

計算機概論A班 實習課

W6

助教:王常在

課程內容

→ git 講解

→ 問答時間

→ 使用 SSH 金鑰與

→ 作業五

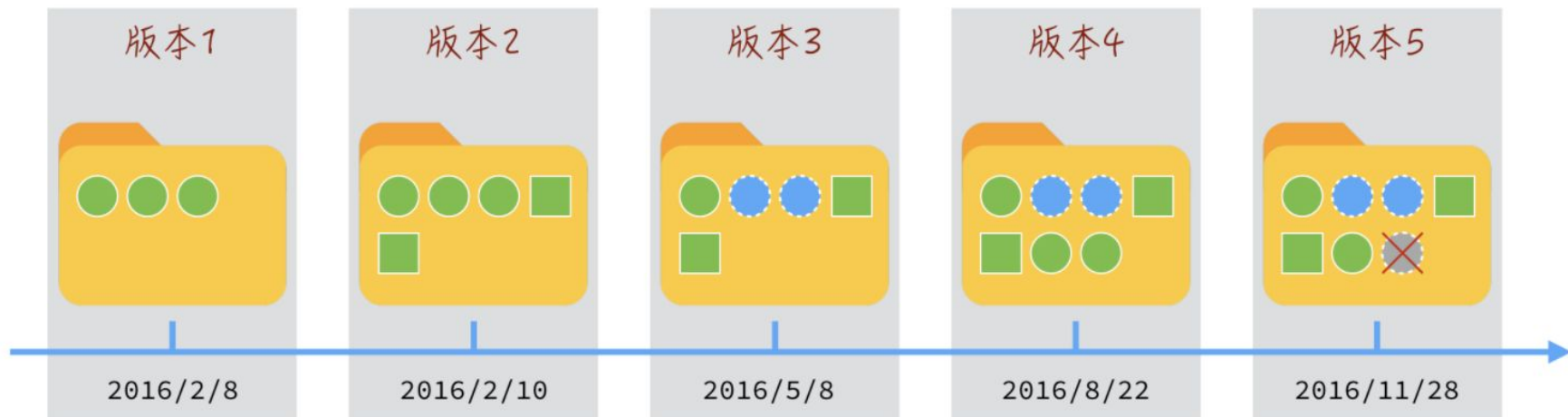
GitHub 連線

→ git 實作

Git 是什麼？

1. 免費、開源專案管理工具
2. 用來做軟體的版本控制與維護
3. 記錄版本更動情形，保留對於檔案的新增、修改或是刪除等操作的歷史紀錄
4. 分散式系統
5. 可離線工作
6. 多人合作專案時

Git-版本控制



Git-版本控制

造成的困擾：

- 當檔案備份過多時
 - 容易忘記每個檔案做了哪些變動，檔案之間的差異
- 當多人共同編輯時
 - 在整合檔案或是不同人在進行反覆編輯時，容易導致原本的檔案被覆蓋掉
 - 修改到別人的程式碼

Git-版本控制的特點

1. 版本儲存

- a. 儲存檔案的重要資訊, EX:編輯者、時間、版本相關資訊

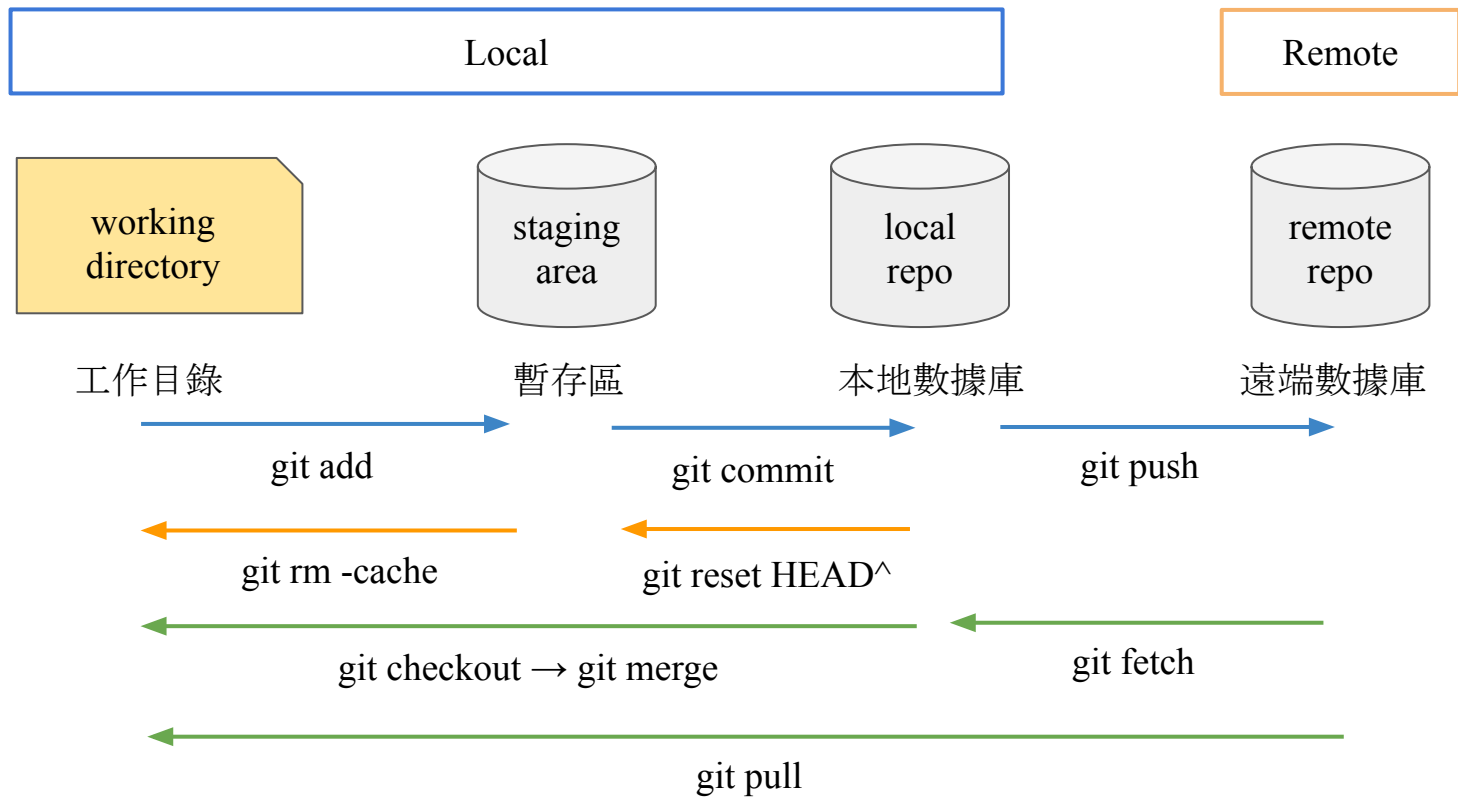
2. 共同編輯

- a. 可藉由repository和共同編輯者分享資料, 不會因為兩人同時編輯, 導致先進行編輯的人的內容被覆蓋掉

3. 儲存空間

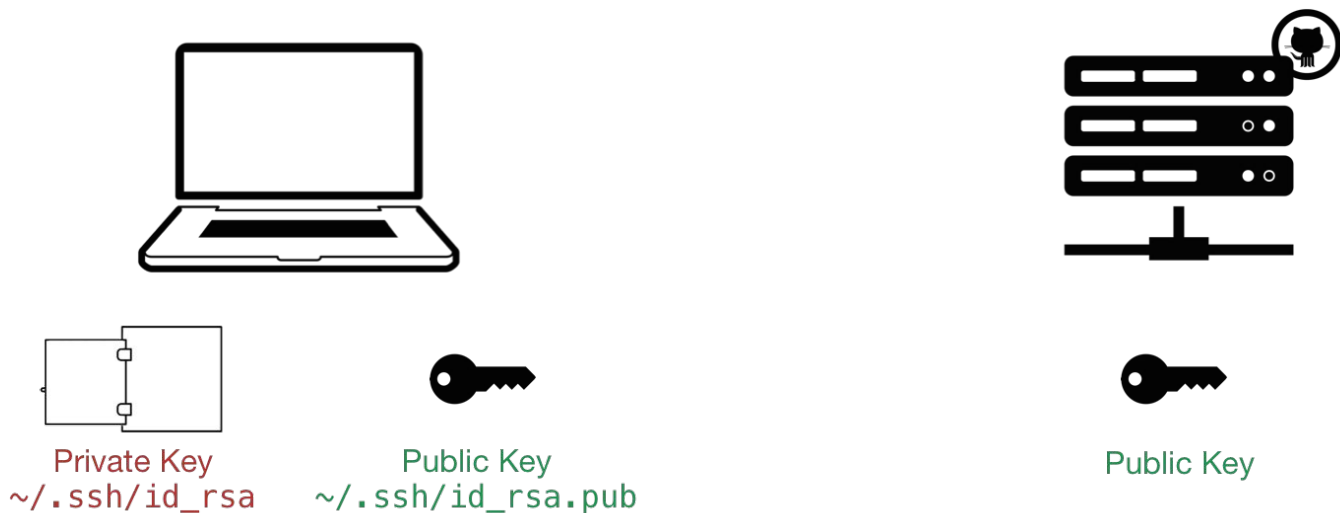
- a. git並不是記錄版本差異, 而是記錄檔案內容的快照, 使git體積小、速度快

Git-版本控制流程



使用 SSH 金鑰與 GitHub 連線

1. 產生金鑰對(公鑰 & 私鑰)
2. 將公鑰放在遠端伺服器(GitHub)

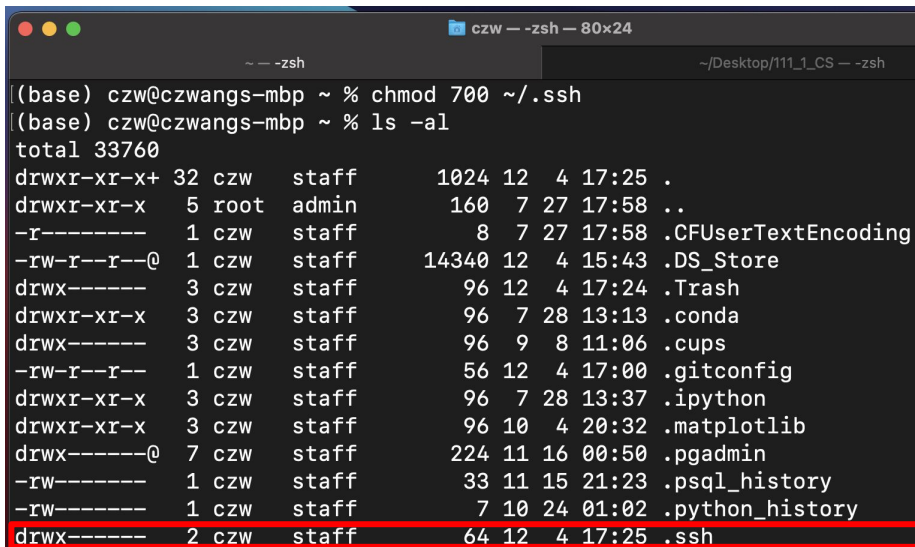


SSH 連線運作方式(圖片來源:[【Git】使用 SSH 金鑰與 GitHub 連線](#))

使用 SSH 金鑰與 GitHub 連線

→ 建立 .ssh 目錄

1. **MacOS**: `mkdir ~/.ssh`
Windows: `mkdir c:/Users/username/.ssh`
2. `chmod 700 ~/.ssh`

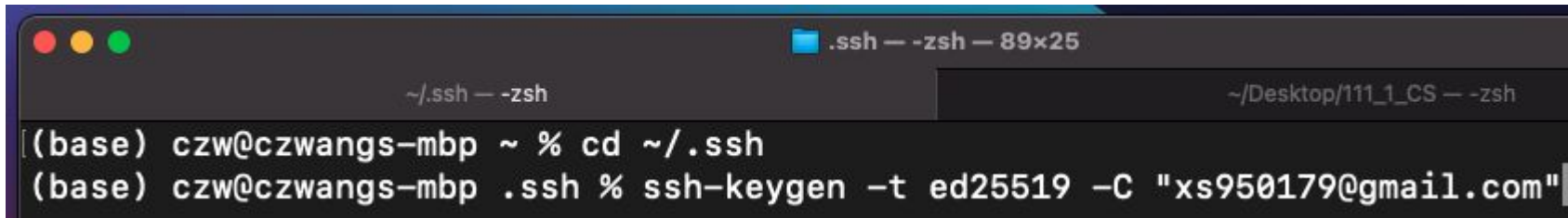


```
czw -- zsh -- 80x24
~ -- zsh
~/Desktop/111_1_CS -- zsh
(base) czw@czwangs-mbp ~ % chmod 700 ~/.ssh
(base) czw@czwangs-mbp ~ % ls -al
total 33760
drwxr-xr-x+ 32 czw  staff      1024 12  4 17:25 .
drwxr-xr-x  5 root  admin       160  7 27 17:58 ..
-r-----   1 czw  staff         8  7 27 17:58 .CFUserTextEncoding
-rw-r--r--@  1 czw  staff    14340 12  4 15:43 .DS_Store
drwx-----  3 czw  staff       96 12  4 17:24 .Trash
drwxr-xr-x  3 czw  staff       96  7 28 13:13 .conda
drwx-----  3 czw  staff       96  9  8 11:06 .cups
-rw-r--r--  1 czw  staff       56 12  4 17:00 .gitconfig
drwxr-xr-x  3 czw  staff       96  7 28 13:37 .ipython
drwxr-xr-x  3 czw  staff       96 10  4 20:32 .matplotlib
drwx-----@  7 czw  staff      224 11 16 00:50 .pgadmin
-rw-----  1 czw  staff       33 11 15 21:23 .psql_history
-rw-----  1 czw  staff        7 10 24 01:02 .python_history
drwx-----  2 czw  staff       64 12  4 17:25 .ssh
```

使用 SSH 金鑰與 GitHub 連線

→ 產生金鑰對

1. `cd ~/.ssh`
2. `ssh-keygen -t ed25519 -C "your_email@example.com"`

A screenshot of a macOS terminal window. The title bar shows a folder icon, ".ssh", and "-zsh - 89x25". The terminal has two tabs: the active one is "~/.ssh - zsh" and the other is "~/Desktop/111_1_CS - zsh". The prompt is "(base) czw@czwangs-mbp ~". The user enters "cd ~/.ssh" and the prompt changes to "(base) czw@czwangs-mbp ~/.ssh". Then the user enters "ssh-keygen -t ed25519 -C 'xs950179@gmail.com'" and the command is being executed.

```
(base) czw@czwangs-mbp ~ % cd ~/.ssh
(base) czw@czwangs-mbp ~/.ssh % ssh-keygen -t ed25519 -C "xs950179@gmail.com"
```

使用 SSH 金鑰與 GitHub 連線

→ 產生金鑰對(過程中會問三個問題)

1. Enter file in which to save the key (/Users/czw/.ssh/id_rsa): **/Users/username/.ssh/github_key**
詢問金鑰儲存的位置和檔名, 預設檔名是id_ed25519。
2. Enter passphrase (empty for no passphrase):
3. Enter same passphrase again:
詢問是否設定密碼保護金鑰。

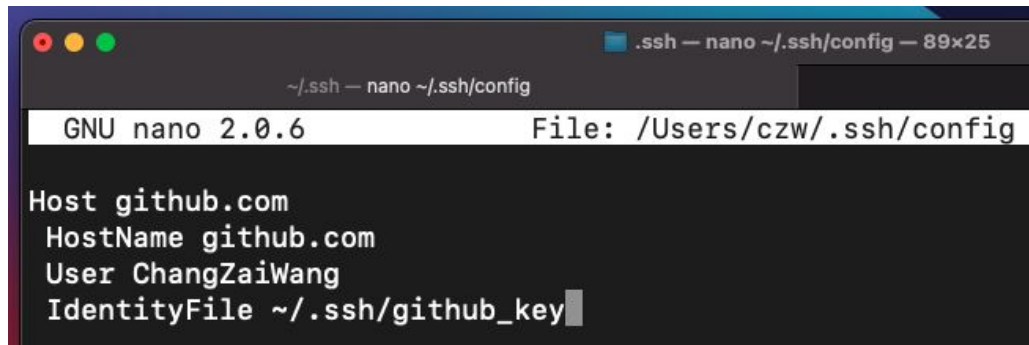
```
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/czw/.ssh/id_ed25519): /Users/czw/.ssh/github_
key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/czw/.ssh/github_key.
Your public key has been saved in /Users/czw/.ssh/github_key.pub.
The key fingerprint is:
SHA256:1Ihyz8L8Alo3gk9UzNg5SGpU8NpfT2ezN0AIjhJgfPs xs950179@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
| .==+*...      |
| 0.0+0*+ + .   |
| 0.=.+.+ 0 .   |
| . =.* + .     |
| 0 *. * S . =  |
|   =E=  o o +  |
| . . 0 . . . 0 |
| .             |
|               |
+----[SHA256]-----+
(base) czw@czwangs-mbp .ssh %
```

使用 SSH 金鑰與 GitHub 連線

→ 本機設定 ssh config

```
nano ~/.ssh/config
```

```
Host github.com
  HostName github.com
  User github_username
  IdentityFile ~/.ssh/github_key
```



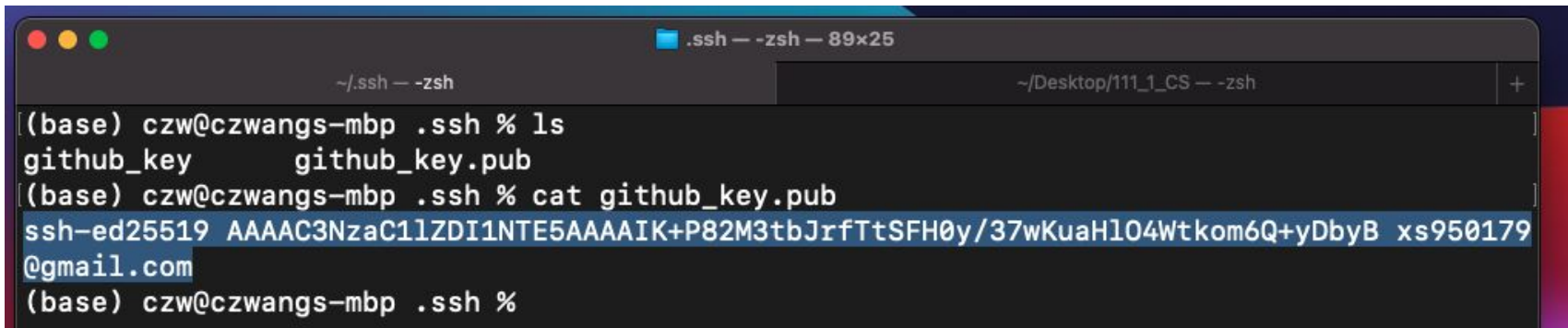
```
~/.ssh - nano ~/.ssh/config
GNU nano 2.0.6 File: /Users/czw/.ssh/config
Host github.com
  HostName github.com
  User ChangZaiWang
  IdentityFile ~/.ssh/github_key
```

1. control + o 儲存檔案
2. enter
3. control + x 離開 nano

使用 SSH 金鑰與 GitHub 連線

→ 新增公鑰至 GitHub 遠端倉庫

1. `ls`
2. `cat github_key.pub`

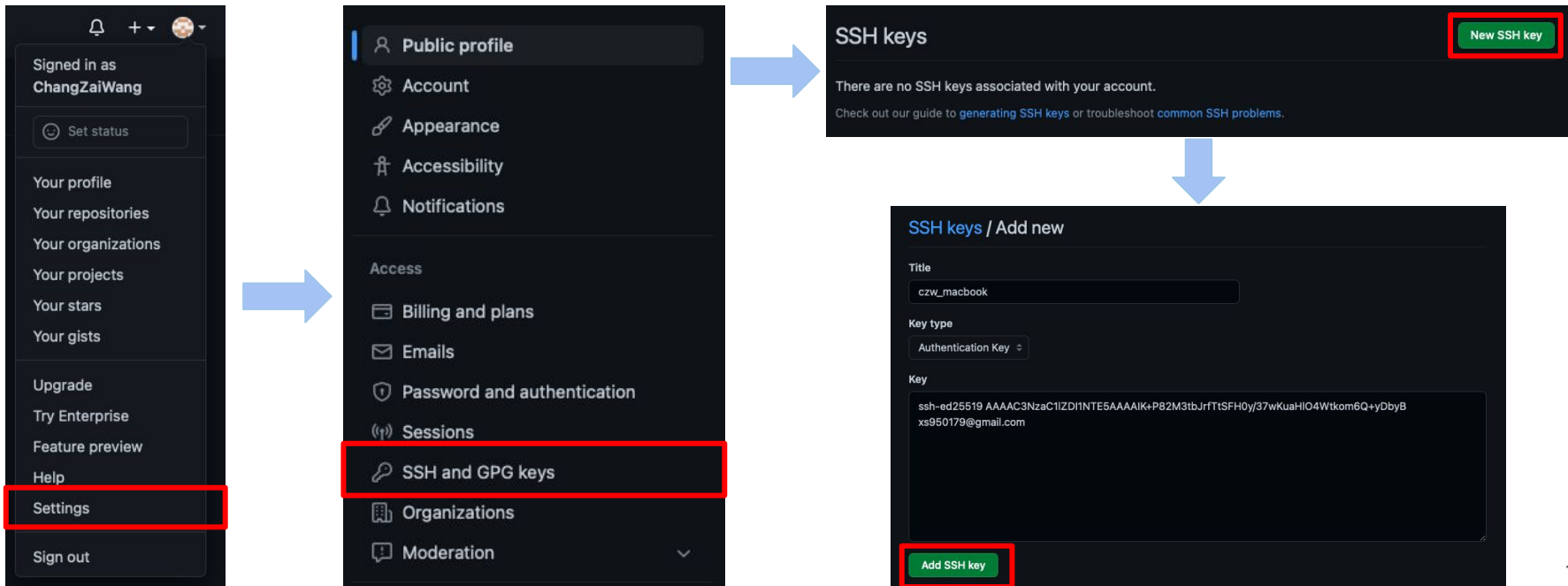
A screenshot of a macOS terminal window with two tabs. The active tab is titled ".ssh — -zsh — 89x25" and shows the following commands and output:

```
(base) czw@czwangs-mbp .ssh % ls
github_key      github_key.pub
(base) czw@czwangs-mbp .ssh % cat github_key.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIK+P82M3tbJrfTtSFH0y/37wKuaHlO4Wtkom6Q+yDbyB xs950179@gmail.com
(base) czw@czwangs-mbp .ssh %
```

The output of the `cat` command is highlighted in blue. The second tab is titled "~ / Desktop / 111_1_CS — -zsh" and is inactive.

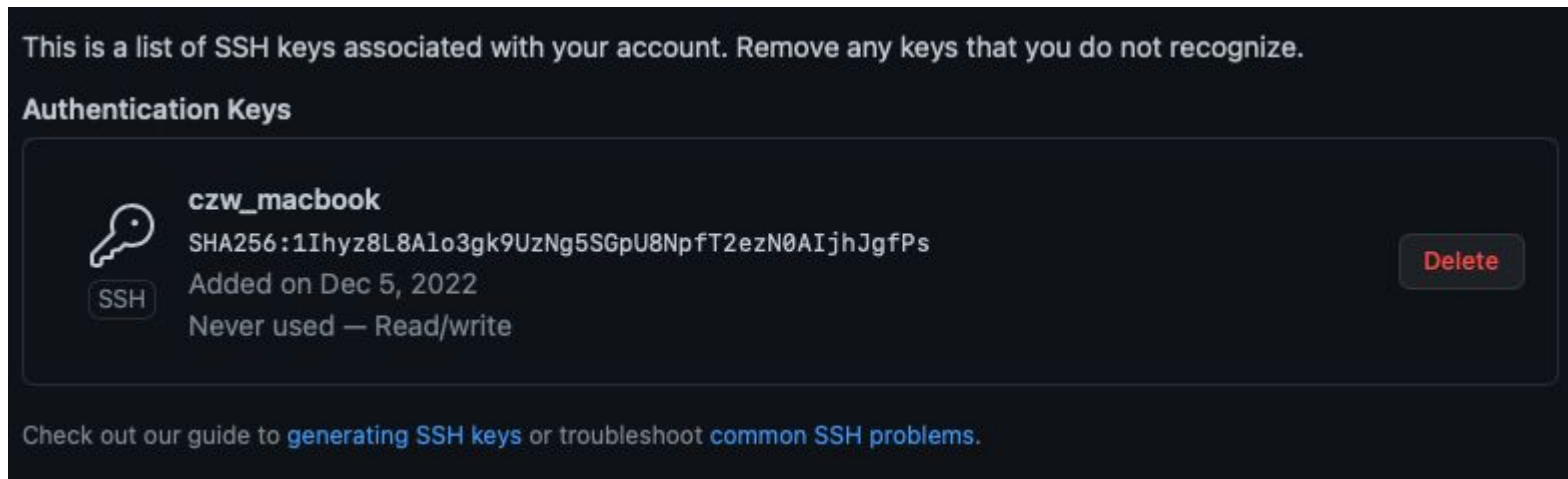
使用 SSH 金鑰與 GitHub 連線

➔ 新增公鑰至 GitHub 遠端倉庫



使用 SSH 金鑰與 GitHub 連線

→ 新增公鑰至 GitHub 遠端倉庫

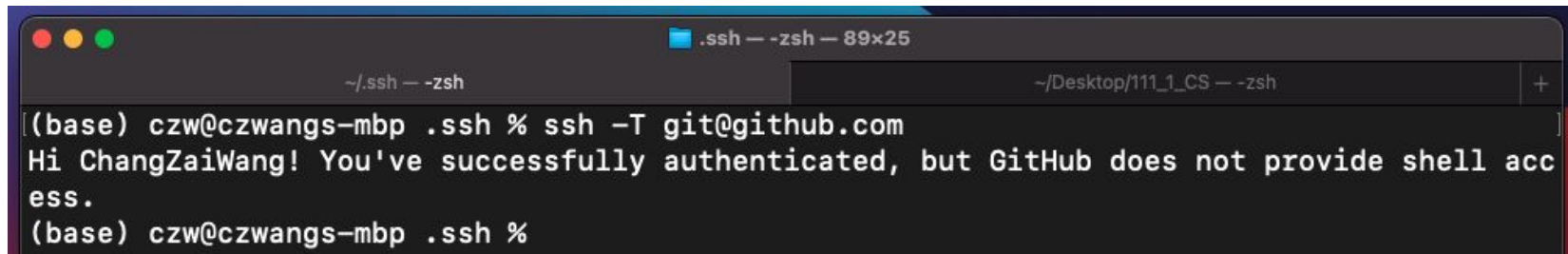


出現上面畫面表示新增成功！

使用 SSH 金鑰與 GitHub 連線

→ 測試連線

```
ssh -T git@github.com
```

A screenshot of a macOS terminal window with a dark background. The window title is ".ssh — -zsh — 89x25". There are two tabs: the active one is "~/.ssh — -zsh" and the other is "~/Desktop/111_1_CS — -zsh". The terminal shows a user in a conda base environment running the command `ssh -T git@github.com`. The output is "Hi ChangZaiWang! You've successfully authenticated, but GitHub does not provide shell access." followed by a new prompt.

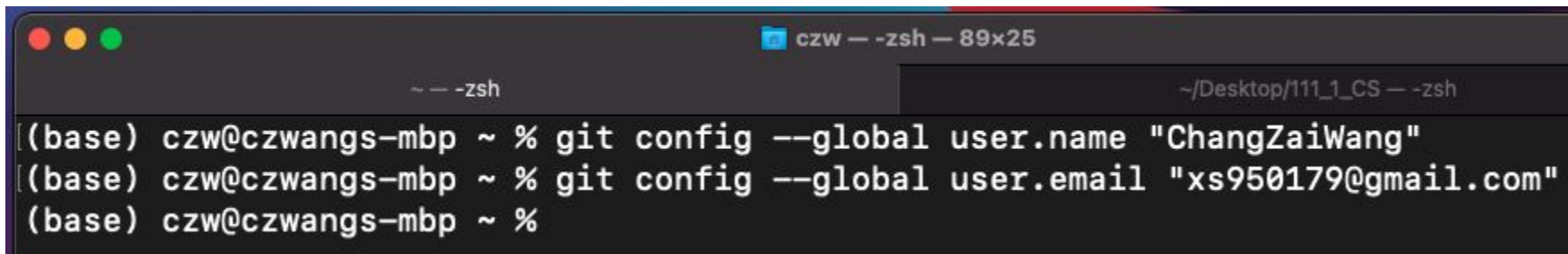
```
(base) czw@czwangs-mbp .ssh % ssh -T git@github.com
Hi ChangZaiWang! You've successfully authenticated, but GitHub does not provide shell access.
(base) czw@czwangs-mbp .ssh %
```


請先準備好

1. github email
2. github user name
3. github repo link

Git 連結帳號至 Github

1. `$ git config --global user.name "Your Name"`
2. `$ git config --global user.email "your@gmail.com"`

A screenshot of a macOS terminal window. The title bar shows 'czw - zsh - 89x25'. The terminal has a dark background with light gray text. The prompt is '(base) czw@czwangs-mbp ~ %'. The user has entered two commands: 'git config --global user.name "ChangZaiWang"' and 'git config --global user.email "xs950179@gmail.com"'. The terminal shows the output of these commands as the prompt returns.

```
(base) czw@czwangs-mbp ~ % git config --global user.name "ChangZaiWang"
(base) czw@czwangs-mbp ~ % git config --global user.email "xs950179@gmail.com"
(base) czw@czwangs-mbp ~ %
```

Git 指令

- 初始化專案: `git init`
- 單一檔案加入索引(暫存區): `git add <檔案名稱>`
- 所有檔案加入索引(暫存區): `git add .`
- 觀看當前狀態: `git status`
- 提交版本: `git commit -m “修改紀錄”`
- 瀏覽歷史紀錄: `git log`

“.” 的意思代表全部

Git-使用方法 Local(本地端)



- I . 新增Working directory(工作目錄)
 1. 建立資料夾(ex:mkdir cs6_git)
 2. 移動到資料夾:\$ cd cs6_git
 3. 將專案資料夾建立成git repository:\$ git init (初始化資料夾)
此時會產生隱藏檔(.git), 而這個隱藏檔會追蹤修改
 4. 新增檔案(ex:index.html)
 5. 查看資料夾內檔案變化:\$ git status
此時檔案尚未被追蹤(Untracked files)

Git-使用方法 Local(本地端)



二. 進入Staging area(暫存區)

1. 將新增或變更的檔案加入追蹤:\$ git add index.html
2. 查看資料夾內檔案變化:\$ git status
#此時檔案已被追蹤
3. 再新增一個檔案(ex:README.md)
4. 查看資料夾內檔案的變化:\$ git status
5. 將所有的檔案加入追蹤:\$ git add --all

Git-使用方法 Local(本地端)



三. 進入Local repository(本地數據庫)

1. 將檔案移入本地repo, 提交新版本:

```
$ git commit -m "First release of Hello"
```

-m: 填寫版本資訊、修改紀錄。方便日後查找

2. 使用\$ git status觀察:\$ git status

出現"nothing to commit, working tree clean", 因為暫存區的檔案已被提交成新的版本了

3. 查看新增的版本(歷史紀錄):\$ git log

會看到版本更新紀錄

Git- 取消追蹤檔案Local(本地端)



四. 檔案從Staging area(暫存區)退回Working directory工作目錄

(不是真的想把這個檔案刪掉, 只是不想讓這個檔案再被 Git 控管)

- 修改index.html: `nano index.html`
- 查看狀態: `$ git status`
- 將檔案加入索引: `$ git add index.html`
- 查看狀態: `$ git status`
- 取消追蹤檔案: `$ git rm --cached index.html`
#此時檔案變回尚未被追蹤(Untracked files)
- 查看狀態: `$ git status`

Git- 取消commit Local(本地端)



五. 將commit拆掉

- 新增檔案: `nano hello.py`
- 將檔案加入索引: `$ git add .`
- 提交版本: `$ git commit -m "print method"`
- 查看歷史紀錄: `$ git log`
- 取消commit: `$ git reset --mixed HEAD^`
 - # `^` 符號表示「前一次」的意思, 回到前n個 `HEAD~n`
- 查看歷史紀錄: `$ git log`
 - # 會發現最後一次commit的紀錄不見了
- 查看狀態: `$ git status`
 - # 會出現hello.py尚未被追蹤, 代表此時檔案在工作目錄

Git- Reset模式

Reset有三種模式：

- **--mixed**: 預設模式, Commit 拆出來的檔案會留在工作目錄, 但不會留在暫存區
- **--soft**: 工作目錄跟暫存區的檔案都不會被丟掉
- **--hard**: 工作目錄以及暫存區的檔案都會丟掉

	--mixed	--soft	--hard
commit拆出來的檔案	丟回工作目錄	丟回暫存區	直接丟掉

Git-使用方法 Remote(雲端)



六. 進入Remote repository(遠端數據庫)

1. 雲端跟本地端連動:

```
$ git remote add origin git@github.com:github_username/project.git
```

2. Push Local master(主幹)進入Remote:

```
$ git push --set-upstream origin main
```

3. 回到Github查看

等同於\$ git push -u origin master

-u : 設定要push到哪裡, 當之後push不加參數時, 會將本地Local repo的master分支, push到遠端的origin節點下

Git-使用方法 Remote(雲端)



七. Remote repository(遠端數據庫)

1. 到github上新增或編輯README.md
2. 把遠端東西拉回來: `$ git fetch` # 執行 Fetch 指令後, Git 看了一下線上版本的內容後, 將目前線上有但本地這邊沒有的內容抓了一份下來
3. 查看狀態: `$ git status`
4. 查看歷史紀錄: `$ git log` # 會發現在github上發送的commit被抓下來了
5. 比較本地分支和遠端分支內容的不同: `$ git diff origin/main`
6. 將遠端和本地端的內容做合併: `$ git merge origin/main`
7. 查看狀態: `$ git status`
8. Pull下載更新: `$ git pull origin` # 將本地main的資料更新, 以防與Remote端不同

作業五

將這學期實習課內容整理成筆記，
並放置在Github Repo中

開一個.md檔，用markdown語法寫
請寫在現在正在使用的repo中

期限:2022-12-31