

Git的使用



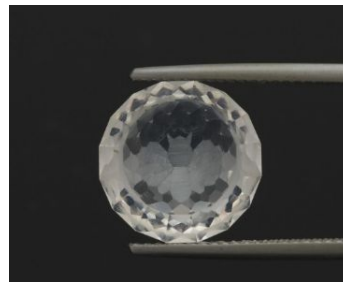
吳如峰

主要經歷:

貓頭鷹藝術科技總工程師

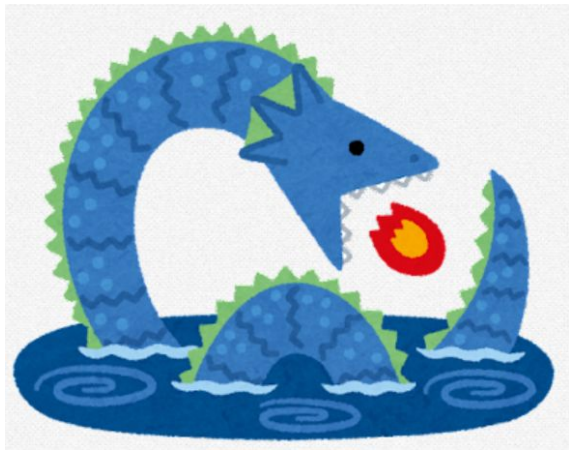
廣達集團高級工程師

成功大學電機系控制組碩士



git是什麼，為什麼要學？

打遊戲的時候



先記錄一下

git是什麼, 為什麼要學?

If a :

#準備寫程式

else:

#準備寫程式

**有沒有辦法做一個紀錄
免得程式寫壞?**

git是什麼，為什麼要學？

1. 很醜
2. 版本差異不明
3. 如果內容很多，很麻煩



my_python_final.py

類型: Python File



my_python_final_1.py

類型: Python File



my_python_real_final.py

類型: Python File



my_python_real_final_1.py

類型: Python File



my_python_real_real_final.py

類型: Python File



my_python_v1.py

類型: Python File



my_python_v2.1.py

類型: Python File



my_python_v2.py

類型: Python File

git是什麼，為什麼要學？

很多開高薪水的公司需要

git

×

地區

▼

職務類別

☐ 只搜尋職務名稱

相關搜尋：API、JavaScript、Java、Python、Linux

職務

公司

更新日期 ▼

出勤制度 ▼

月薪4萬up ▼

經歷要求 ▼

公司相關 ▼

更多條件 ▼

排除條件 ▼

全部(1063)

全職(1060)

兼職(0)

高階(0)

其他 ▼

第 1 / 54 頁 ▼

符合度排序 ▼

git是甚麼，為甚麼要學？

1000多個薪水4萬以上職缺是甚麼概念

A screenshot of a job search website. The search bar at the top contains the word "python" and a "地區" (Region) dropdown. Below the search bar, there is a checkbox for "只搜尋職務名稱" and a list of related search terms: "軟體設計工程師", "C++", "Linux", "Java", and "資訊工程". The "職務" (Job Title) tab is selected. Below the tabs, there are several filter dropdowns: "更新日期", "出勤制度", "月薪4萬up", "經歷要求", "公司相關", and "更多條件". At the bottom, there are filters for "全部(897)", "全職(881)", "兼職(0)", "高階(0)", and "其他". A page indicator shows "第 1 / 45 頁".

python × | 地區 ▾

☐ 只搜尋職務名稱 相關搜尋：軟體設計工程師、C++、Linux、Java、資訊工程

職務 公司

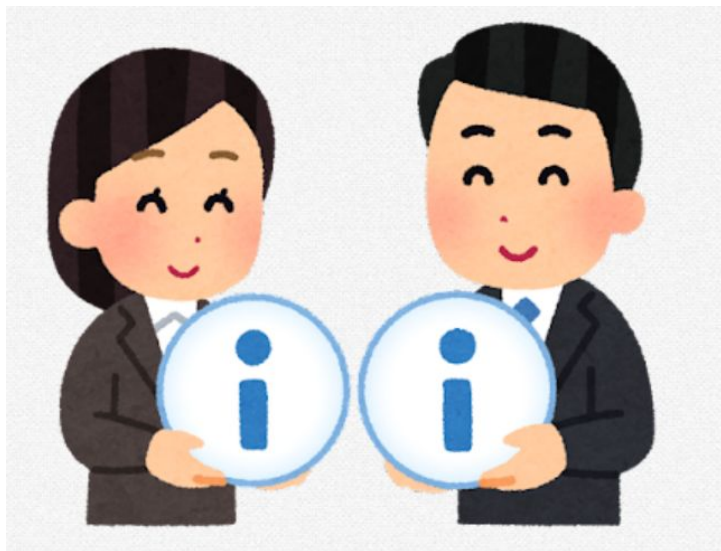
更新日期 ▾ 出勤制度 ▾ 月薪4萬up ▾ 經歷要求 ▾ 公司相關 ▾ 更多條件 ▾

全部(897) 全職(881) 兼職(0) 高階(0) 其他 ▾ 第 1 / 45 頁 ▾

Git是高階人才必備的能力!

你對於版本管控的期待？

你覺得版本管控，需要什麼樣的功能？



來安裝Git吧!

Git for Windows

Download for Windows

[Click here to download](#) the latest (2.35.1) 64-bit version of **Git for Windows**. This is the most recent [maintained build](#). It was released **16 days ago**, on 2022-02-01.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

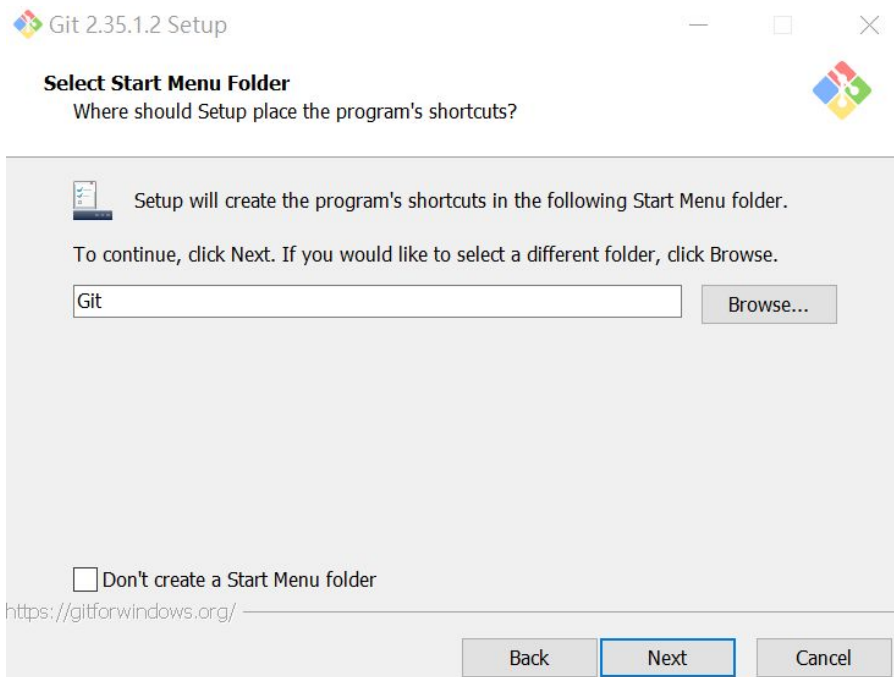
Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

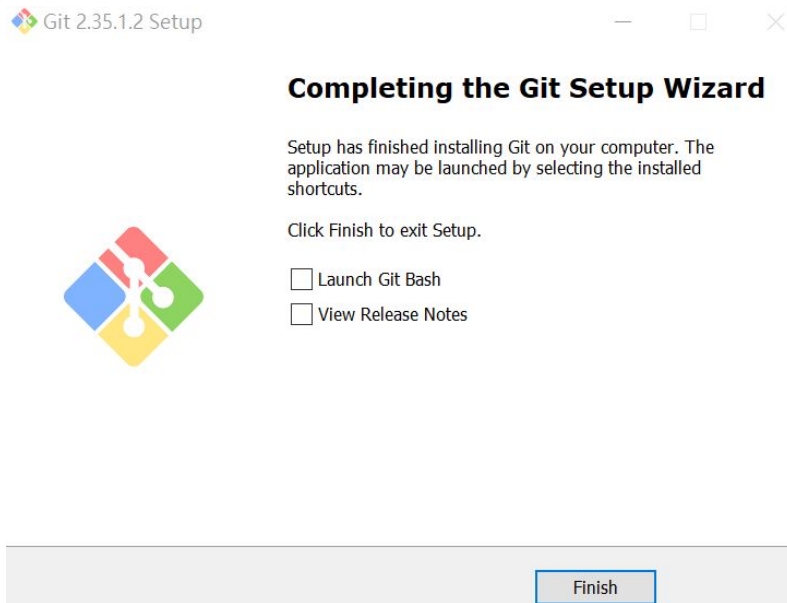
來安裝Git吧!

按Next就對了



來安裝Git吧!

看到這樣就表示成功了



請大家來一起安裝吧!

來開始使用GIT吧

我們會發現多了兩個東西

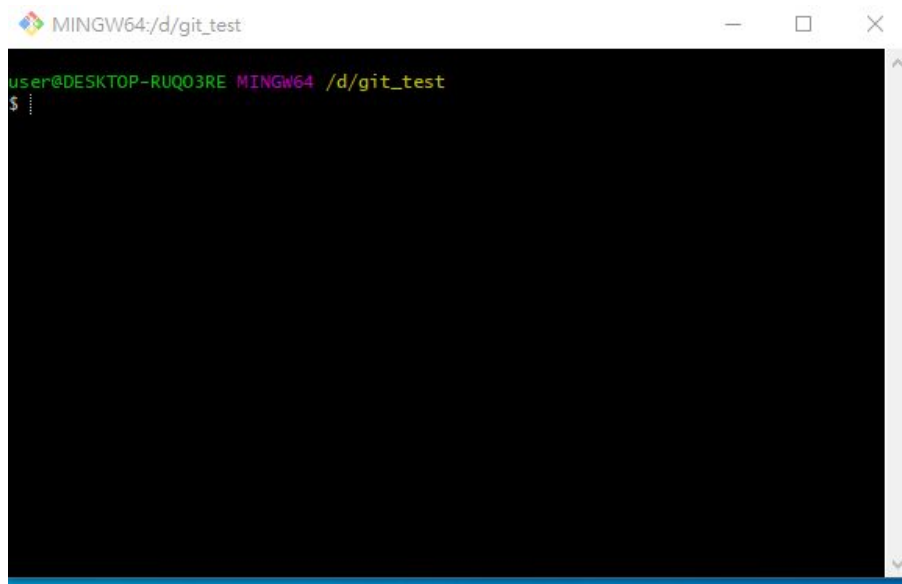


git_test.txt



Git Bash

可以讓你下命令用的

A screenshot of a Git Bash terminal window. The title bar at the top reads "MINGW64:/d/git_test" and includes standard window control buttons (minimize, maximize, close). The terminal area has a black background with green text. The first line shows the prompt "user@DESKTOP-RUQ03RE MINGW64 /d/git_test". The second line shows a dollar sign "\$" followed by a vertical ellipsis "..." and a cursor, indicating the terminal is ready for input.

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test
$ ...
```

Git GUI

按下GIT GUI會出現的樣式



表示這裡沒有GIT的倉儲(repository)

建立GIT的第一個倉儲

在建立之前，請先輸入你的email和名稱

```
git config --global user.email "你的email"
```

```
git config --global user.name "你的名字"
```

讓未來團隊合作的時候知道是誰做了什麼動作

如果要看自己輸入了什麼

在建立之前, 請先輸入你的email和名稱

git config user.name -> 顯示你的名字

git config user.email -> 顯示你的email

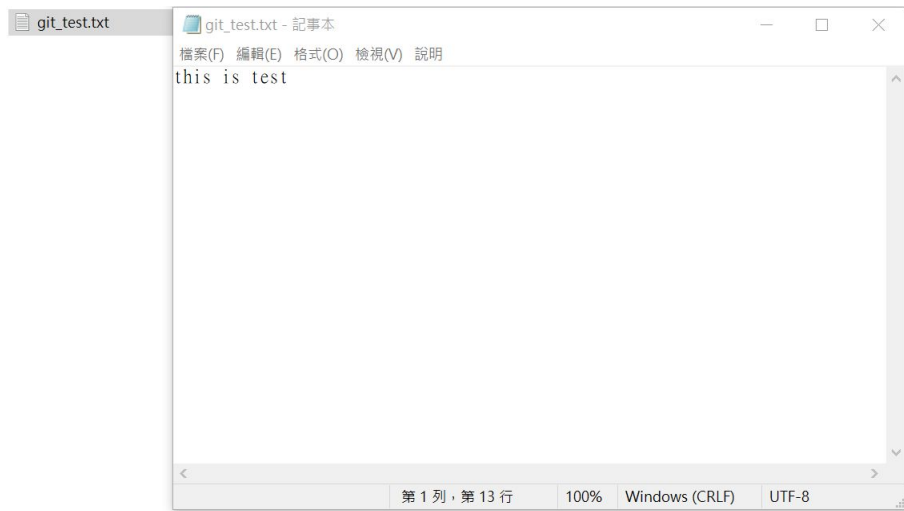
如果你要重新修改自己的名字和email

git config --global user.email "你的email"

git config --global user.name "你的名字"

建立GIT的第一個倉儲

終於要建立第一個倉儲了!



先產生一個txt檔案, 裡面寫一些內容

建立GIT的第一個倉儲

名稱

git_test

類型

文字文件

MINGW64:/d/git_test

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test
$ git init
Initialized empty Git repository in D:/git_test/.git/

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ |
```

如何知道建立好了？

名稱	大小	類型
.git		檔案資料夾
git_test.txt	1 KB	文字文件

如果你有開隱藏資料夾
裡面就會有一個.git的隱藏資料夾

如何知道建立好了？

比較好的做法

看現在的狀態

看狀態

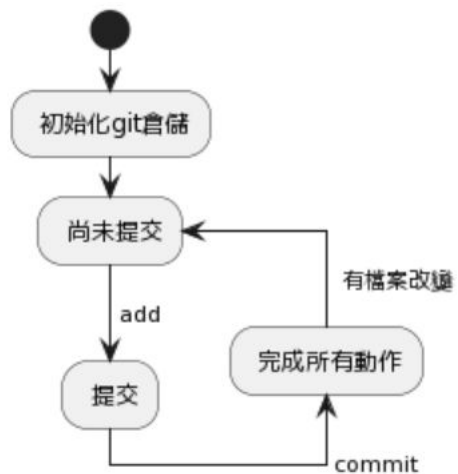
```
$ git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    git_test.txt

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

尚未commit?

GIT 的狀態

你的倉儲/檔案會有四個狀態



Git status 可以讓你看在哪一個狀態內

看狀態

```
$ git status
On branch master

No commits yet

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

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

紅色是表示檔案在尚未提交的狀態

轉到提交的狀態

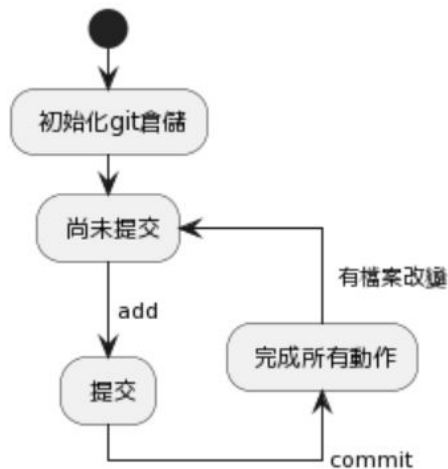
```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git add git_test.txt

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   git_test.txt
```

利用add <file>的指令改變狀態到“提交”



轉到提交的狀態

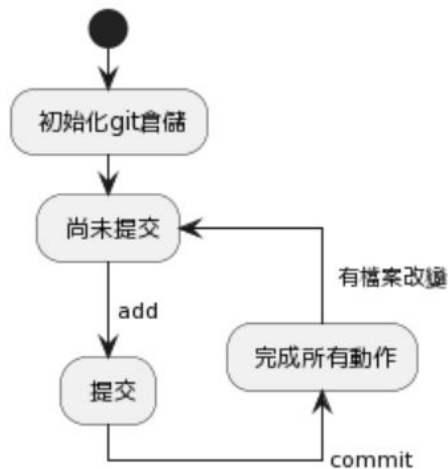
```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git add git_test.txt

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   git_test.txt
```

綠色是指檔案到“提交”的狀態



取消提交的狀態

如果要反悔

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   git_test.txt

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git reset git_test.txt

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git status
On branch master

No commits yet

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

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

`git reset <file_name>`

轉到提交的狀態

如果要一次提交所有的東西

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git add .

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git status
On branch master

No commits yet

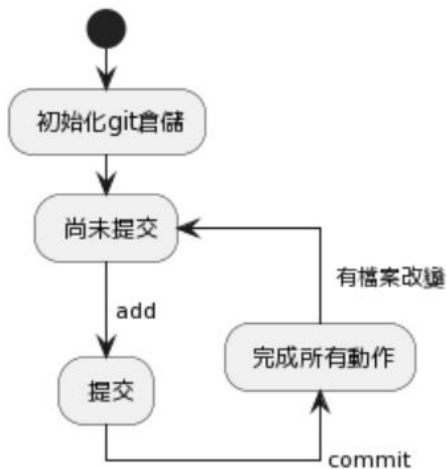
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   git_test.txt
```

add和.之間要有空格

git add .

準備完成所有動作

利用commit完成所有的動作



下commit錯誤

如果沒有輸入email和姓名會這樣

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git commit -m 'first commit'
Author identity unknown

*** Please tell me who you are.

Run

    git config --global user.email "you@example.com"
    git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'user@DESKTOP-RUQ03RE.(none)')
```

準備完成所有動作

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git commit -m 'first commit'
[master (root-commit) c495e8e] first commit
1 file changed, 1 insertion(+)
create mode 100644 git_test.txt
```

‘’內的是這個版本的說明

```
$ git status
On branch master
nothing to commit, working tree clean
```

再看一次狀態，沒有檔案有變化

再試一次

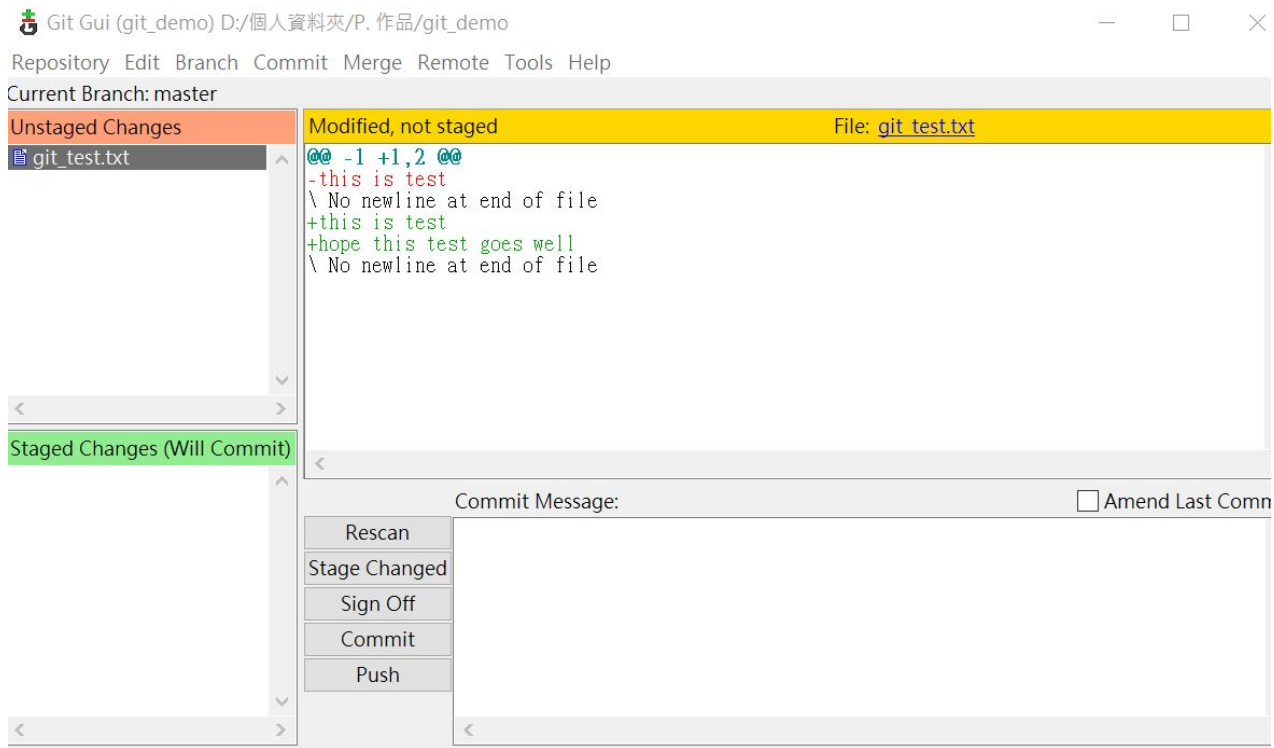
打開檔案，隨便改一下，看看狀況

```
$ 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:   git_test.txt

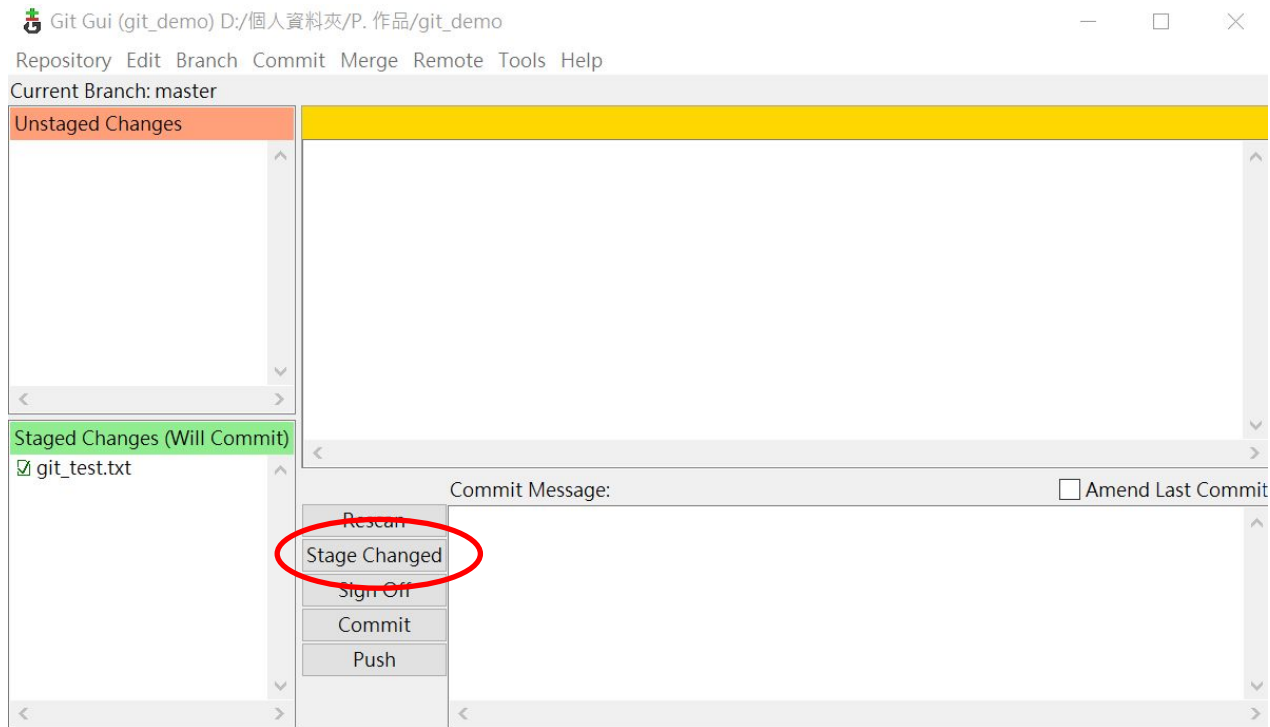
no changes added to commit (use "git add" and/or "git commit -a")
```

這一個檔案被修改
又準備下次的循環了

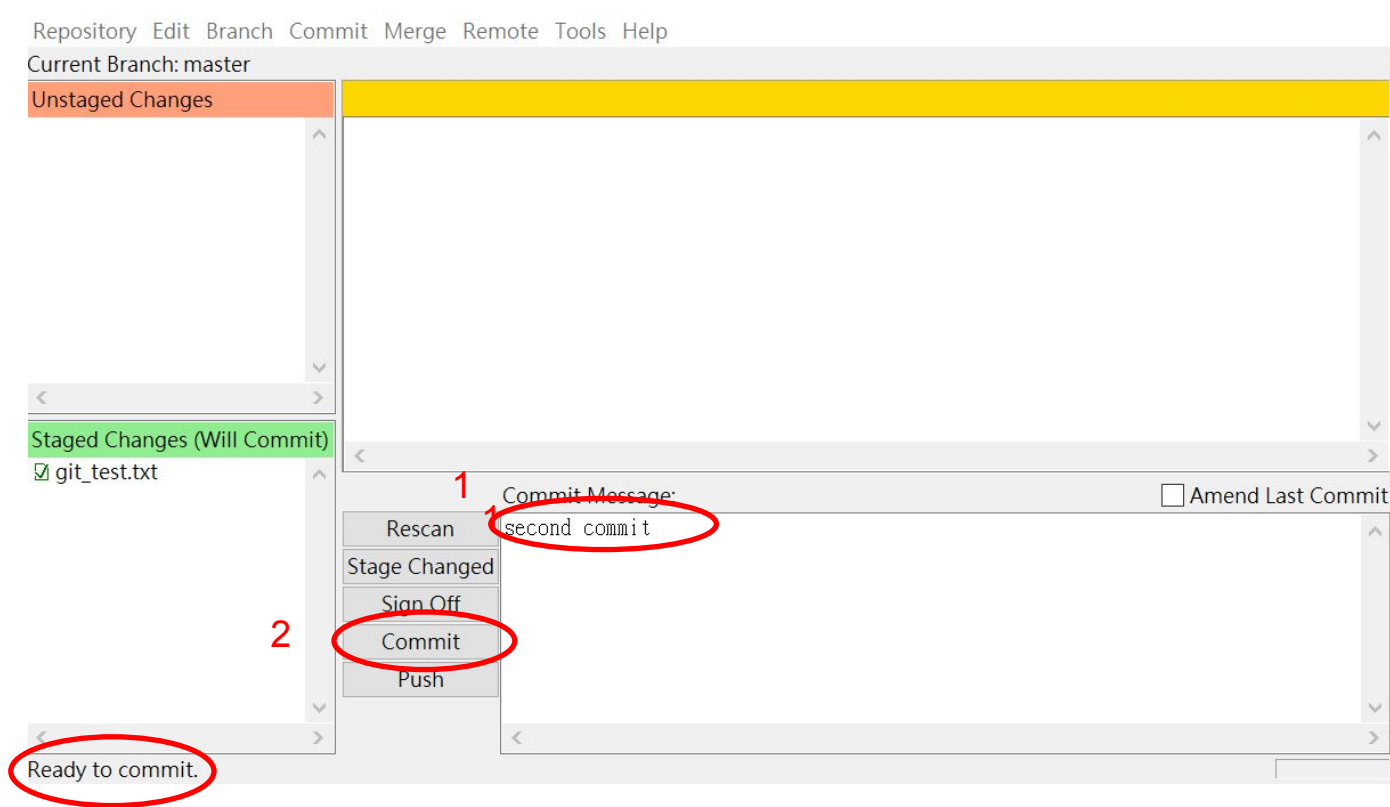
這次用GUI來用吧



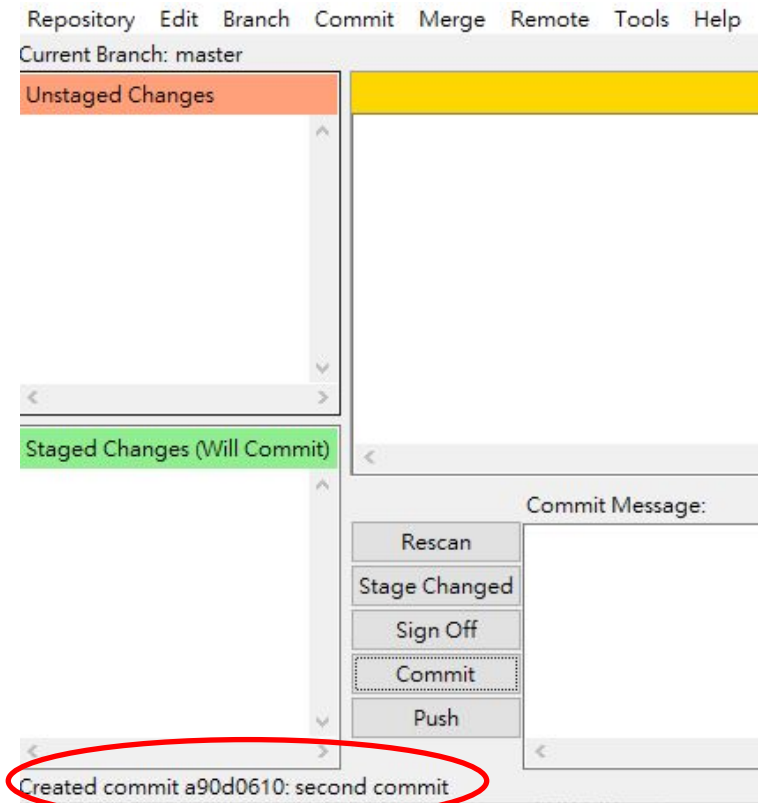
要變換到提交的狀態



準備完成所有動作



完成所有動作



問題-我要怎麼看到所有的版本?

```
$ git log
commit a90d06102a68cd38b39b836ba27483d4ab352a62 (HEAD -> master)
Author: leo.wu <leo.wu>
Date: Mon Mar 14 10:33:25 2022 +0800

    second commit

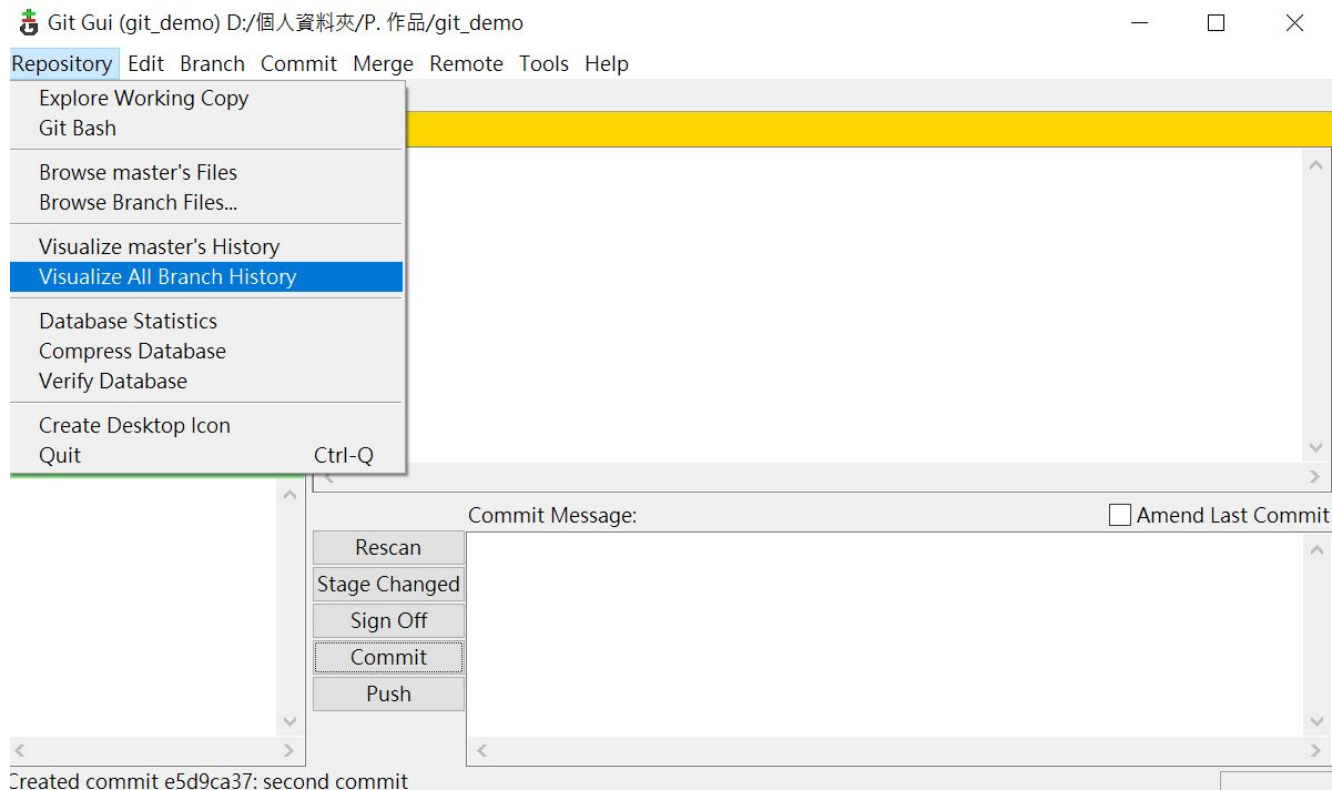
commit c495e8ee510ce299cdb873bb0406e0fa29af9648
Author: leo.wu <leo.wu>
Date: Mon Mar 14 10:31:05 2022 +0800

    first commit
```

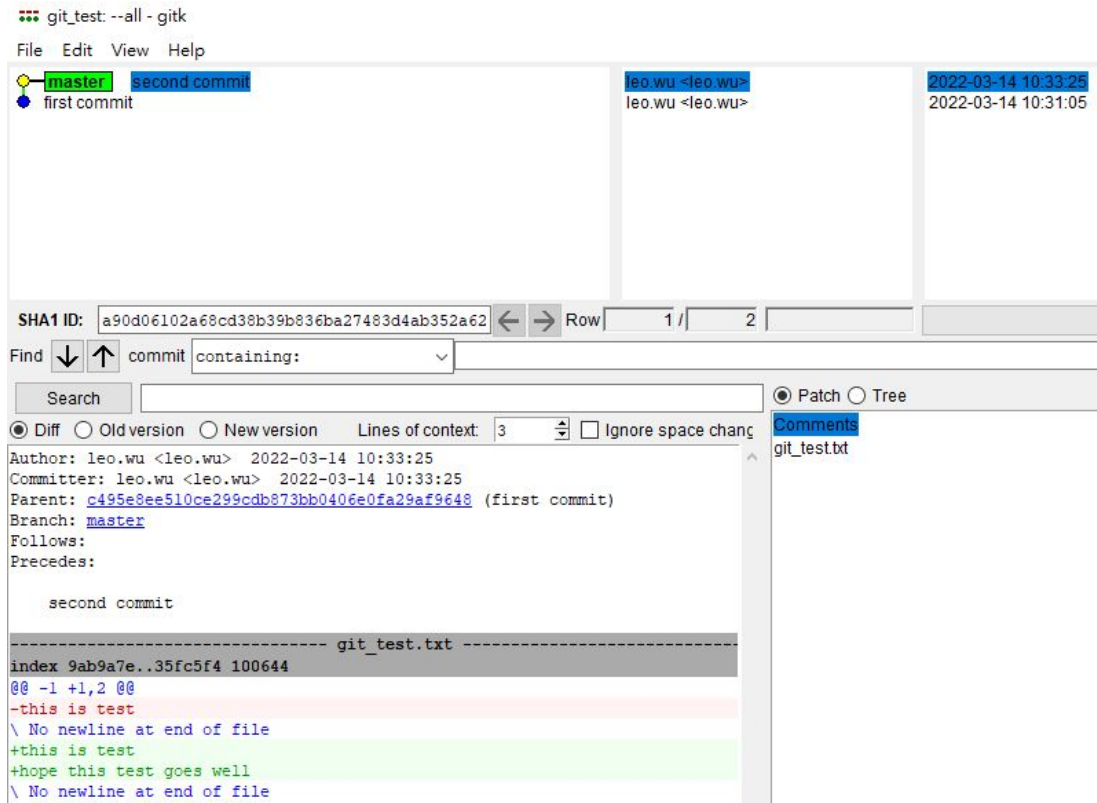
剛剛的版本全部都出來了

16進位版本號碼

有沒有更好的方式？



gitk – 可以看到所有的狀態



這有哪一些資訊?

黃色是目前的位置

16進位版本號碼

誰?
什麼時候?
版本的敘述

上一版本
跟這一版差別

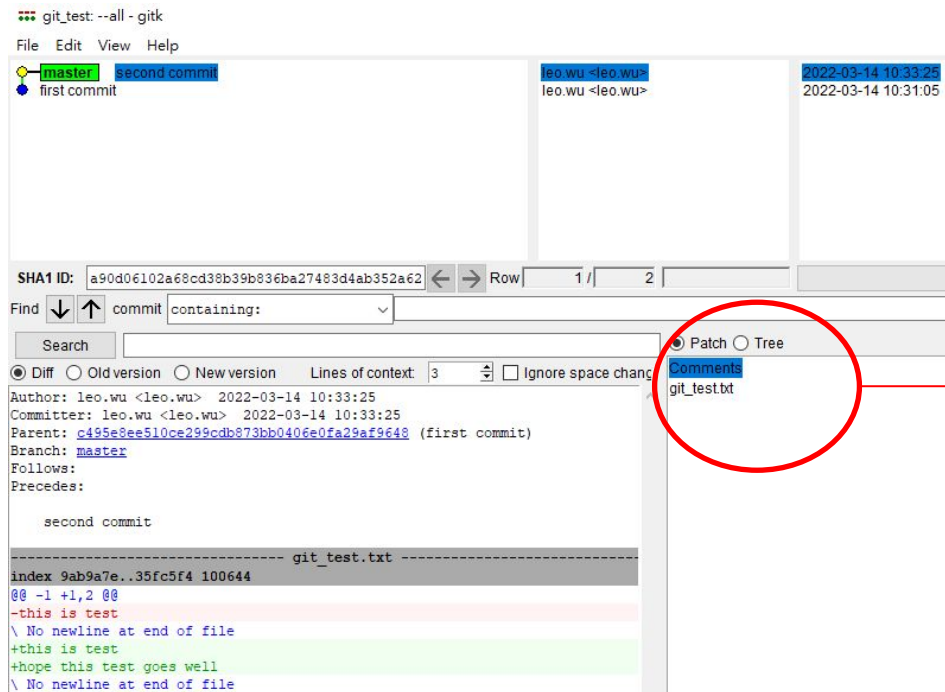
```
git_test: --all - gitk
File Edit View Help
master second commit
first commit
leo.wu <leo.wu>
leo.wu <leo.wu>
2022-03-14 10:33:25
2022-03-14 10:31:05

SHA1 ID: la90d06102a68cd38b39b836ba27483d4ab352a62 Row 1 / 2
Find commit containing:
Search
Diff Old version New version Lines of context: 3 Ignore space changes
Patch Tree
Comments
git_test.txt
Author: leo.wu <leo.wu> 2022-03-14 10:33:25
Committer: leo.wu <leo.wu> 2022-03-14 10:33:25
Parent: c495e8ee510ce299cdb873bb0406e0fa29af9648 (first commit)
Branch: master
Follows:
Precedes:

second commit

----- git_test.txt -----
index 9ab9a7e..35fc5f4 100644
@@ -1 +1,2 @@
- this is test
\ No newline at end of file
+ this is test
+ hope this test goes well
\ No newline at end of file
```

這有哪一些資訊？



哪一些檔案被修改了

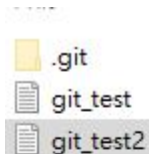
要如何回到之前的狀態？



我們做紀錄的重點是
要能回到過去

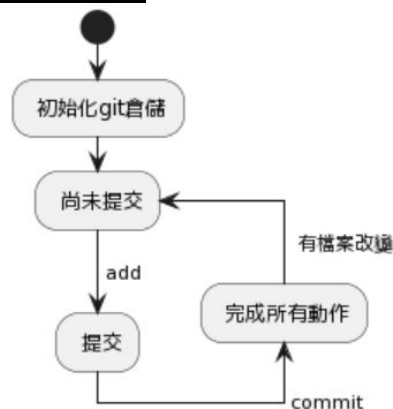
再多設定一版

首先，我們增加一個檔案並且記錄



```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  git_test2.txt

nothing added to commit but untracked files present (use "git add" to track)
g
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git add .
git
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git commit -m 'third commit'
[master a9c74bd] third commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 git_test2.txt
```



準備回到過去



回到過去

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git reset --hard a90d
HEAD is now at a90d061 second commit
```




恭喜搭乘時光機回到過去

可是，我後悔了

如果你想要回到第三版

```
$ git reflog  
a90d861 (HEAD -> master) HEAD@{0}: reset: moving to a90d  
a9c74bd HEAD@{1}: commit: third commit  
a90d061 (HEAD -> master) HEAD@{2}: commit: second commit  
c495e8e HEAD@{3}: commit (initial): first commit
```



找出第三版的版本編號

可是，我後悔了

```
$ git reflog
a90d061 (HEAD -> master) HEAD@{0}: reset: moving to a90d
a9c74bd HEAD@{1}: commit: third commit
a90d061 (HEAD -> master) HEAD@{2}: commit: second commit
c495e8e HEAD@{3}: commit (initial): first commit

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git reset --hard a9c74bd
HEAD is now at a9c74bd third commit
```



我們又回來了!

Git 分支

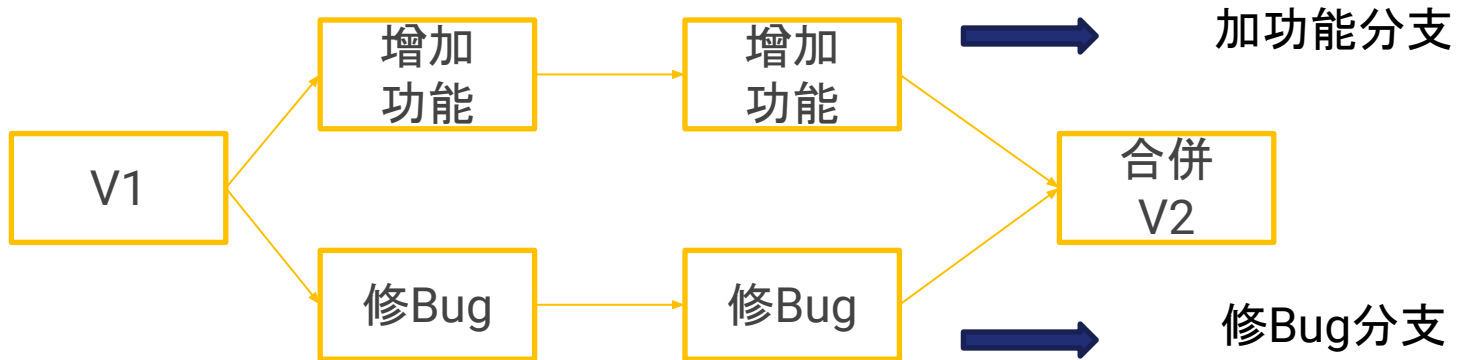
分支算是Git的最大特色



為什麼要使用分支？

Git 分支

舉個例子



確定分支

先來看看有哪一些分支

```
$ git branch  
* master
```

master

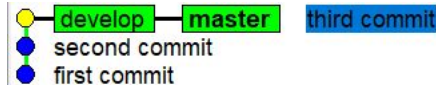
目前有一個叫做master的分支

開分支

開一個develop的分支

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git branch develop

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git branch
  develop
* master
```



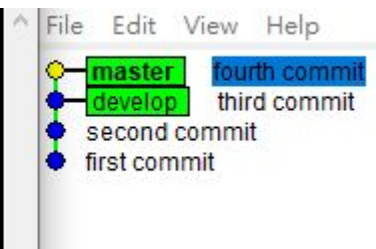
```
2022-02-21 11:20:28
2022-02-18 15:53:04
2022-02-18 15:35:29
```

多了一個叫做develop的分支
但是目前在master這一個分支內

如果這時候提交

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git add .

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git commit -m 'fourth commit'
[master 22dfb18] fourth commit
1 file changed, 2 insertions(+)
```



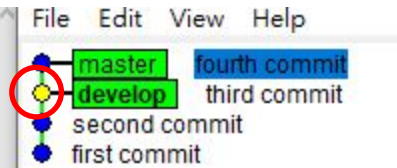
提交的內容是在master的分支無誤

切換分支

切換成develop分支

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git checkout develop
Switched to branch 'develop'

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (develop)
$
```

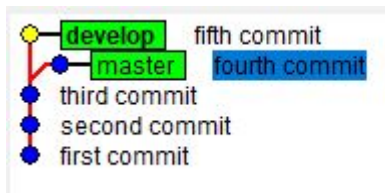


可以看到黃色的點回到develop上

如果這時候提交

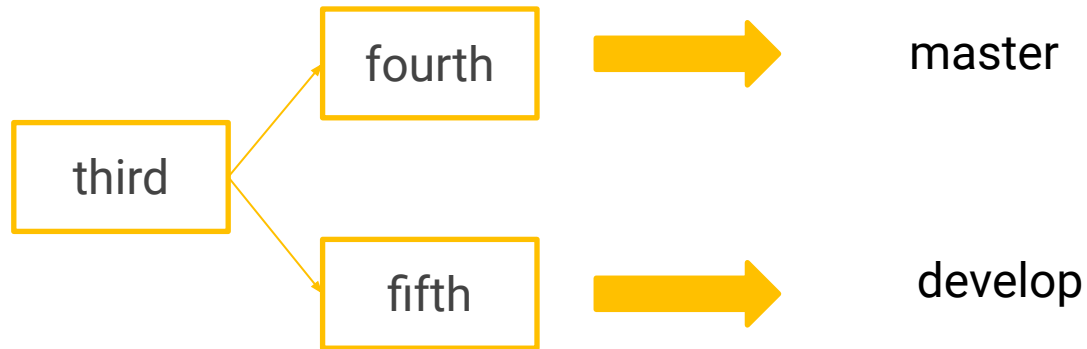
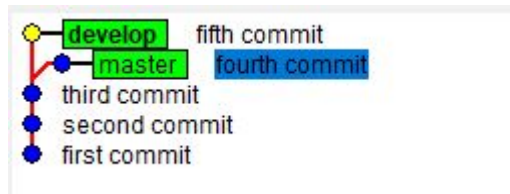
```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   git_test2.txt

no changes added to commit (use "git add" and/or "git commit -a")
g
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (develop)
$ git add .
git
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (develop)
$ git commit -m 'fifth commit'
[develop ace0734] fifth commit
1 file changed, 2 insertions(+)
```

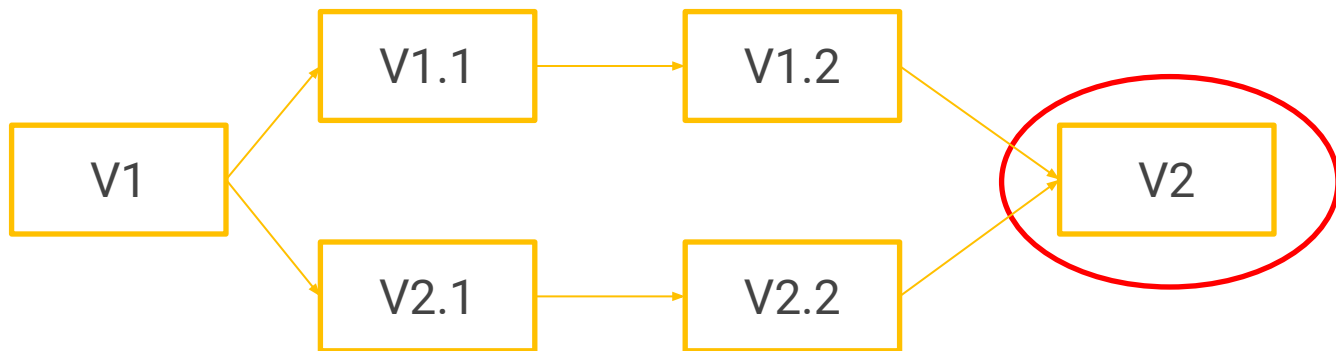


終於有分支的感覺了？

現在的情況



分支合併



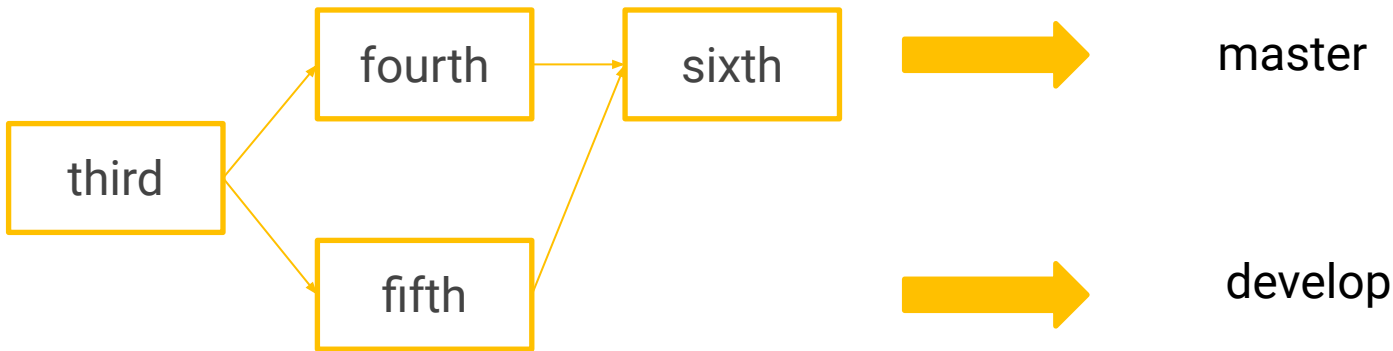
開發完成之後，我們需要合併分支

分支合併

1. 回到master的分支
2. 將develop 合併到master

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (develop)
$ git checkout master
Switched to branch 'master'

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git merge develop
```

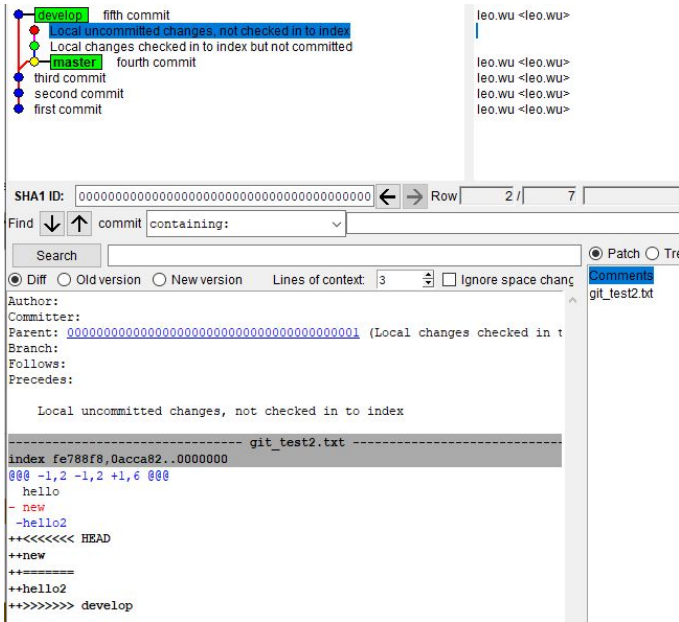


結果自動合併失敗

合併有衝突, 所以失敗了

```
$ git merge develop
Auto-merging git_test2.txt
CONFLICT (content): Merge conflict in git_test2.txt
Automatic merge failed; fix conflicts and then commit the result.
```

利用gitk來看看為甚麼失敗



master

Develop

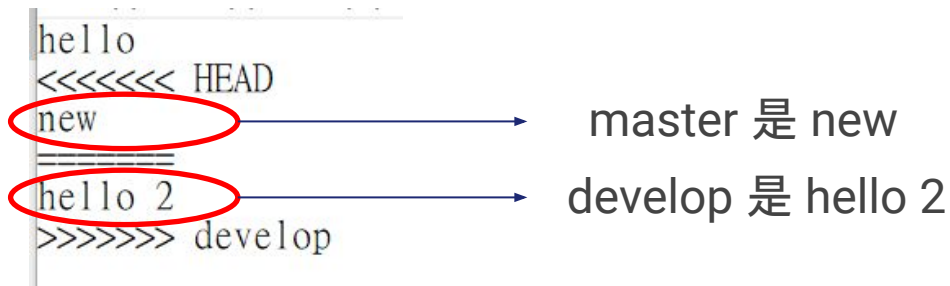
hello
new

hello
hello2

Hello後面到底是new還是hello2

合併衝突

Git 很貼心地告訴你哪裡衝突



HEAD = 現在的位置 = master

手動修改衝突

我們人工修改衝突

 git_test2.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
hello
new
hello 2

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master|MERGING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

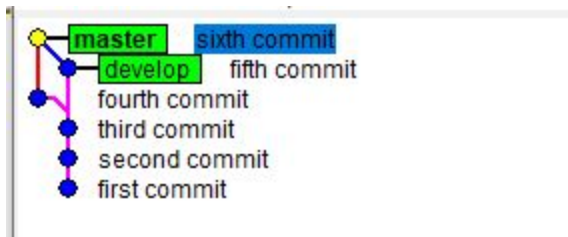
Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   git_test2.txt

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

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master|MERGING)
$ git add .
gi
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master|MERGING)
$ git commit -m 'sixth commit'
[master 3753fd7] sixth commit
```

手動修改衝突之後

Six commit 成功了!



什麼樣的情況之下, git 會自動合併?

Git 可以自動合併的條件 - 1

不同檔案修改

master

Develop



First.txt



Second.txt



First.txt



Second.txt

猜猜看自動合併會變成怎樣？

Git 可以自動合併的條件 - 1

不同檔案修改

自動合併

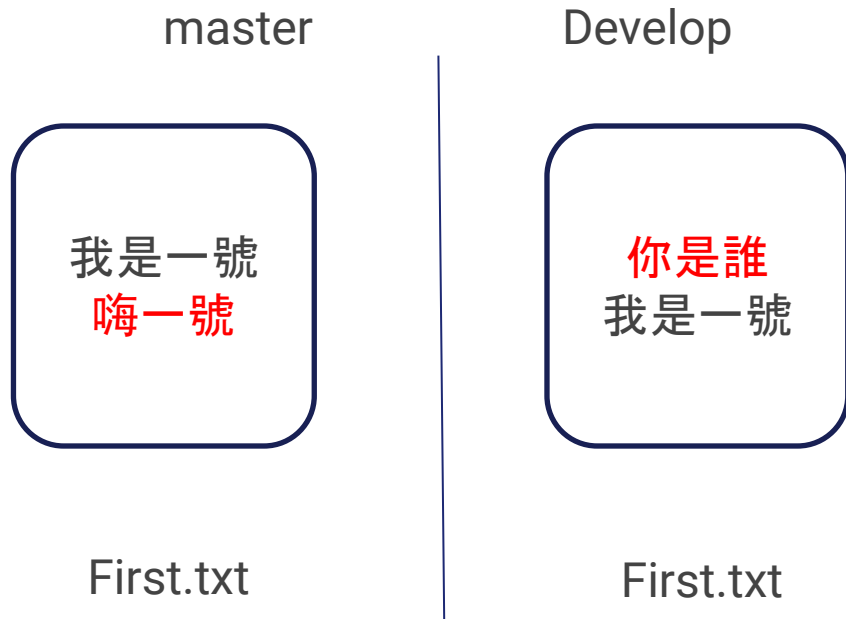
我是一號
嗨一號

我是二號
嗨二號

還滿合理的

Git 可以自動合併的條件 - 2

同檔案不同位置修改



Git 可以自動合併的條件 - 2

同檔案不同位置修改

自動合併

你是誰
我是一號
嗨一號

不用再人工合併程式

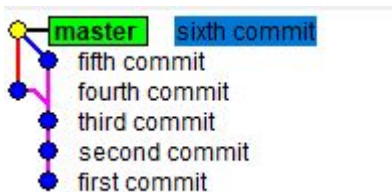
非常好用!

Git 分支刪除

最後，將不用的分支刪除

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git branch -d develop
Deleted branch develop (was ace0734).

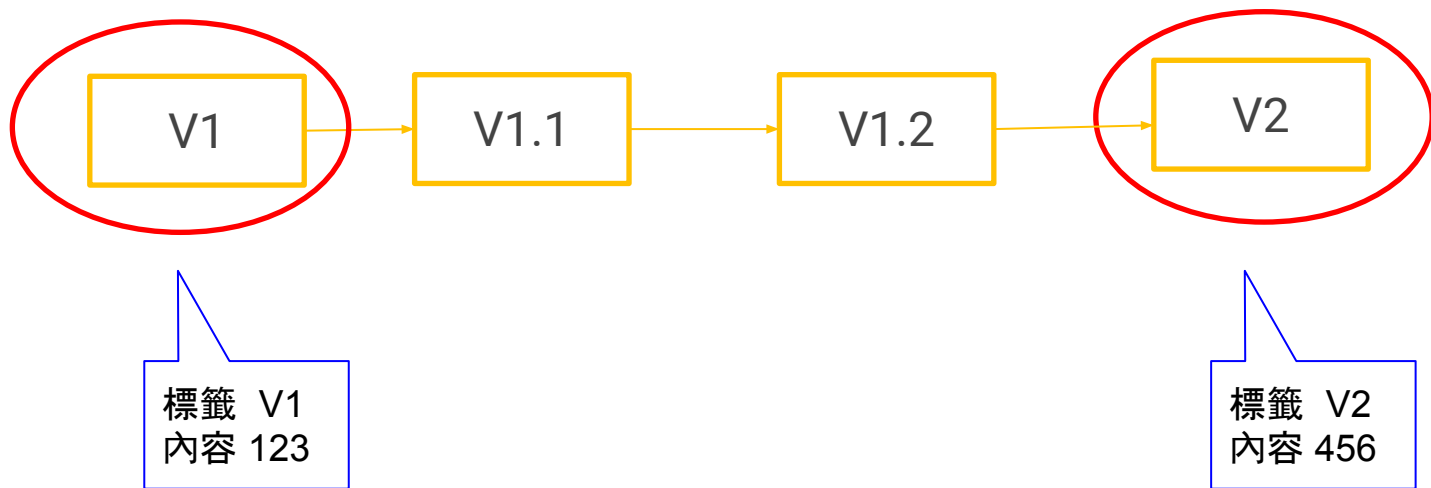
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git branch
* master
```



Develop的分支被刪除了

Git標籤

如果要發行版本，可以在特定的版本放上標籤，方便快速搜尋



標籤好用的地方

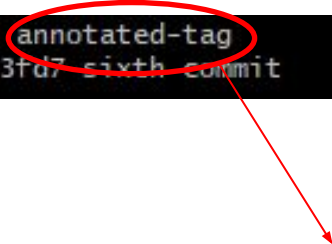
因為標籤是版號的別名

所以可以搭配reset -hard使用

git tag -> 看資訊

git reset -hard <tag_name> -> 切換到此版本

```
$ git reset --hard annotated-tag  
HEAD is now at 3753fd7 sixth commit
```



不用再輸入複雜的版號

輕量標籤 VS 標示標籤

輕量標籤
(lightweight tag)

一個commit的別名
資訊少
較少用

標示標籤
(annotated tag)

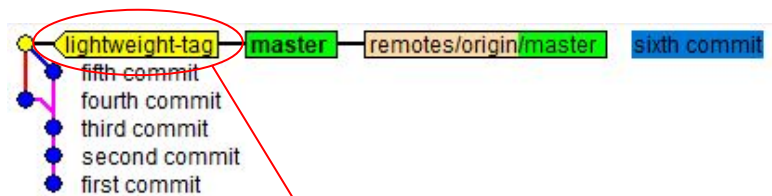
一個正式的標籤
資訊多
較常用

輕量標籤

加入輕量標籤

`git tag <tag_name>`

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git tag lightweight-tag
```



多了一個標籤

看標籤狀態

git tag -> 看所有的標籤(包含輕量標籤跟標示標籤)

git tag -n ->看所有的標籤和敘述(包含輕量標籤跟標示標籤)

```
$ git tag
lightweight-tag

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git tag -n
lightweight-tag sixth commit
```

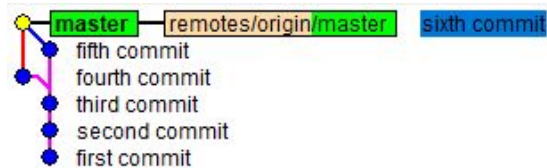
輕量標籤的狀態是commit的敘述

刪除標籤

刪除特定標籤

`git tag -d <tag_name>`

```
$ git tag -d lightweight-tag  
Deleted tag 'lightweight-tag' (was 3753fd7)
```



標示標籤

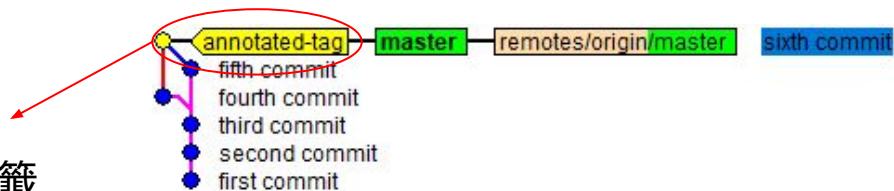
新增標示標籤

`git tag -a <tag_name> -m <tag_description>`

```
user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git tag -a annotated-tag -m 'this is annotated tag'
```

`git push origin --tags`

一樣多了一個標籤



看標籤狀態

git tag -> 看所有的標籤(包含輕量標籤跟標示標籤)

git tag -n ->看所有的標籤和敘述(包含輕量標籤跟標示標籤)

```
$ git tag
annotated-tag

user@DESKTOP-RUQ03RE MINGW64 /d/git_test (master)
$ git tag -n
annotated-tag this is annotated tag
```

標示標籤是標示時候的敘述

經驗分享

沒有使用Git:

星期三下午之前 -> 大家給小組長code, 星期三下午小組長合程式
星期四上午 -> 小組長給經理code, 經理用一天合大家的程式

用了Git: 大家再也不用合程式, 多的時間可以放空休息做更多事

下集預告

章魚貓是甚麼？

如何下載好用的code？



我要如何證明我很強？

答案都在下一堂課
Github

補充 Git 誰做了甚麼？

到底這一行是誰寫的

```
$ git blame git_test.txt
e5d9ca37 (leo.wu 2022-02-18 15:53:04 +0800 1) this is test
e5d9ca37 (leo.wu 2022-02-18 15:53:04 +0800 2) hope this test goes well
```

git blame
非常好用!

補充 Git Flow

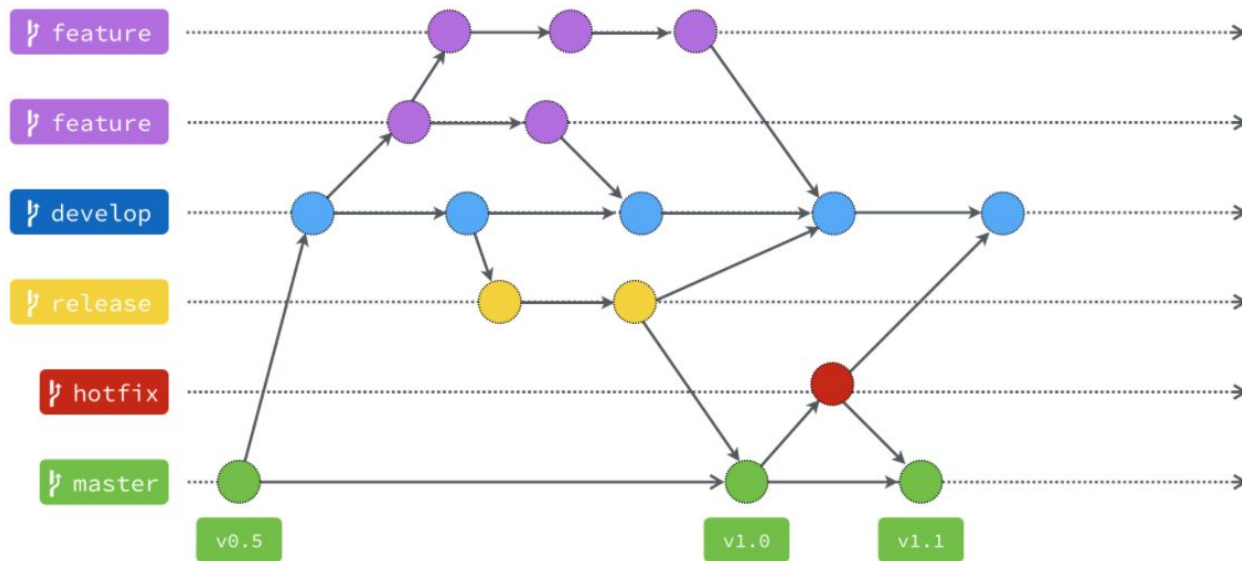
新的東西要開發

暫時版本

發版前的最後測試

突然有bug要改

發行的版本



舉例 Git Flow

補充 比較程式軟體

以前還沒有Git的時候，會用比程式軟體來看程式的差別

```
#install Openscad python
RUN pip3 install solidpython
RUN pip3 install viewscad

#install google test and google mock
RUN apt install libgtest-dev -y
RUN cd /usr/src/googletest/googletest && cmake CMakeLists.txt && make
RUN cd /usr/src/googletest/googletest && cmake CMakeLists.txt && make
RUN cp /usr/src/googletest/googletest/lib/libgtest.a /usr/lib
RUN cp /usr/src/googletest/googletest/lib/libgtest_main.a /usr/lib
RUN cp /usr/src/googletest/googletest/lib/libgmock_main.a /usr/lib
RUN cp /usr/src/googletest/googletest/lib/libgmock.a /usr/lib

#install install yaml
RUN echo "y"| apt-get install libyaml-cpp-dev

#install lib modbus
RUN mkdir /root/modbus_lib
COPY libmodbus-3.1.6.tar.gz /root/modbus_lib
RUN cd /root/modbus_lib && tar -zxvf libmodbus-3.1.6.tar.gz && cd ./libmodbus-3.1.6 && ./configure

#install voice
# RUN echo "y"| apt install sox
# RUN echo "Y"| apt-get install libsox-fmt-mp3
COPY libxml_lib.so /usr/local/lib/libxml_lib.so

#install Openscad
RUN add-apt-repository ppa:openscad/releases
RUN apt-get update
RUN echo "y"| apt-get install openscad

#install Openscad python
RUN pip3 install solidpython
RUN pip3 install viewscad

#install google test and google mock
RUN apt install libgtest-dev -y
RUN cd /usr/src/googletest/googletest && cmake CMakeLists.txt && make
RUN cd /usr/src/googletest/googletest && cmake CMakeLists.txt && make
RUN cp /usr/src/googletest/googletest/lib/libgtest.a /usr/lib
RUN cp /usr/src/googletest/googletest/lib/libgtest_main.a /usr/lib
RUN cp /usr/src/googletest/googletest/lib/libgmock_main.a /usr/lib
RUN cp /usr/src/googletest/googletest/lib/libgmock.a /usr/lib

RUN echo "y"| apt-get install libyaml-cpp-dev
RUN apt-get update
RUN echo "y"| apt install ros-dashing-rqt*

RUN mkdir /root/modbus_lib
COPY libmodbus-3.1.6.tar.gz /root/modbus_lib
RUN cd /root/modbus_lib && tar -zxvf libmodbus-3.1.6.tar.gz && cd ./libmodbus-
RUN apt install sox
RUN apt-get install libsox-fmt-mp3
```

