Lab of Object-Oriented Programming:

Git & Github

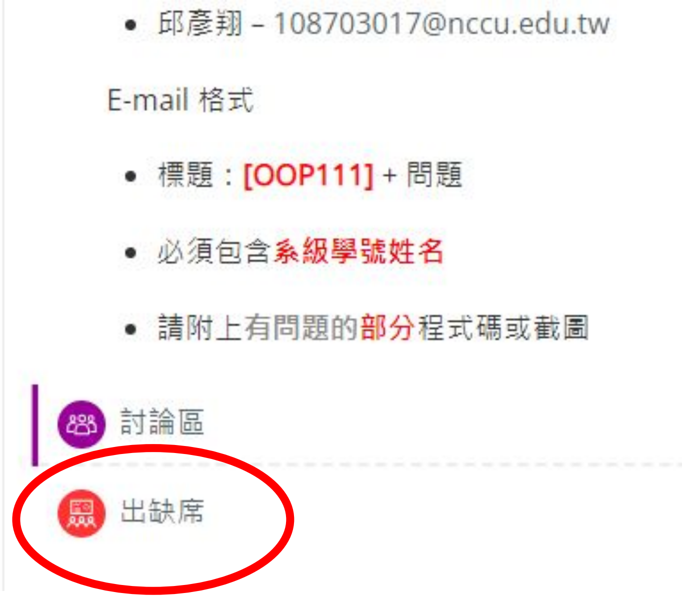
黃威、陳岳紘、邱彥翔

2022

使用 moodle 點名

請登入實習課的 moodle 課程

點擊出缺席並完成今日的點名



- 邱彥翔 – 108703017@nccu.edu.tw

E-mail 格式

- 標題：[OOP111] + 問題
- 必須包含系級學號姓名
- 請附上有問題的部分程式碼或截圖

Agenda

- Version Control
- Git
- Github

Version Control

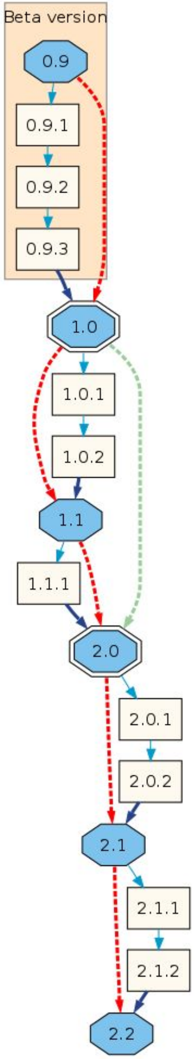
Version Control

Major.Minor(.Build)

Major(主版本號)：重大更新

Minor(子版本號)：小更新，不至於到 Major 的程度

Build(修正版本號)：錯誤修正



Version Control

1. 主版本號為 0 時為 Beta 測試版

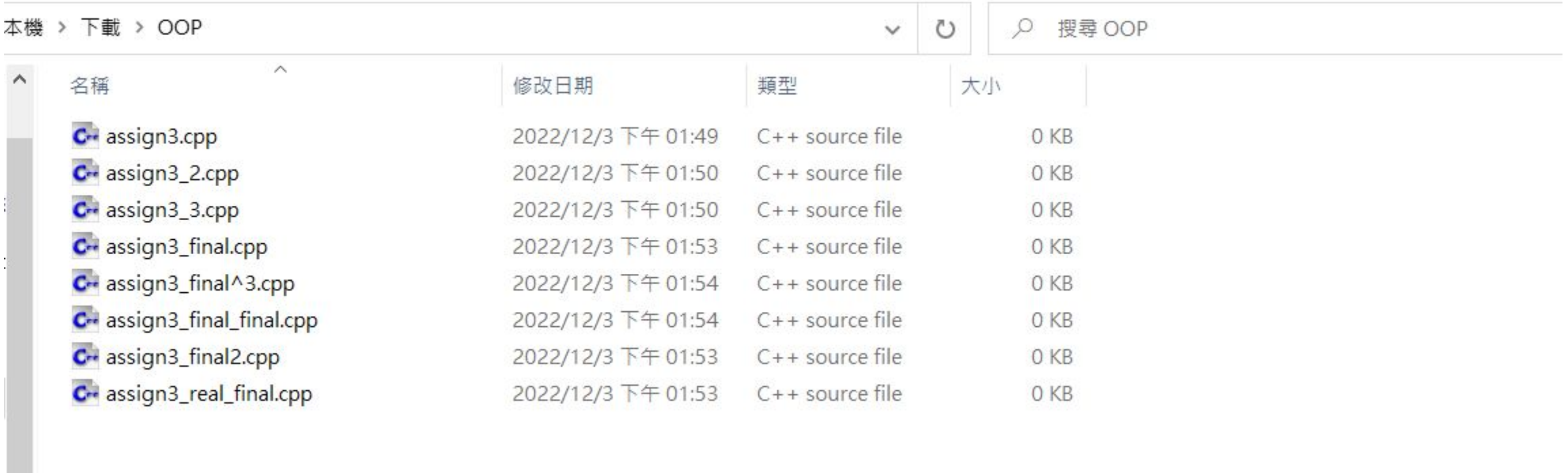
2. 當項□在進□了局部修改或 bug 修正時，修正版本號加 1

3. 當項□在原有的基礎上增加了部分功能時，□版本號加 1，當修正版本號為 0 可以忽略

4. 當項□在進□了重□修改或局部修正累積較多，導致項□整體發□全局變化時，主版本號加 1

Git

Why Use Git?



名稱	修改日期	類型	大小
assign3.cpp	2022/12/3 下午 01:49	C++ source file	0 KB
assign3_2.cpp	2022/12/3 下午 01:50	C++ source file	0 KB
assign3_3.cpp	2022/12/3 下午 01:50	C++ source file	0 KB
assign3_final.cpp	2022/12/3 下午 01:53	C++ source file	0 KB
assign3_final^3.cpp	2022/12/3 下午 01:54	C++ source file	0 KB
assign3_final_final.cpp	2022/12/3 下午 01:54	C++ source file	0 KB
assign3_final2.cpp	2022/12/3 下午 01:53	C++ source file	0 KB
assign3_real_final.cpp	2022/12/3 下午 01:53	C++ source file	0 KB

Install Git

Install Git - Linux

Fedora

```
$ sudo yum install git-all
```

Debian Series (like Ubuntu)

```
$ sudo apt-get install git-all
```

https://git-scm.com/download/linux

Install Git - Mac

```
$ brew install git
```

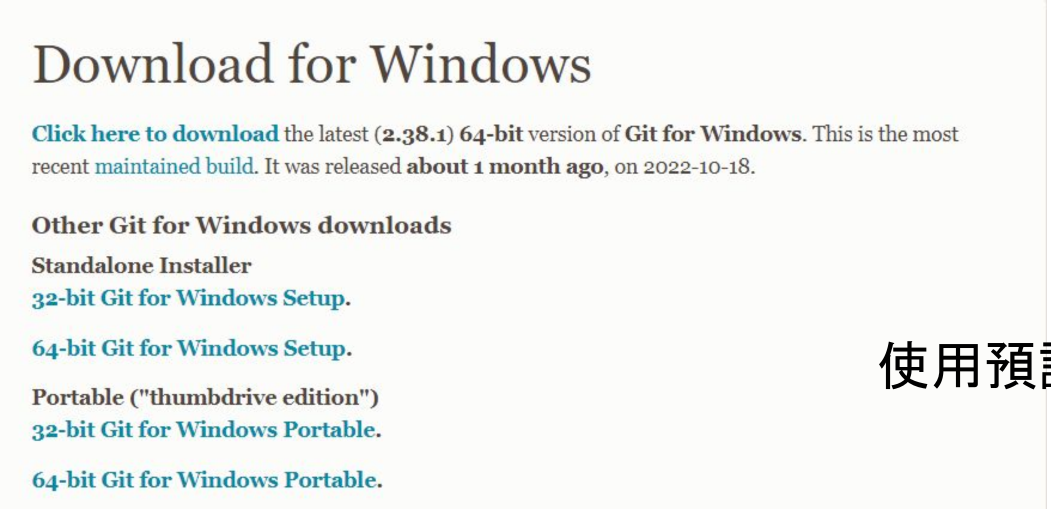
沒有 homebrew? install homebrew:

```
    $ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

https://git-scm.com/download/mac

Install Git - Windows

https://git-scm.com/download/win



Download for Windows

Click here to download the latest (**2.38.1**) **64-bit** version of **Git for Windows**. This is the most recent maintained build. It was released **about 1 month ago**, on 2022-10-18.

Other Git for Windows downloads

Standalone Installer
32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

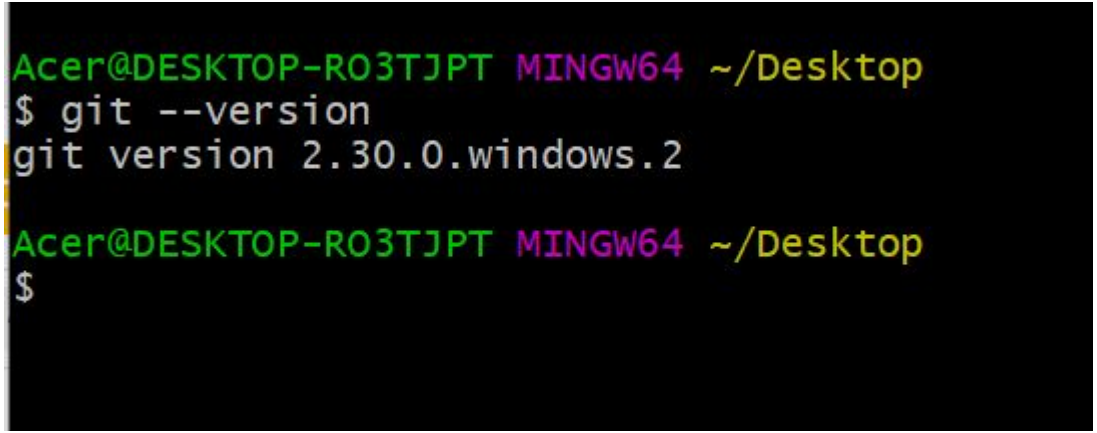
Portable ("thumbdrive edition")
32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

使用預設配置，按 Next 即可

Install Git - git --version

輸入 git --version，以下畫面代表安裝成功



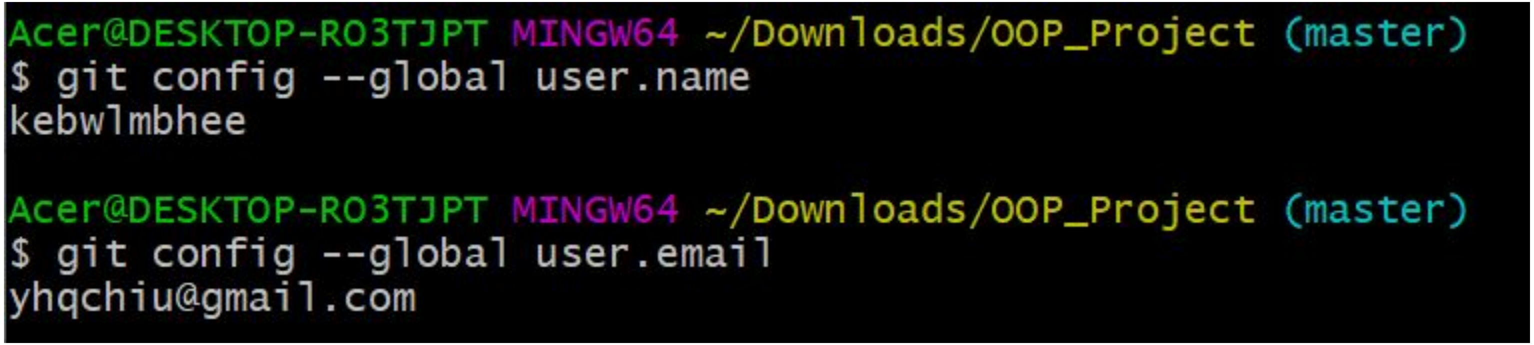
```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Desktop
$ git --version
git version 2.30.0.windows.2

Acer@DESKTOP-RO3TJPT MINGW64 ~/Desktop
$
```

Start Using Git

Git - git config

git config --global user.name *YourUserName*

git config --global user.email *YourUserEmail*



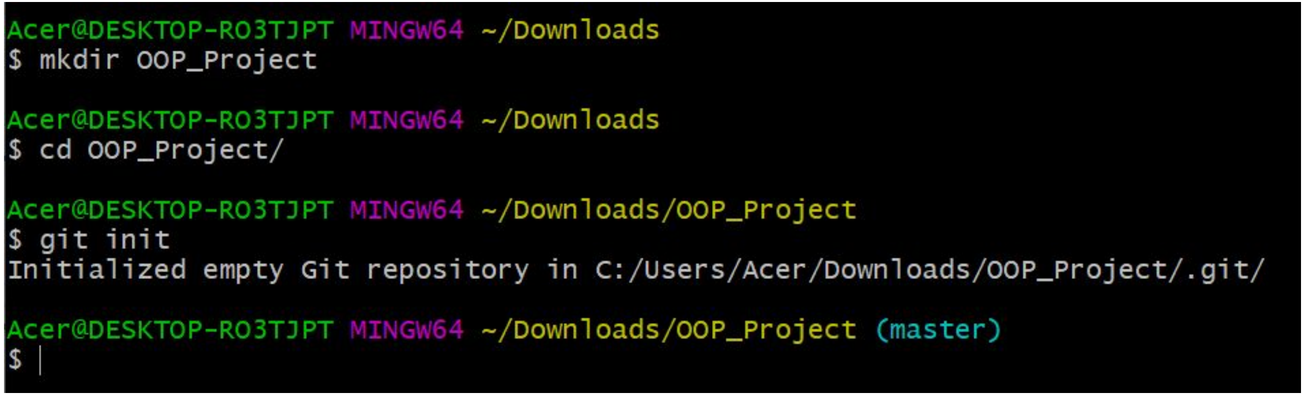
```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git config --global user.name
kebwlmbhee

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git config --global user.email
yhqchiu@gmail.com
```

Git - git init

建立□個名為 OOP_Project 的directory

進入 OOP_Project directory

在 OOP_Project 底下執□ git init

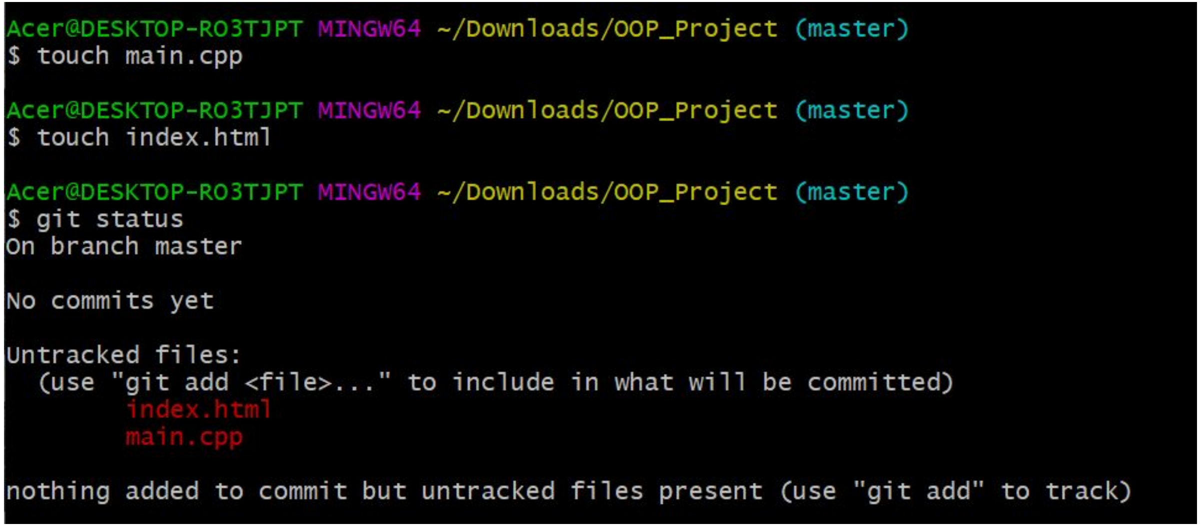


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads
$ mkdir OOP_Project

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads
$ cd OOP_Project/

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project
$ git init
Initialized empty Git repository in C:/Users/Acer/Downloads/OOP_Project/.git/

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ 
```

Git - git status

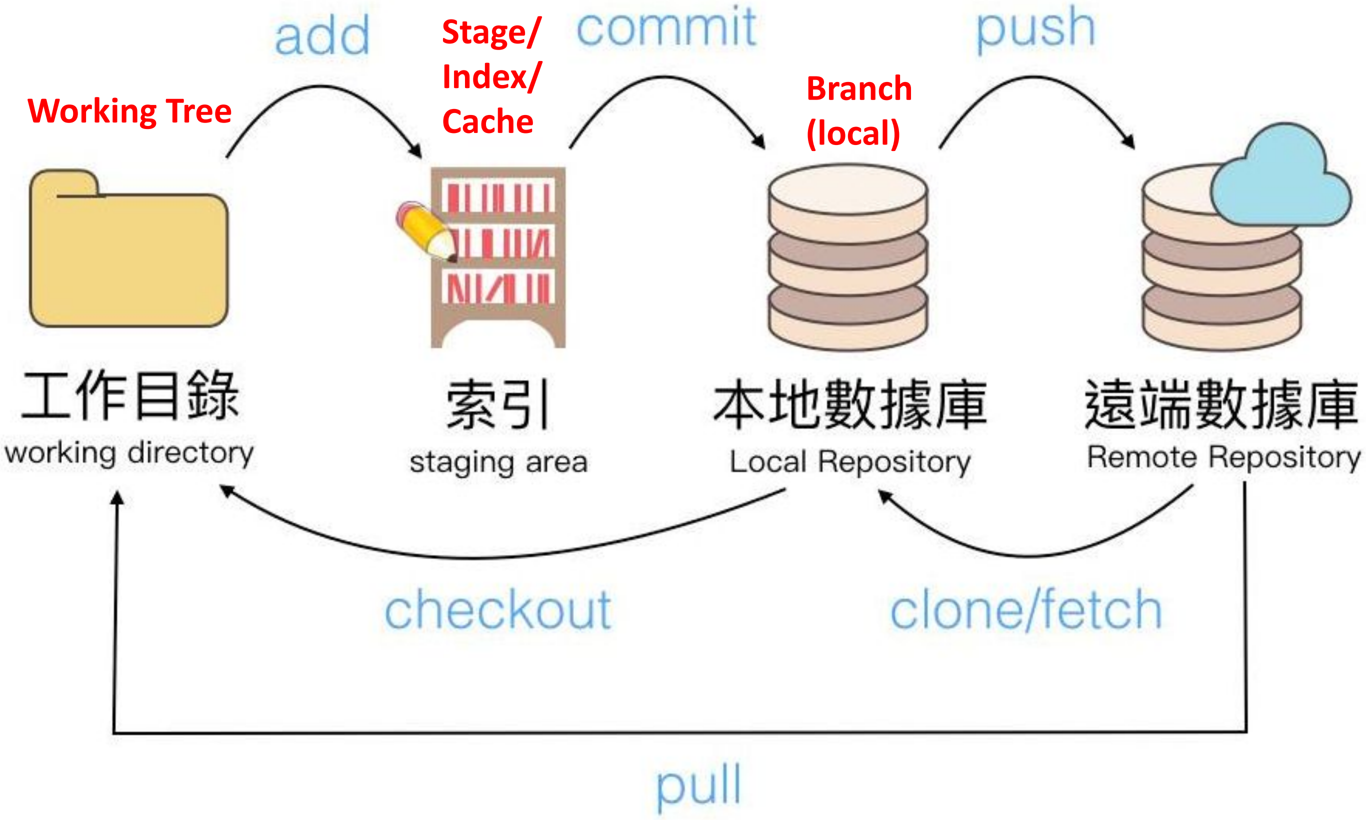
touch 在目錄下新增檔案

git status 查詢當前目錄的 git 狀態

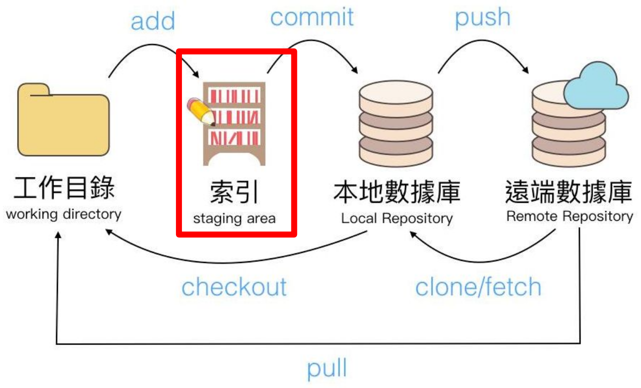


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ touch main.cpp

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ touch index.html

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        main.cpp

nothing added to commit but untracked files present (use "git add" to track)
```

Git - git status

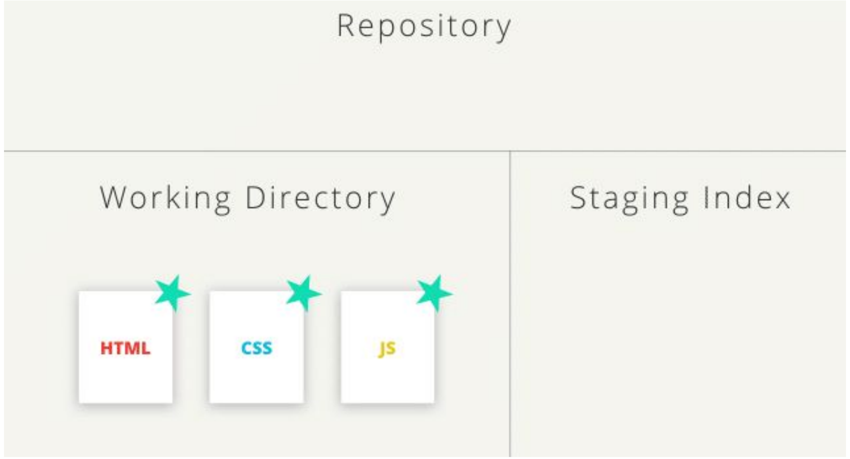


Git - git add

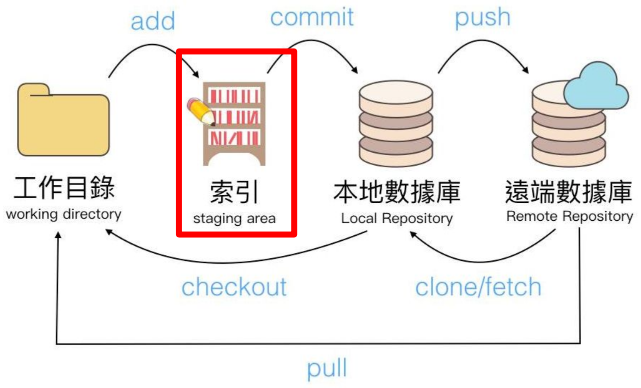


git add <files> 追蹤檔案，將其加入暫存區

或使用 git add . 追蹤當前目錄下所有檔案並加入暫存區

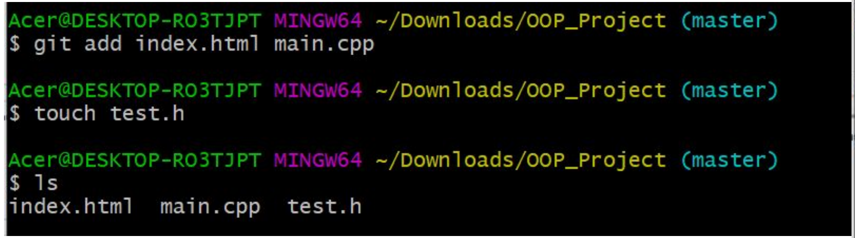


Git - git add

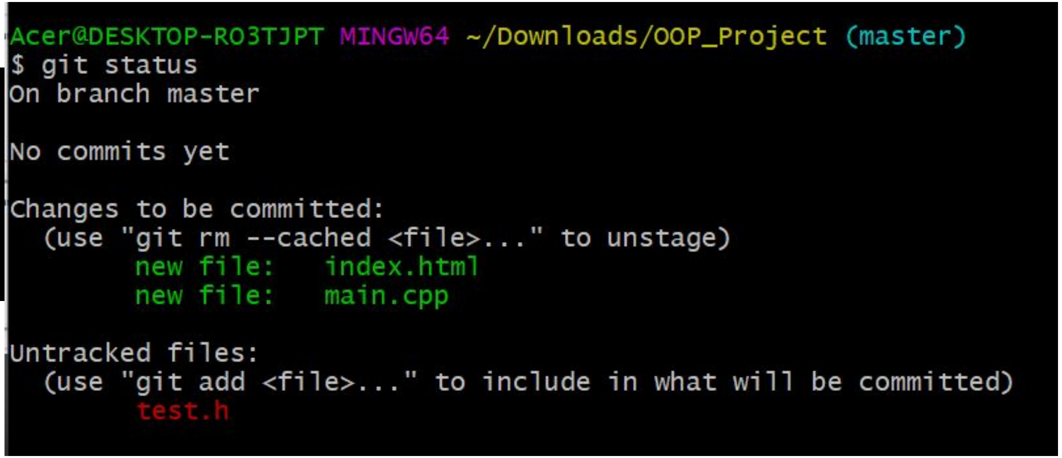


git add <files> 追蹤檔案，將其加入暫存區

或使用 git add . 追蹤當前目錄下所有檔案並加入暫存區



```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git add index.html main.cpp

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ touch test.h

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ ls
index.html  main.cpp  test.h
```

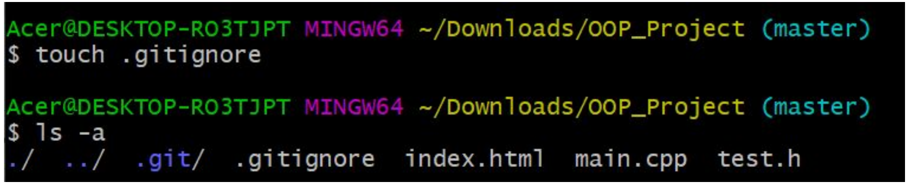


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   main.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.h
```

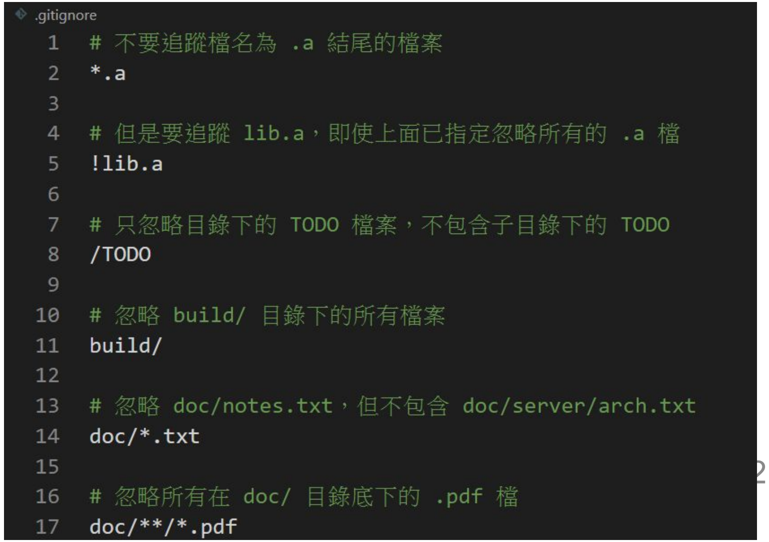
Git - .gitignore file

在目錄下建立 .gitignore 檔案來忽略追蹤

符合 .gitignore 裡規定的檔案在狀態(git status)中無法被看到



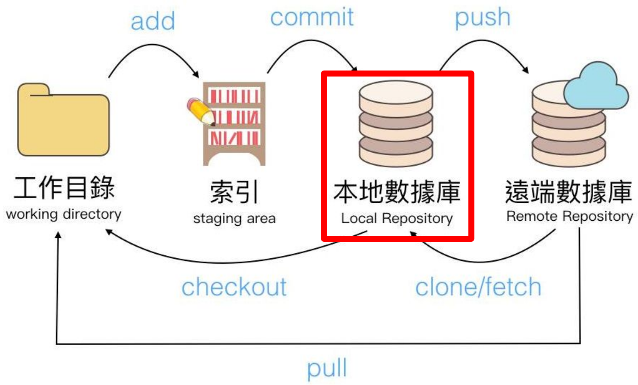
```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ touch .gitignore

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ ls -a
./  ../  .git/  .gitignore  index.html  main.cpp  test.h
```



.gitignore

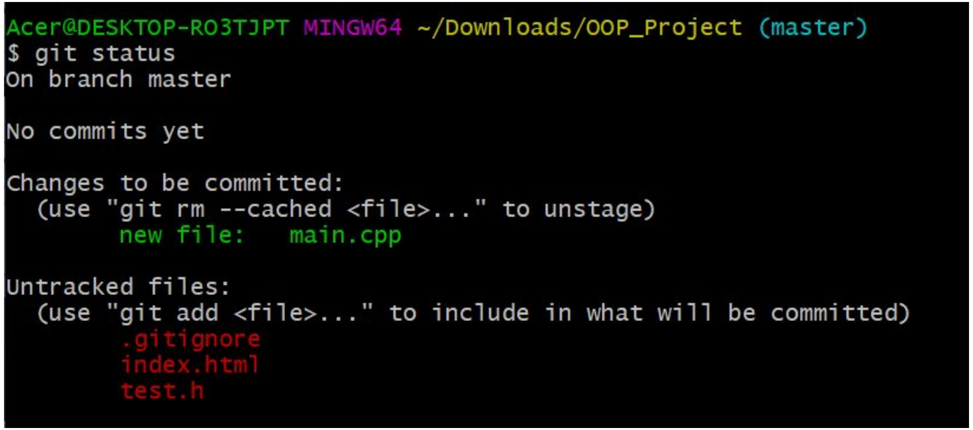
```
# 不要追蹤檔名為 .a 結尾的檔案
*.a

# 但是要追蹤 lib.a，即使上面已指定忽略所有的 .a 檔
!lib.a

# 只忽略目錄下的 TODO 檔案，不包含子目錄下的 TODO
/TODO

# 忽略 build/ 目錄下的所有檔案
build/

# 忽略 doc/notes.txt，但不包含 doc/server/arch.txt
doc/*.txt

# 忽略所有在 doc/ 目錄底下的 .pdf 檔
doc/**/*.pdf
```

Git - git commit

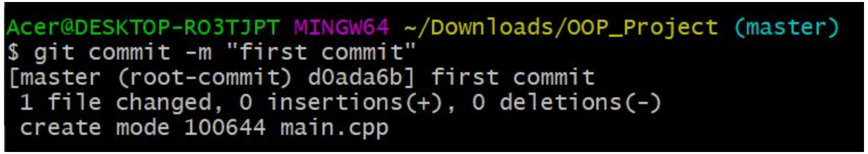


git commit 提交 new branch

-m 代表 one-line Message

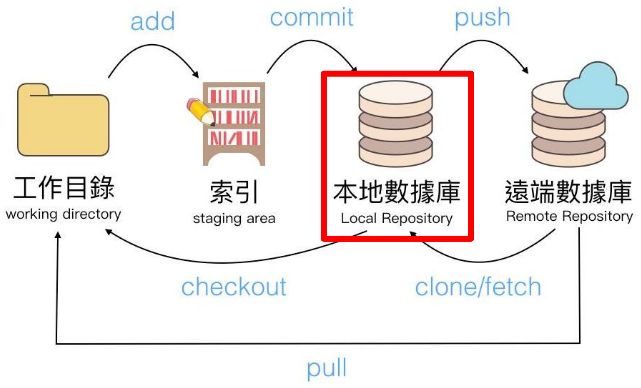


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   main.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        index.html
        test.h
```

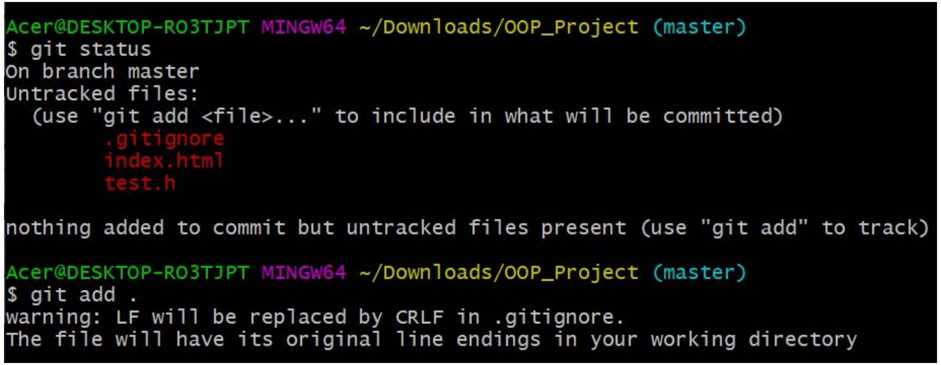


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git commit -m "first commit"
[master (root-commit) d0ada6b] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 main.cpp
```

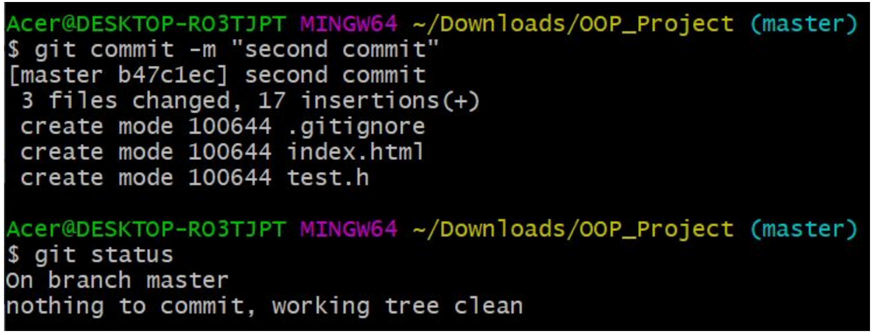
Git - git commit



git commit 只會提交已加入追蹤(git add)的檔案



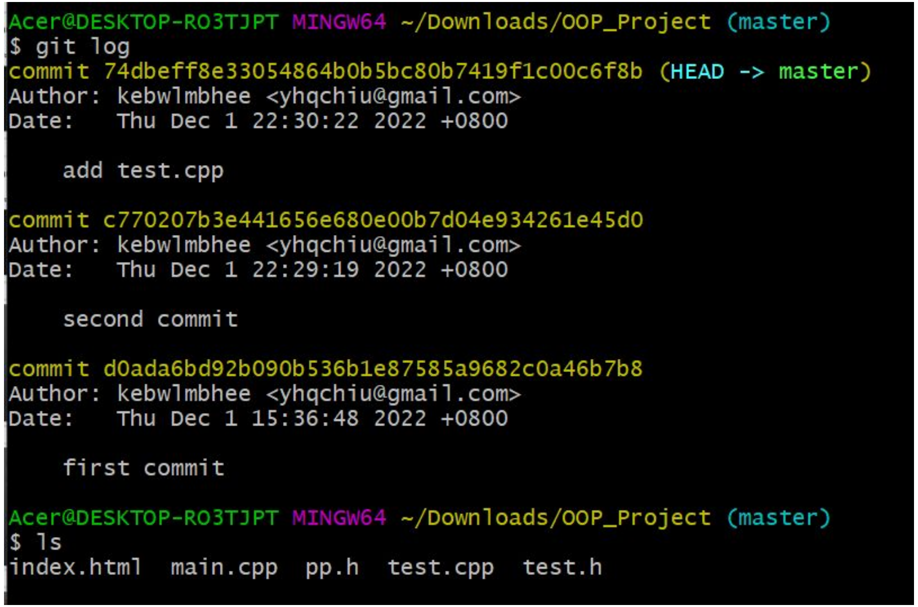
```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        index.html
        test.h

nothing added to commit but untracked files present (use "git add" to track)

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory
```



```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git commit -m "second commit"
[master b47c1ec] second commit
 3 files changed, 17 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 index.html
 create mode 100644 test.h

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git status
On branch master
nothing to commit, working tree clean
```

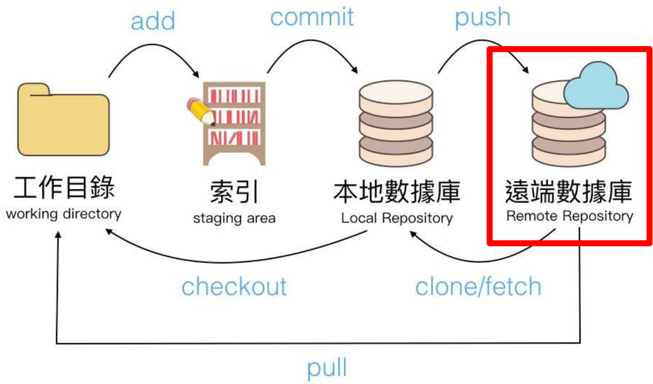
Git - git log

git log 查看提交(git commit)記錄



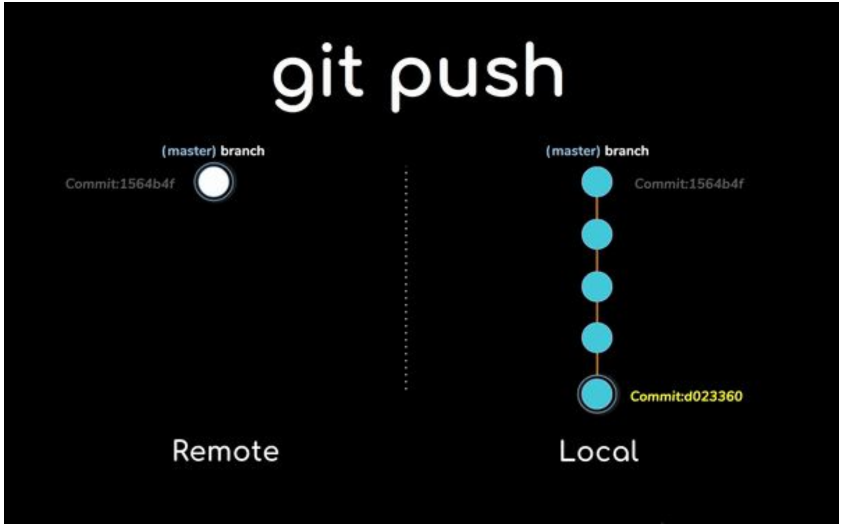
```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git log
commit 74dbeff8e33054864b0b5bc80b7419f1c00c6f8b (HEAD -> master)
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 22:30:22 2022 +0800

    add test.cpp

commit c770207b3e441656e680e00b7d04e934261e45d0
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 22:29:19 2022 +0800

    second commit

commit d0ada6bd92b090b536b1e87585a9682c0a46b7b8
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 15:36:48 2022 +0800

    first commit

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ ls
index.html  main.cpp  pp.h  test.cpp  test.h
```

Git - git push

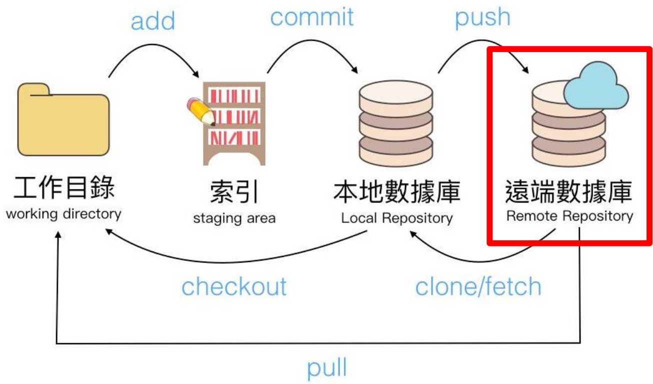


在設定好遠端儲存庫後

使用 git push 將本地端提交記錄推至遠端

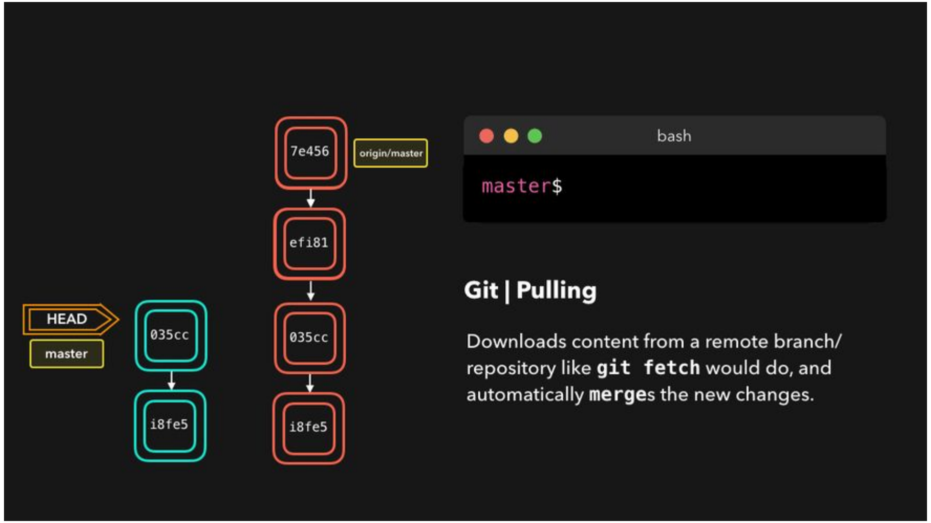


Git - git pull



在設定好遠端儲存庫後

遠端提交記錄超前本地端時，使用 git pull 將遠端記錄拉回本地端

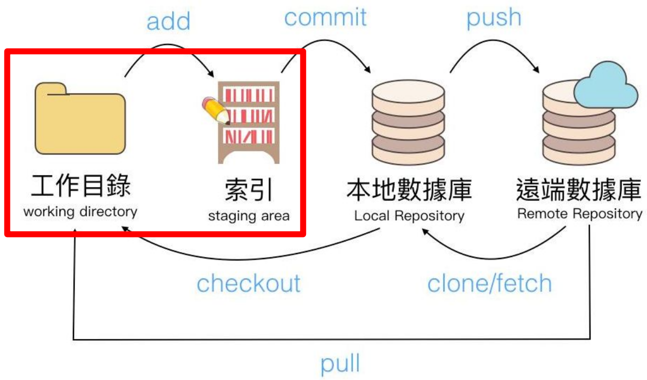


```
master$
```

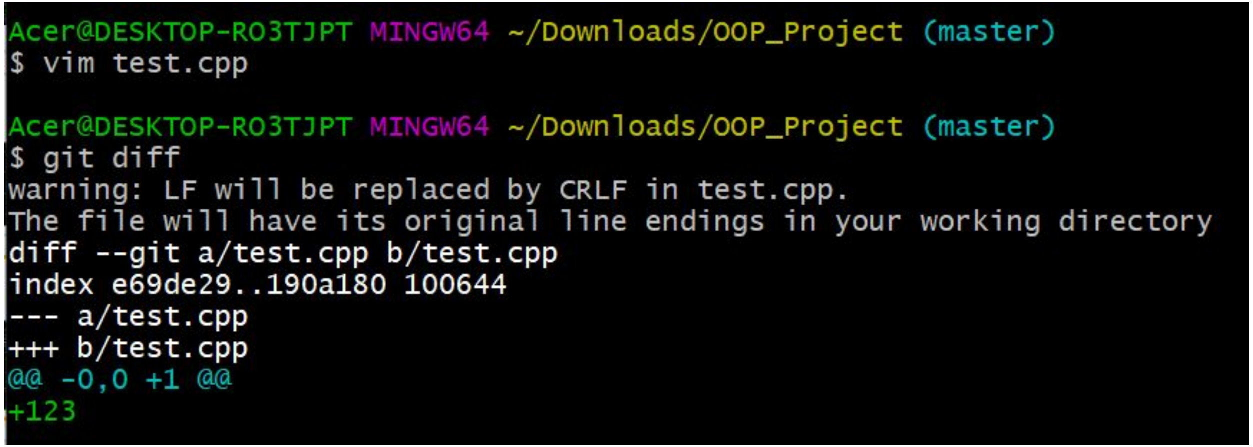
Git | Pulling

Downloads content from a remote branch/ repository like **git fetch** would do, and automatically **merge**s the new changes.

Git - git diff

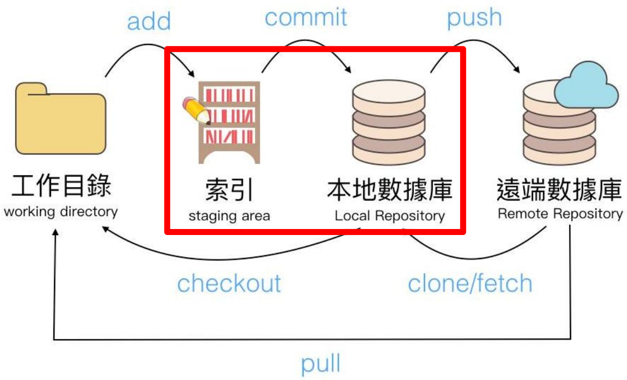


git diff 查看當前未加入追蹤(git add)的變更

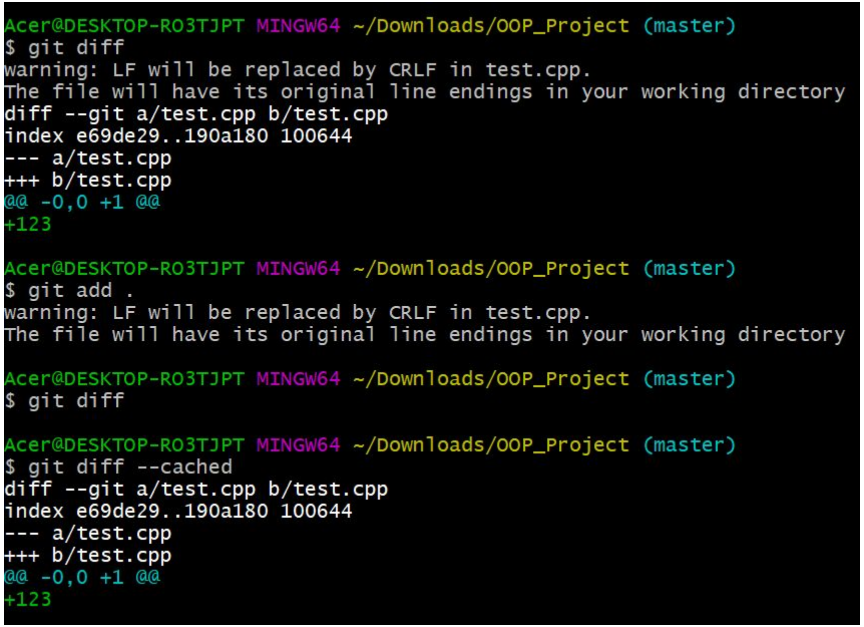


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ vim test.cpp

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff
warning: LF will be replaced by CRLF in test.cpp.
The file will have its original line endings in your working directory
diff --git a/test.cpp b/test.cpp
index e69de29..190a180 100644
--- a/test.cpp
+++ b/test.cpp
@@ -0,0 +1 @@
+123
```

Git - git diff --cached

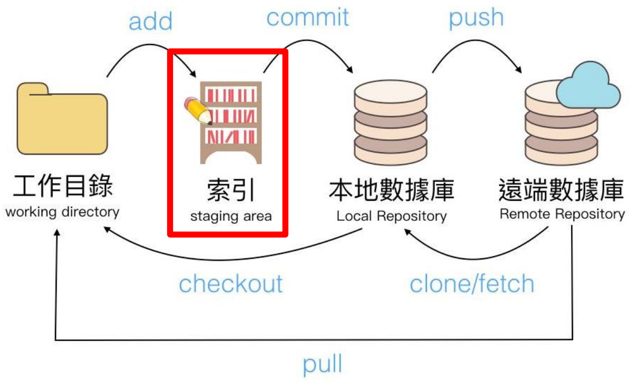


git diff --cached 查看暫存區與 HEAD 相比的變更

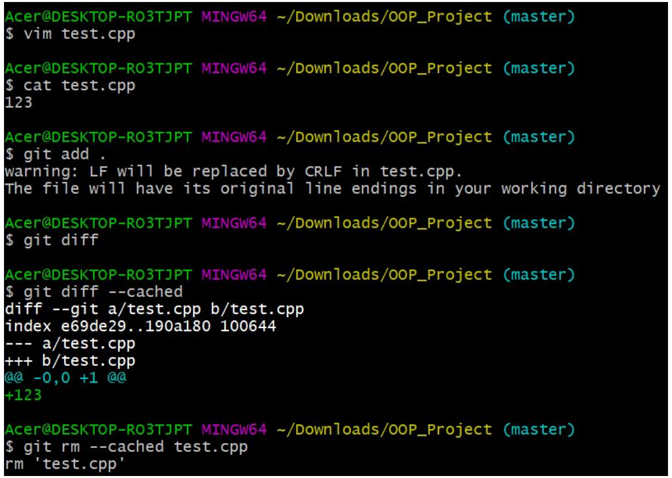


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff
warning: LF will be replaced by CRLF in test.cpp.
The file will have its original line endings in your working directory
diff --git a/test.cpp b/test.cpp
index e69de29..190a180 100644
--- a/test.cpp
+++ b/test.cpp
@@ -0,0 +1 @@
+123

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git add .
warning: LF will be replaced by CRLF in test.cpp.
The file will have its original line endings in your working directory

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff --cached
diff --git a/test.cpp b/test.cpp
index e69de29..190a180 100644
--- a/test.cpp
+++ b/test.cpp
@@ -0,0 +1 @@
+123
```

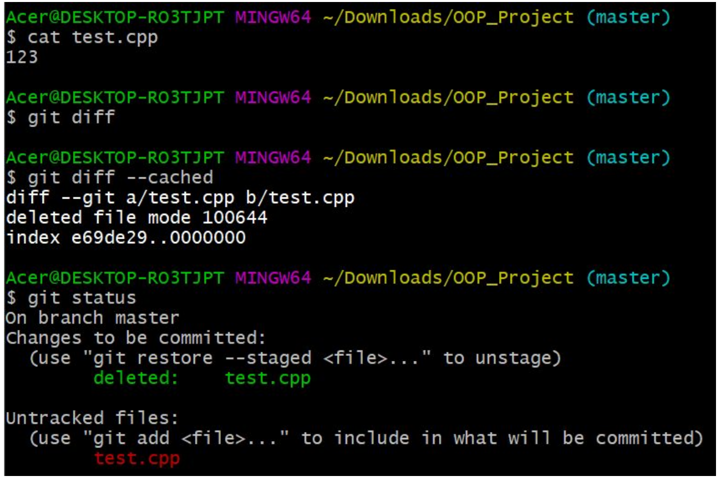
Git - git rm --cached



git rm --cached <files> 將檔案取消追蹤,
工作目錄裡仍留存更改，移除位於暫存區裡的檔案複本(git diff --cached)
若沒加 --cached 會直接刪除檔案

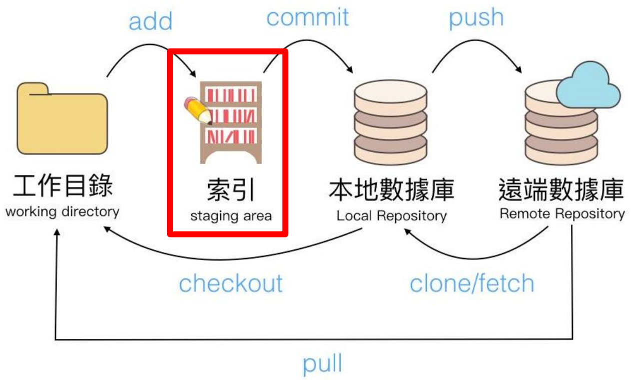


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ vim test.cpp

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ cat test.cpp
123

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git add .
warning: LF will be replaced by CRLF in test.cpp.
The file will have its original line endings in your working directory

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff --cached
diff --git a/test.cpp b/test.cpp
index e69de29..190a180 100644
--- a/test.cpp
+++ b/test.cpp
@@ -0,0 +1 @@
+123

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git rm --cached test.cpp
rm 'test.cpp'
```

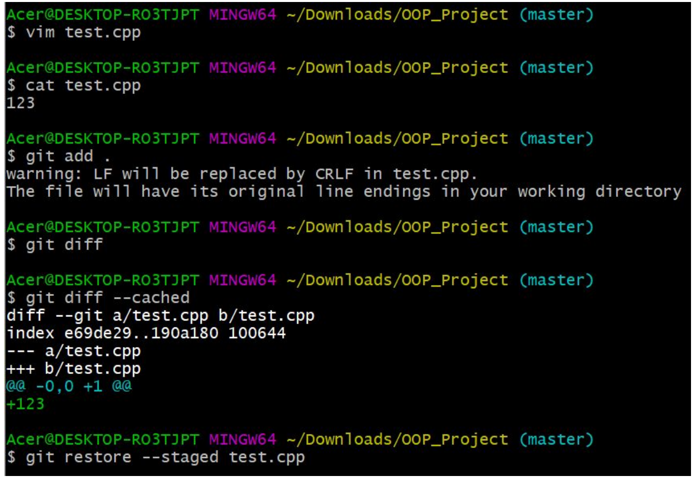


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ cat test.cpp
123

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff --cached
diff --git a/test.cpp b/test.cpp
deleted file mode 100644
index e69de29..0000000

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    test.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.cpp
```

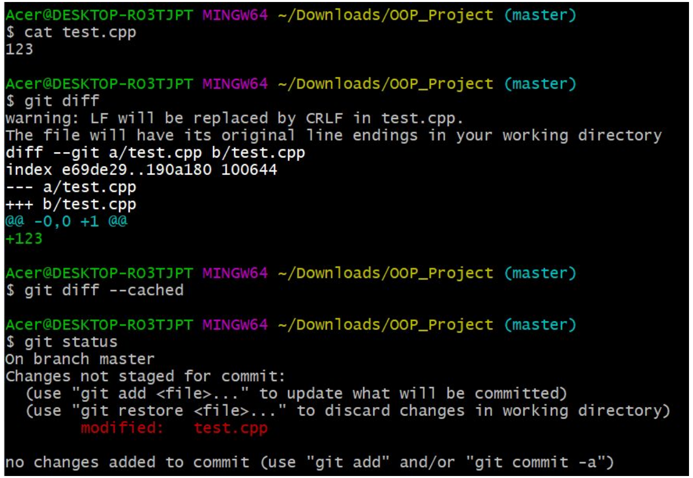
Git - git restore --staged



git restore --staged <files> 將檔案取消追蹤，
工作目錄裡仍留存更改，還原暫存區裡的檔案複本使其維持與上次 commit 時相同(git diff --cached)



```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ vim test.cpp

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ cat test.cpp
123

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git add .
warning: LF will be replaced by CRLF in test.cpp.
The file will have its original line endings in your working directory

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff --cached
diff --git a/test.cpp b/test.cpp
index e69de29..190a180 100644
--- a/test.cpp
+++ b/test.cpp
@@ -0,0 +1 @@
+123

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git restore --staged test.cpp
```

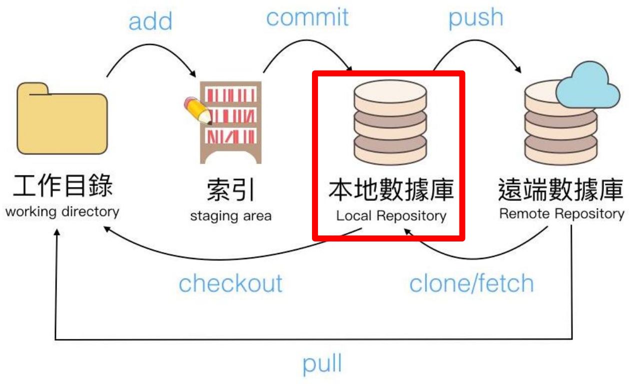


```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ cat test.cpp
123

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff
warning: LF will be replaced by CRLF in test.cpp.
The file will have its original line endings in your working directory
diff --git a/test.cpp b/test.cpp
index e69de29..190a180 100644
--- a/test.cpp
+++ b/test.cpp
@@ -0,0 +1 @@
+123

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git diff --cached

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.cpp

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

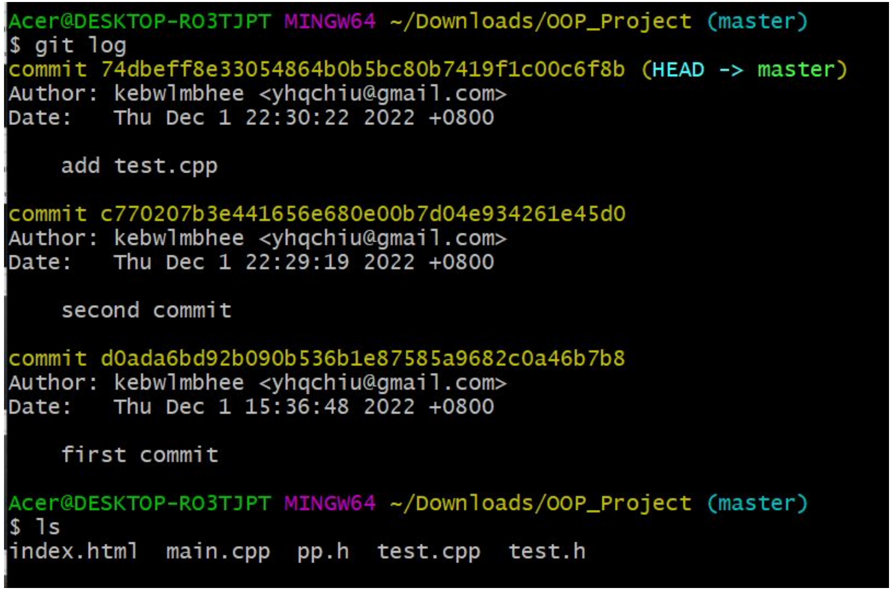
Git - git rm --cached & git restore --staged

git restore --staged 若當前 commit 的是新文件(new file)而非修改文件(modified)，撤銷當前提交即為刪除文件，此時其作用等同於 git rm --cached

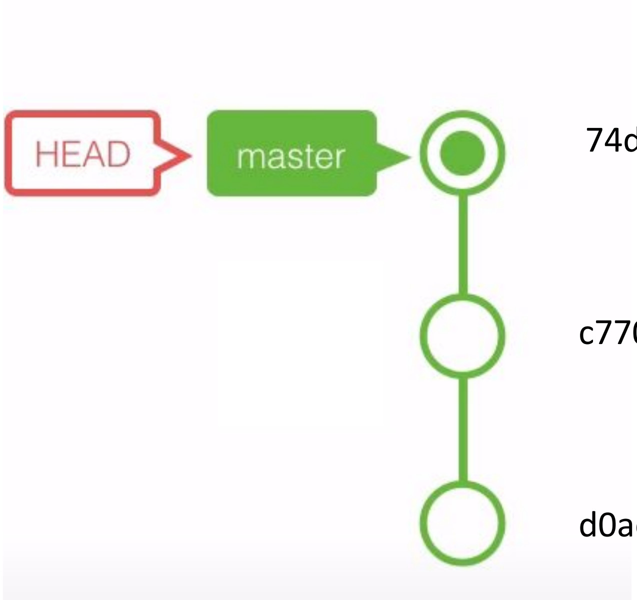
Git - git checkout



git checkout commit/branch => 移動 Head 至某次提交後的工作目錄/分支)



```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git log
commit 74dbeff8e33054864b0b5bc80b7419f1c00c6f8b (HEAD -> master)
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 22:30:22 2022 +0800

    add test.cpp

commit c770207b3e441656e680e00b7d04e934261e45d0
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 22:29:19 2022 +0800

    second commit

commit d0ada6bd92b090b536b1e87585a9682c0a46b7b8
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 15:36:48 2022 +0800

    first commit

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ ls
index.html  main.cpp  pp.h  test.cpp  test.h
```

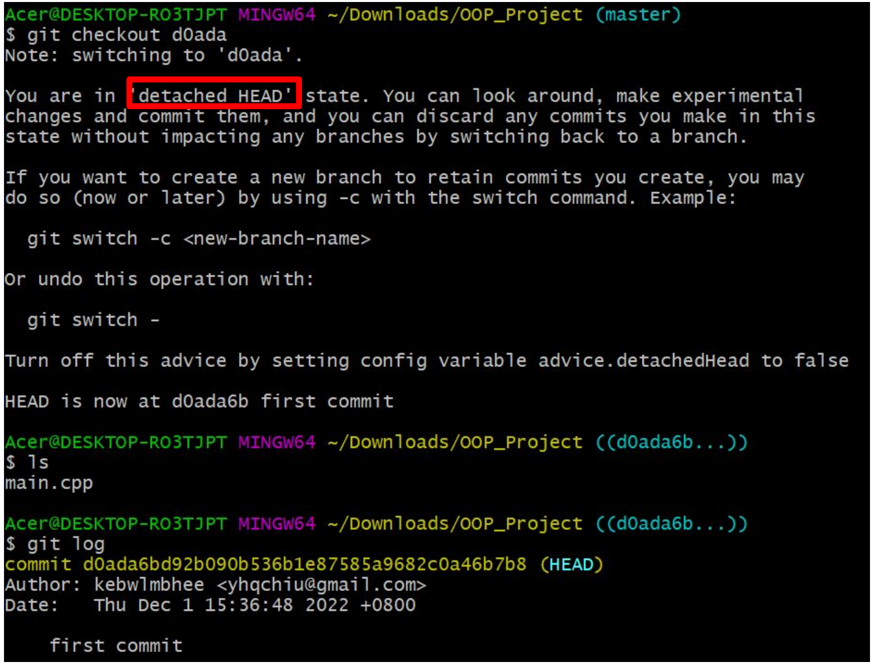


74dbeff

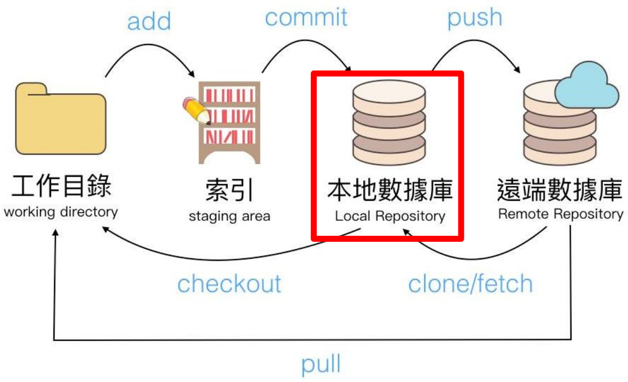
c770207

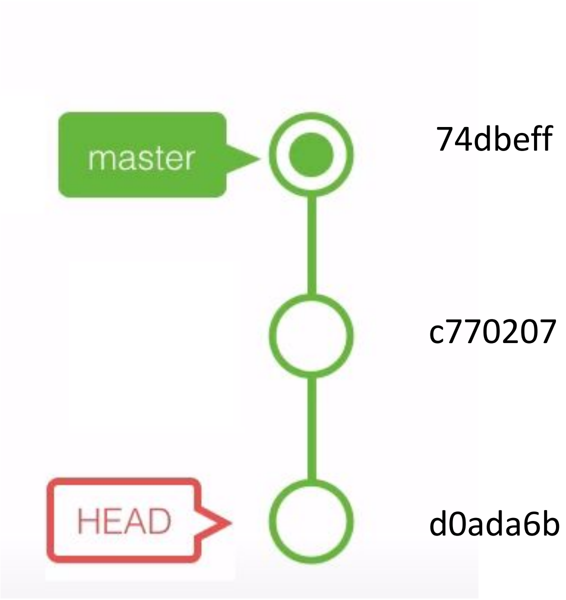
d0ada6b

Git - git checkout



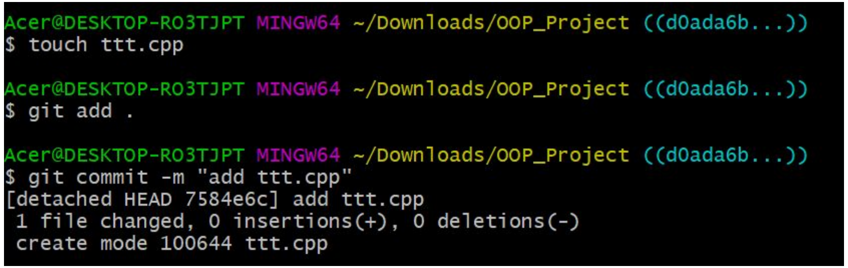
```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git checkout d0ada
Note: switching to 'd0ada'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at d0ada6b first commit

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project ((d0ada6b...))
$ ls
main.cpp

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project ((d0ada6b...))
$ git log
commit d0ada6bd92b090b536b1e87585a9682c0a46b7b8 (HEAD)
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 15:36:48 2022 +0800

    first commit
```



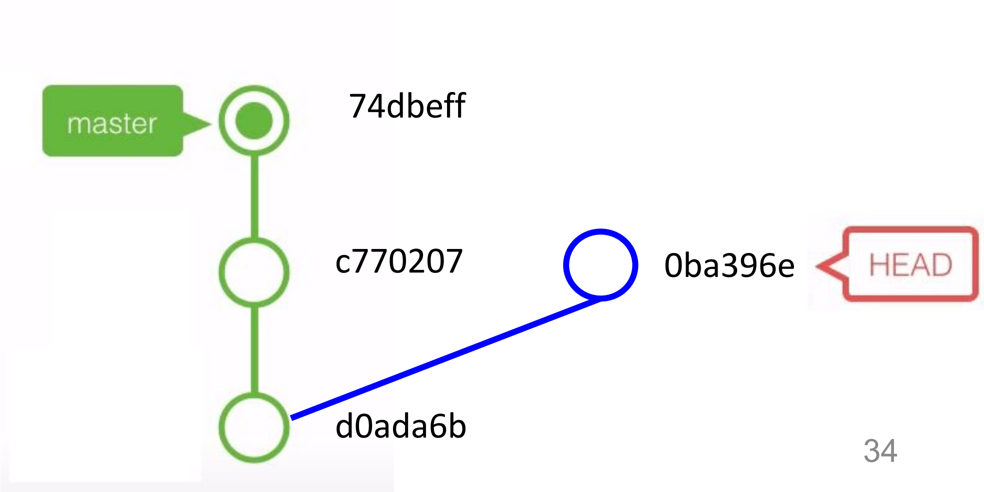


Git - detached HEAD

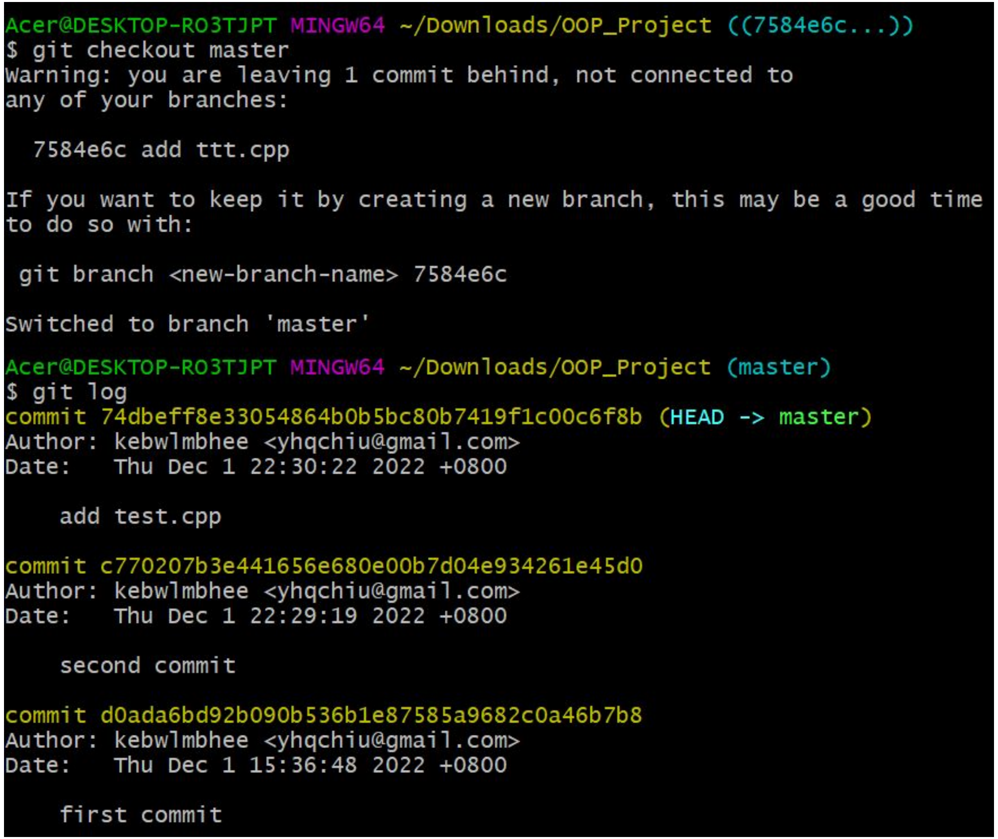
detached HEAD => 當使用 detached HEAD 進行 commit 時，因為沒有分支指向它，除非你記住它的 hash value 或是建立一個新分支，否則從 detached HEAD 開始所進行的 commit，都會遺失在茫茫大海中



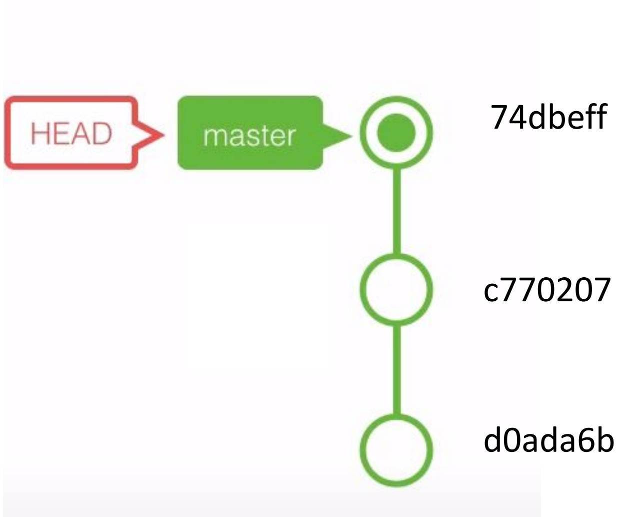
```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project ((d0ada6b...))
$ touch ttt.cpp

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project ((d0ada6b...))
$ git add .

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project ((d0ada6b...))
$ git commit -m "add ttt.cpp"
[detached HEAD 7584e6c] add ttt.cpp
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ttt.cpp
```



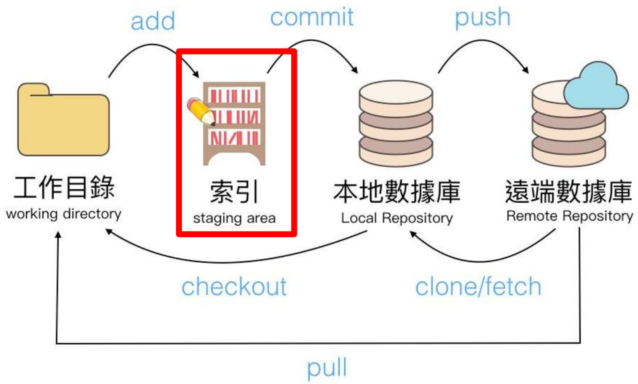
Git - detached HEAD



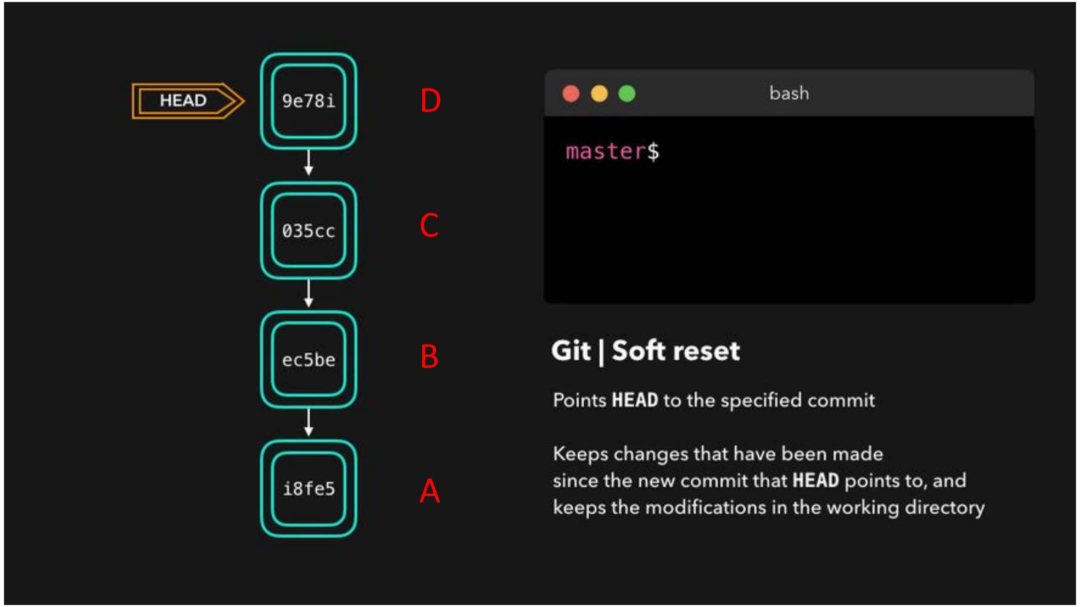
```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project ((7584e6c...))
$ git checkout master
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

  7584e6c add ttt.cpp

If you want to keep it by creating a new branch, this may be a good time
to do so with:

 git branch <new-branch-name> 7584e6c

Switched to branch 'master'

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOP_Project (master)
$ git log
commit 74dbeff8e33054864b0b5bc80b7419f1c00c6f8b (HEAD -> master)
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 22:30:22 2022 +0800

    add test.cpp

commit c770207b3e441656e680e00b7d04e934261e45d0
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 22:29:19 2022 +0800

    second commit

commit d0ada6bd92b090b536b1e87585a9682c0a46b7b8
Author: kebwlmbhee <yhqchiu@gmail.com>
Date:   Thu Dec 1 15:36:48 2022 +0800

    first commit
```



Git - git reset --soft



git reset --soft HEAD~n/commit_id



```
master$
```

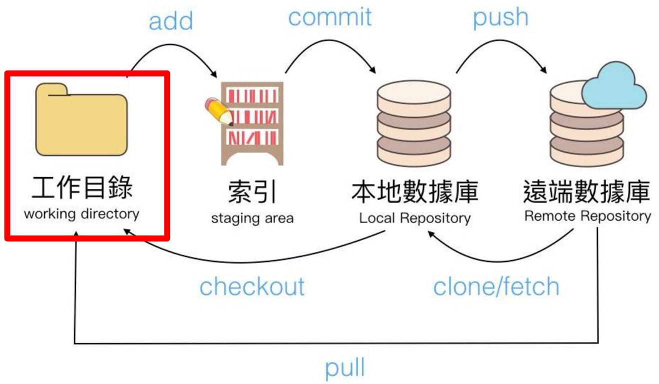
Git | Soft reset

Points **HEAD** to the specified commit

Keeps changes that have been made since the new commit that **HEAD** points to, and keeps the modifications in the working directory

HEAD 與 master 從 D 回到 B，C 和 D 的節點會消失，但 C 和 D 的更改仍將留存於工作目錄和暫存區

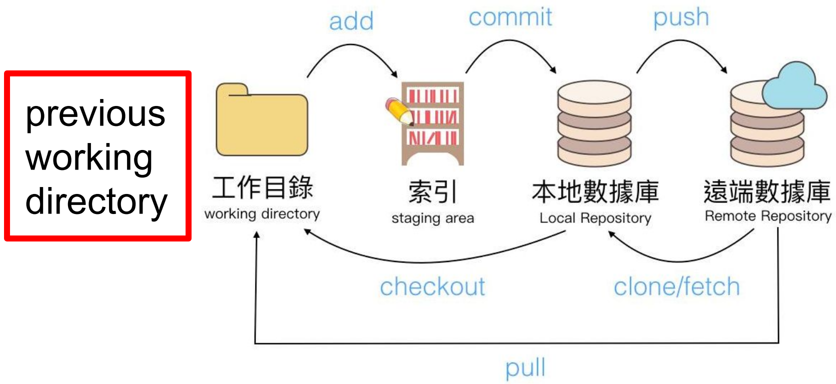
Git - git reset (--mixed)



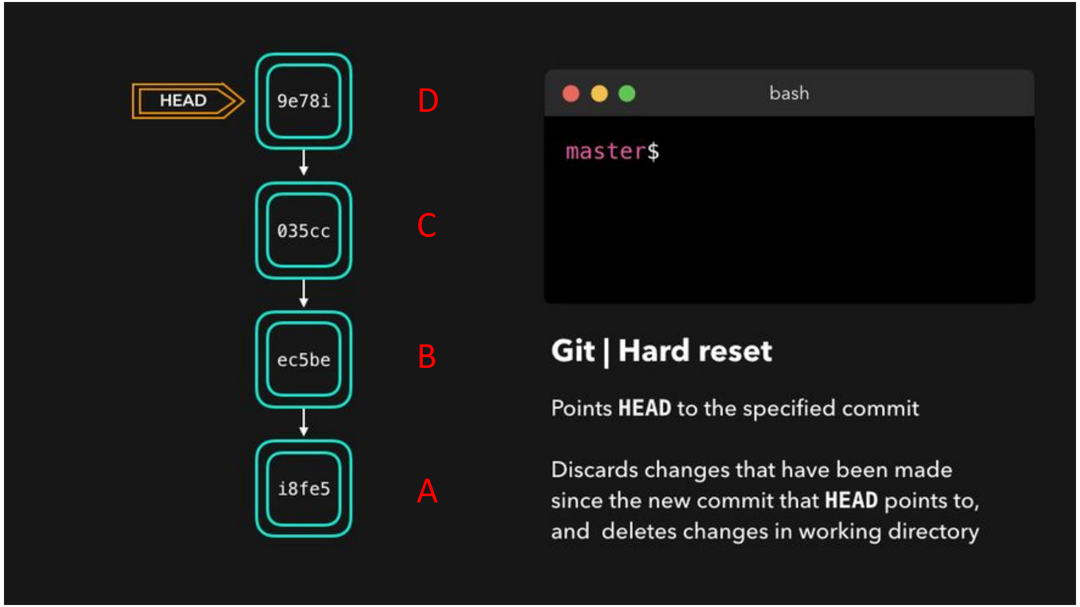
git reset --mixed HEAD~n/commit_id (default)

HEAD 與 master 從 D 回到 B，C 和 D 的節點會消失，但 C 和 D 的更改仍將留存於工作目錄，但暫存區裡不會出現

Git - git reset --hard



git reset --hard HEAD~n/commit_id



```
master$
```

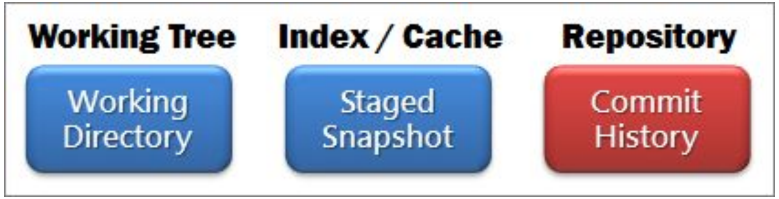
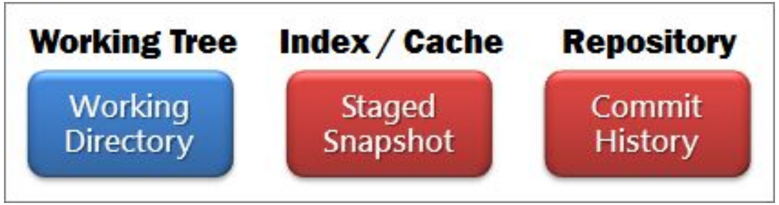
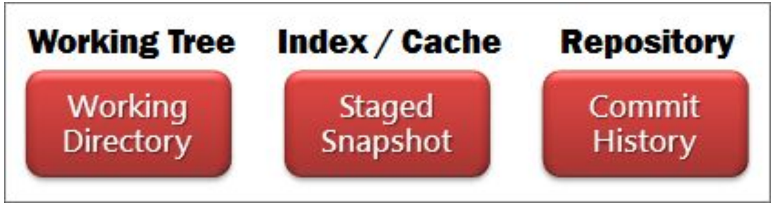
Git | Hard reset

Points **HEAD** to the specified commit

Discards changes that have been made since the new commit that **HEAD** points to, and deletes changes in working directory

HEAD 與 master 從 D 回到 B，C 和 D 的節點會消失，C 和 D 的更改**也將消失於工作目錄**

Git - git reset

	Status	Description
soft	 Working Tree Index / Cache Repository Working Directory Staged Snapshot Commit History	保留 Working Tree 和 Index 資料，只有 Repository 中的檔案內容變更為與 reset 目標節點一致
mixed	 Working Tree Index / Cache Repository Working Directory Staged Snapshot Commit History	只保留 Working Tree 的資料，將 Index 與 Repository 中的檔案內容變更為與 reset 目標節點一致
hard	 Working Tree Index / Cache Repository Working Directory Staged Snapshot Commit History	放棄目前本地的所有變更，將全部內容都重置成與 reset 目標節點的資料一致

Git - git reset vs. git checkout

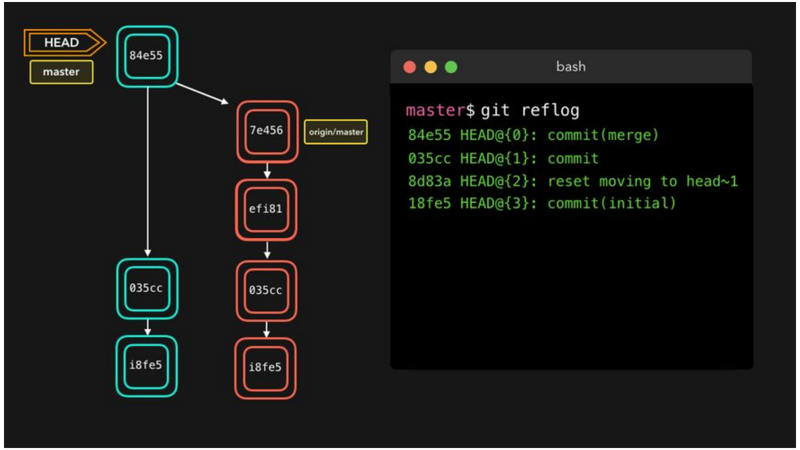
git reset 同時移動 HEAD 和 master，用於還原以前的版本

git checkout 只有移動 HEAD，僅用於查看 commit 記錄和分支節點下的檔案

Git - git reflog

把重要的檔案 reset 掉了？reset 錯地方？

1. 使用 git reflog 查看所有 commit 記錄

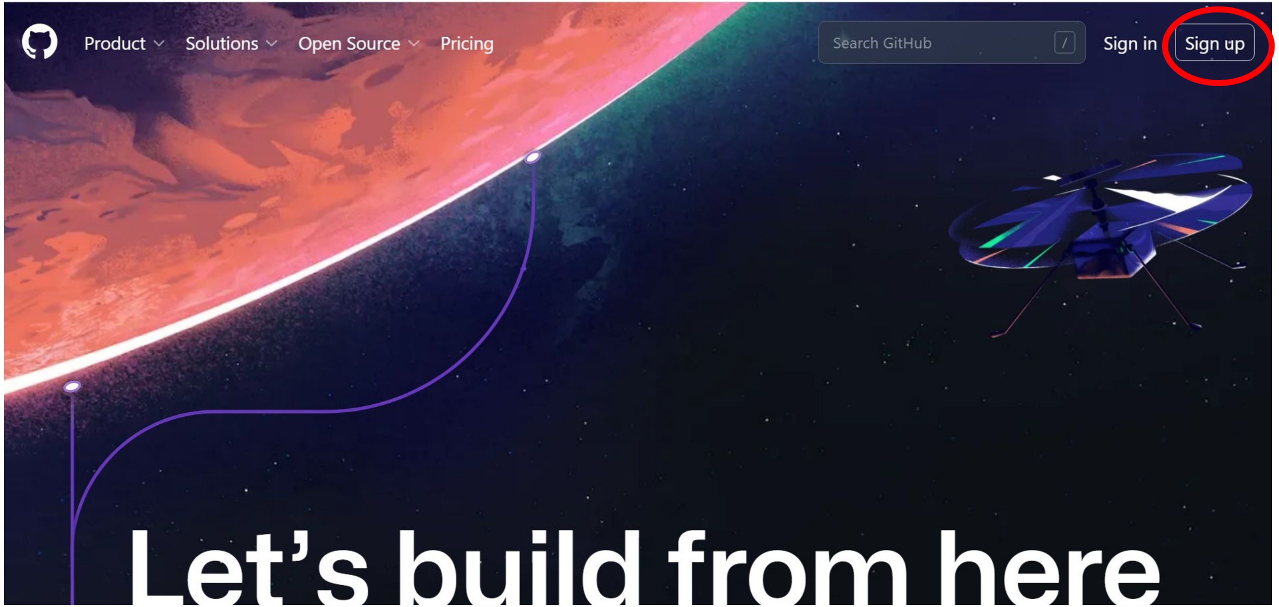


```
master$ git reflog
84e55 HEAD@{0}: commit(merge)
035cc HEAD@{1}: commit
8d83a HEAD@{2}: reset moving to head~1
18fe5 HEAD@{3}: commit(initial)
```

2. 找到想要回復的變更使用 git reset (--hard/--soft) HEAD@{n}將 HEAD 和 master 一起移動到該 commit

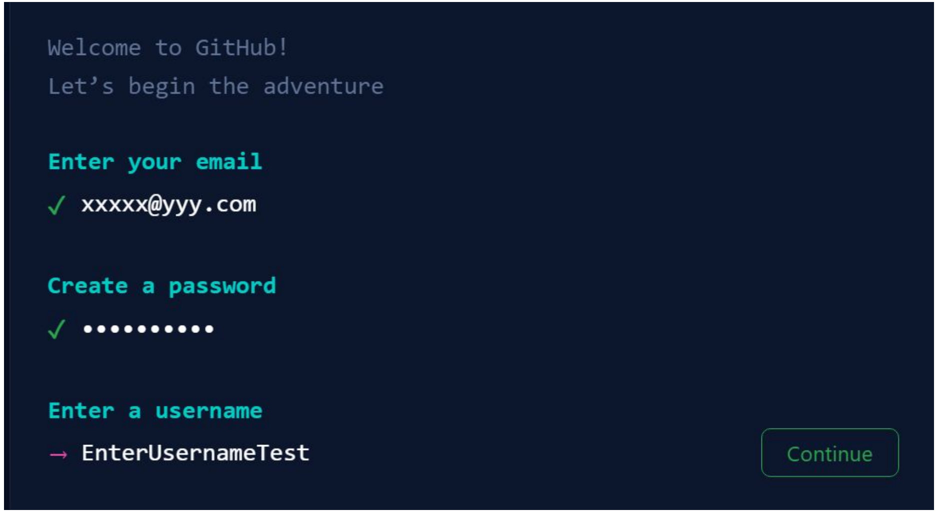
Github

Github - Sign Up

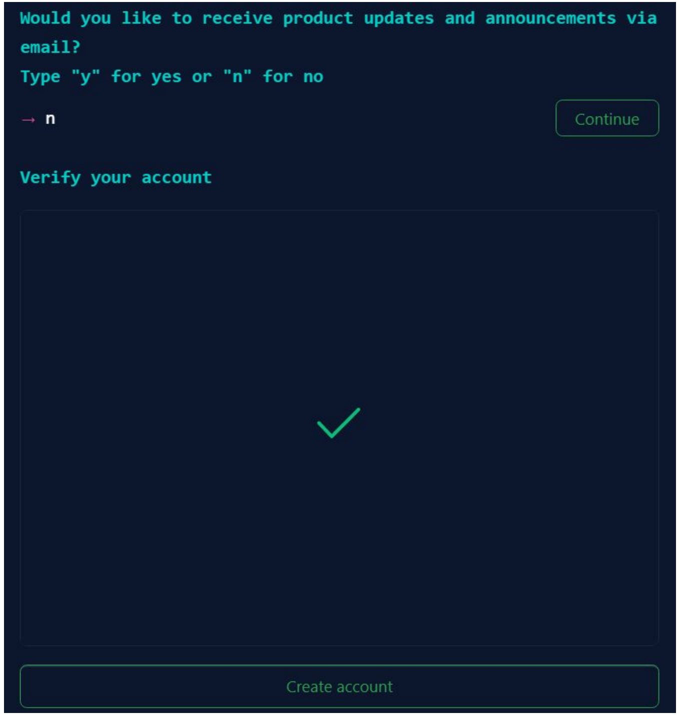


https://github.com/

Github - Sign Up

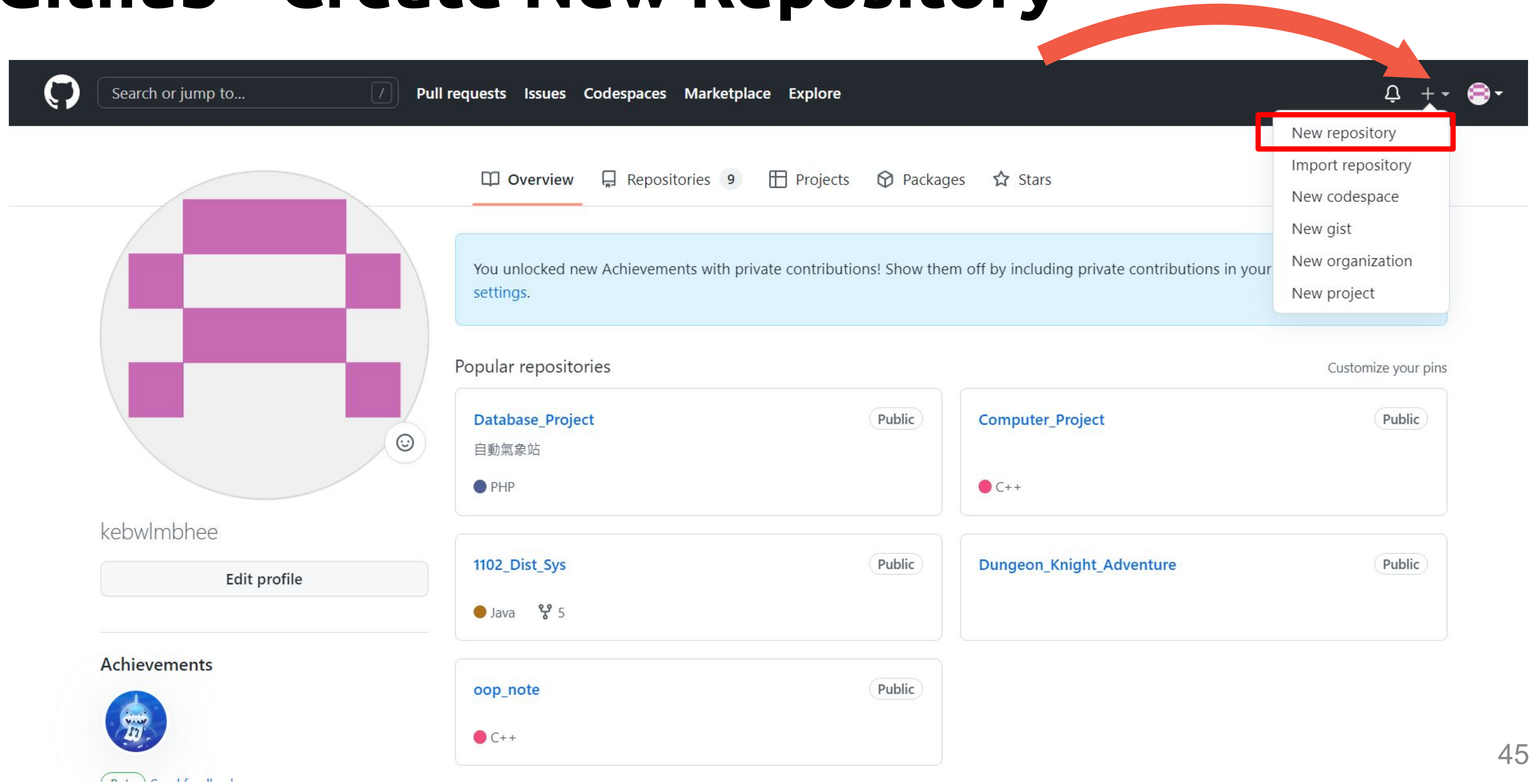


```
Welcome to GitHub!
Let’s begin the adventure

Enter your email
✓ xxxxx@yyy.com

Create a password
✓ ••••••••••

Enter a username
→ EnterUsernameTest
```



```
Would you like to receive product updates and announcements via
email?
Type "y" for yes or "n" for no
→ n

Verify your account
```

Github - Create New Repository



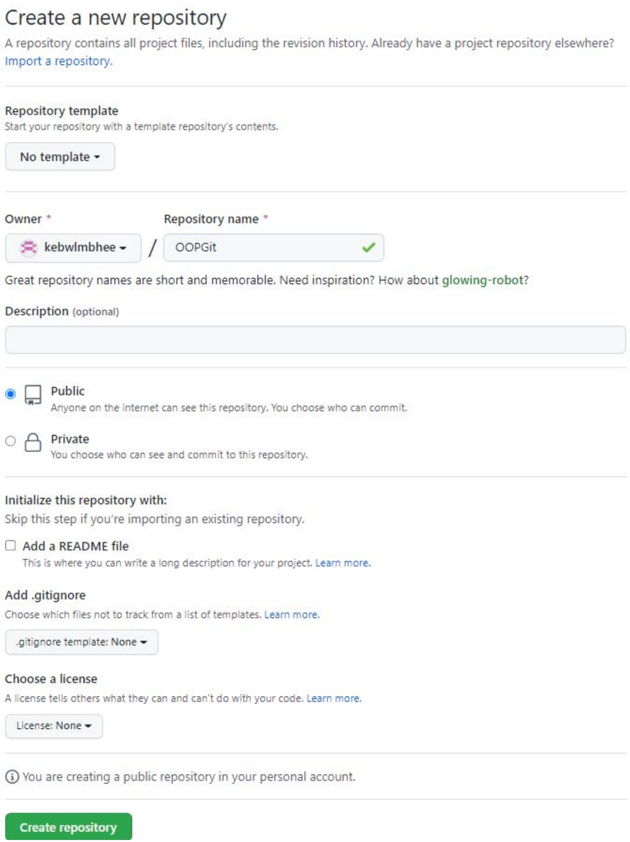
You unlocked new Achievements with private contributions! Show them off by including private contributions in your settings.

Popular repositories

Customize your pins

Database_Project Public 自動氣象站 PHP	Computer_Project Public C++
1102_Dist_Sys Public Java 5	Dungeon_Knight_Adventure Public
oop_note Public C++	

Github - Create New Repository



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner * Repository name *

kebwlmbhee ▾ / OOPGit ✓

Great repository names are short and memorable. Need inspiration? How about glowing-robot?

Description (optional)

● 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.

□ 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: None ▾

Choose a license

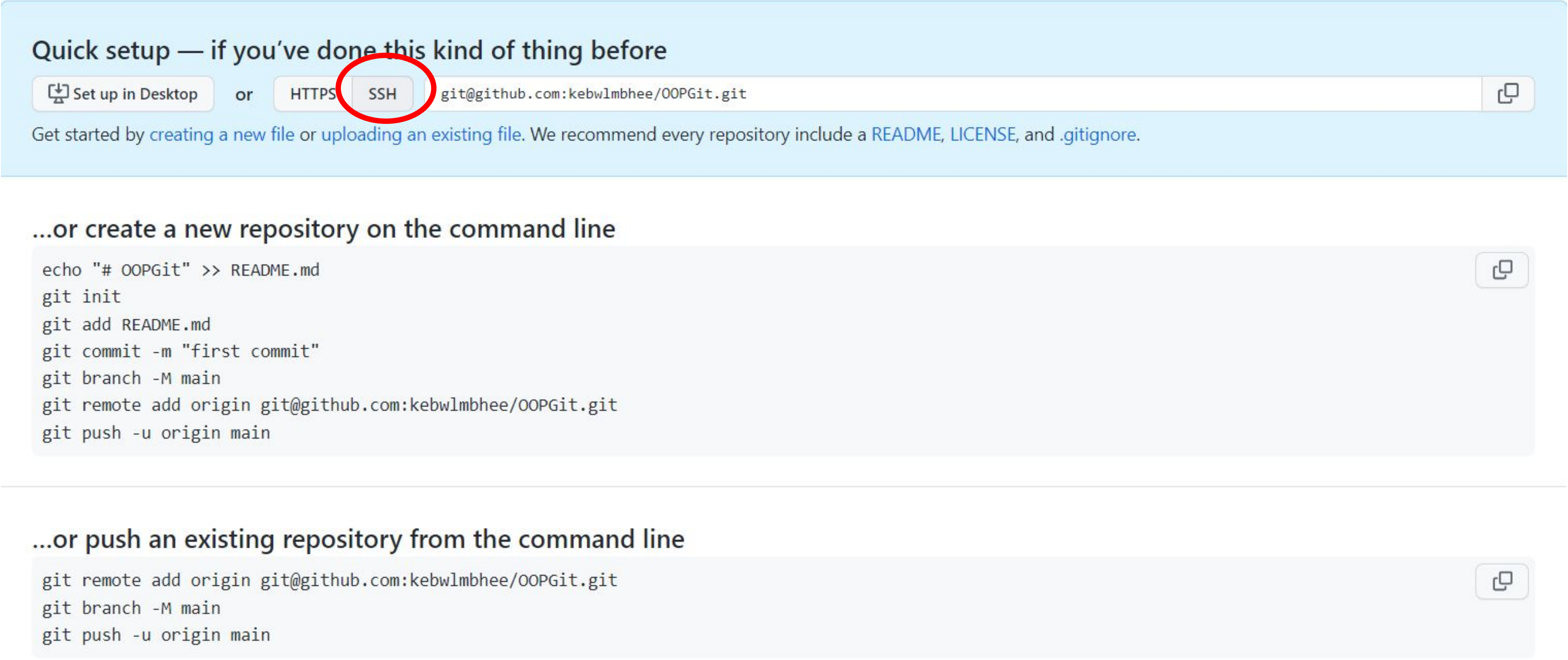
A license tells others what they can and can't do with your code. Learn more.

License: None ▾

ⓘ You are creating a public repository in your personal account.

Create repository

Github - Create New Repository



Quick setup — if you've done this kind of thing before

git@github.com:kebwlmbhee/OOPGit.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# OOPGit" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:kebwlmbhee/OOPGit.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:kebwlmbhee/OOPGit.git
git branch -M main
git push -u origin main
```

Github - Check SSH Key

檢查是否已擁有 SSH key 存放在 ~/.ssh 目錄下

/home/user_name/.ssh 或是 C:\Users\user_name\.ssh

如果已擁用則無需再次生成，可直接跳至 Setting SSH Key

Github - Generate SSH Key

Windows 使用 Git bash, Mac/Linux 使用 Terminal

```
$ ssh-keygen -C "Your_Email"              // Email 辨別創建者，也可不加
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/user_name/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

passphrase 可讓你的 private key 更安全，但每次存取時需輸入 passphrase，或使用 ssh-agent

Github - Generate SSH Key

```
Your identification has been saved in
/c/Users/jamal/.ssh/id_rsa.
Your public key has been saved in
/c/Users/jamal/.ssh/id_rsa.pub.                // 記住此 路徑
The key fingerprint is:
SHA256:****************************************
jamal@fabrikam.com
The key's randomart image is:
...
```

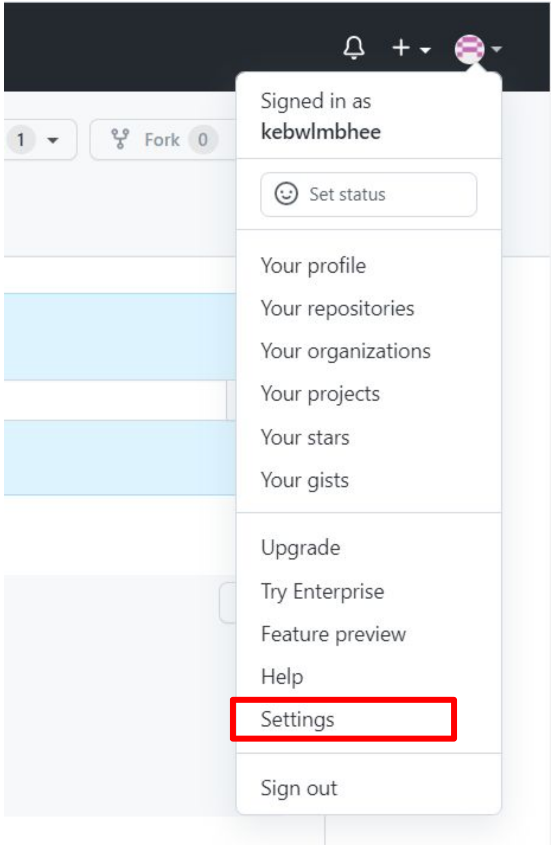
Github - Generate SSH Key

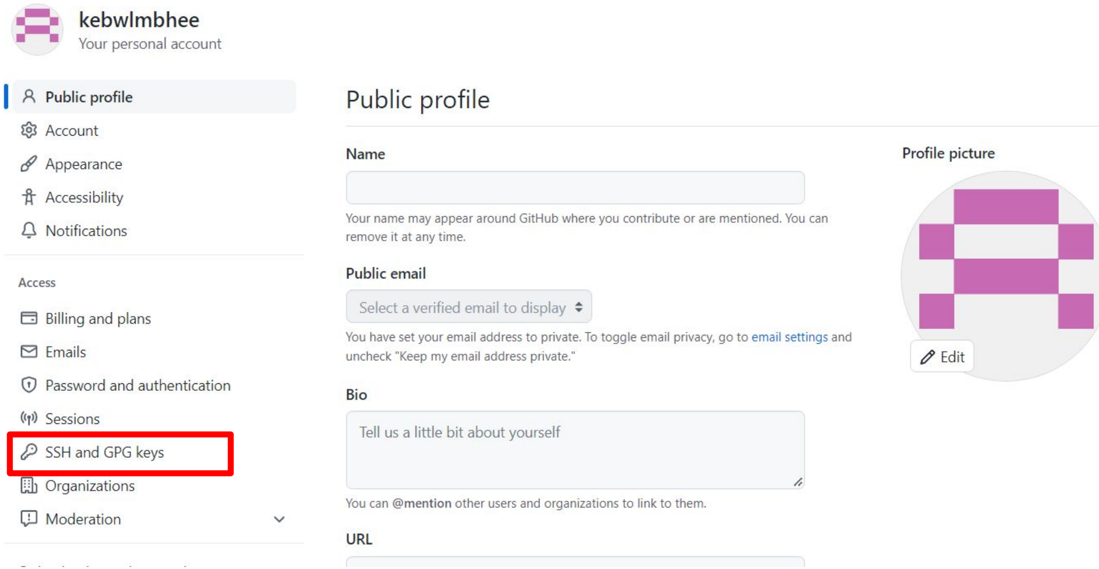
進入 路徑 下找到 id_rsa.pub

```
$ cat id_rsa.pub
```

把印出來的公鑰全選後複製

Github - Setting SSH Key





Public profile

Name

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email

Select a verified email to display

You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."

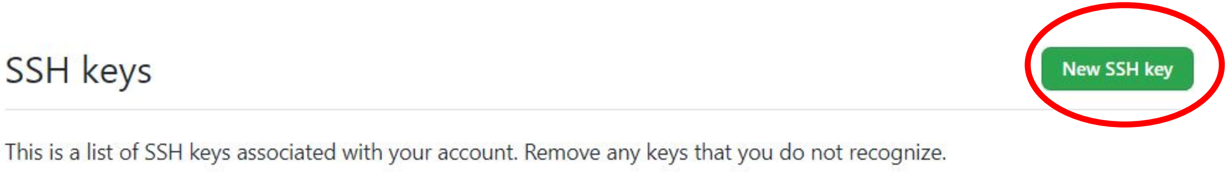
Bio

Tell us a little bit about yourself

You can **@mention** other users and organizations to link to them.

URL

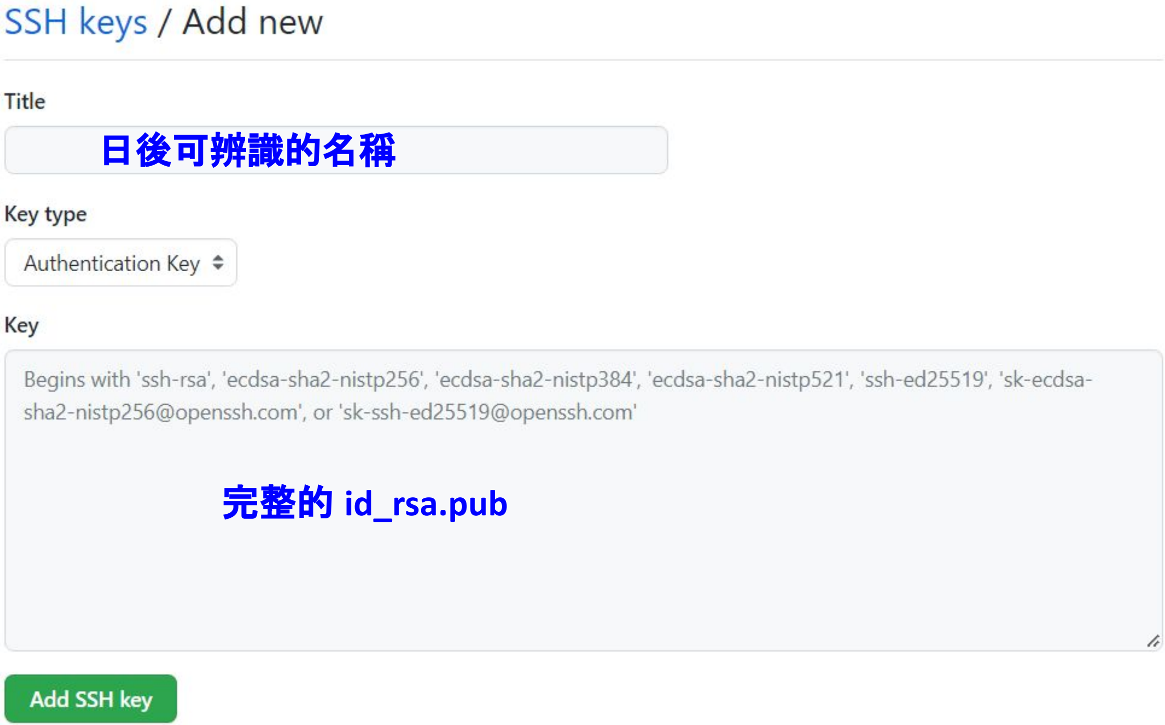
Github - Setting SSH Key



SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Github - Setting SSH Key



SSH keys / Add new

Title

日後可辨識的名稱

Key type

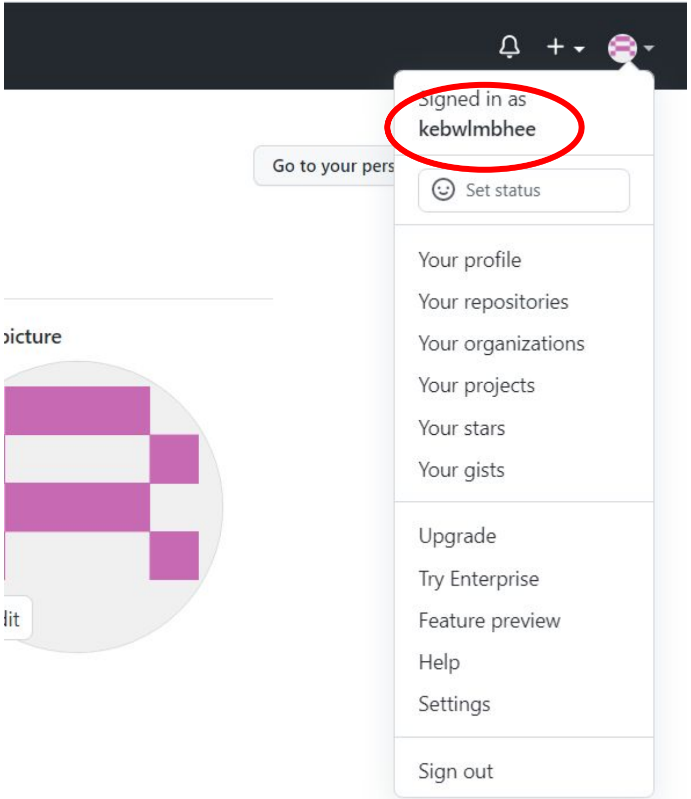
Authentication Key

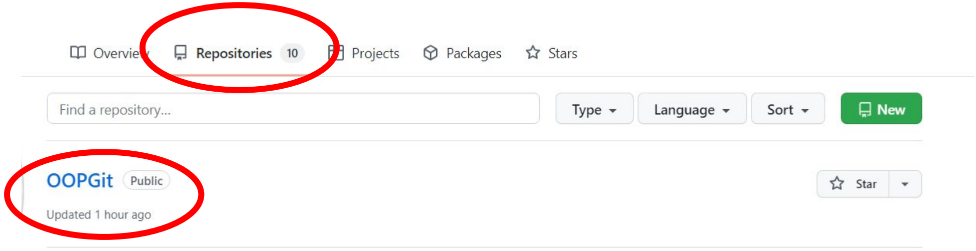
Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

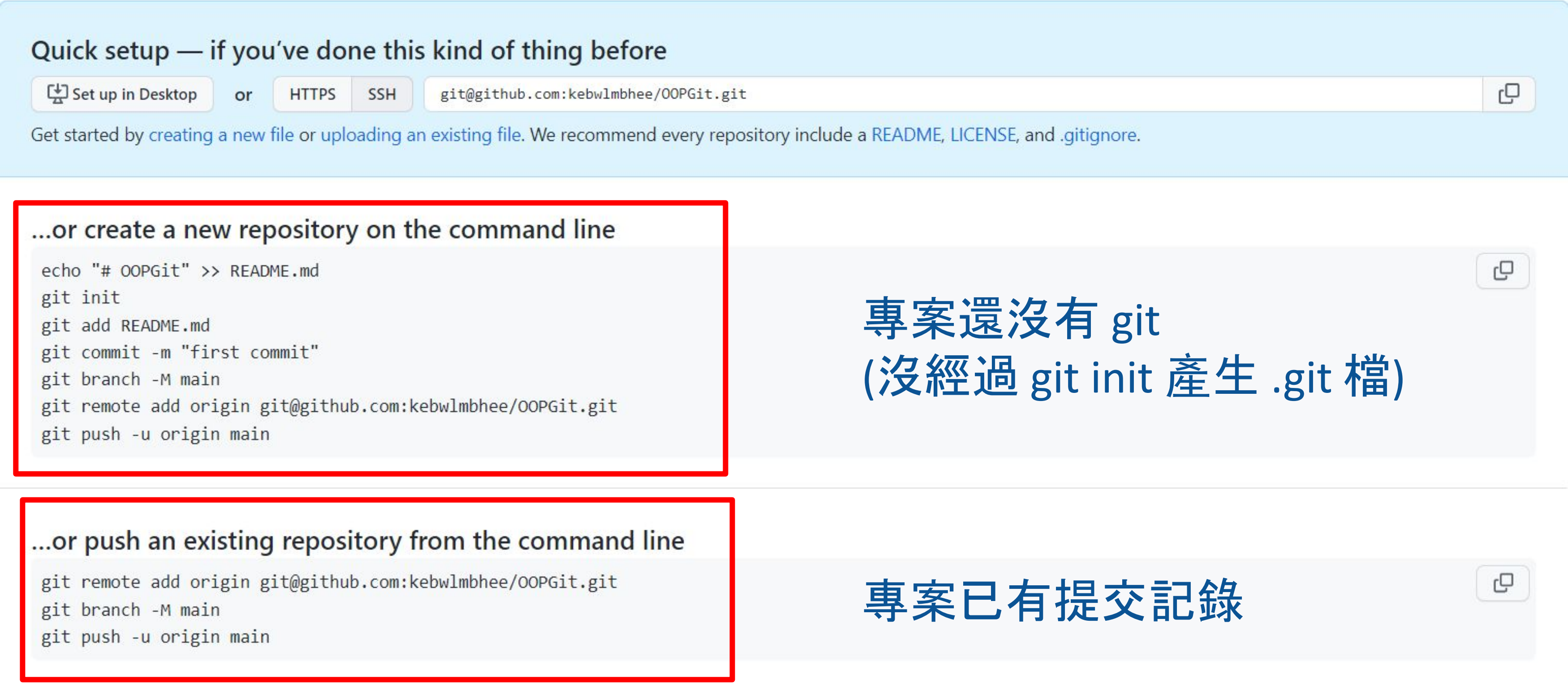
完整的 id_rsa.pub

Github - 回到 Repository





Github - Setup Repository



Quick setup — if you've done this kind of thing before

git@github.com:kebwlmbhee/OOPGit.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# OOPGit" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:kebwlmbhee/OOPGit.git
git push -u origin main
```

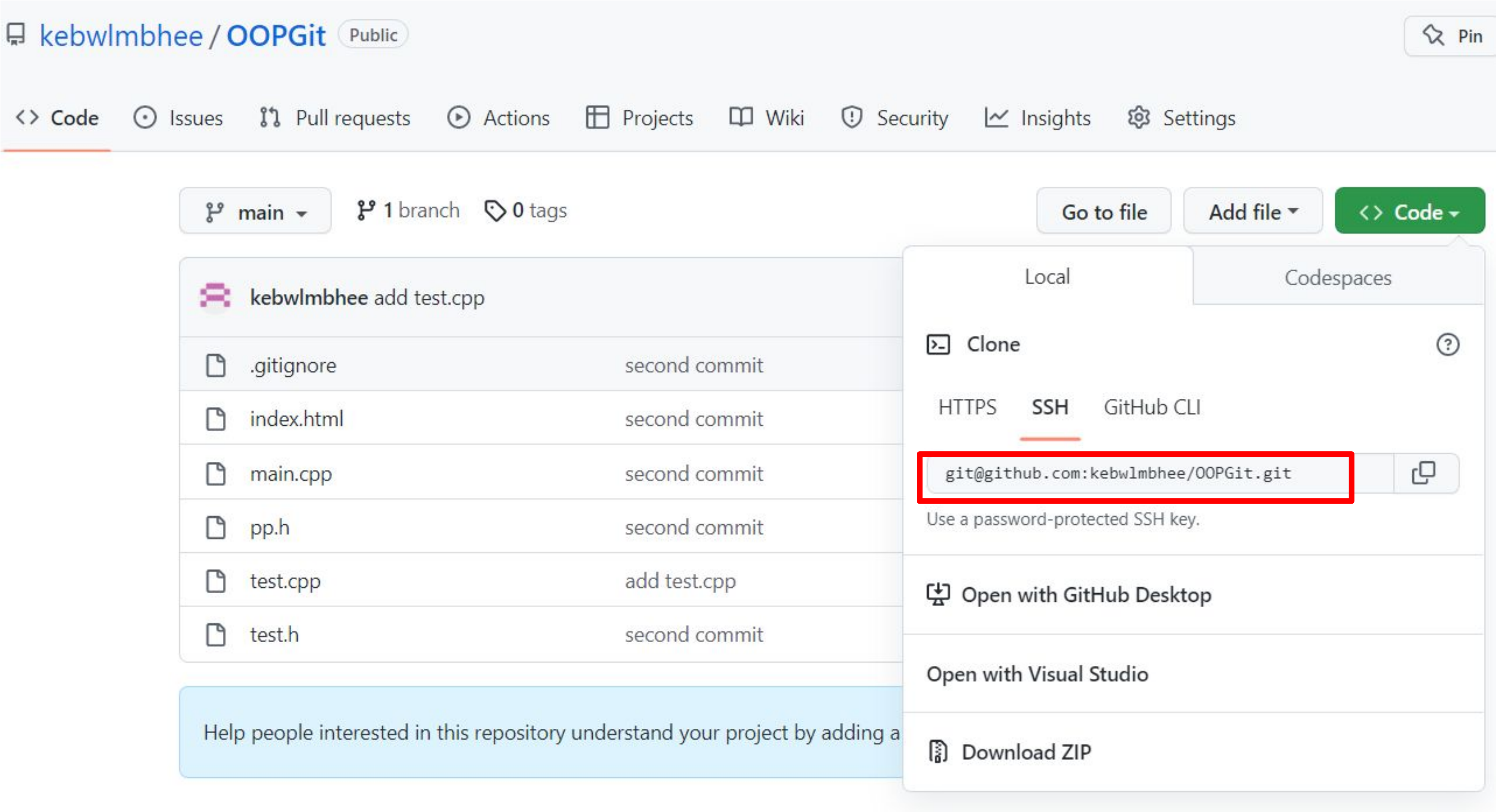
專案還沒有 git
(沒經過 git init 產生 .git 檔)

...or push an existing repository from the command line

```
git remote add origin git@github.com:kebwlmbhee/OOPGit.git
git branch -M main
git push -u origin main
```

專案已有提交記錄

Github - git clone



kebwlmbhee / OOPGit Public

Pin

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

kebwlmbhee add test.cpp

.gitignore	second commit
index.html	second commit
main.cpp	second commit
pp.h	second commit
test.cpp	add test.cpp
test.h	second commit

Help people interested in this repository understand your project by adding a

Local Codespaces

Clone

HTTPS SSH GitHub CLI

```
git@github.com:kebwlmbhee/OOPGit.git
```

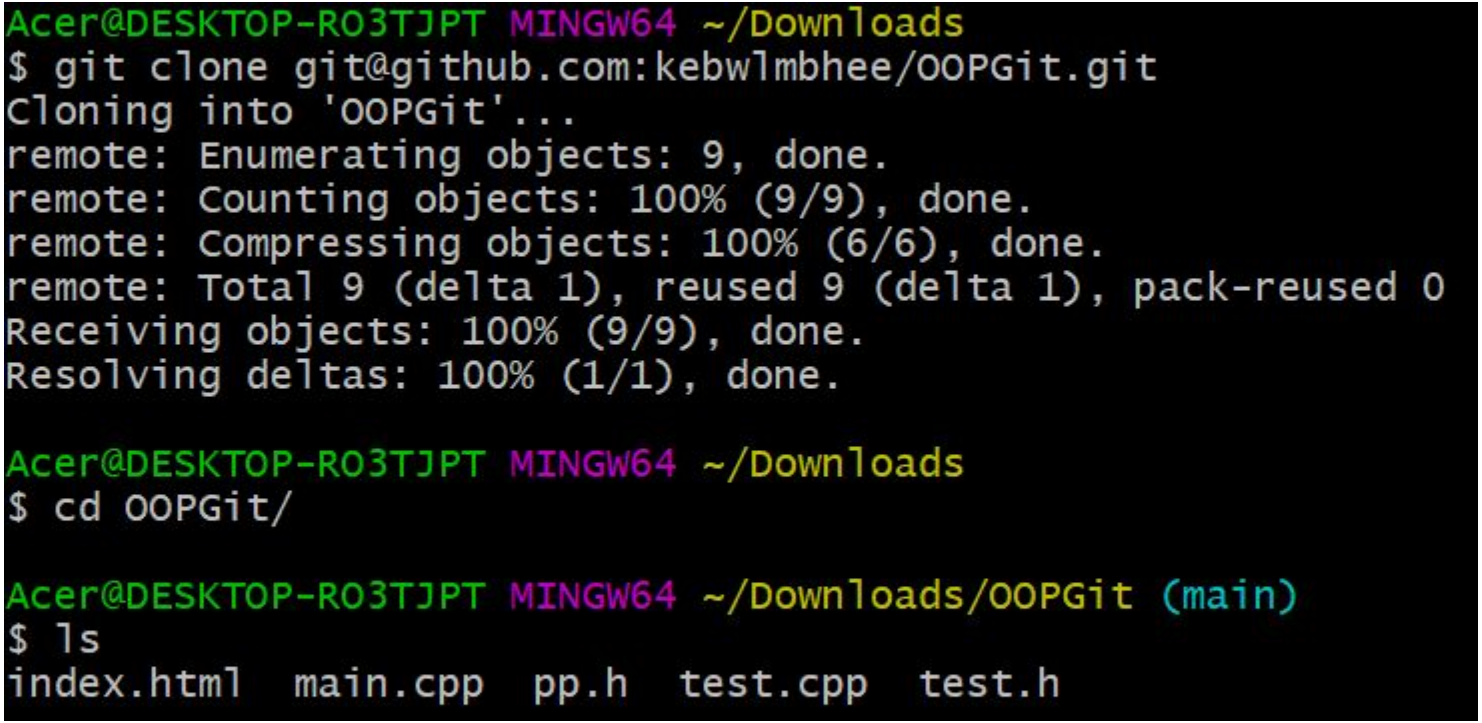
Use a password-protected SSH key.

Open with GitHub Desktop

Open with Visual Studio

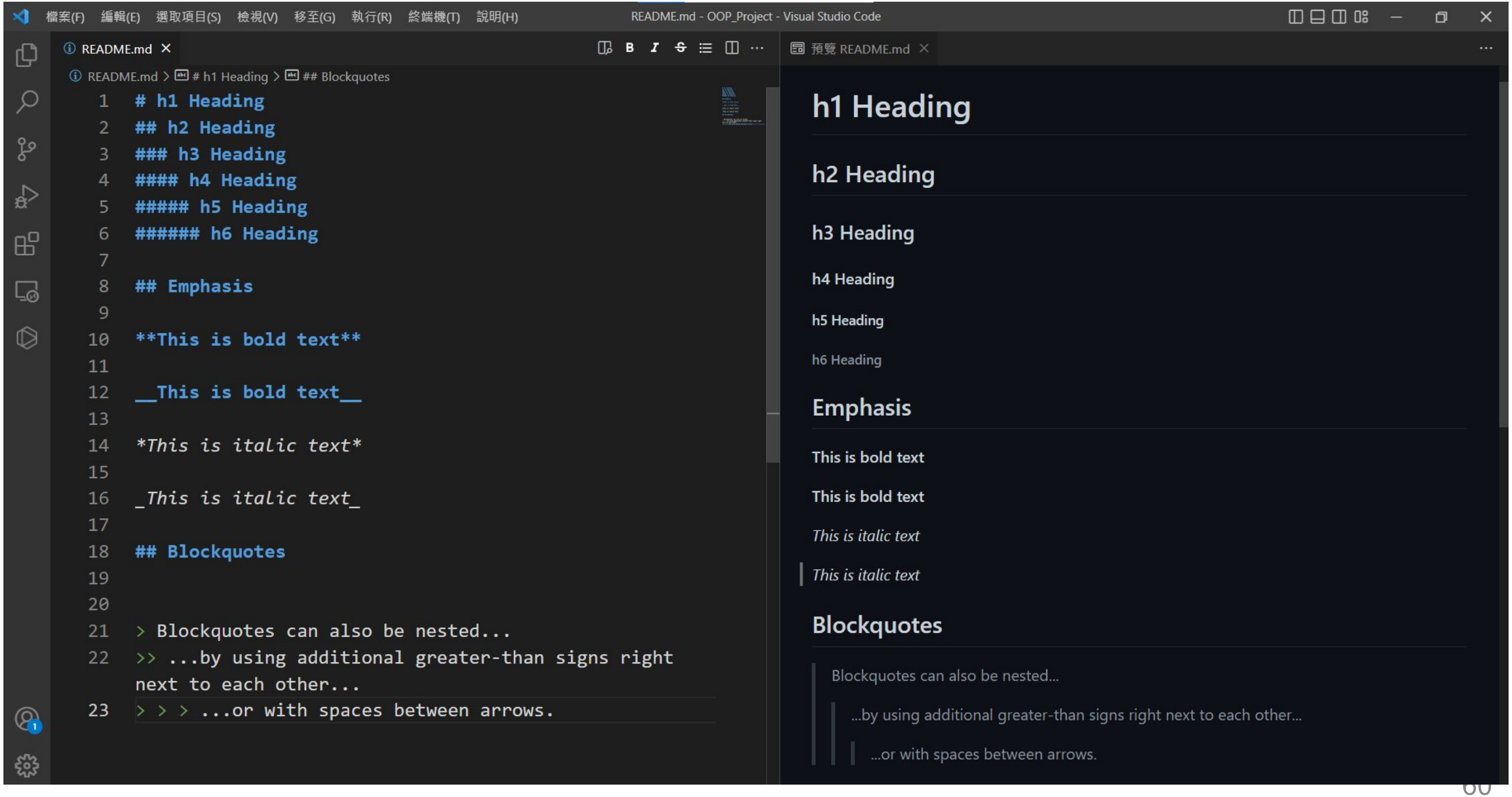
Download ZIP

Github - git clone



```
Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads
$ git clone git@github.com:kebwlmbhee/OOPGit.git
Cloning into 'OOPGit'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 1), reused 9 (delta 1), pack-reused 0
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (1/1), done.

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads
$ cd OOPGit/

Acer@DESKTOP-RO3TJPT MINGW64 ~/Downloads/OOPGit (main)
$ ls
index.html  main.cpp  pp.h  test.cpp  test.h
```

Markdown


檔案(F)
編輯(E)
選取項目(S)
檢視(V)
移至(G)
執行(R)
終端機(T)
說明(H)
README.md - OOP_Project - Visual Studio Code
README.md
README.md > # h1 Heading > ## Blockquotes
預覽 README.md

```
# h1 Heading
## h2 Heading
### h3 Heading
#### h4 Heading
##### h5 Heading
###### h6 Heading

## Emphasis

**This is bold text**

__This is bold text__

*This is italic text*

_This is italic text_

## Blockquotes


> Blockquotes can also be nested...
>> ...by using additional greater-than signs right next to each other...
> > > ...or with spaces between arrows.
```

h1 Heading

h2 Heading

h3 Heading

h4 Heading

h5 Heading

h6 Heading

Emphasis

This is bold text

This is bold text

This is italic text

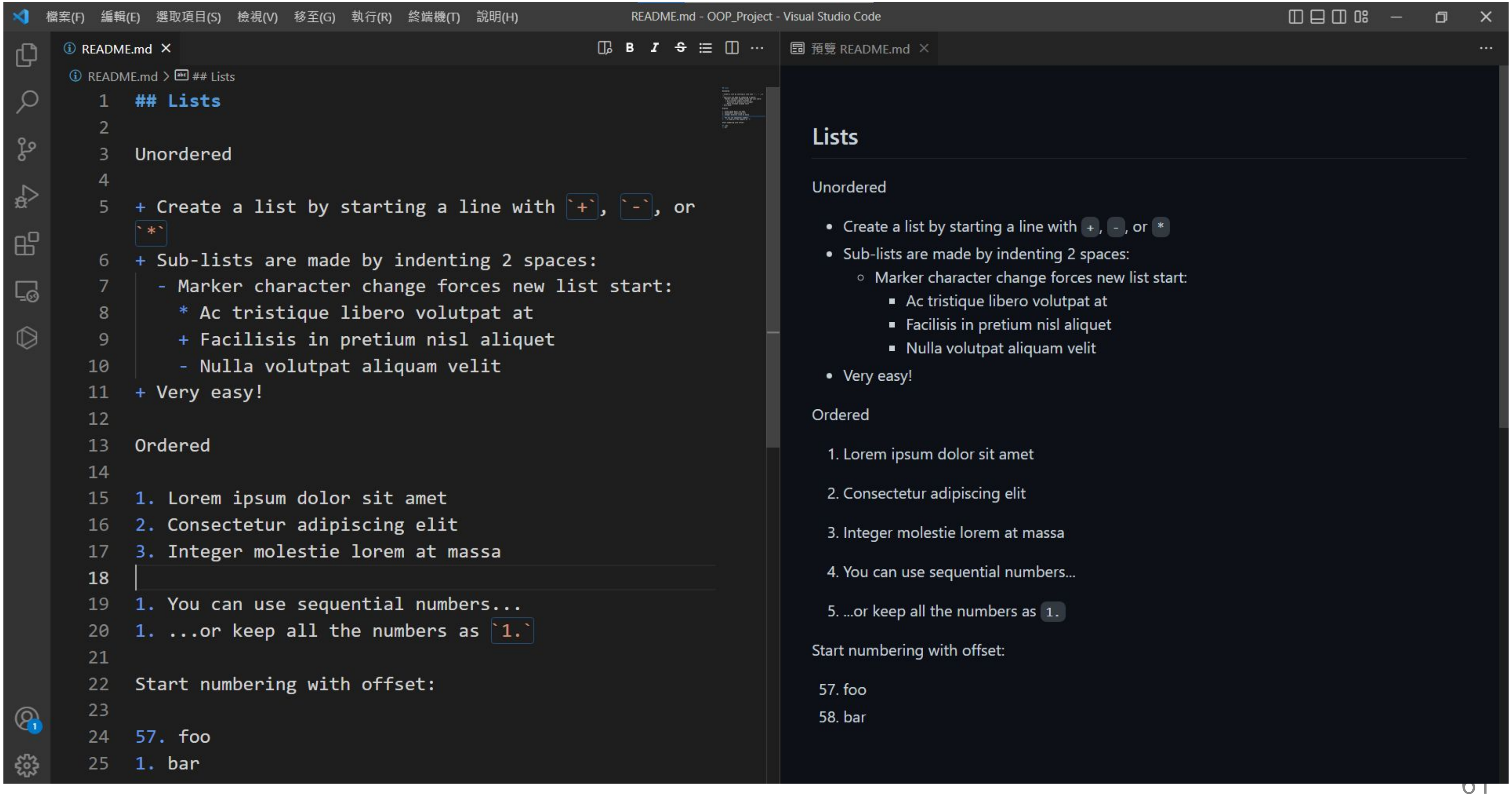
This is italic text

Blockquotes

Blockquotes can also be nested...

...by using additional greater-than signs right next to each other...

...or with spaces between arrows.



```
## Lists

Unordered

+ Create a list by starting a line with `+`, `-`, or
`*`
+ Sub-lists are made by indenting 2 spaces:
  - Marker character change forces new list start:
    * Ac tristique libero volutpat at
    + Facilisis in pretium nisl aliquet
    - Nulla volutpat aliquam velit
+ Very easy!

Ordered

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa

1. You can use sequential numbers...
1. ...or keep all the numbers as `1.`

Start numbering with offset:

57. foo
1. bar
```

Lists

Unordered

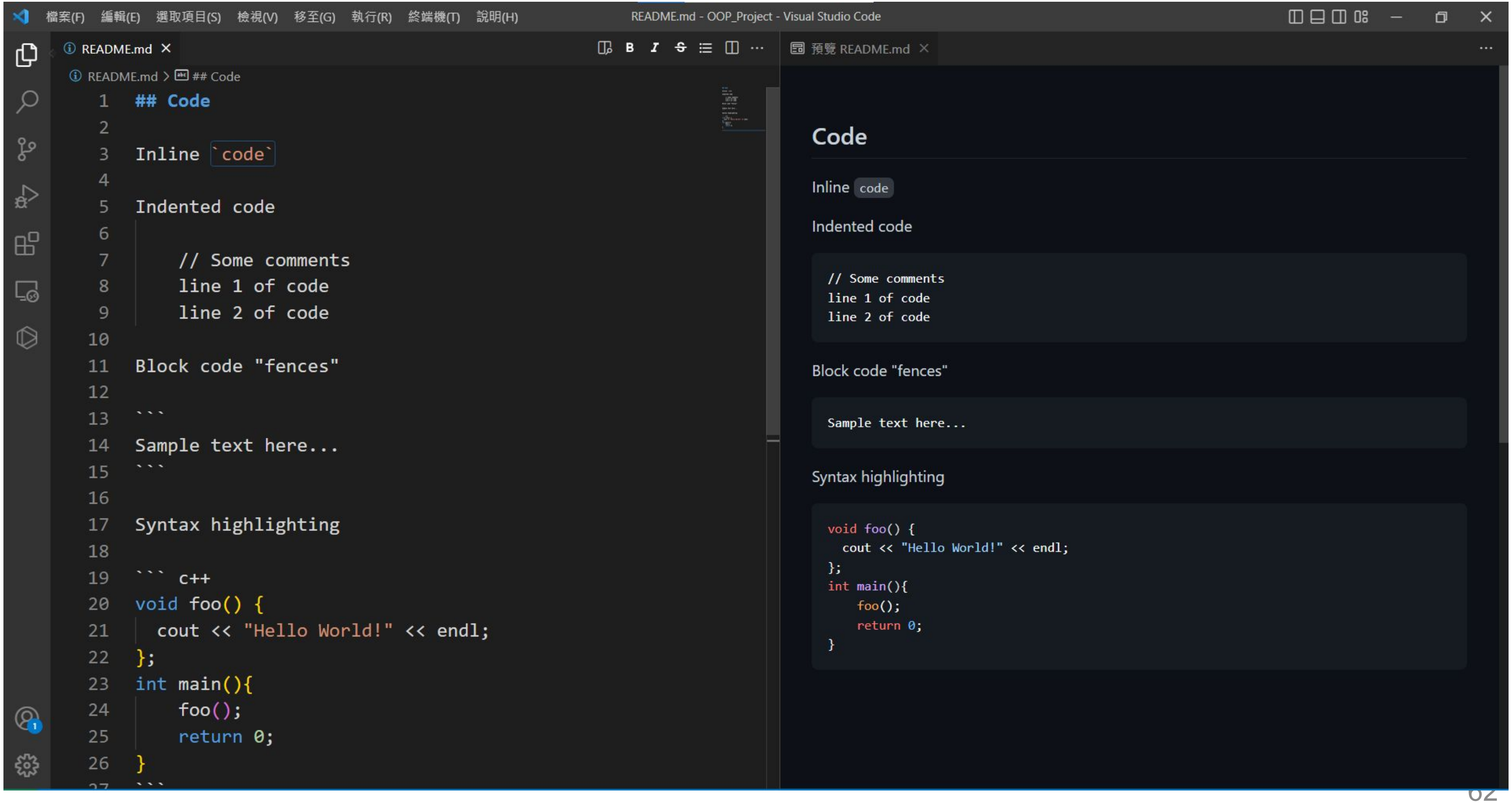
- Create a list by starting a line with `+`, `-`, or `*`
- Sub-lists are made by indenting 2 spaces:
  - Marker character change forces new list start:
    - Ac tristique libero volutpat at
    - Facilisis in pretium nisl aliquet
    - Nulla volutpat aliquam velit
- Very easy!

Ordered

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa
4. You can use sequential numbers...
5. ...or keep all the numbers as `1.`

Start numbering with offset:

57. foo
58. bar



~~~~
## Code

Inline `code`

Indented code

    // Some comments
    line 1 of code
    line 2 of code

Block code "fences"

```
Sample text here...
```

Syntax highlighting

``` c++
void foo() {
  cout << "Hello World!" << endl;
};
int main(){
    foo();
    return 0;
}
```
~~~~

Code

Inline `code`

Indented code

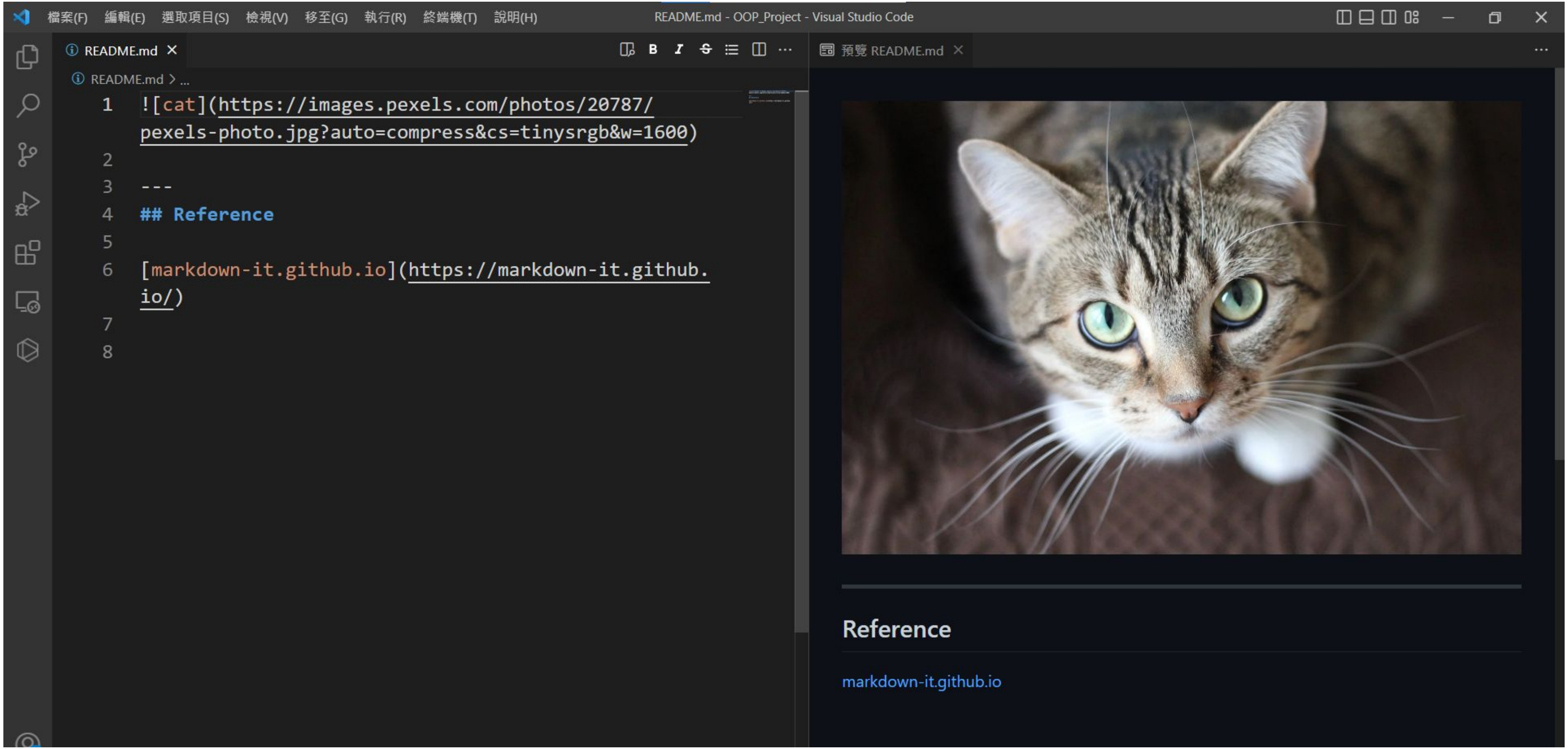
```
// Some comments
line 1 of code
line 2 of code
```

Block code "fences"

```
Sample text here...
```

Syntax highlighting

```
void foo() {
  cout << "Hello World!" << endl;
};
int main(){
    foo();
    return 0;
}
```



```
![cat](https://images.pexels.com/photos/20787/pexels-photo.jpg?auto=compress&cs=tinysrgb&w=1600)

---
## Reference

[markdown-it.github.io](https://markdown-it.github.io/)


```

Reference

markdown-it.github.io

接下來的課程安排

https://docs.google.com/forms/d/e/1FAIpQLSdJN2u44xMNcDRIjgR50aq8pUNRwxGfR52VmPhgDNp0qEsqWQ/viewform

Reference

Git - Book：https://git-scm.com/book/zh-tw/v2

Git & Github tutorial：https://www.youtube.com/playlist?list=PLYrA-SsMvTPOZeB6DHvB0ewl3miMf-2tj

Markdown：https://markdown-it.github.io/

Any questions?