MMS Lab MultiMedia Systems Laboratory

Understand the basic of Git & GitHub

國立台北科技大學電子工程系
黃士嘉老師

What is Git

- What is Git ?

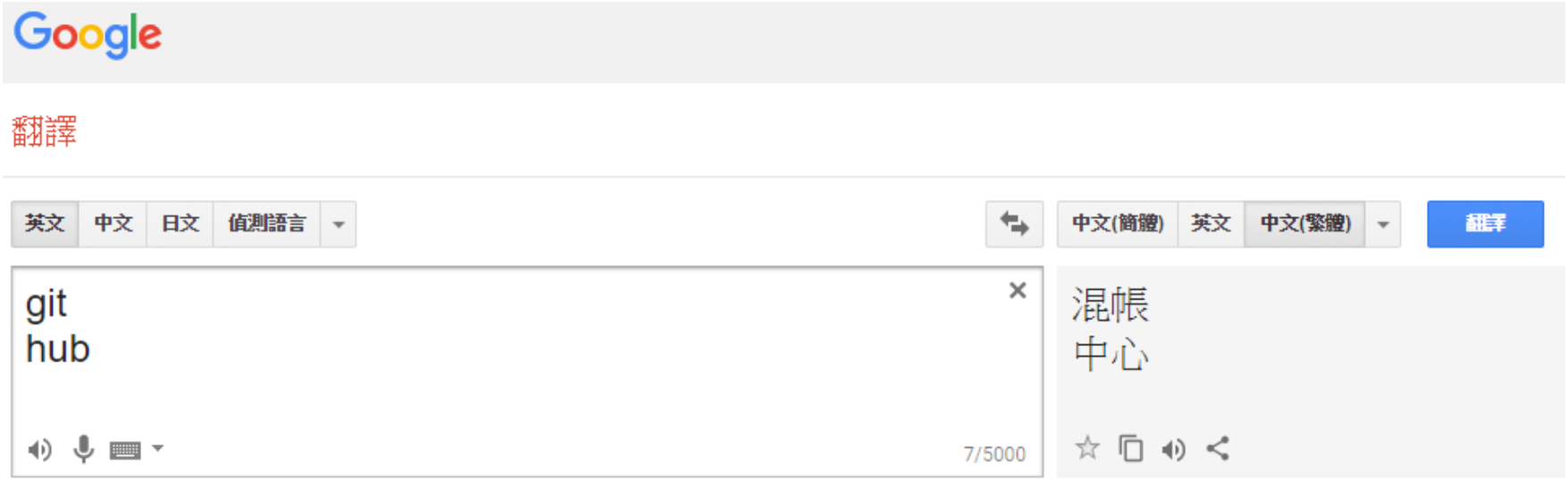
- What is Hub?

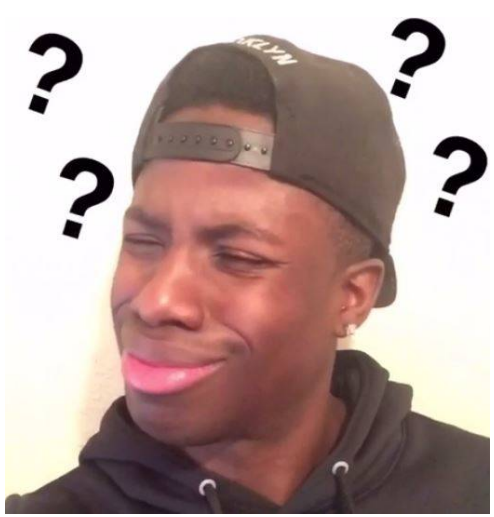
「hub」的翻譯

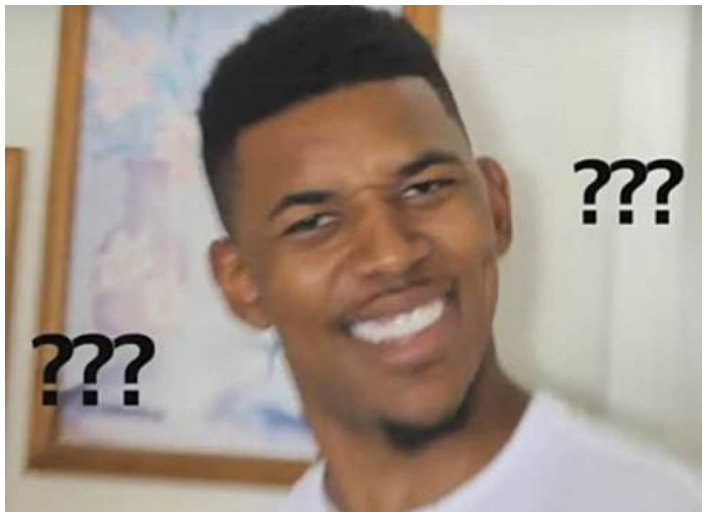
名詞

樞紐	hub, pivot, axis
中心	center, heart, hub, core, centre
轂	hub, wheel, nave, hub of wheel
樞	pivot, hub, hinge, center, centre
中轉站	hub

- Github?







- What is Git ?


Google
git
全部
圖片
影片
新聞
書籍
更多
設定
工具
約有 215,000,000 項結果 (搜尋時間：0.38 秒)
git（/ɡɪt/， 音訊（說明·資訊））是一個分散式版本控制軟體，最初由林納斯·托瓦茲（Linus Torvalds）創作，於2005年以GPL釋出。 最初目的是為更好地管理Linux內核開發而設計。
git - 維基百科，自由的百科全書 - Wikipedia
https://zh.wikipedia.org/zh-tw/Git
進一步瞭解這項結果
意見回饋

- What is Git ?
  - A version control software

- What is GitHub?

GitHub [編輯]

維基百科，自由的百科全書

GitHub是一個透過Git進行版本控制的軟體原始碼代管服務，由GitHub公司（曾稱Logical Awesome）的開發者Chris Wanstrath、PJ Hyett和Tom Preston-Werner使用Ruby on Rails編寫而成。

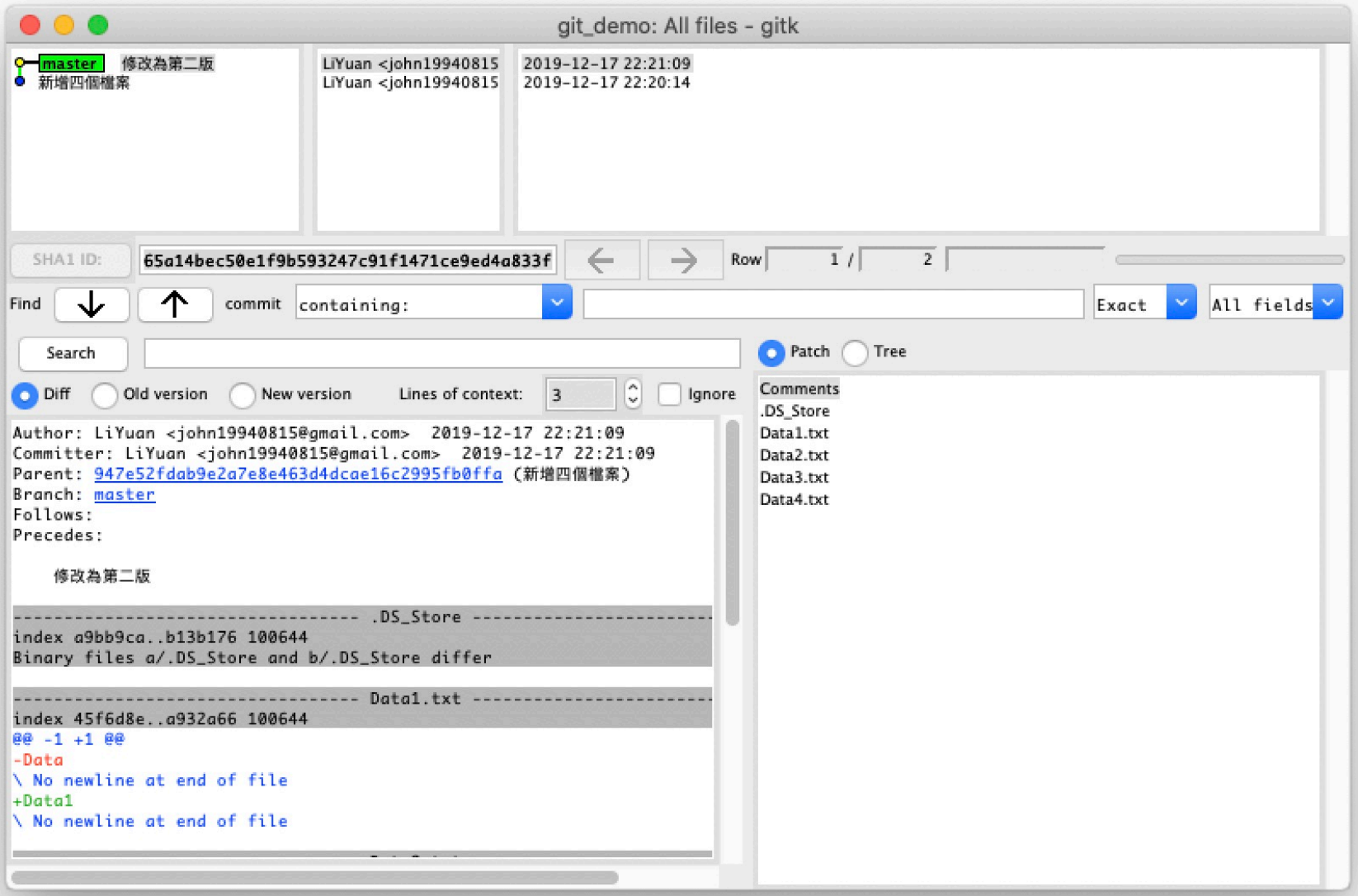
- Is there other software for version control?

Open source [edit]

- ArX – written by Walter Landry, started as a fork of GNU arch, but has been completely rewritten
- Bazaar – written in Python, originally by Martin Pool and sponsored by Canonical; decentralised, and aims to be fast and easy to use; can losslessly import Arch archives
- BitKeeper – was used in Linux kernel development (2002 – April 2005) until it was abandoned due to being proprietary. It was open-sourced in 2016 in an attempt to broaden its appeal again.
- Codeville – written in Python originally by Ross Cohen; uses an innovative merging algorithm
- Darcs – written in Haskell and originally developed by David Roundy; can keep track of inter-patch dependencies and automatically rearrange and "cherry-pick" them using a "theory of patches"
- DCVS – decentralized and CVS-based
- Fossil – written by D. Richard Hipp for SQLite; distributed revision control, wiki, and bug-tracking (all-in-one solution) with console and web interfaces. Single portable executable and single repository file.
- Git – written in a collection of Perl, C, and various shell scripts, designed by Linus Torvalds based on the needs of the Linux kernel project; decentralized, and aims to be fast, flexible, and robust
- GNU arch
- Mercurial – written in Python as an Open Source replacement to BitKeeper; decentralized and aims to be fast, lightweight, portable, and easy to use
- Monotone – developed by the Monotone Team; decentralized in a peer-to-peer way
- SVK – written in Perl by Kao Chia-liang; built on top of Subversion to allow distributed commits
- Veracity - Is another distributed version control system which includes bug tracking and Agile software development tools integrated with the version control features.

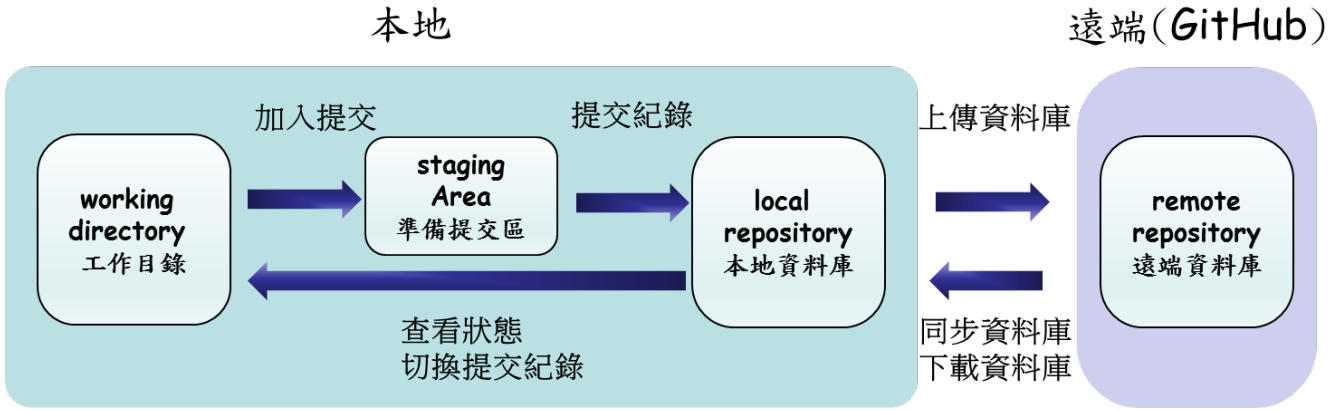
https://en.wikipedia.org/wiki/List_of_version_control_software

- Why version control?

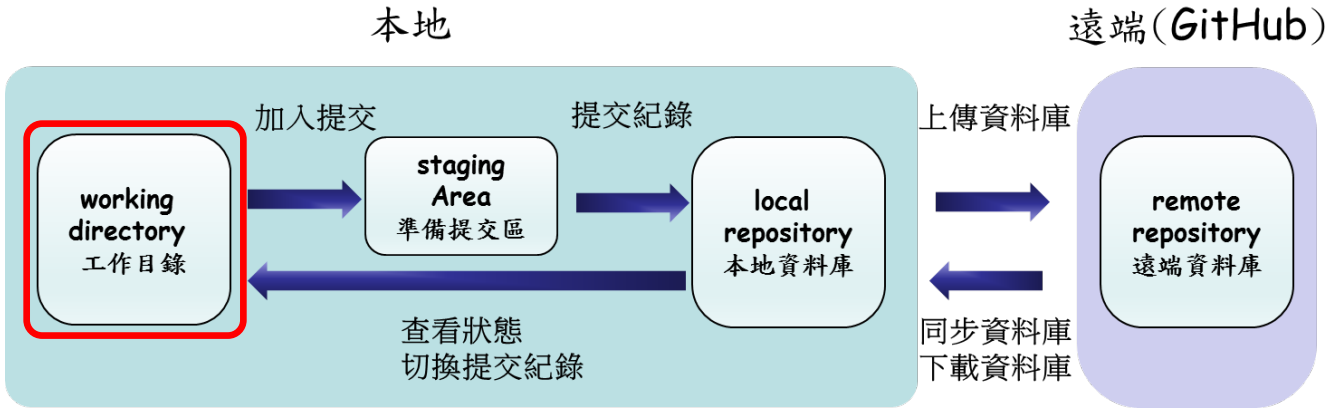
- 
git_demo: All files - gitk
master 修改為第二版
新增四個檔案
LiYuan <john19940815
LiYuan <john19940815
2019-12-17 22:21:09
2019-12-17 22:20:14
SHA1 ID:
65a14bec50e1f9b593247c91f1471ce9ed4a833f
Row 1 / 2
Find
commit
containing:
Exact
All fields
Search
Patch
Tree
Diff
Old version
New version
Lines of context:
3
Ignore
Author: LiYuan <john19940815@gmail.com> 2019-12-17 22:21:09
Committer: LiYuan <john19940815@gmail.com> 2019-12-17 22:21:09
Parent: 947e52fdab9e2a7e8e463d4dcae16c2995fb0ffa (新增四個檔案)
Branch: master
Follows:
Precedes:
修改為第二版
---------- .DS_Store ----------
index a9bb9ca..b13b176 100644
Binary files a/.DS_Store and b/.DS_Store differ
---------- Data1.txt ----------
index 45f6d8e..a932a66 100644
@@ -1 +1 @@
-Data
\ No newline at end of file
+Data1
\ No newline at end of file
Comments
.DS_Store
Data1.txt
Data2.txt
Data3.txt
Data4.txt

Workflow of Git

- The work flow of Git



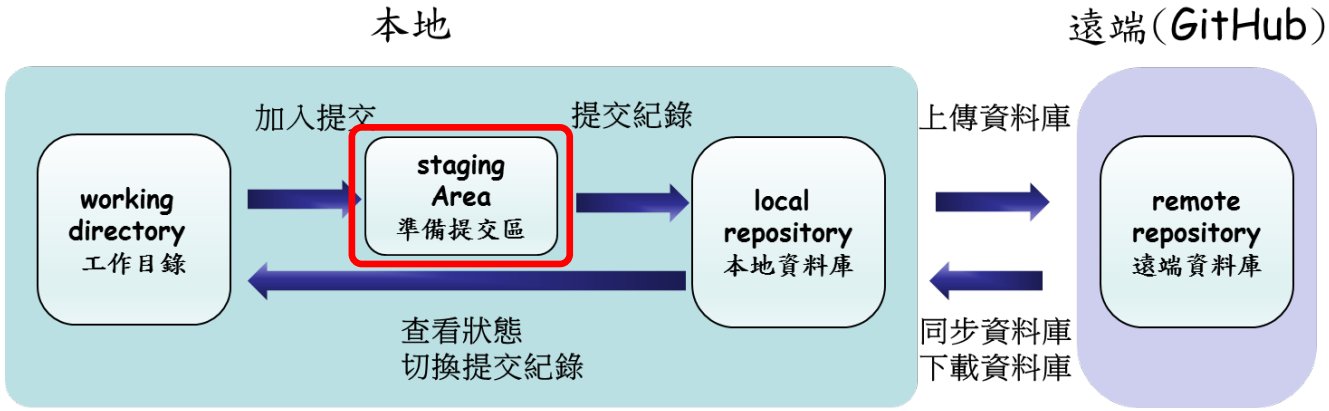
- The work flow of Git



  - Working directory

    工作目錄主要是存放要被版本控制的檔案資料夾。我們可以選擇一個一般資料夾並在資料夾內建立 git 的資料庫(Repository)，就能把該資料夾變成 git 的工作目錄

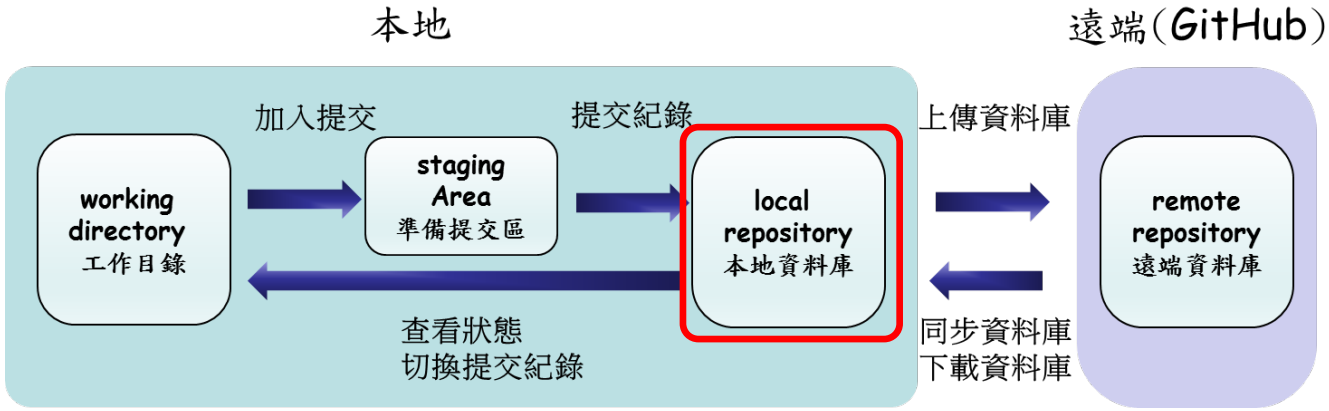
- The work flow of Git



- Staging area

  準備提交區用於記錄即將要被提交的資料。當在工作目錄下檔案的有進行變更，且我們希望能提交這些變更，我們會將這些資料存入準備提交區之中，這狀態下只有標記那些資料要被提交，但還並未實際的做提交紀錄。

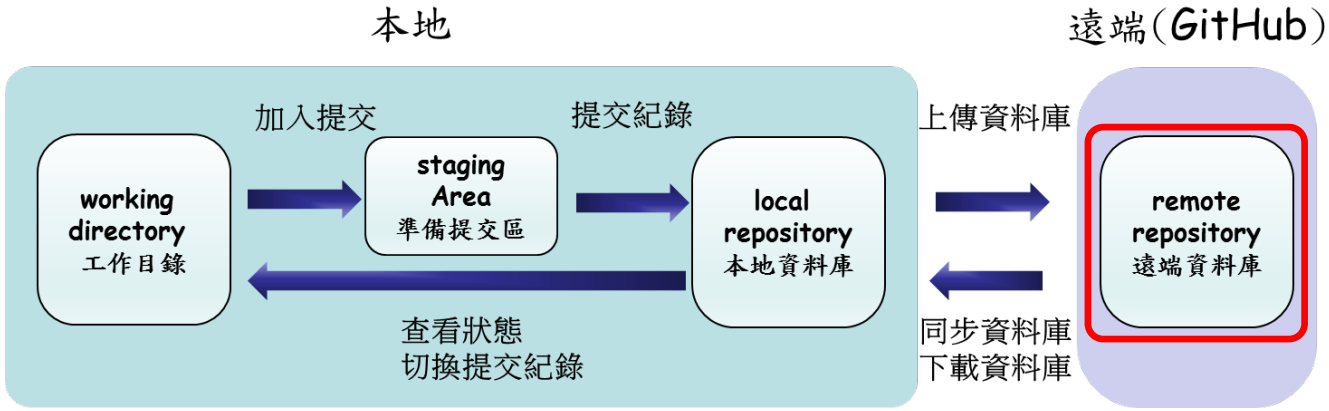
- The work flow of Git



- Local repository

  即為自己電腦端上的資料庫。當確定好所有要提交的資料都加入到準備提交區之後，可以將準備提交區的資料做提交紀錄，提交後的資料會被記錄成一個提交紀錄保存於資料庫中。

- The work flow of Git

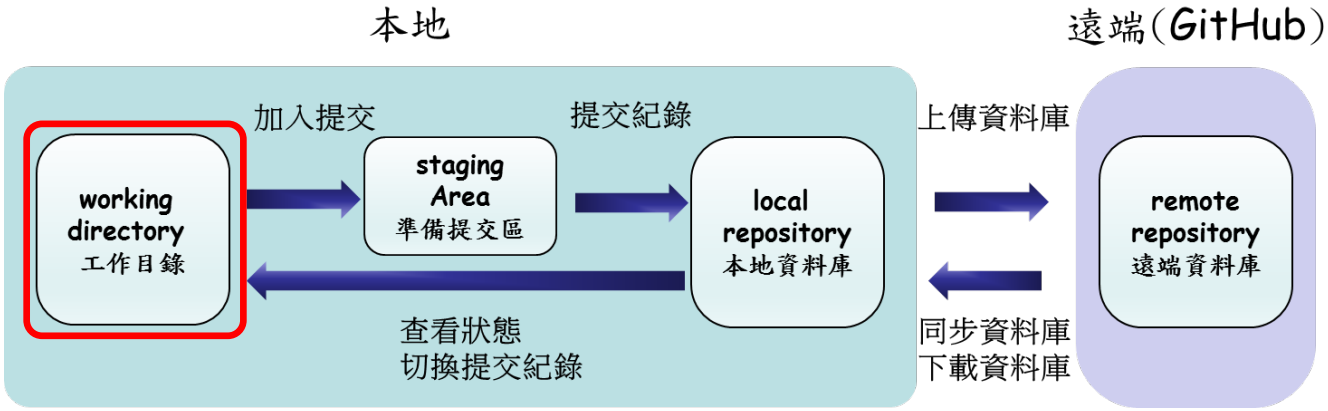


- Remote repository

  即為遠端伺服器上的資料庫。當本地資料備齊後，我們可以透過上傳將資料保存到遠端資料庫，也可以反過來將遠端資料庫的紀錄同步下來。

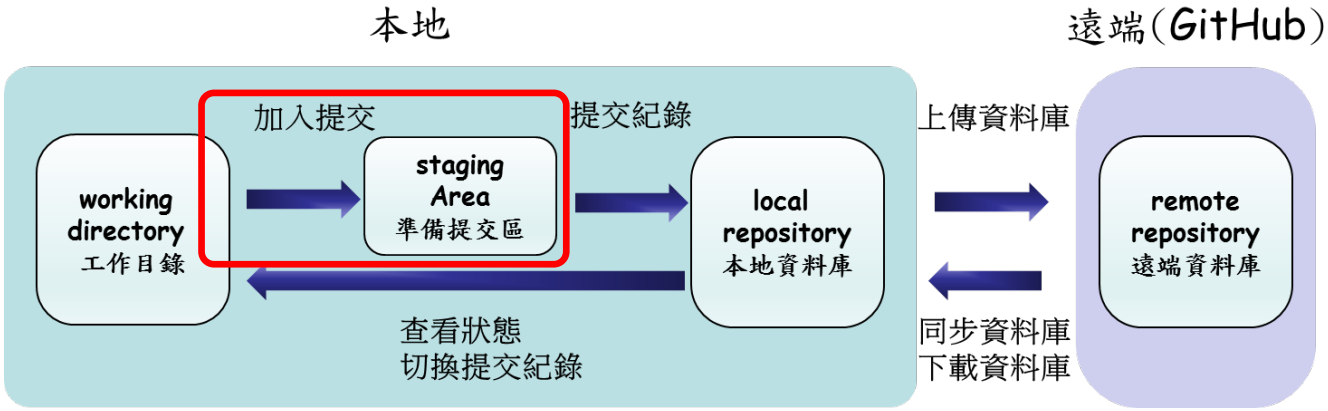
Basic command of Git

- Git init
  - 將目前資料夾初始化為working directory




```
Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab0
$ git init
Initialized empty Git repository in C:/Users/Jdway/Desktop/Lab0/.git/

Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab0 (master)
$
```

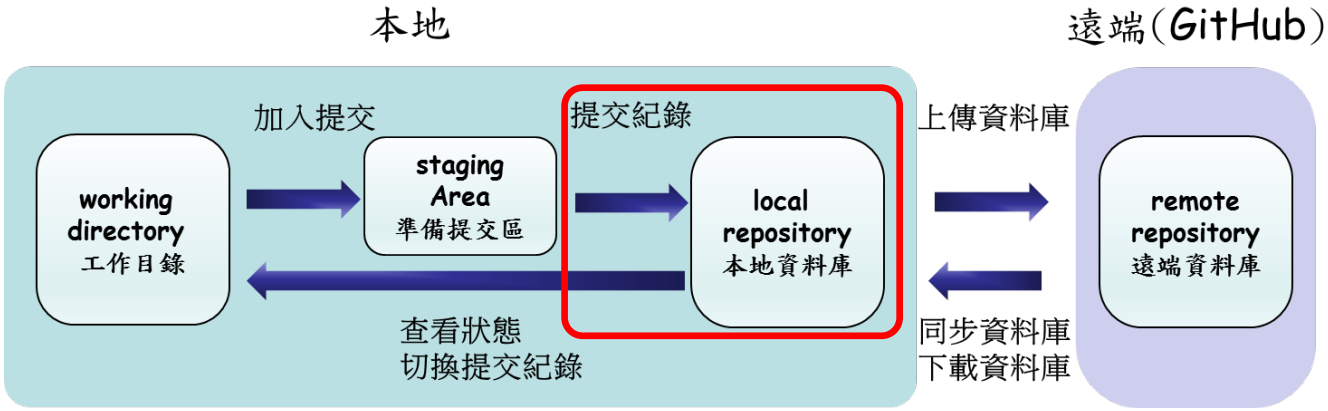
- Git add
  - 將資料的變更加入到準備提交區
  - 語法：git add "添加到準備區的檔案名稱"




```
Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab0 (master)
$ git add .

Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab0 (master)
$
```

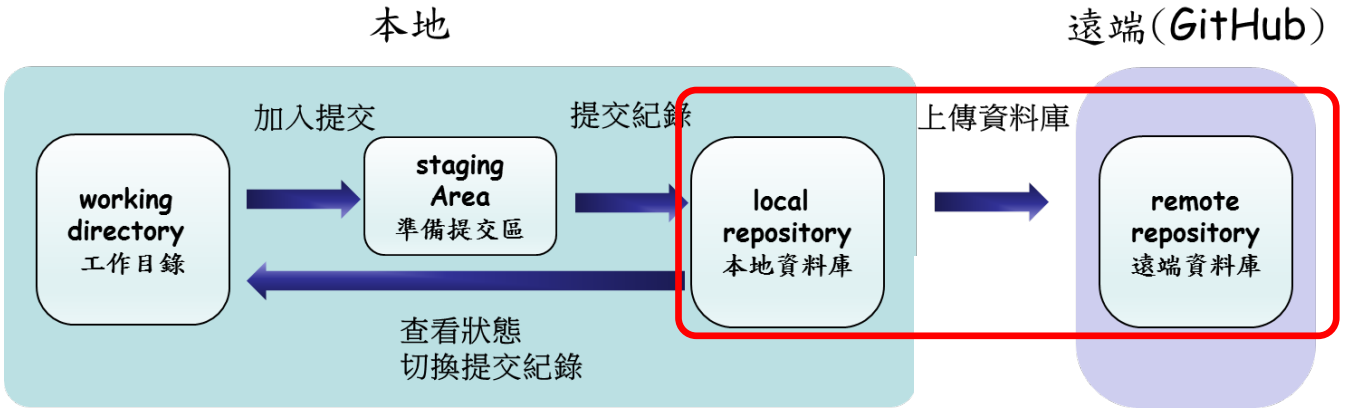
. 代表當前目錄下的所有資料變更

- Git commit
  - 準備提交區的變更寫入到本地資料庫中
  - 語法：git commit -m "本次提交的備註"



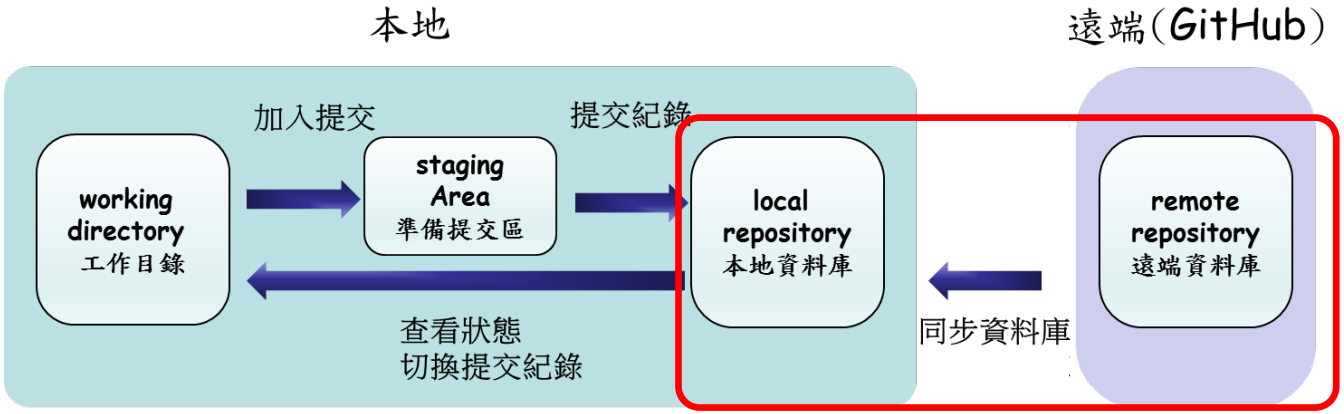

```
Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab0 (master)
$ git commit -m "Commit informations"
[master (root-commit) 4da7ccf] Commit informations
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 asdf

Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab0 (master)
$
```

- Git push
  - 將本地資料庫的資料同步到遠端資料庫
  - 語法：git push 或 git push origin 「分支名稱」



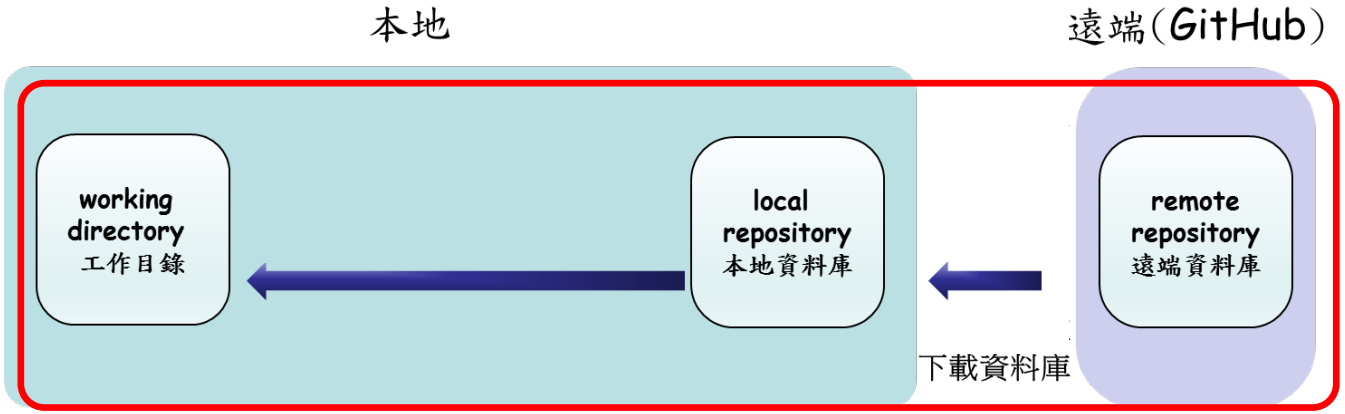

```
Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab1 (master)
$ git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 274 bytes | 274.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/davidjaw/Ch0-Lab-1.git
   29c6bed..497653d  master -> master

Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab1 (master)
$
```

- Git pull
  - 將遠端資料庫的資料同步到本地資料庫
  - 語法：git pull 或 git pull origin 「分支名稱」




```
Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab1 (master)
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/davidjaw/Ch0-Lab-1
   497653d..13a4109  master     -> origin/master
Updating 497653d..13a4109
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)

Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop/Lab1 (master)
$
```

- Git clone
  - 將遠端資料庫的資料同步到本地資料庫
  - 語法：git clone「遠端資料庫網址」



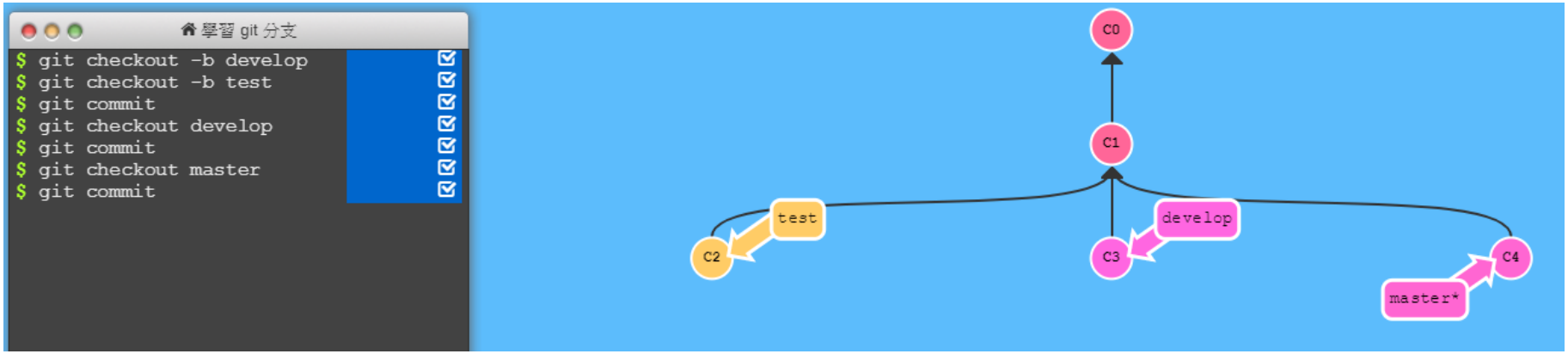

```
Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop
$ git clone https://github.com/davidjaw/Ch0-Lab-1.git
Cloning into 'Ch0-Lab-1'...
remote: Counting objects: 34, done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 34 (delta 7), reused 30 (delta 6), pack-reused 0
Unpacking objects: 100% (34/34), done.

Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop
$
```

Advance

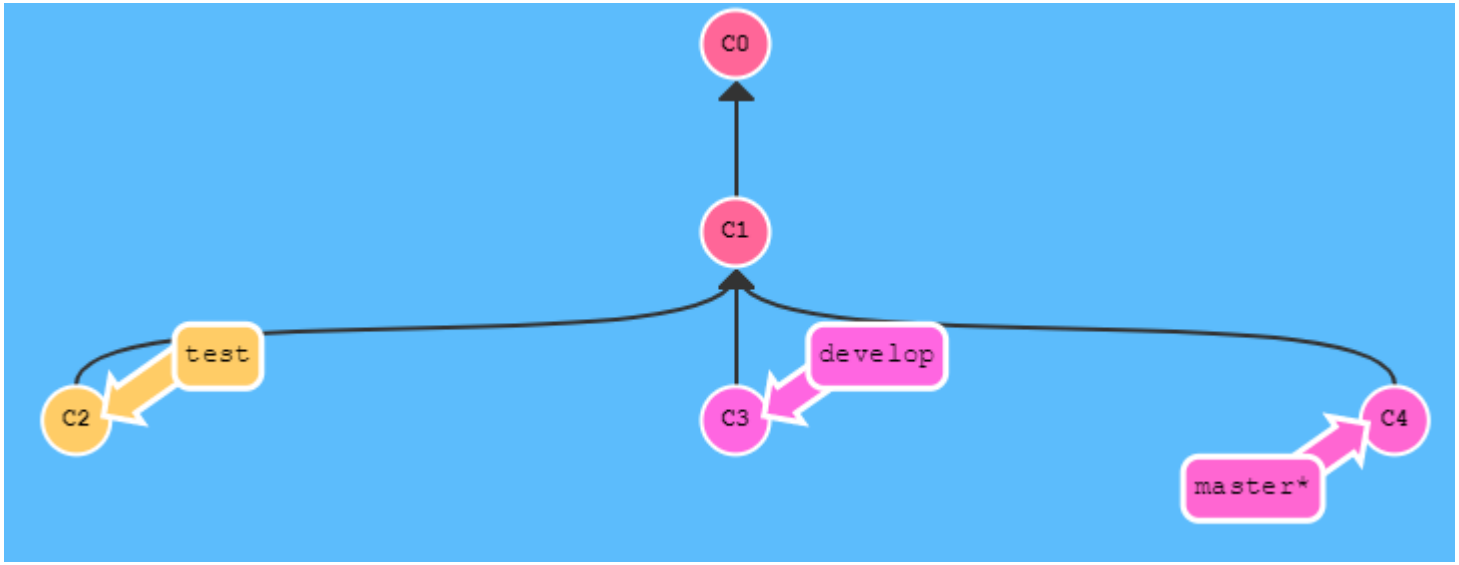
- Branch (分支)
  - 在大型專案中，產品會根據不同功能的發展方向做開發，例如：
    - 系統既有的程式加速
    - 系統未有的功能開發
    - 系統當前的程式除錯
  - 因此，多人開發時將會切分出不同的分支以及主分支來進行，如此不會因為其中誰更改了某個功能而導致同步後無法使用

- 視覺化學習網站：http://learngitbranching.js.org/
  - 此網站僅提供local指令練習
  - 有許多練習題可以線上做
  - Homework將會在這個網站上完成

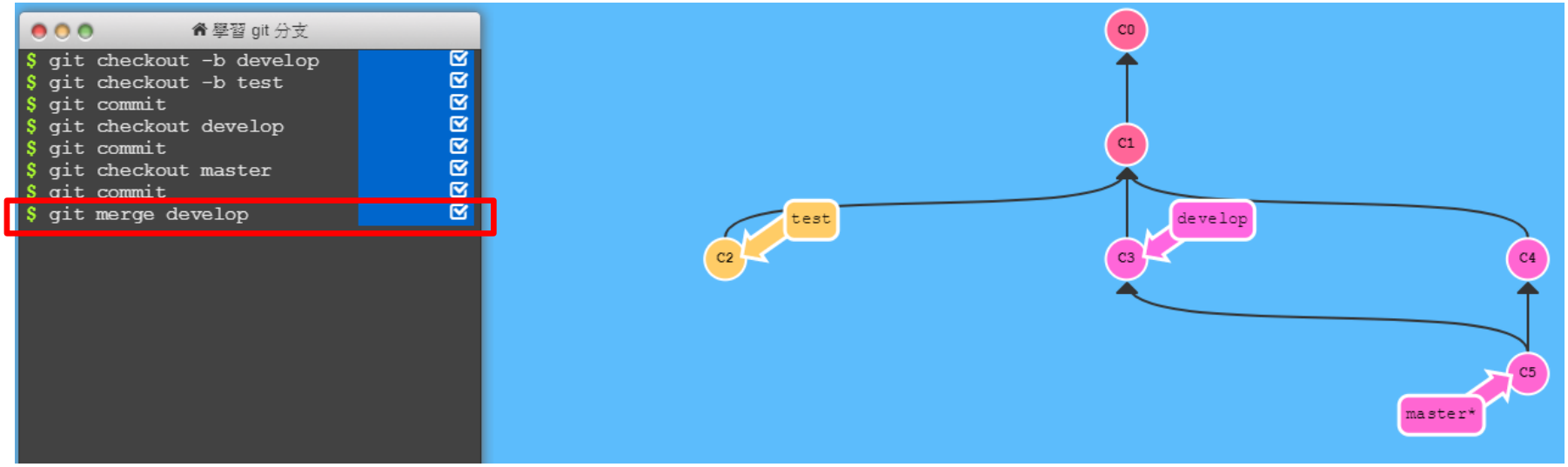
- Git checkout 「分支名稱」
  - 切換分支：將本地的工作環境切換到該分支上
- Git checkout -b 「分支名稱」
  - 創建分支：將本地的工作環境儲存到目前分支上
- Git branch
  - 查看本地端分支：將本地的分支顯示於終端機



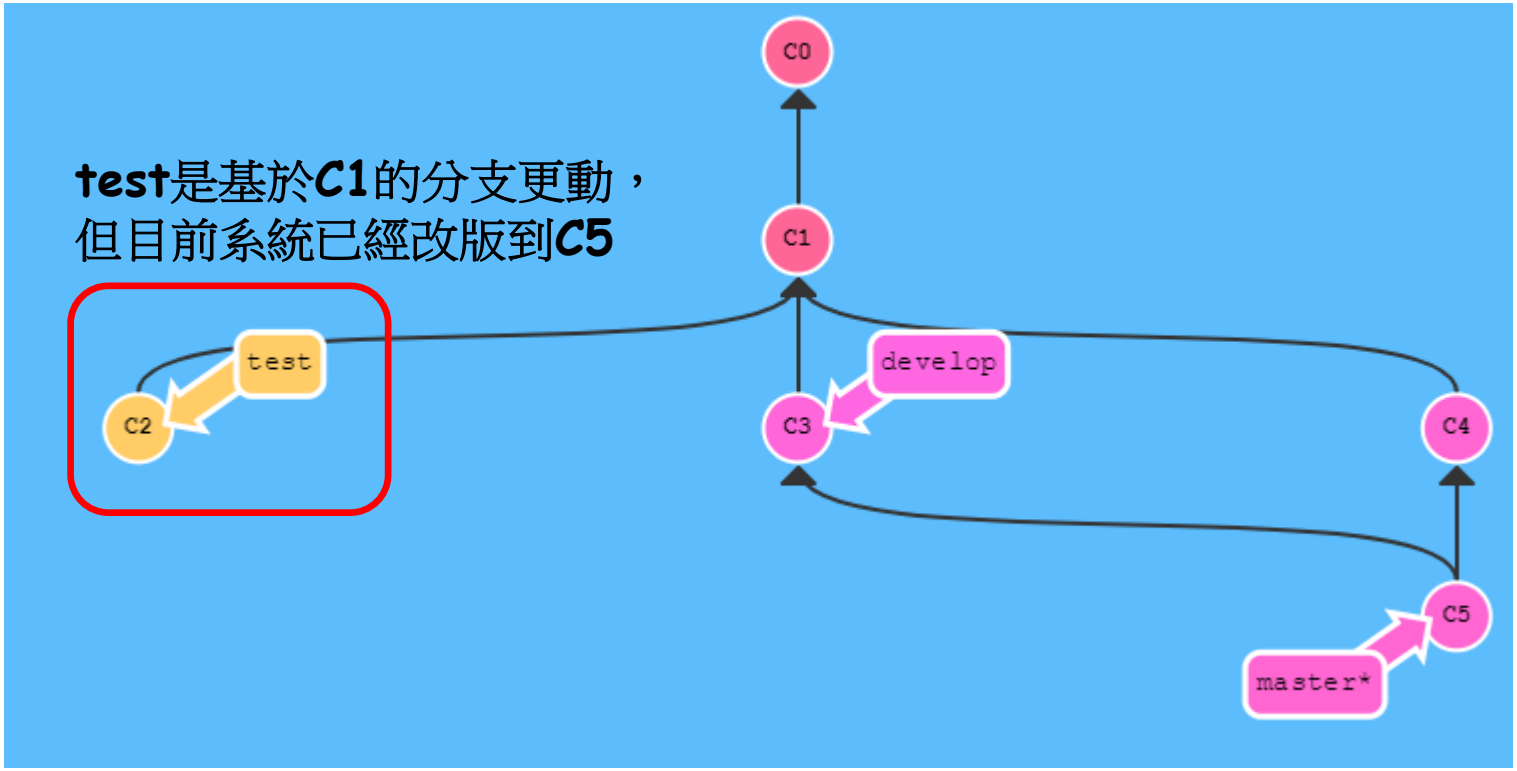
- Git merge 「分支名稱」
  - 將其他分支所做的改動整合到目前所在的分支上
    - 如下圖，目前分支為master, 我們要將develop所做的改動整合到master分支上，其指令為「git merge develop」



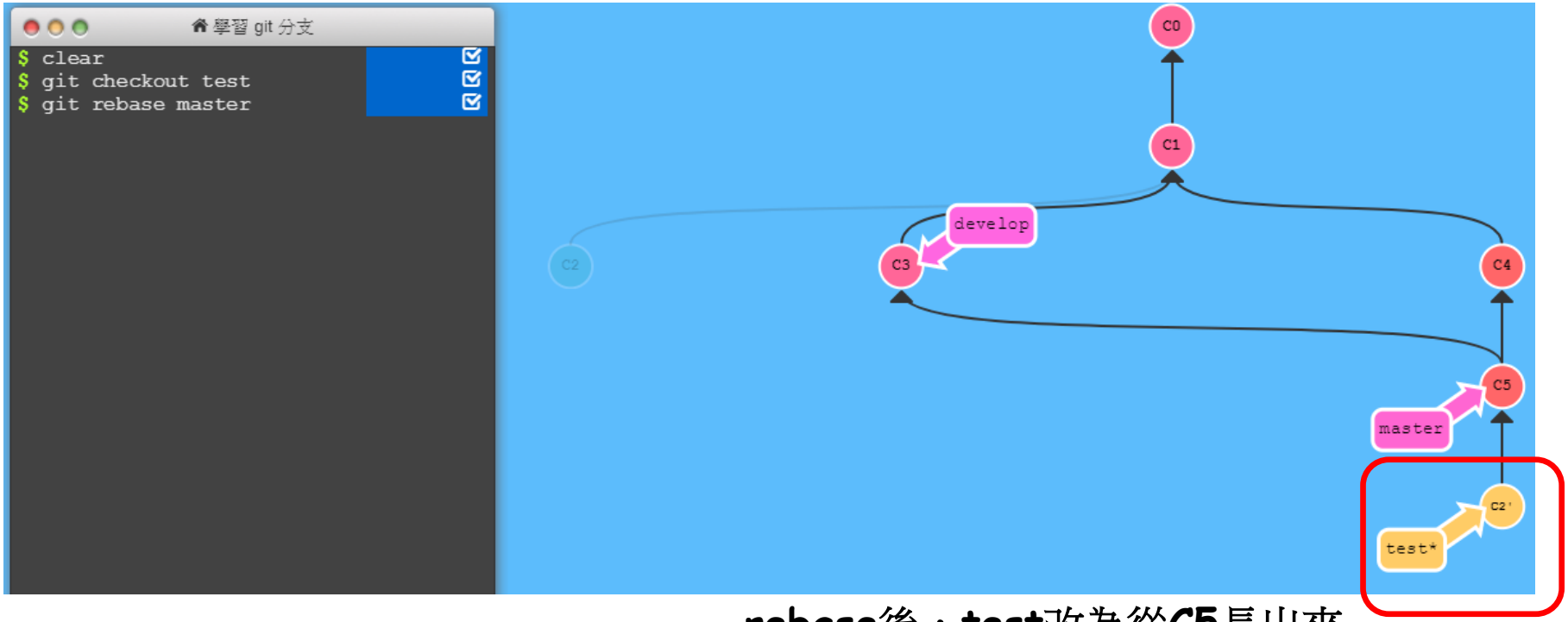
- Git merge 「分支名稱」
  - 將其他分支所做的改動整合到目前所在的分支上
    - 如下圖，目前分支為master, 我們要將develop所做的改動整合到master分支上



- Git rebase 「分支名稱」
  - 將目前基於A點的更動換成基於B點
    - 例如：你開發某個功能但尚未完成時，系統進行了改版，你必須要基於改版後的系統繼續進行開發才能確保最後上線時不會有問題



- Git rebase 「分支名稱」
  - 將目前基於A點的更動換成基於B點
    - 例如：你開發某個功能但尚未完成時，系統進行了改版，你必須要基於改版後的系統繼續進行開發才能確保最後上線時不會有問題



rebase後，test改為從C5長出來