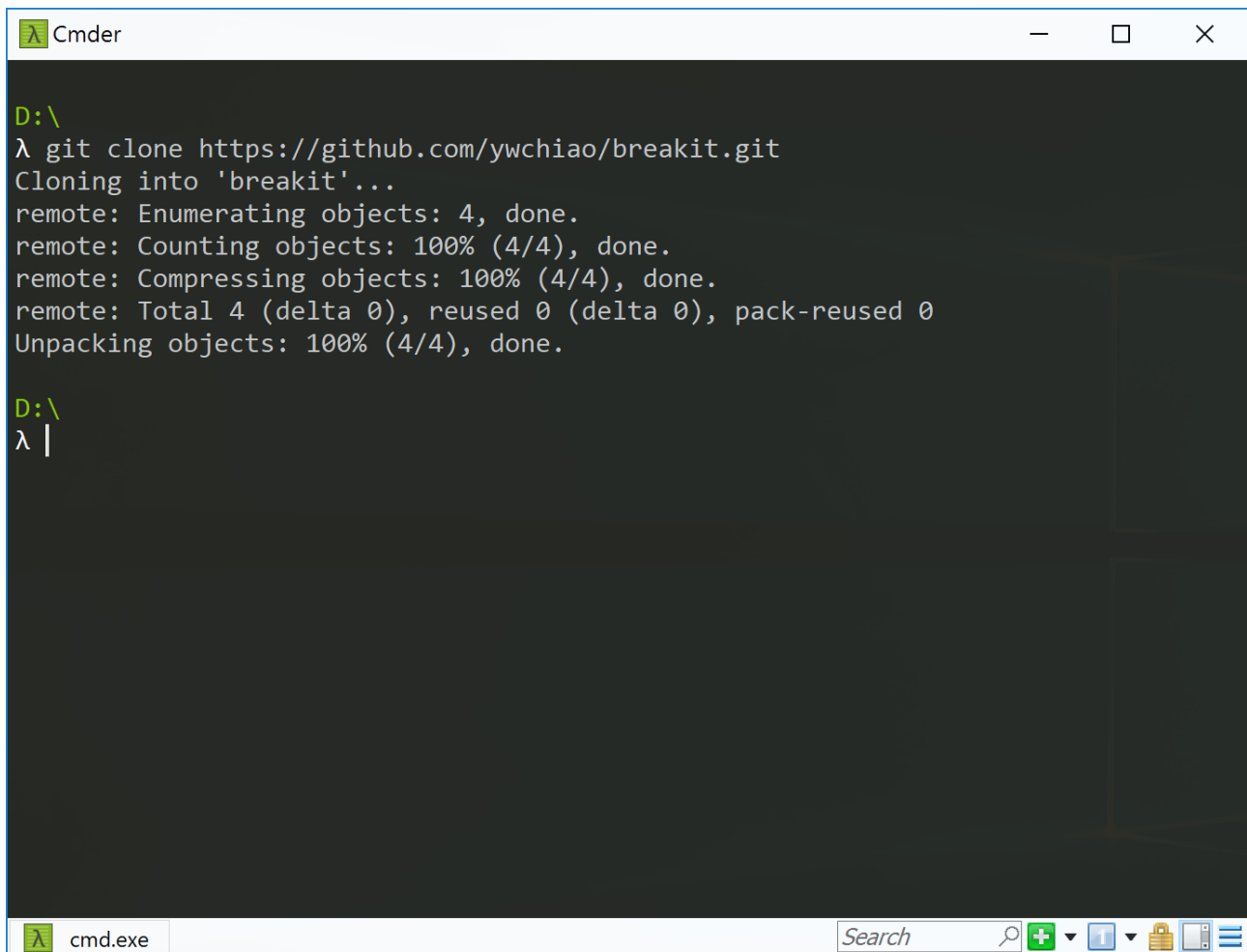
這裡可以選擇將整個專案壓成一個.zip 檔案下載，或者用 [GitHub](#) 的客戶端下載。但目前我們關心的是 Figure 11 中畫紅線的 url。

開啟**命令提示字元** (或者，如果有安裝的話，[cmdr](#)；如果是 Linux/MacOS，開啟 *terminal*。)

輸入：`git clone your_url;`

比如：`git clone https://github.com/ywchiao/breakit.git`

然後，專案源碼就下載到本地端 (local) 的硬碟裡了。如 Figure 12。



```
D:\
λ git clone https://github.com/ywchiao/breakit.git
Cloning into 'breakit'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.

D:\
λ |
```

Figure 12: git clone url 示意圖

之後的專案開發就會在本地端的這個資料夾下工作；再利用 [git](#) 同步到 [GitHub](#) 上。

4.4 *README.md*

README (讀我) 檔案用來記錄的專案的介紹，說明，注意事項等事宜。理想上，是專案使用者首先要閱讀的文件。因為使用者通常缺少耐心，所以 *README* 檔案裡最好只放關於專案的**簡單介紹**和一定要事先知道的**注意事項**。

README.md 裡的 `.md` 指的則是 [Markdown](#)²⁸ 標記語言。

4.4.1 [Markdown](#)

[Markdown](#) 是一個易學，易讀，易寫，易傳播的文件寫作標記語言。目前這份文件就是利用 [Markdown](#) 語言寫的。

²⁸<https://en.wikipedia.org/wiki/Markdown>

作為一個在網路叢林裡野蠻生長的個體，[Markdown](#) 和 Javascript 有類似的問題，就是**方言** (*dialect*) 太多，不同的實作都在某些細節有些微妙的不同；或者有不同的語言擴充。

標準化的努力稱作 [CommonMark](#)²⁹，嘗試定義一個最小的共通語言核心，並嚴格定義這個核心標準的實作細節。[GitHub](#) 已宣佈在核心部份支援 [CommonMark](#)，另外再加上 [GitHub](#) 自己定義的**語言擴充** (*extensions*)。GitHub 使用的 [Markdown](#) 方言就稱作 [GFM](#)³⁰: [GitHub Flavored Markdown](#)。

幸運的是，不像 JavaScript，[Markdown](#) 不是程式語言；方言不同的影響只是當文件格式需要轉換的時候，如 md <-> HTML，md <-> pdf，md <-> epub 等，呈現的效果會有差異，基本上對閱讀的影響不大。

[GitHub](#) 要求/假定專案說明的 *README* (讀我) 檔案需要以 [Markdown](#) (*GFM*) 撰寫。

²⁹<http://commonmark.org>

³⁰<https://github.github.com/gfm>