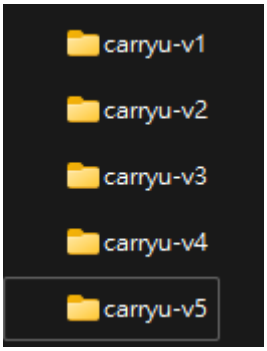
Git基礎教學

written by https://github.com/xxrjun

為甚麼要用 Git？

方便維護，不想發生下面麻煩又不方便的狀況



版本控制能夠紀錄檔案的內容變化，並能查詢每個版本更動的內容

Git 怎麼用？

Prerequisite

安裝 Git 與 VSCode 工具

1. 依照對應的作業系統安裝 Git
2. 安裝 VSCode 以及其視覺化套件
    a. GitLens (Student Developer Package 即可免費使用，我自己也是用這個)
    b. 如果不想付費或用學生帳號那也可以使用 Git Graph

安裝完後打開終端機輸入指令檢查是否安裝成功

```
git --version
```

註冊 GitHub 帳號

相信大家都註冊好了ㄅ

設定識別資料

打開終端機設定使用者名稱及電子郵件。 這一點非常重要，因為每次 Git 的提交會使用這些資訊，以我的帳號為例就是：

```
$ git config --global user.name "xxrjun"
$ git config --global user.email rjun0729@gmail.com
```

檢查一下有沒有 `user.name` 跟 `user.email`

```
$ git config --list
```

按 `q` 退出。想了解更詳細可以看這

設定 SSH 連接

官方文件 **Connecting to GitHub with SSH**

1. 在終端機輸入指令以產生 SSH 金鑰，信箱部份放你 GitHub 註冊的信箱

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
```

2. 一直按 Enter 直到輸出，不需要輸入東西
3. 確認一下 ssh-agent (甭管這啥)有沒有在背景執行，只要跑出類似下面的東西就好。

   (第一行是指令；第二行是輸出)

```
$ eval "$(ssh-agent -s)"
> Agent pid 59566
```

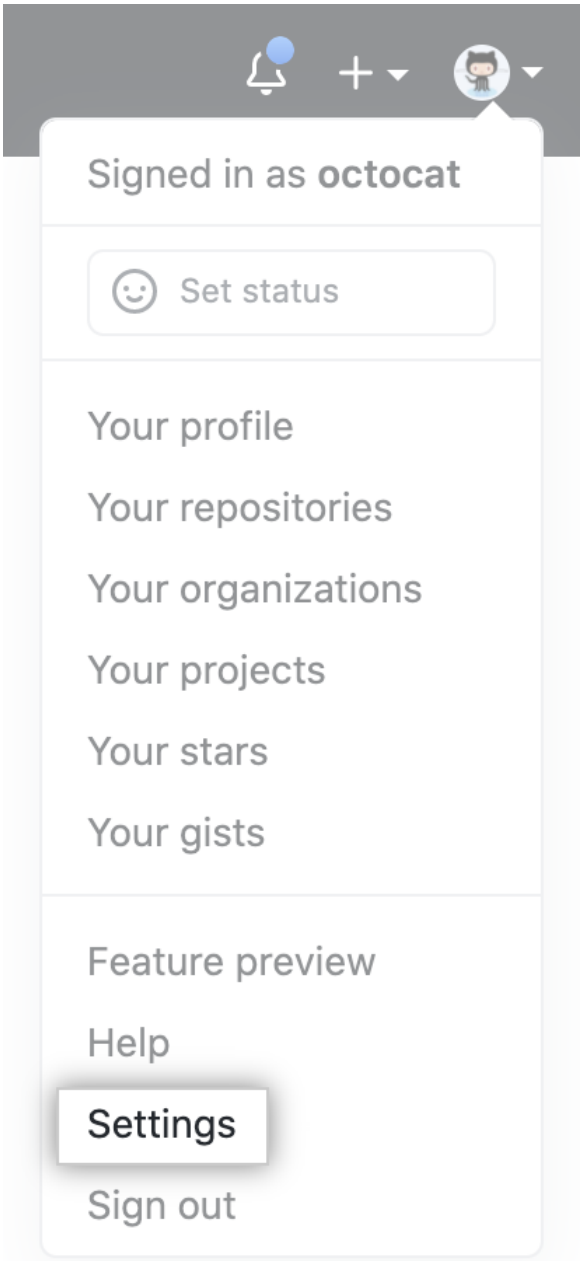
4. 新增 SSH 金鑰到 ssh-agent

```
$ ssh-add ~/.ssh/id_ed25519
```

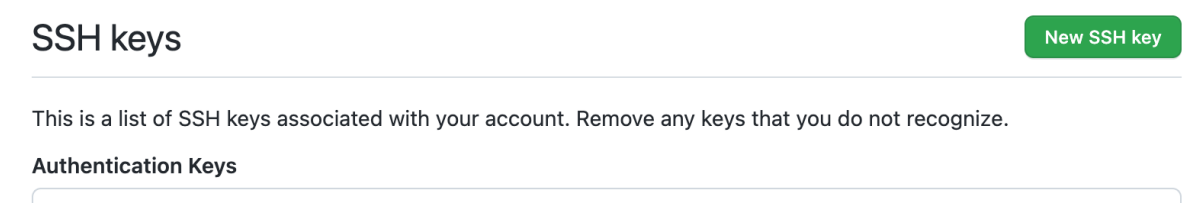
5. 輸入指令並複製內容 (Private Key)

```
cat ~/.ssh/id_ed25519.pub
```

6. 到 GitHub 網頁照圖片做

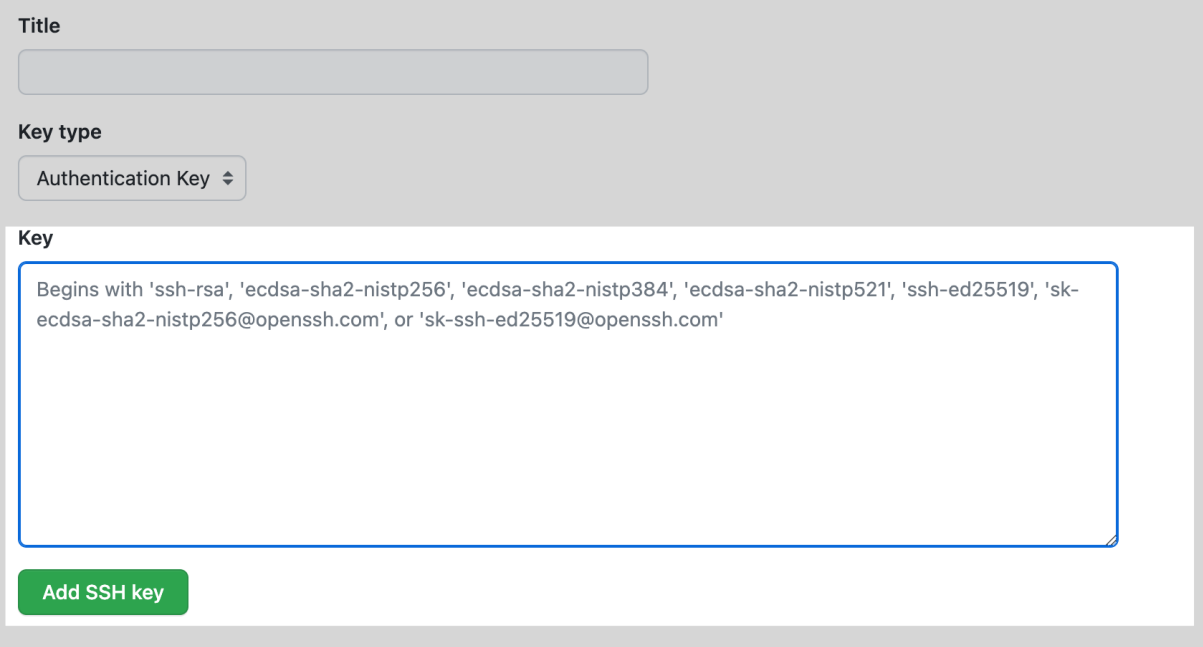


點選 SSH Key and GPG Keys，再按 New SSH key



`Title` 看你想設甚麼，像我是設成自己電腦的名稱如 `rjun-desktop` 之類的

`Key` 的地方填入你剛剛複製起來的 private key



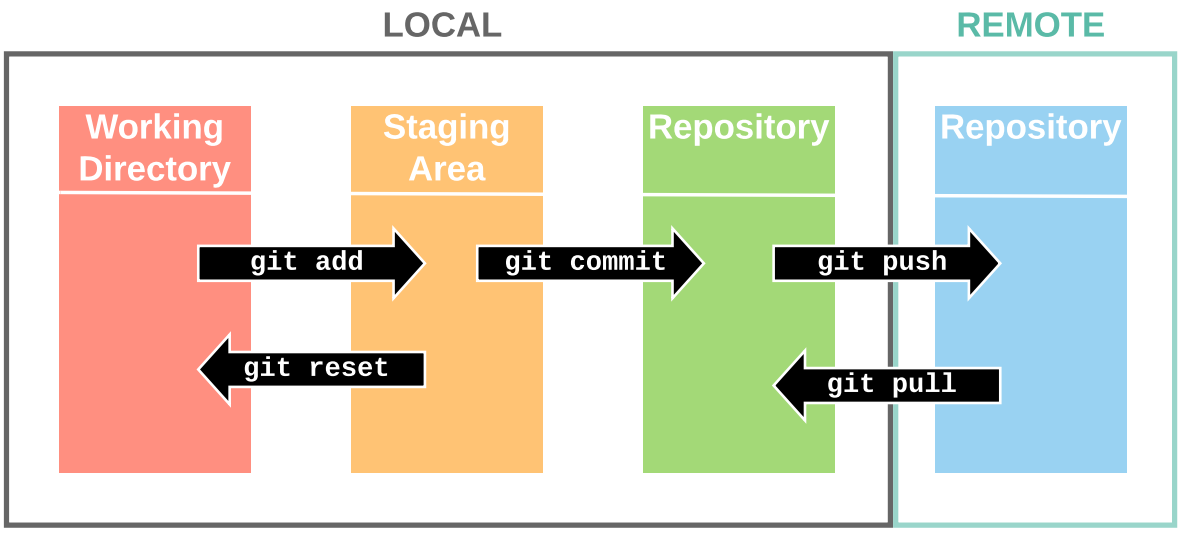
按下 Add SSH Key 就完成囉!

Git基本指令

Command	Description
`git init`	初始化儲存庫。到你想要的資料夾目錄中輸入即可建立。
`git status`	查詢當前工作目錄的詳細狀態。
`git add <files>`	將修改過的指定檔案放入暫存區。
`git reset <files>`	將指定檔案從暫存區移出。
`git commit`	將暫存區的檔案添加commit message並提交至儲存庫。語法為 `git commit -m <message for this commit.>`
`git push`	將本地儲存庫推送至遠端儲存庫。
`git pull`	將儲存庫從遠端拉到本地。因為遠端的儲存庫版本可能比較新，因此在修改專案時記得要先執行此指令讓本地儲存庫的內容是最新版本，否則會出現conflict。如果發生conflict也不用緊張，可以透過 `git merge` 做合併。
`git clone`	抓取(複製)遠端儲存庫至本地。後面放該遠端儲存庫的網址(HTTPS、SSH)。

更多可參考: Git命令快速參考 或是直接餵狗

Git Workflow



Git commit message 怎麼寫才好呢？

可以參照國外 AngularJS 團隊的規範： AngularJS Git Commit Message Conventions

以下也有一些規範引用

▼ **Good Commit**

1. 用一行空白行分隔標題與內容
2. 限制標題最多只有 50 字元
3. 標題開頭要大寫
4. 標題不以句點結尾
5. 以祈使句撰寫標題
6. 內文每行最多 72 字
7. 用內文解釋 what 以及 why vs. how

▼ **標準化的 conventional commits**

1. `feat`: 新增/修改功能 (feature)。
2. `fix`: 修補 bug (bug fix)。
3. `docs`: 文件 (documentation)。
4. `style`: 格式 (不影響程式碼運行的變動 white-space, formatting, missing semi colons, etc)。
5. `refactor`: 重構 (既不是新增功能，也不是修補 bug 的程式碼變動)。

6. `perf`: 改善效能 (A code change that improves performance)。
7. `test`: 增加測試 (when adding missing tests)。
8. `chore`: 建構程序或輔助工具的變動 (maintain)。
9. `revert`: 撤銷回覆先前的 commit 例如：revert: type(scope): > - subject (回覆版本：xxxx)。

原文引用: Commit Message Guidelines、 Conventional Commits

GitHub又是甚麼鬼?

以 Git 為核心技術基礎的雲端服務平台

市面上常見的有: GitHub、GitLab、Bitbucket、Gitee(大陸)，你可以把程式碼上傳到平台上與人交流，也可以在這些平台上找到許多開源專案，觀摩其他人的程式碼或學習新知，甚至可以加入開源專案的協作。

實作練習

請先完成 Prerequisite!

遠端 Repository

1. 在GitHub上建立一個名為 first-repo 的Repository
2. 建立本地資料夾

```
mkdir first-repo
```

3. 新增一個檔案 `README.md`

```
echo "# first-repo" >> README.md
```

4. 初始化Git。輸入完這個指令就代表你這個整個資料夾已經是一個Git的儲存庫了

```
git init
```

5. 檢查一下當前目錄的狀態

```
git status
```

6. 將檔案加入 Staging Area

```
git add README.md
```

7. 再檢查一下當前目錄的狀態，看看檔案是不是追蹤了

```
git status
```

8. 提交 Staging Area 中的檔案到 (.git) Repository

```
git commit -m "first commit"
```

9. 將預設分支改為 `main` (原本叫master不過github為了響應黑人運動因此逐漸遷移)

```
git branch -M main
```

10. 與遠端Repository做連結

```
git remote add origin https://github.com/leontzou/first-repo.git
```

11. 將本地內容push至GitHub上

```
git push -u origin main
```

忽略不需要的檔案 `.gitignore`

範例如下。在這樣的狀況下，以 `.env` 與 `.exe` 為副檔名的檔案都會被git忽略

```
.env
.exe
```

不同版本？ 建立 Branch!

鼓勵大家了解原理看這 → Click me

1. 複製 `practice-git` 專案到本地並進入該專案

   ```
   $ git clone git@github.com:CARRYUU/practice-git.git
   $ cd practice-git
   ```

2. 檢查一下本地 branch 有哪些

   ```
   $ git branch
   ```

   理論上只會有 main。想要看遠端 repo 全部的 branch 就要打

   ```
   $ git branch -a
   ```

3. 讓本地也追蹤 develop 這個 branch 吧 (開發時大家會 PR 到 develop，等到確定都可以執行才會跟 main 合併)

   ```
   $ git checkout develop
   ```

4. 建立一個以自己帳號名稱命名新分支(=在目前提交上新增了一個指標)。

   像我就會是 `xxrjun-branch`

   ```
   $ git branch YOURNAME-branch
   ```

5. 切換分支

   ```
   $ git checkout YOURNAME-branch
   ```

   可以使用 `git branch -b` 同時達成建立與切換分支

6. 更新自己的分支與 `develop` 一樣 (為避免未來共同開發時你的分支落後太多 commit 導致合併衝突，平常要繼續開發的第一步驟就是先做更新)

```
$ git checkout -a
$ git pull --rebase origin develop
```

如果 rebase 後，你修改的程式碼於提交的這段期間上游的程式碼也被改了，那仍然會發生衝突，這時候打開文字編輯器如(VSCode)處理衝突。接著將更改預存(Stage) 起來，然後執行 `git rebase --continue` 。

`--rebase` 引數可以用來確保在本機 Commit 過的更新被重新套用在 Pull 的分支的 **最前端**，這也是我們在 PR 工作流程裡通常會做的方式。這樣一來，在開啟 Pull Request 時，你的分支與上游 `develop` 分支的差別就只會有你的 Commit。

完成第一次PR吧!

記得要在自己的 branch上哦，因為 `main` 、 `develop` 都是受保護分支，連我都不能直接 push!

1. 打開把自己的github加入contributors的行列吧! (檔案位置 `./README.md`)。範例如下

```
- [xxrjun](https://github.com/xxrjun)
```

2. `git add` → `git commit`

```
$ git add README.md
$ git commit -m "docs: add <PLACE_YOUR_GITHUB_NAME> to `README.md`"
```

3. (Optional) 打開 `index.html` 到約 40 行的地方複製貼上並將**名稱**跟**連結**改成自己的 GitHub

```
<a
   href="https://github.com/xxrjun">
  <button class="btn btn-theme-color modal-toggle" style="font-size: 20px;
  ">
    xxrjun
  </button>
</a>
```

一樣 add → commit

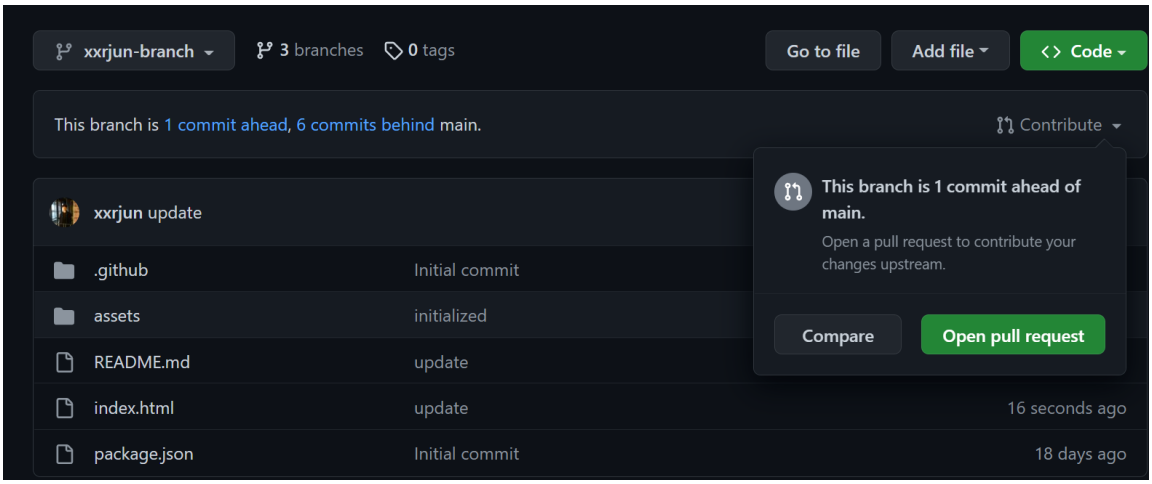
```
$ git add index.js
$ git commit -m "feat: add xxrjun into index.html
```

4. 這邊 push 需要設定路徑，之後遠端 repo 才會追蹤這個 branch，範例如下

```
$ git push origin xxrjun-branch
```

5. 到 GitHub 上發 PR!

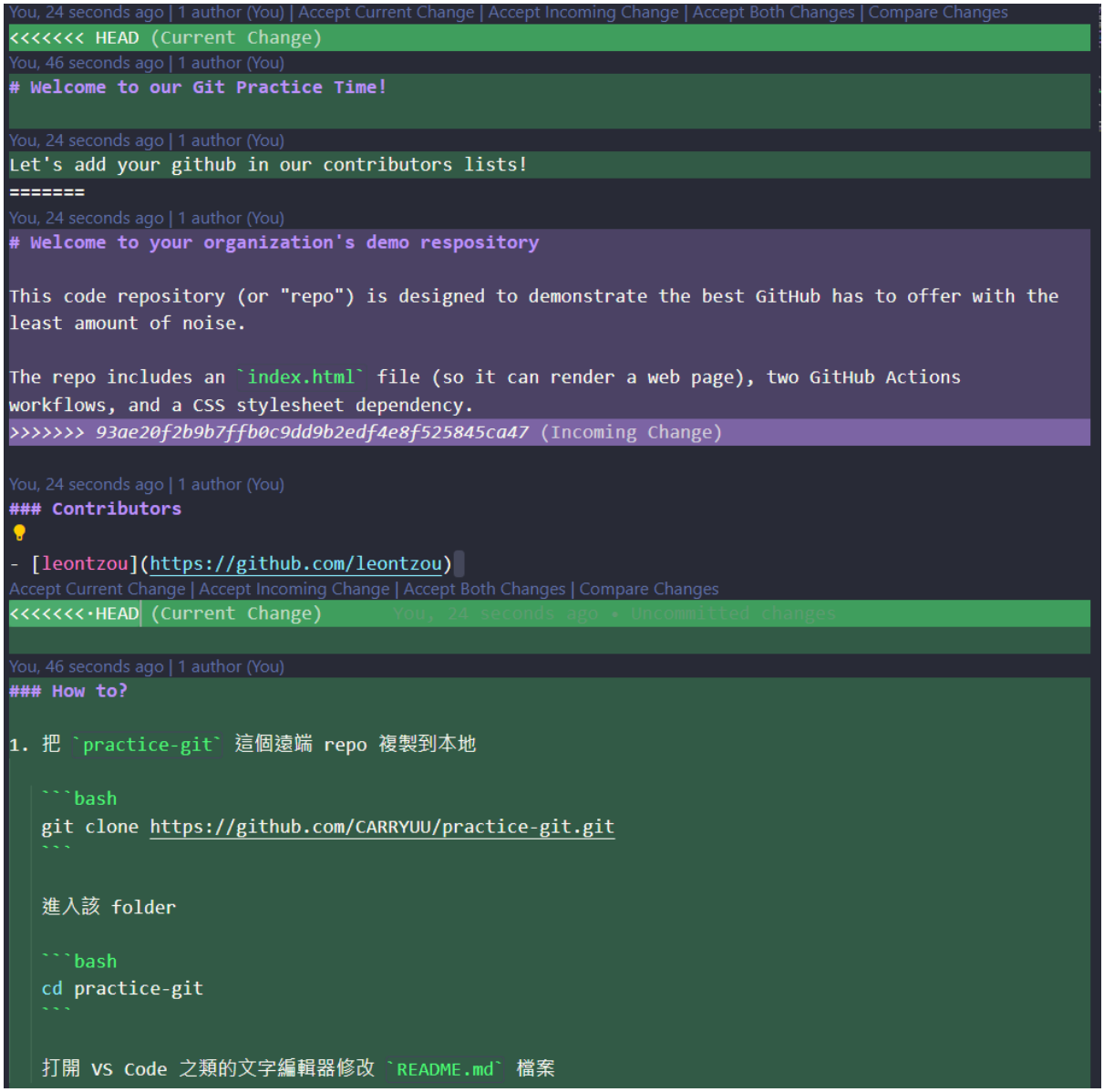
   切換到自己的分支 → Open pull request → 留個訊息 → Create pull request → 等待機器檢查一下 → Merge pull request



遇到版本衝突不用怕(Conflict)

到VSCode開啟檔案

可以選擇接受你已經修改過的本地版本(Accept Current Change)，用於你覺得自己改過的檔案才是正確時。如果遠端抓下來的版本才是正確的話就用選擇 Accept Incoming Change



SourceTree(Optional)

SourceTree安裝: link

參考資料&延伸閱讀

1. https://git-scm.com/book/en/v2
2. 猴子都能懂的Git入門指南
3. 保哥的【30 天精通 Git 版本控管】: **Learn-Git-in-30-days**
4. 什麼是 Git？為什麼要學習它？

5. **Git 基礎教學 -** contributed by <steven1lung> and <Niomoo>