

# Git ( 個人使用 )

Sunny <kyshen@nlplab.cc>

# 版本紀錄

- 本地端數據庫 & 遠端數據庫

在儲存任何資料到遠端數據庫前，  
你需要開一個「空的」遠端數據庫

本地端數據庫  
( 你的電腦 )

遠端數據庫  
( Github )

# 開始操作！

建立 Github 上的 repo，  
並下載下來

# 步驟一

- 建立 Github 上的 repository

The screenshot shows the GitHub profile of Kuan-Yu, Shen (username: sunlight0602). The page layout includes a header with navigation links (Pull requests, Issues, Marketplace, Explore) and a search bar. The main content area is divided into two columns. The left column displays the user's profile picture (a blue square with white code symbols), name, and a list of organizations. The right column shows the 'Repositories' tab, which is highlighted with a red dashed box and a yellow '1.' label. Below the tab, there is a search bar and a 'New' button, which is also highlighted with a red dashed box and a yellow '2.' label. The list of repositories includes '11010ISS\_SOA\_FlipFlap' (Public, Ruby, Updated 10 days ago), 'how\_to\_express\_anger\_to\_your\_professor' (Public, Python, Updated 13 days ago), 'GCP\_places\_api' (Private, Python, Updated 20 days ago), and 'test' (Public, Updated on 5 Sep).


# 步驟二

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

1.

Owner \*

 sunlight0602 ▾

/

Repository name \*

cs\_hw

✓


Great repository names are short and memorable. Need inspiration? How about [urban-goggles?](#)


2.

Description (optional)

git lecture

3.

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

### Initialize this repository with:

Skip this step if you're importing an existing repository.

4.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

5.

☒ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Actionscript ▾

6.

## 步驟二

### Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

☒ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Actionscript ▼

☒ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

License: Apache License 2.0 ▼

This will set  **main** as the default branch. Change the default name in your [settings](#).

Create repository

[Terms](#)

[Privacy](#)

[Security](#)

[Status](#)

[Docs](#)



[Contact GitHub](#)

[Pricing](#)

[API](#)

# 介紹

sunlight0602 / cs\_hw Public

Unwatch

1

Star

0

Fork

0

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

main

1 branch

0 tags

某些不想上傳到 Github 上的資料（帳號密碼、超大的訓練資料集、虛擬環境……），寫進 gitignore 後就不會上傳上來

.gitignore

Initial commit

像一張契約，規範他人如何使用你的開源專案（能不能商用？）

LICENSE

Initial commit

README.md

Initial commit

README.md

cs\_hw

git lecture

字面上的意思，請他人在使用你的專案時，先閱讀此篇文件。是 markdown 格式，會顯示在下面

點進這三份文件看看！

Readme

Releases

No releases published

[Create a new release](#)

[Publish your first package](#)

## 步驟三

```
(base) └─sunny@chenguanyudeMacBook-Pro ~  
└─$ cd /Users/sunny/Desktop  
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop  
└─$ █
```

- 開啟 terminal / 命令提示字元 ( cmd ) ，並移動當前位置到桌面
- terminal 的使用者，輸入：
  - `cd /Users/[你電腦的使用者名稱]/Desktop`
- cmd 的使用者，輸入：
  - `cd C:/Users/[你電腦的使用者名稱]/Desktop/`

cmd 的使用者，可以參考：<https://opensource.com/windows-programming/command-prompt-primer/>



步騫四

- 觀察當前路徑（桌面）上有什麼
- terminal：輸入
  - `dir`
- 命令提示字元：輸入
  - `ls`

```
(base) └─sunny@chenguanyudeMacBook-Pro ~
└─$ cd /Users/sunny/Desktop
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop
└─$ ls
1595164209-7696-Keroro.png          截圖 2021-09-23 上午1.02.09.png
FlipFlap_HW                        截圖 2021-09-24 下午6.05.48.png
GCP_places_api                     截圖 2021-09-24 下午6.08.23 2 2.png
Keroro.png                          截圖 2021-09-24 下午6.08.23 2.png
Papers                             截圖 2021-09-24 下午6.08.23.png
ROCLING2021_34.pptx                截圖 2021-09-24 下午6.08.48.png
basic_info.jsonl.txt               截圖 2021-09-24 下午6.33.11.png
database_operation                  截圖 2021-09-24 下午6.34.20.png
python-archetypes                  截圖 2021-09-24 下午6.34.49.png
recommendation-engine              截圖 2021-10-02 下午1.26.10.png
hw_fizzbuzz                         截圖 2021-10-12 下午1.37.15.png
hw_serializer                       截圖 2021-10-12 上午11.55.45.png
hw_serializer                       截圖 2021-10-12 下午2.02.57.png
```

# 指令介紹

- `cd [path]`
  - 移動當前位址
- `dir / ls`
  - 顯示當前位址中的資料

# 步驟五

- 回到 Github 頁面，複製遠端數據庫位址

The screenshot shows the GitHub repository page for 'sunlight0602 / cs\_hw'. The repository is public and has a 'main' branch with 1 branch and 0 tags. The repository contains three files: '.gitignore', 'LICENSE', and 'README.md', all from the 'Initial commit'. The 'README.md' file is open, showing the title 'cs\_hw' and the content 'git lecture'. A 'Code' button is highlighted with a red dashed box and a yellow '1.' label. A 'Clone' dropdown menu is open, showing options for 'HTTPS', 'SSH', and 'GitHub CLI'. The 'HTTPS' option is selected, and the URL 'https://github.com/sunlight0602/cs\_hw' is displayed. The URL is highlighted with a red dashed box and a yellow '3.' label. A yellow '2.' label is placed next to the repository name 'sunlight0602'.

sunlight0602 / cs\_hw Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file Code 1.

Clone ?

HTTPS SSH GitHub CLI

https://github.com/sunlight0602/cs\_hw 3.

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

sunlight0602 Initial commit 2.

File	Commit
.gitignore	Initial commit
LICENSE	Initial commit
README.md	Initial commit

README.md

## cs\_hw

git lecture

## 步驟六

- 回到 Terminal / CMD，輸入：

```
git clone [你剛才複製的位址]
```

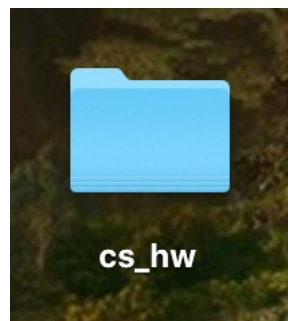
```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop
└─$ git clone https://github.com/sunlight0602/cs_hw.git
Cloning into 'cs_hw'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
```

# 指令介紹

- `git clone [path]`
  - 將 Github 上的 repo 下載到本地端

# 步驟七

- 這時到你的桌面，即可看到 `cs_hw` 資料夾



- 在你的 terminal / CMD 上進入該資料夾：
- Terminal: `cd cs_hw`
- CMD: `cd cs_hw`

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop
└─$ cd cs_hw
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main>
└─$
```

# 操作到這邊結束

你學會了如何在 github 上開一個 repo 、  
如何 clone 到本地端 、  
如何在 terminal / CMD 上移動 ( cd ) 、  
如何觀察當前位址的資料 ( ls / dir )

# 版本紀錄

- 本地端數據庫 & 遠端數據庫

本地端數據庫  
( 你的電腦 )

遠端數據庫  
( Github )



# 版本紀錄

- 本地端數據庫 & 遠端數據庫

## 1. 在本地端開發



v1

本地端數據庫  
( 你的電腦 )

遠端數據庫  
( Github )

# 版本紀錄

- 本地端數據庫 & 遠端數據庫

1. 在本地端開發



本地端數據庫  
( 你的電腦 )

2. 本地端開發完後，上傳至遠端



遠端數據庫  
( Github )

# 版本紀錄

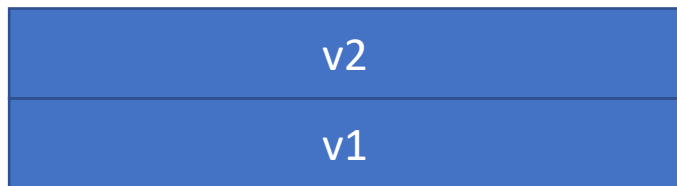
- 本地端數據庫 & 遠端數據庫

在沒有回到舊版的情況下，

本地端的版本通常  $\geq$  遠端的版本，

因為你都在本地端開發

## 3. 繼續在本地端開發



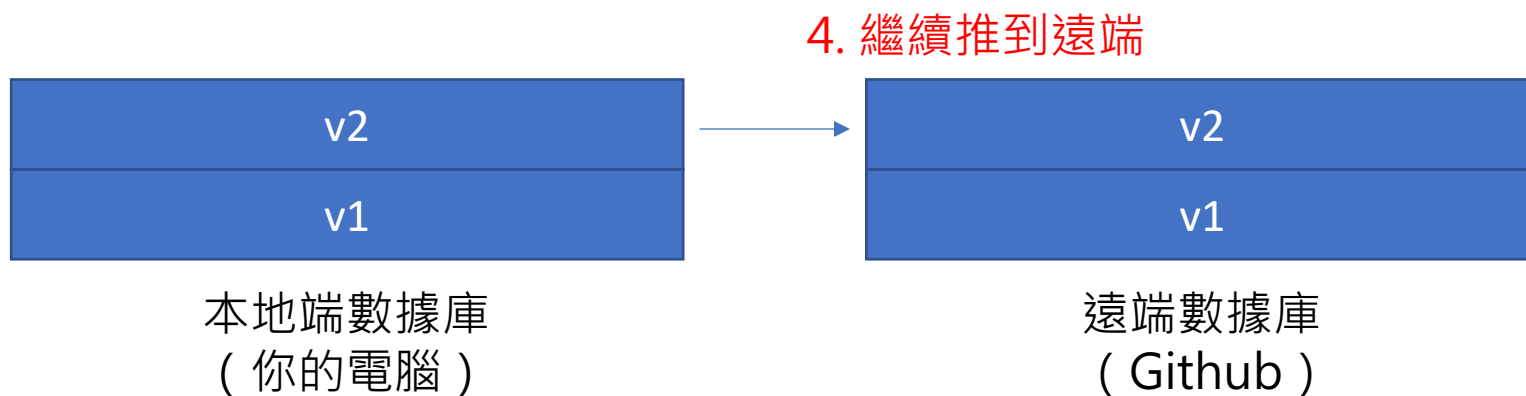
本地端數據庫  
( 你的電腦 )



遠端數據庫  
( Github )

# 版本紀錄

- 本地端數據庫 & 遠端數據庫

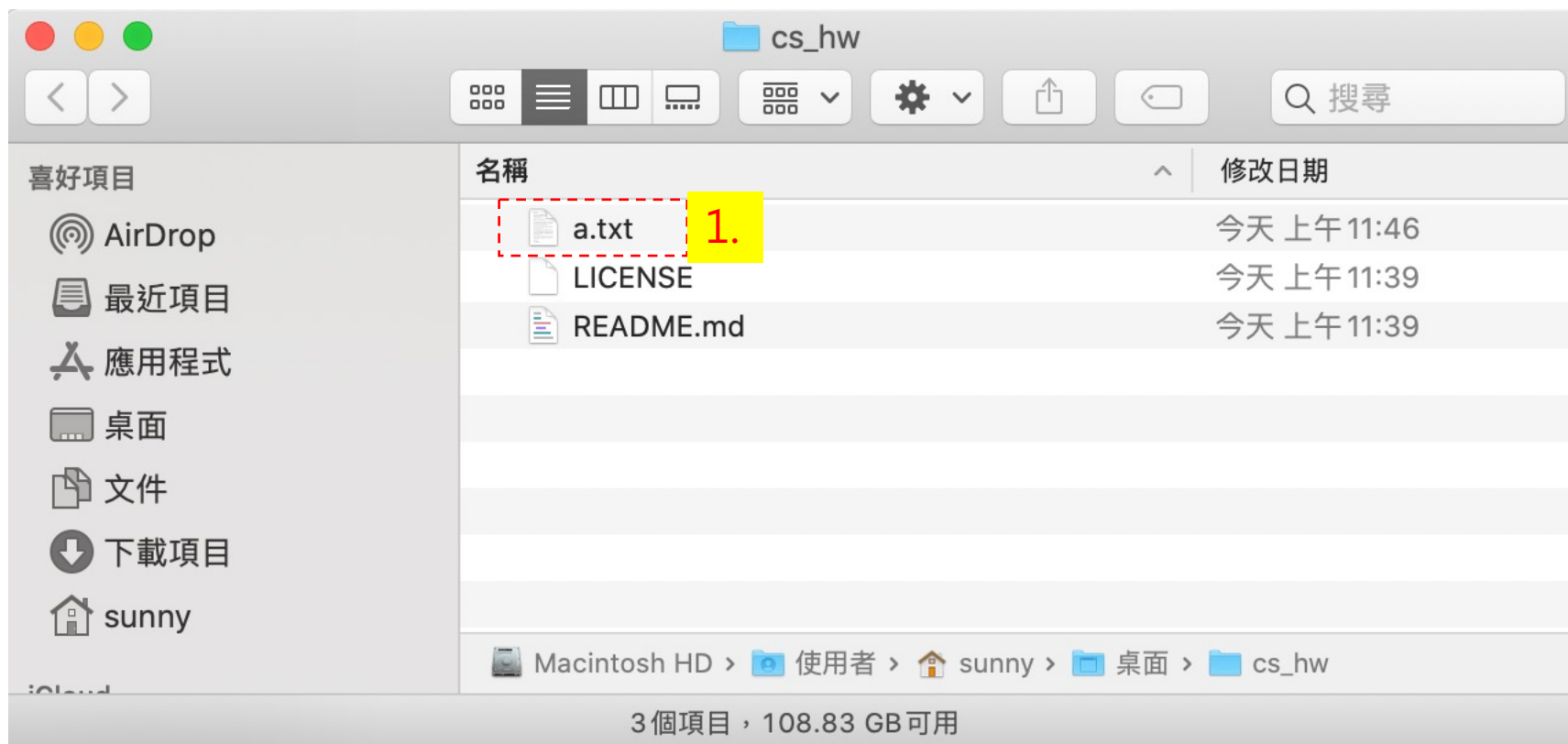


# 開始操作！

將本地端的版本推到遠端

# 步驟一

- 我們來模擬寫程式的過程
- 在 cs\_hw 資料夾中，創造一個 a.txt 的記事本



## 步驟二

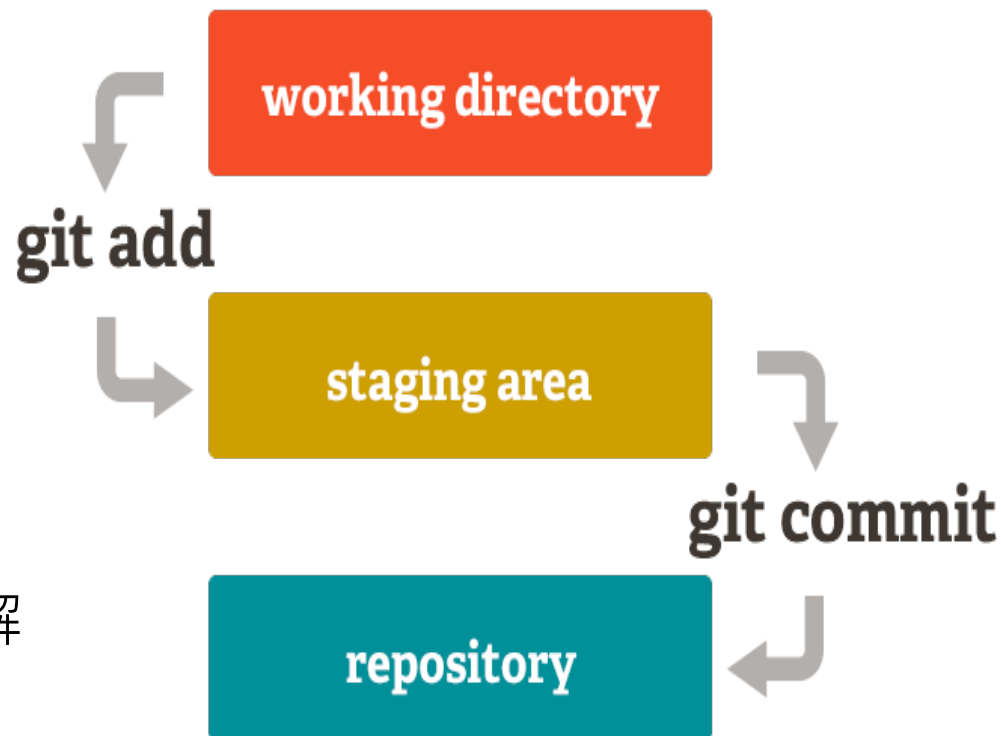
- 回到 terminal / CMD，輸入以下三個指令

- `git add .`
- `git commit -m "create a.txt"`
- `git push`

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git add .
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git commit -m "create a.txt"
[main 88f2d90] create a.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 a.txt
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main>
└─$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 263 bytes | 263.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/sunlight0602/cs_hw.git
bd368dc..88f2d90 main -> main
```

# 指令介紹

- `git add .`
  - 將檔案加入 staging area
  - `.` 代表該位址底下的全部檔案
- `git commit -m "create a.txt"`
  - 將檔案加入本地端的數據庫
  - “create a.txt” 是訊息，為本次版本的註解
- `git push`
  - 將數據庫中的版本上傳至 Github





# 步驟三

- 到 terminal / CMD，觀看本地端的 commit history
- 輸入以下指令：

`git log`

剛才的 commit 紀錄，  
每次 commit 都會生成一筆紀錄

- 按 q 離開

```
commit 88f2d902fd73c7ea9add2f81dc5cc319fc2f56ae (HEAD -> main, origin/main, origin/HEAD)
Author: Sunny <shensunny594@gmail.com>
Date: Tue Oct 19 11:53:56 2021 +0800

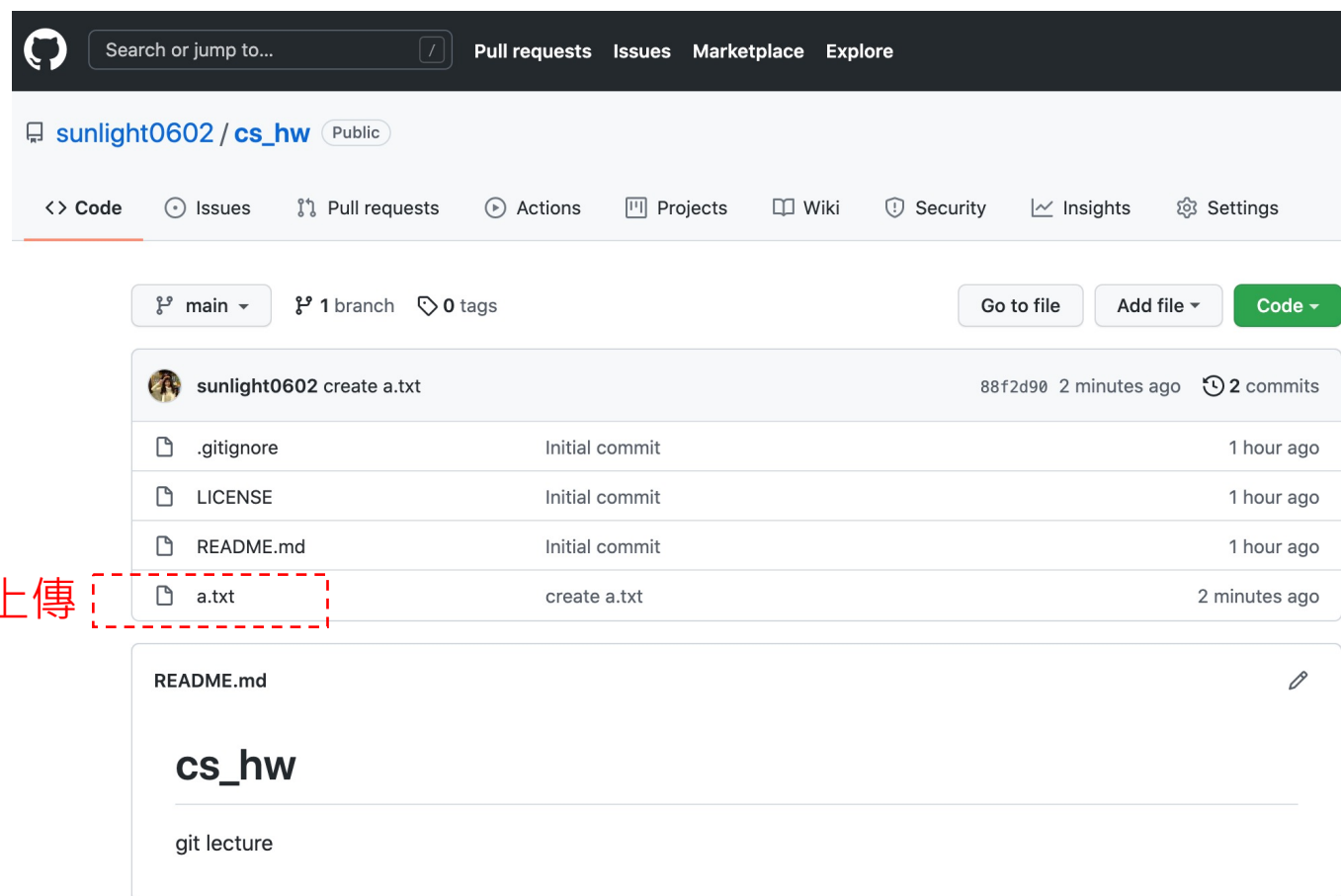
    create a.txt

commit bd368dc542b1dccf83e1899da53cb1bc391326ce
Author: Kuan-Yu, Shen <shensunny594@gmail.com>
Date: Tue Oct 19 11:10:56 2021 +0800

    Initial commit
~
~
```

# 步驟四

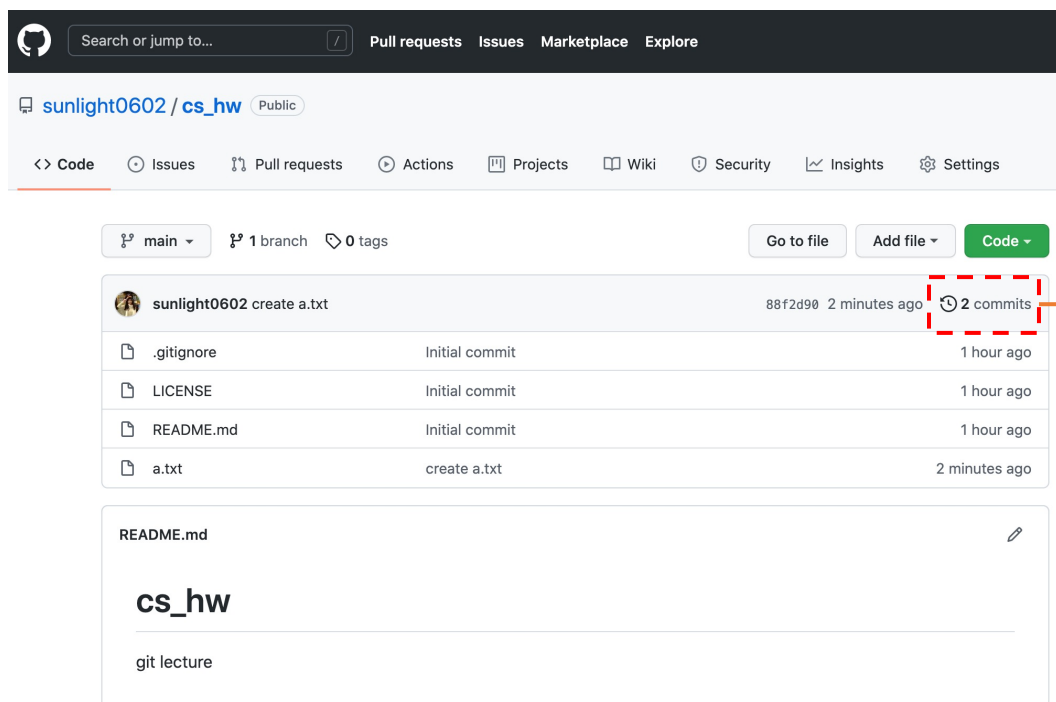
- 到 Github 上，觀看你 push 上去的記事本
- ( 記得要重新整理 )



新的，有看到代表成功上傳

# 步驟五

- 到 Github 上，觀看你的 commit history
- ( 記得要重新整理 )



sunlight0602 / cs\_hw (Public)

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

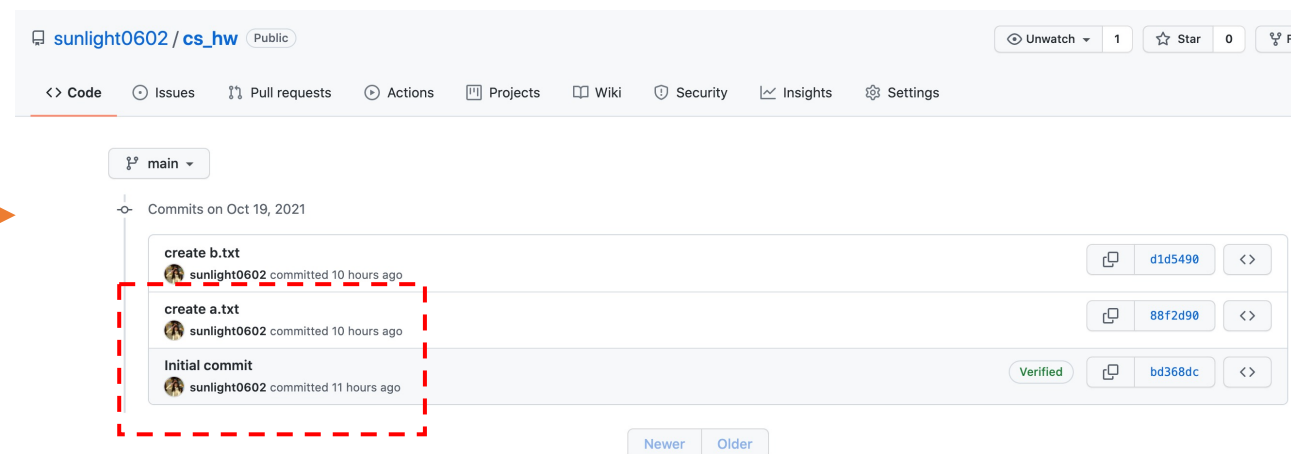
Go to file Add file Code

sunlight0602	create a.txt	88f2d90	2 minutes ago	2 commits
.gitignore	Initial commit		1 hour ago	
LICENSE	Initial commit		1 hour ago	
README.md	Initial commit		1 hour ago	
a.txt	create a.txt		2 minutes ago	

README.md

cs\_hw

git lecture



sunlight0602 / cs\_hw (Public)

Unwatch 1 Star 0 Fork

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

Commits on Oct 19, 2021

create b.txt	sunlight0602 committed 10 hours ago	d1d5490	<>
create a.txt	sunlight0602 committed 10 hours ago	88f2d90	<>
Initial commit	sunlight0602 committed 11 hours ago	bd368dc	<>

Verified

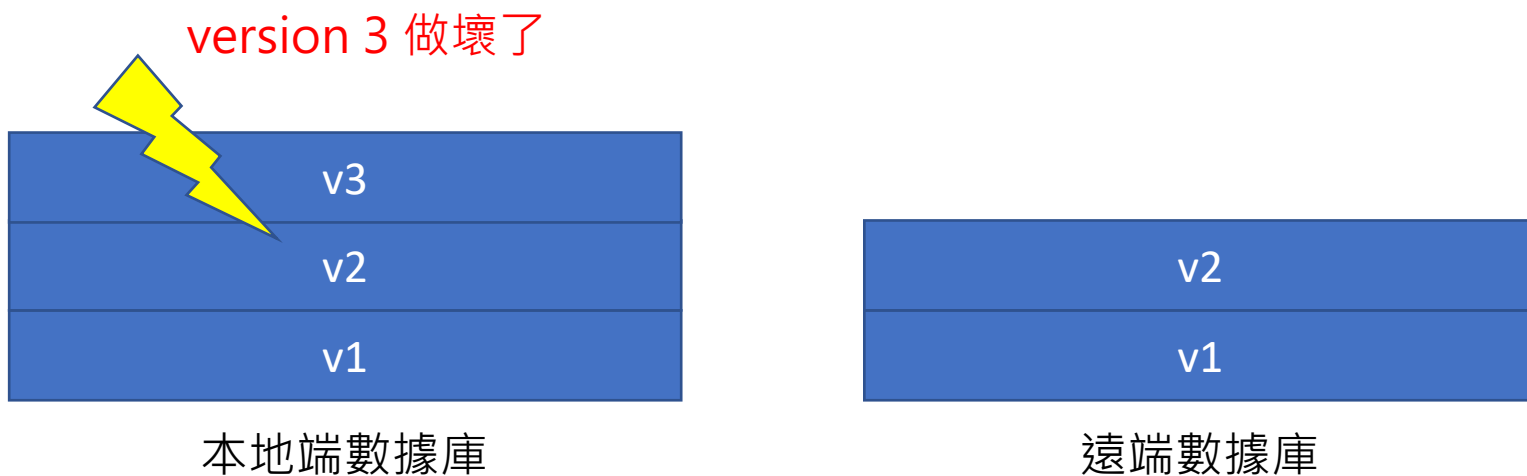
Newer Older

# 操作到這邊結束

你學會了如何將本地端的版本上傳至 Github  
( `git add` / `git commit` / `git push` ) 、  
如何觀看 commit history ( `git log` ) 、  
如何觀看 Github 上的 commit history

# 版本紀錄

- 當你在本地端把程式寫壞了，會需要回到上一個版本

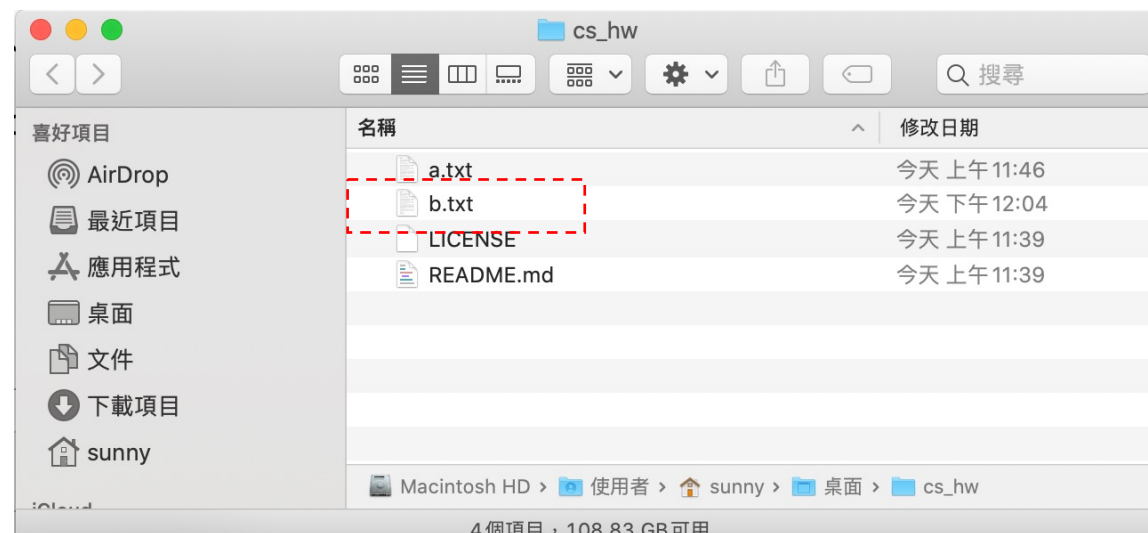


# 開始實作！

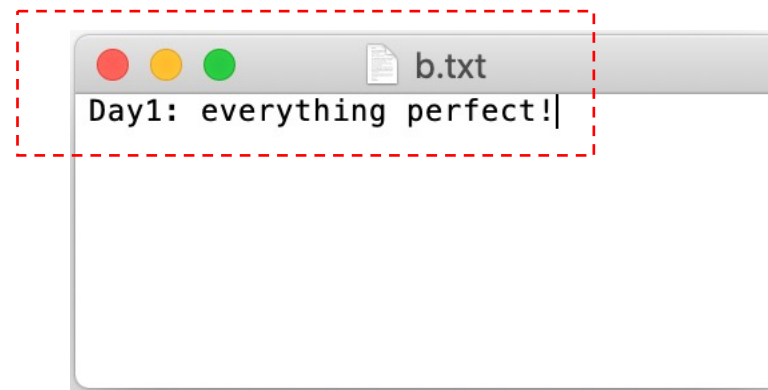
將本地端的版本復原

# 步驟一

- 先來模擬一下程式寫壞的狀況









- 在 cs\_hw 資料夾中，新增一個 b.txt
- 在 b.txt 中，輸入一段文字：
  - Day1: everything perfect!



## 步驟二

- 運用前面所學的 `add`, `commit`, `push` ,  
將現有版本推上 Github

 sunlight0602 create b.txt		
	.gitignore	Initial commit
	LICENSE	Initial commit
	README.md	Initial commit
	a.txt	create a.txt
	b.txt	create b.txt

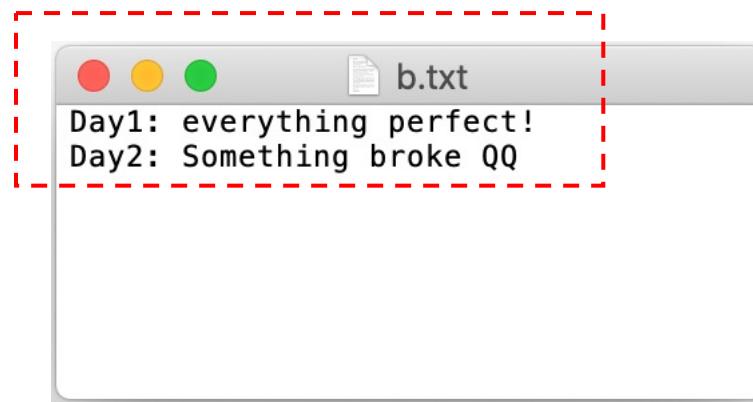
```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git add .
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git commit -m "create b.txt"
[main d1d5490] create b.txt
1 file changed, 1 insertion(+)
create mode 100644 b.txt
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main>
└─$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 290 bytes | 290.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/sunlight0602/cs_hw.git
88f2d90..d1d5490 main -> main
```



# 步驟三

- 模擬程式寫壞的情況
- 回到桌面，打開 `b.txt`，加入第二行：
  - `Day2: Something broke QQ`
- 提醒：記得儲存檔案

第二天開發，有程式寫壞了！



## 步驟四

- 程式寫壞了，先檢查當前狀態
- 在 terminal / CMD 上輸入：
- `git status`
- `git diff` （也是按 `q` 離開）

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   b.txt

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

```
diff --git a/b.txt b/b.txt
index fdec74..76e1568 100644
--- a/b.txt
+++ b/b.txt
@@ -1,2 @@
- Day1: everything perfect!
\ No newline at end of file
+ Day1: everything perfect!
+ Day2: Something broke QQ
(END)
```

- 更動很少的話，你可以看 `git diff` 手動刪除更動的部分

# 指令介紹

- **git status**

- 可以看哪些檔案被改動過（跟上一次 commit 比較）
- 會提供有什麼指令可以使用
- 比較本地端跟遠端的 commit history

- **git diff**

- 看那些被改動過的檔案裡，被改了什麼

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git status
On branch main
Your branch is up to date with 'origin/main'.

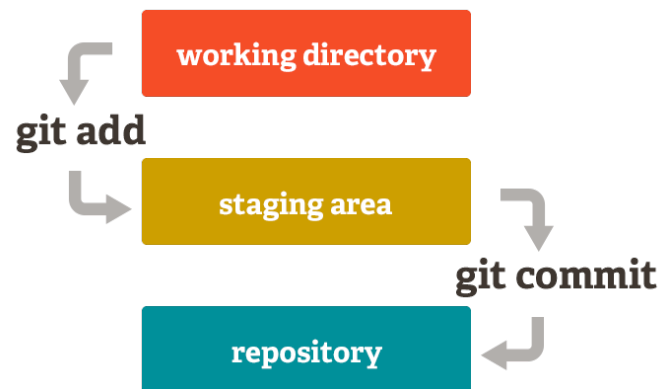
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   b.txt

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

# 步驟五

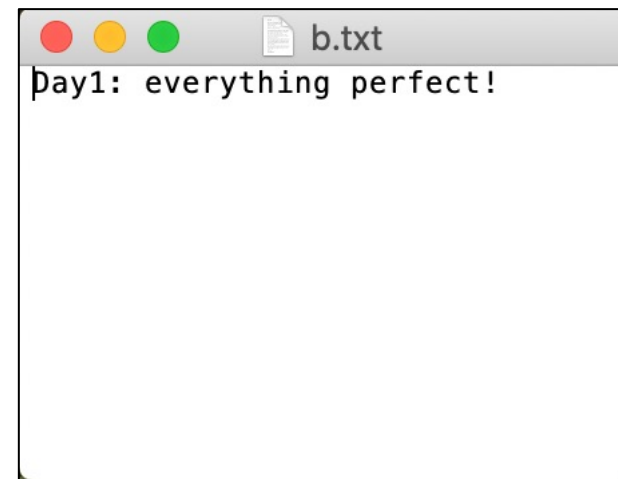
- 新的 “Day2: Something broke QQ” 還沒被 `git add` 過
- 輸入以下指令：
  - `git checkout -- .`



# 步驟六

- 打開 b.txt，會發現 “Day2: Something broke QQ” 這行不見了
- 再檢查當前狀態，輸入：

- `git status`



```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git checkout -- .
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main>
└─$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

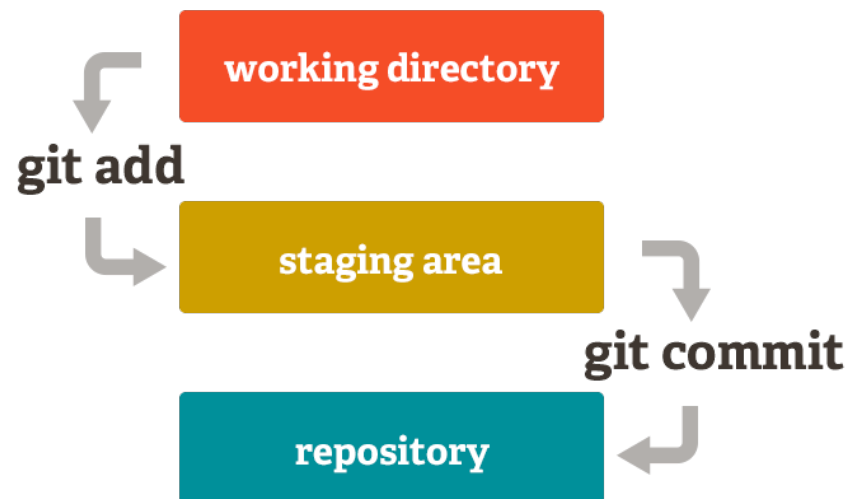
→ 沒有異動過的檔案了！

# 步驟七

- 再模擬一次程式寫壞的狀況（重複步驟三、四）
- 這次，再輸入指令：
  - `git add .`

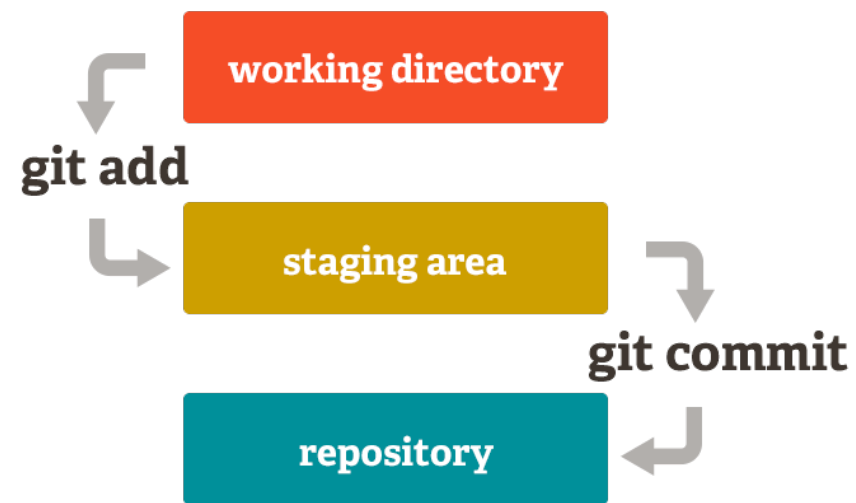
# 步驟八

- 在已經進入 staging area 的情況 ( 已經 `git add .` ) ,  
無法用 `git checkout -- .` 來回復檔案狀況
- 輸入指令 :
- `git reset`



# 指令介紹

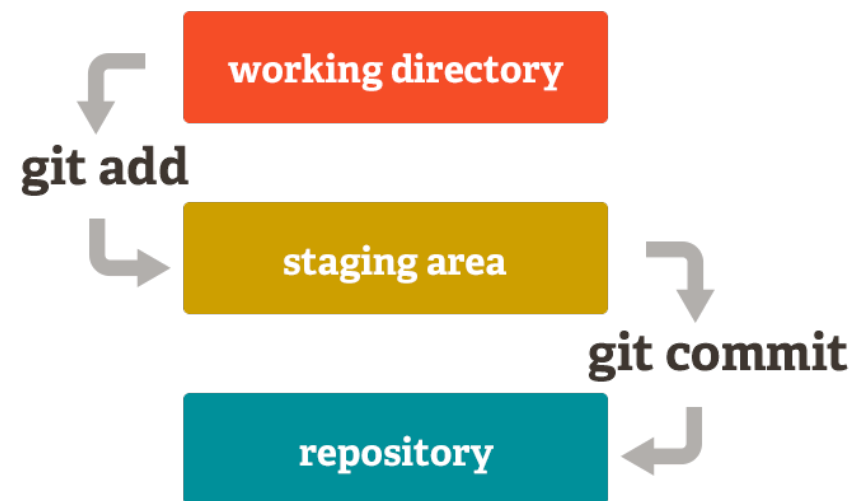
- **git reset**
  - 將檔案移出 staging area



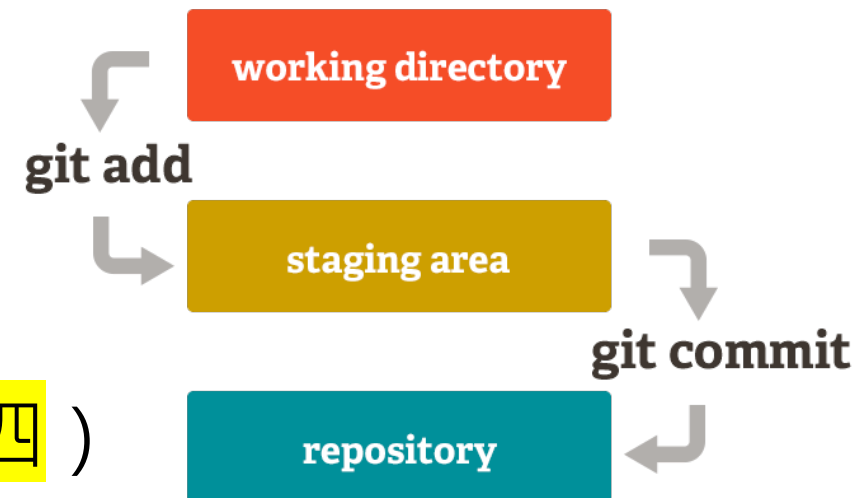


# 步驟九

- 這時，可以就可以使用 `git checkout --` 來回復檔案了
- 重複步驟五、六



# 步驟十



- 再模擬一次程式寫壞的狀況（重複步驟三、四）

- 這次，再輸入指令：

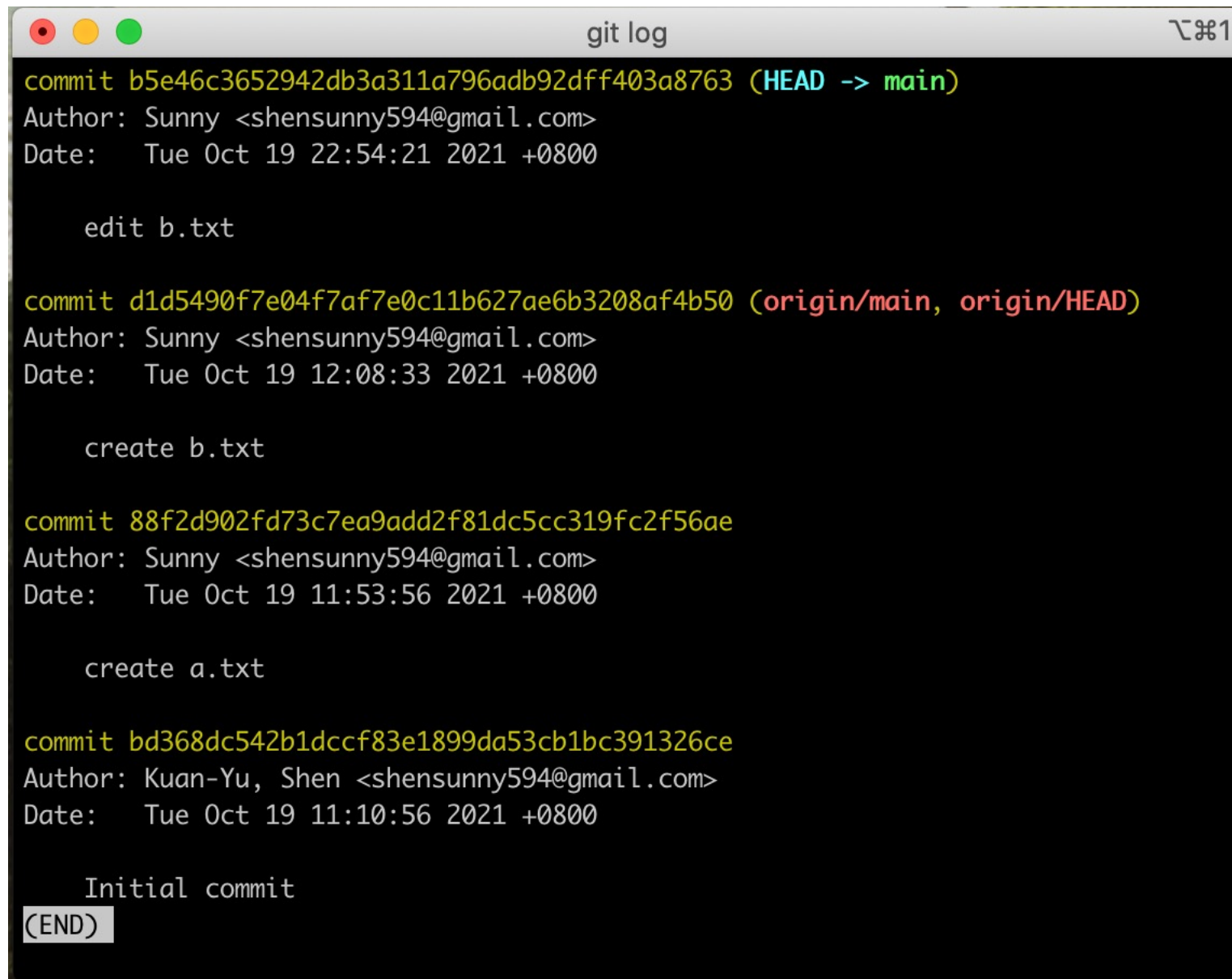
- `git add .`

- `git commit -m "edit b.txt"`

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git add .
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git commit -m "edit b.txt"
[main b5e46c3] edit b.txt
1 file changed, 2 insertions(+), 1 deletion(-)
```

# 步驟十一

- 觀察 `git log`

A terminal window titled 'git log' with a dark background and light-colored text. The window shows the output of the 'git log' command, listing four commits in reverse chronological order. Each commit entry includes the commit hash, the branch it is on, the author's name and email, and the date. The commits are: 1. 'commit b5e46c3652942db3a311a796adb92dff403a8763 (HEAD -> main)' by Sunny, dated Tue Oct 19 22:54:21 2021, with the message 'edit b.txt'. 2. 'commit d1d5490f7e04f7af7e0c11b627ae6b3208af4b50 (origin/main, origin/HEAD)' by Sunny, dated Tue Oct 19 12:08:33 2021, with the message 'create b.txt'. 3. 'commit 88f2d902fd73c7ea9add2f81dc5cc319fc2f56ae' by Sunny, dated Tue Oct 19 11:53:56 2021, with the message 'create a.txt'. 4. 'commit bd368dc542b1dccf83e1899da53cb1bc391326ce' by Kuan-Yu, Shen, dated Tue Oct 19 11:10:56 2021, with the message 'Initial commit'. The window ends with '(END)' in a grey box.

```
git log
commit b5e46c3652942db3a311a796adb92dff403a8763 (HEAD -> main)
Author: Sunny <shensunny594@gmail.com>
Date: Tue Oct 19 22:54:21 2021 +0800

    edit b.txt

commit d1d5490f7e04f7af7e0c11b627ae6b3208af4b50 (origin/main, origin/HEAD)
Author: Sunny <shensunny594@gmail.com>
Date: Tue Oct 19 12:08:33 2021 +0800

    create b.txt

commit 88f2d902fd73c7ea9add2f81dc5cc319fc2f56ae
Author: Sunny <shensunny594@gmail.com>
Date: Tue Oct 19 11:53:56 2021 +0800

    create a.txt

commit bd368dc542b1dccf83e1899da53cb1bc391326ce
Author: Kuan-Yu, Shen <shensunny594@gmail.com>
Date: Tue Oct 19 11:10:56 2021 +0800

    Initial commit

(END)
```

## 步驟十二

- 在已經 commit 的情況下，輸入以下指令：
  - `git reset HEAD~1`

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main>
└─$ git reset HEAD~1
Unstaged changes after reset:
M      b.txt
```

# 步驟十三

- 再觀察一下 `git log`
- 發現上一次 `commit` 不見了

```
git log
commit d1d5490f7e04f7af7e0c11b627ae6b3208af4b50 (HEAD -> main, origin/main, origin/HEAD)
Author: Sunny <shensunny594@gmail.com>
Date: Tue Oct 19 12:08:33 2021 +0800

    create b.txt

commit 88f2d902fd73c7ea9add2f81dc5cc319fc2f56ae
Author: Sunny <shensunny594@gmail.com>
Date: Tue Oct 19 11:53:56 2021 +0800

    create a.txt

commit bd368dc542b1dccf83e1899da53cb1bc391326ce
Author: Kuan-Yu, Shen <shensunny594@gmail.com>
Date: Tue Oct 19 11:10:56 2021 +0800

    Initial commit
~
~
```

# 步驟十四

- 觀察 `git status`

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

       modified:   b.txt

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

可以使用 `git checkout -- .` 了

# 步驟十五

- 重複步驟五、六

# 步驟十六

- 再模擬一次程式寫壞的情況（最後一次了！）

- 重複步驟三、四

- 這次，再輸入以下三個指令：

- `git add .`

- `git commit -m "edit b.txt"`

- `git push`

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git add .
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main*>
└─$ git commit -m "edit b.txt"
[main 5e32b8a] edit b.txt
1 file changed, 2 insertions(+), 1 deletion(-)
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main>
└─$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 298 bytes | 298.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/sunlight0602/cs_hw.git
d1d5490..5e32b8a main -> main
```



# 步驟十七

- 在已經 `git push` 的情況下，  
除了本地端外，你還需要把遠端的版本復原
- 輸入以下指令：
- `git reset --hard HEAD~1`
- `git push --force`

```
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main>
└─$ git reset --hard HEAD~1
HEAD is now at d1d5490 create b.txt
(base) └─sunny@chenguanyudeMacBook-Pro ~/Desktop/cs_hw <main>
└─$ git push --force
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/sunlight0602/cs_hw.git
+ 5e32b8a...d1d5490 main -> main (forced update)
```

# 步驟十八

- 觀察 Github 的 commit history
- 觀察 git log

# 操作到這邊結束

你學會了如何在各種時候回到上一個版本

# 指令表，複習用

- `git clone`
- `git add .`
- `git commit -m "some_message"`
- `git push`
- `git log`
- `git status`
- `git diff`
- `git checkout -- .`
- `git reset`
- `git reset HEAD~1`
- `git reset --hard HEAD~1`
- `git push --force`

# Reference

- [連猴子都能懂的 Git 入門指南](#)

網路上很多 git 教學，可以自己 google

- [安裝 WSL \( for Windows \)](#)

- [安裝 iterm2 \( for macOS \)](#)

你早晚要使用 git 的，

何不現在就先把環境處理好？

- [Git Commit Message 這樣寫會更好](#)

寫 commit message 有一定的規格