



軟體工程課程： 版本控管 (Version Control)

海大資工 馬尚彬



單元大綱

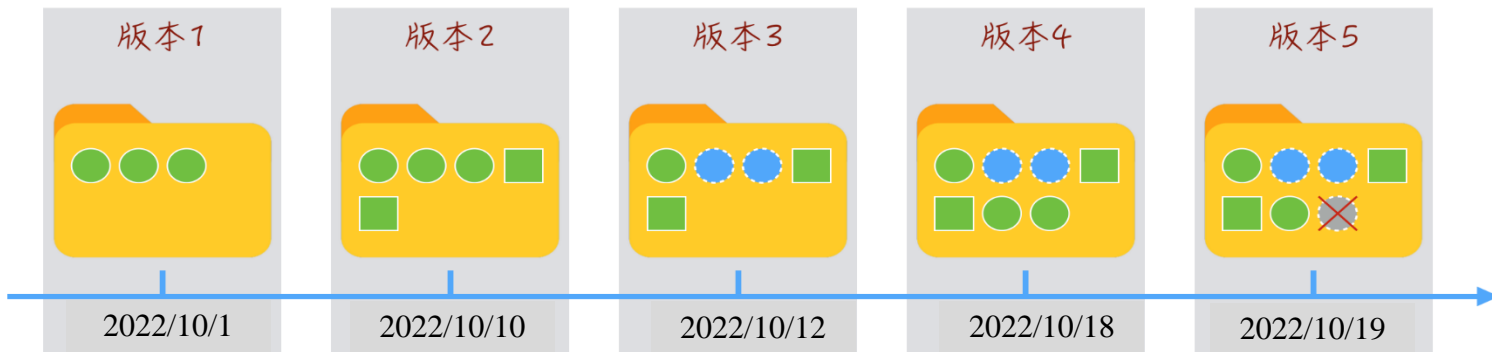
2

- 版本控管概念
- GIT基本指令
- GIT分支作法
- 使用遠端儲存庫GitHub
- GitHub Pull Request
- Git/GitHub開發流程

甚麼是GIT?

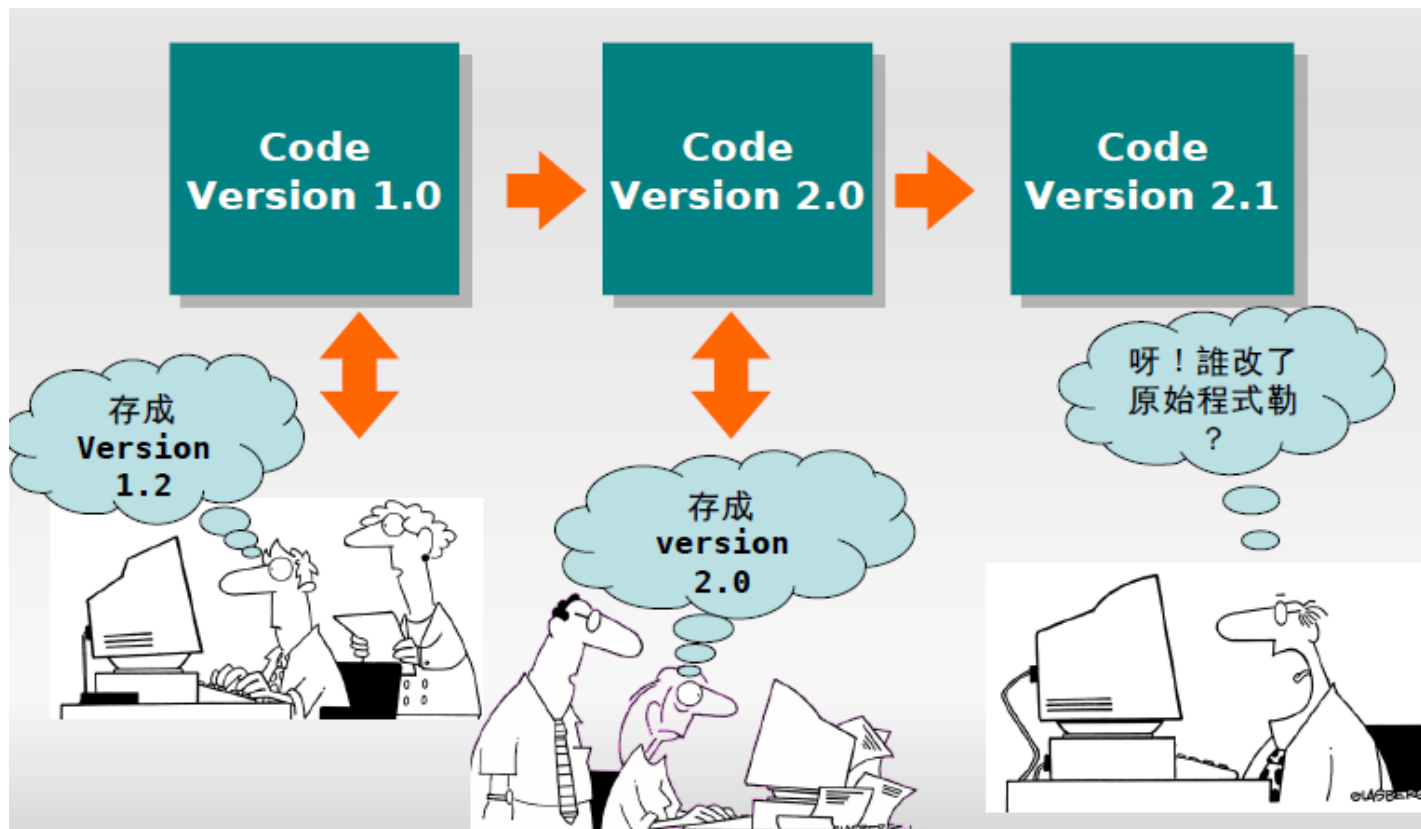
3

- Git 是一種分散式版本的版本控制(Version Control)系統
- 甚麼是版本？
 - ▣ 你的(軟體)專案，不管是新增或刪除檔案，亦或是修改檔案內容，都稱之為一個「版本」
 - ▣ 「版本控制系統」，會記錄這些所有的狀態變化，並且可以像搭乘時光機一樣，隨時切換到過去某個「版本」時候的狀態。



協同開發的問題

4



資料來源:自由軟體鑄造場王家薰，如何使用OpenSource的工具-協助軟體的開發及快速架站

Git的優點

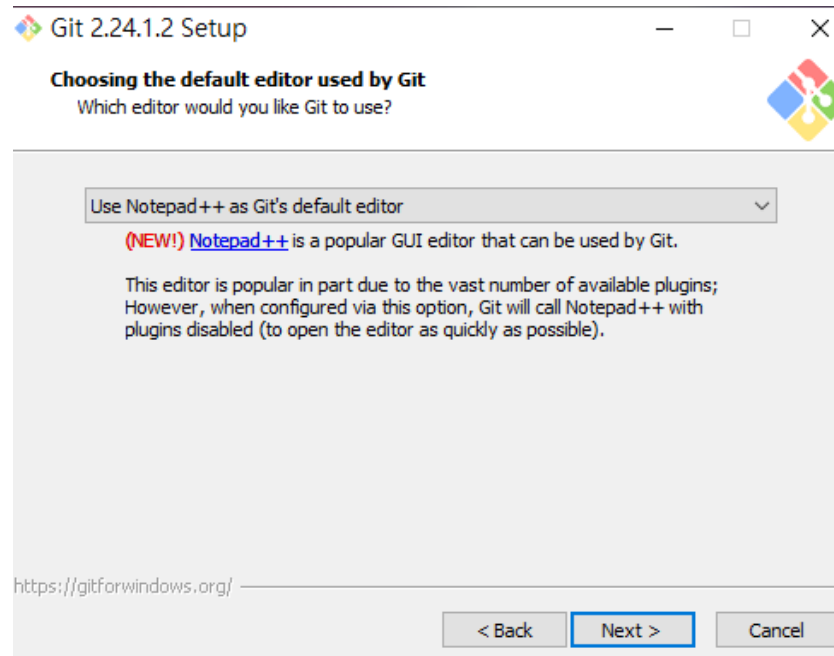
5

- 1. 免費、開源
 - ▣ Git 是由 Linux 核心的作者 Linus Torvalds 在 2005 年為了管理 Linux 核心程式碼，僅花了 10 天所開發出來的
- 2. 速度快、檔案體積小
 - ▣ Git 特別的設計，在於它並不是記錄版本的差異，而是記錄檔案內容的「快照」（snapshot）。
- 3. 分散式系統
 - ▣ Git 是一款分散式的版控系統（Distributed Version Control）。
 - ▣ 大多的 Git 操作也都是在自己電腦本機就可以完成。

Git 系統安裝

6

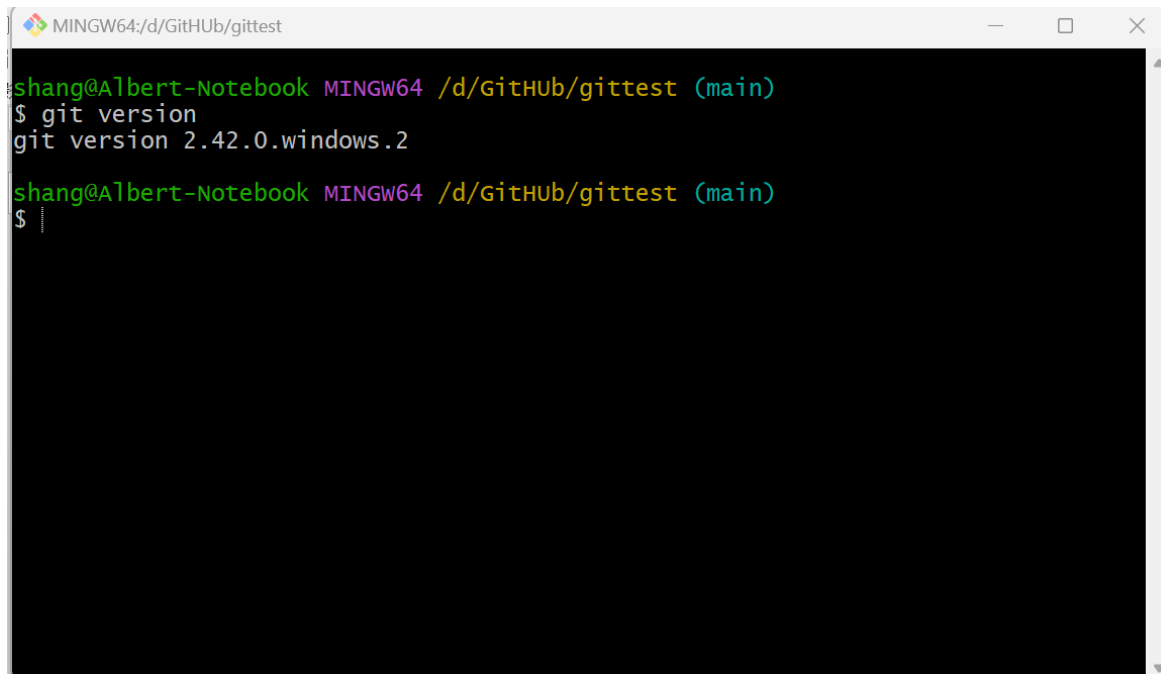
- 請到官方網站下載合適的Git版本：
- <https://git-scm.com/download/win>
- 安裝過程中可以更換預設編輯器為Notepad++或其他你熟悉的編輯器



Git Bash

7

- 安裝完成之後，請啟動「Git Bash」
 - ▣ Windows系統版本模擬了一個在Linux世界的Bash



```
MINGW64:/d/GitHub/gittest
shang@Albert-Notebook MINGW64 /d/GitHub/gittest (main)
$ git version
git version 2.42.0.windows.2
shang@Albert-Notebook MINGW64 /d/GitHub/gittest (main)
$ |
```

可以右鍵選[Options]->[Text]
改字型(Font)大小

可以試打指令看看：
git version

常用終端機命令列指令

8

指令	說明
cd	切換目錄
pwd	取得目前所在的位置
ls	列出目前的檔案列表
mkdir	建立新的目錄
touch	建立檔案
cp	複製檔案
mv	移動檔案
rm	刪除檔案
clear	清除畫面上的內容

設定Git

9

- 要開始使用 Git，首先要設定使用者的 Email 信箱以及使用者名稱：

```
Albert@Albert-HP MINGW64 /e/myprj  
$ git config --global user.name "Albert Ma"
```

```
Albert@Albert-HP MINGW64 /e/myprj  
$ git config --global user.email "albert@ntou.edu.tw"
```

- 可以看看Git的目前設定

```
Albert@Albert-HP MINGW64 /e/myprj  
$ git config --list
```

(移除設定：\$ git config --global --unset XX.YY)

(Albert@Albert-HP是老師的帳號與機器名稱)

新增與初始Repository

10

- 建立目錄
 - ▣ Git預設工作目錄是在C:\Users\{你的名字}
 - ▣ 可以先以檔案總管建立專案目錄(用命令列也可)
 - ▣ 再切換至專案目錄(如底下範例是simple專案目錄)
- 使用 *git init* 指令初始化這個目錄
 - ▣ 讓 Git 開始對這個目錄進行版本控制

```
Albert@Albert-HP MINGW64 ~  
$ cd simplegame
```

```
Albert@Albert-HP MINGW64 ~/simplegame  
$ git init  
Initialized empty Git repository in  
C:/Users/Albert/simplegame/.git/
```

查看目錄狀態

11

- *git status*指令: 查詢現在這個目錄的「狀態」。

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
```

```
$ git status
```

```
On branch master
```

```
No commits yet
```

```
nothing to commit (create/copy files and use "git add" to track)
```

- 若我們把檔案複製到這個目錄，再執行*git status*
 - ▣ 會多看到「Untracked files」資訊(未被「追蹤」)

將檔案交給Git追蹤₁

12

- 讓 Git 開始「追蹤」檔案：
 - ▣ 用 `git add` 指令後面加上檔案名稱
 - ▣ 可再執行 `git status` 看狀態

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git add Role.java
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git status
On branch master
```

```
No commits yet
```

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

剛才那個檔案從 Untracked 變成 new file 狀態了。表示這個檔案已經被安置到暫存區(Staging Area，也被稱為index)。

將檔案交給Git追蹤₂

13

- 使用萬用字元 (wildcard character) :
 - ▣ `$ git add *.java`
 - ▣ `$ git add *.*` (目前目錄層全部檔案)
- `--all` 參數 (把全部的檔案加到暫存區) :
 - ▣ `$ git add --all` (連子目錄都會加入)

把暫存區的內容提交到倉庫

14

□ *git commit*指令

- ▣ 讓暫存區的內容永久的存下來(才算正式進入到版本控管)

```
Albert@Albert-HP MINGW64 ~/simplegame  
(master)  
$ git commit -m "init commit"  
[master (root-commit) f34a359] init commit  
1 file changed, 57 insertions(+)  
create mode 100644 Role.java
```

▣ -m 參數:

- 在 Commit 的時候，要以-m參數輸入訊息(message)。
- 主要的目的就是告訴你自己以及其它人「這次的修改做了什麼」。
- 如果沒有給，Git會開預設的編輯器(如Notepad++)讓你寫訊息。
- 如果檔案中還是沒加訊息，git不會讓commit成功執行。

約定式提交(Conventional Commit)

15

- Conventional Commit: 提供一些簡單的條件集合用於建立明確的提交歷史
- Commit Types: feat, fix, chore, docs, refactor, test, etc.
- 範例：

feat: allow provided config object to extend other configs

BREAKING CHANGE: `extends` key in config file is now used for extending other config files

fix: correct minor typos in code

see the issue for details on the typos fixed

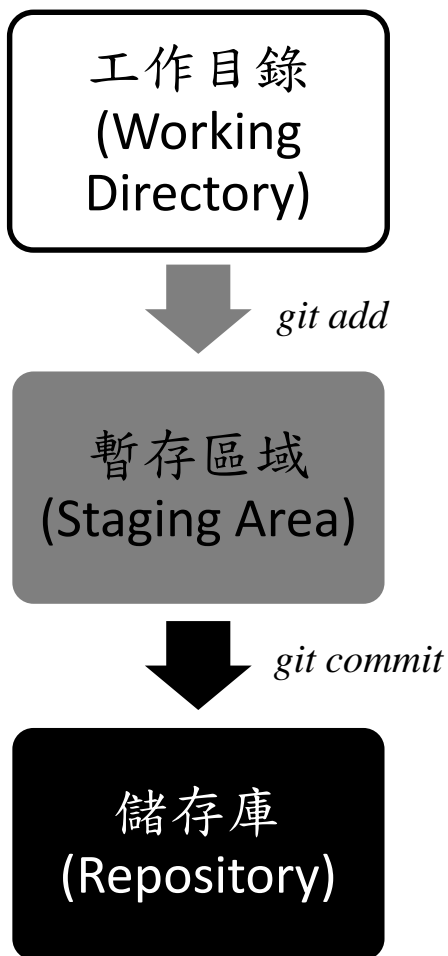
closes issue #12

<https://www.conventionalcommits.org/en/v1.0.0/>

<https://www.conventionalcommits.org/zh-hant/v1.0.0-beta.4/>

工作區、暫存區與儲存庫¹

16



- Git有「工作目錄(Working Directory)」、「暫存區(Staging Area)」、「儲存庫(Repository)」三個區塊。
 - 透過不同的 Git 指令，可以把檔案移往不同的區域。
 - 走完才是完整流程。
- git add 指令把檔案從工作目錄移至暫存區（或索引）。
- git commit 指令會把暫存區的內容移至儲存庫。

工作區、暫存區與儲存庫₂

17

□ 一定要二段式嗎？

- ▣ 你可以在 Commit 的時候多加一個 -a 的參數，縮短這個流程：

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git commit -a -m "commit again"
[master 983fbbb] commit again
1 file changed, 1 insertion(+), 1 deletion(-)
```

- ▣ Add與commit的類比：先把要回收的東西逐一放到推車上(add)，等累積到一定份量後再拿去回收場(commit)。

檢視Git紀錄₁

18

- `git log` 指令：檢視紀錄 (誰、何時、做了甚麼)

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
```

```
$ git log
```

```
commit 983fbbb6a4af820ddd7276121e793647a7be1d86  
(HEAD -> master)
```

```
Author: Albert Ma <albert@ntou.edu.tw>
```

```
Date:    Wed Dec 18 10:53:20 2019 +0800
```

```
commit again
```

```
commit f34a359493addd876c1c45010afbb07d8da244fc
```

```
Author: Albert Ma <albert@ntou.edu.tw>
```

```
Date:    Wed Dec 18 10:29:12 2019 +0800
```

```
init commit
```

983fbbb6a4af820ddd7276121e793647a7be1d86是使用SHA-1 (Secure Hash Algorithm 1) 演算法所計算的結果，每個Commit都有一個這樣的值。

檢視Git紀錄₂

19

□ *git log*的結果可以更精簡

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git log --oneline --graph
* 983fbbb (HEAD -> master) commit again
* f34a359 init commit
```

刪除檔案¹

20

- 如果把檔案自檔案總管刪除了：

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
```

```
$ git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add/rm <file>..." to update what will be committed)
```

```
  (use "git restore <file>..." to discard changes in working directory)
```

```
    deleted:    Role.java
```

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

可以看到 *Role.java* 這個檔案目前的狀態是 *deleted*

刪除檔案₂

21

- 接著還是一樣進行add和commit:

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git add Role.java
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git commit -m "delete Role.java"
[master 01463f9] delete Role.java
1 file changed, 57 deletions(-)
delete mode 100644 Role.java
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git status
On branch master
nothing to commit, working tree clean
```

刪除檔案₃

22

- *git rm* 指令：讓git幫忙刪，會直接把變更送到暫存區

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git rm Role.java
rm 'Role.java'
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:      Role.java
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git commit -m "delete Role.java"
[master 8081fa8] delete Role.java
1 file changed, 57 deletions(-)
delete mode 100644 Role.java
```

改檔案名稱

23

- `git mv`指令可以改名字，也會把變更送到暫存區

```
Albert@Albert-HP MINGW64
```

```
~/simplegame (master)
```

```
$ git mv Role.java Player.java
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
```

```
$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)
```

```
renamed:    Role.java -> Player.java
```

查看是誰做的

24

- *git blame* 可以看出每一行程式是誰寫的

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
```

```
$ git blame Role.java
```

```
b5c896d7 Player.java (Albert Ma 2020-10-18 16:55:14 +0800 1)
```

```
public class Role {
```

```
b5c896d7 Player.java (Albert Ma 2020-10-18 16:55:14 +0800 2)
```

```
private String name; // it is name
```

```
b5c896d7 Player.java (Albert Ma 2020-10-18 16:55:14 +0800 3)
```

```
private int hp;
```

```
b5c896d7 Player.java (Albert Ma 2020-10-18 16:55:14 +0800 4)
```

```
private int offense;
```

```
b5c896d7 Player.java (Albert Ma 2020-10-18 16:55:14 +0800 5)
```

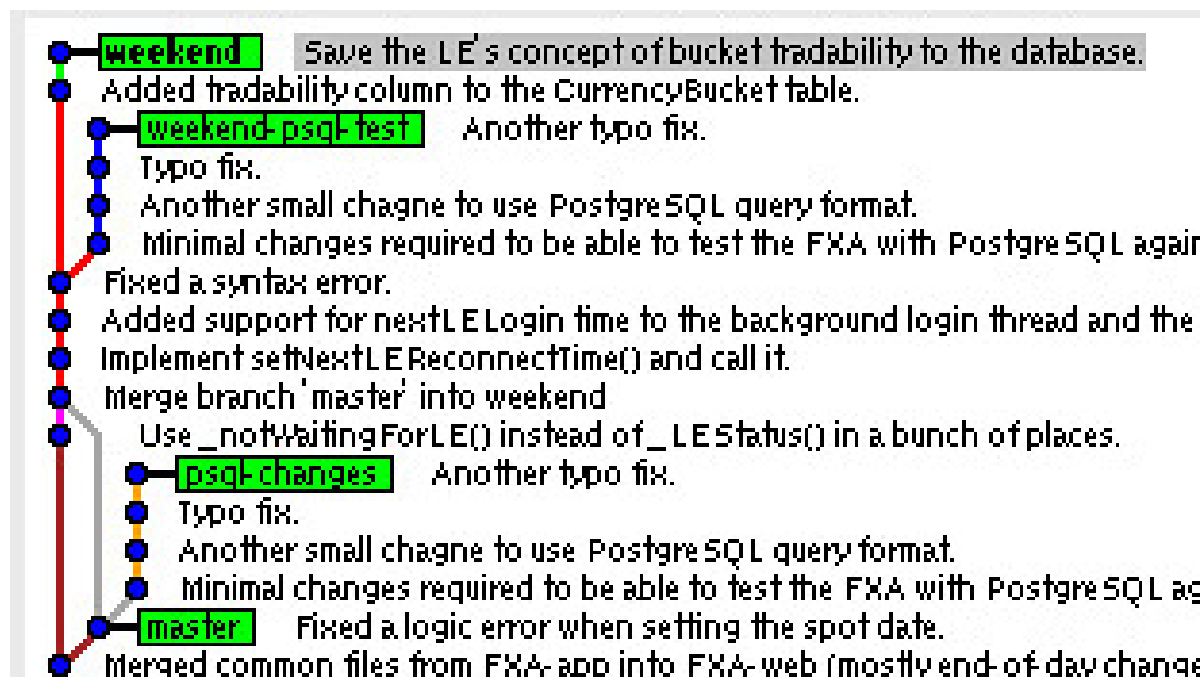
```
private int defense;
```

```
...
```


內建圖形化介面: gitk

25

- 可輸入 *gitk* 觀看圖形化的版本紀錄。



修改 Commit 紀錄

26

- 可使用 `--amend` 參數來進行 Commit，去修改 commit 訊息

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git commit --amend -m "intial commit again"
[master b5c896d] intial commit again
Date: wed Dec 18 16:55:14 2019 +0800
1 file changed, 57 insertions(+)
create mode 100644 Player.java
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git log
commit b5c896d7a0295f969400bed0ff00258d486fc2bc (HEAD ->
master)
Author: Albert Ma <albert@ntou.edu.tw>
Date:   wed Dec 18 16:55:14 2019 +0800
```

intial commit again

若不小心進入到vim畫面，可按[esc] 離開編輯模式，並輸入 :wq 存檔離開

回到過去的Commit

27

- 如果想要取消最後這次的 Commit，可使用 *git revert* (檔案內容會回到過去)
 - ▣ 最常使用指令：*git revert HEAD --no-edit*

```
Albert@Albert-HP MINGW64 ~/git
(master)
$ git log --oneline
e975cb8 (HEAD -> master) 3rd version
5eae413 2nd version
9702f72 1st version
```

```
Albert@Albert-HP MINGW64 ~/git
(master)
$ git revert HEAD --no-edit
[master 2ed4f76] Revert "3rd version"
Date: Mon May 4 13:22:19 2020 +0800
1 file changed, 1 insertion(+), 1
deletion(-)
```

```
Albert@Albert-HP MINGW64 ~/git
(master)
$ git log --oneline
2ed4f76 (HEAD -> master) Revert
"3rd version"
e975cb8 3rd version
5eae413 2nd version
9702f72 1st version
```

```
Albert@Albert-HP MINGW64 ~/git
(master)
$ git status
On branch master
nothing to commit, working tree
clean
```

拆掉Commit

28

- 可用 *git reset* 拆掉Commit (目前的檔案內容不會變)
 - ▣ 把目前的狀態設定成某個指定的 Commit 的狀態，通常適用於尚未推出去的 Commit。
 - ▣ 從目前的版本往前退一版: `$ git reset HEAD^`
 - ▣ 從目前的版本往前退兩版: `$ git reset HEAD~2`

```
Albert@Albert-HP MINGW64
~/simplegame (master)
$ git log --oneline
d7e0f69 (HEAD -> master) 3rd
commit
9fd93f5 2nd version
e30c139 1st version
```

```
Albert@Albert-HP MINGW64
~/simplegame (master)
$ git reset master^
Unstaged changes after reset:
M    role.java
```



```
Albert@Albert-HP MINGW64
~/simplegame (master)
$ git log --oneline
9fd93f5 (HEAD -> master) 2nd version
e30c139 1st version
```

```
Albert@Albert-HP MINGW64 ~/git
(master)
$ git status
On branch master
Changes not staged for commit:
...
```

為什麼要使用分支(Branch)？

29

- 當開始越來越多同伴一起在同一個專案工作的時候，彼此的Commit會互相影響。
- 新分支可讓與主分支暫時隔離
 - ▣ 增加新功能，或是修正 Bug，或是想實驗看看某些新的做法。
 - ▣ 待做完確認沒問題之後再合併回來，不會影響正在運行的產品線。

查看分支

30

- `git branch` 指令可以查看目前分支
 - ▣ 一開始應該只有 master

```
Albert@Albert-HP MINGW64 ~/simplegame  
(master)  
$ git branch  
* master
```

分支新增、改名與刪除

31

- *git branch* 指令、以及運用 *-m* 與 *-d* 參數

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git branch hotfix
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git branch -m hotfix feature
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git branch
feature
* master
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git branch -d feature
Deleted branch feature (was b5c896d).
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git branch
* master
```

切换分支

32

- *git checkout* 指令可以切换分支

```
Albert@Albert-HP MINGW64 ~/simplegame  
(master)  
$ git branch hotfix
```

```
Albert@Albert-HP MINGW64 ~/simplegame  
(master)  
$ git checkout hotfix  
Switched to branch 'hotfix'
```


合併分支

33

- 若新分支已經新增多個commit，要併回原分支，要使用 *git merge* 指令

```
Albert@Albert-HP MINGW64 ~/simplegame  
(hotfix)  
$ git checkout master  
Switched to branch 'master'
```

```
Albert@Albert-HP MINGW64 ~/simplegame  
(master)  
$ git merge hotfix  
Updating 0defb56..4be2cad  
Fast-forward  
  Role.java | 4 +++-  
  1 file changed, 3 insertions(+), 1  
  deletion(-)
```

合併發生衝突該如何解決¹

34

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git merge hotfix
Auto-merging Role.java
CONFLICT (content): Merge conflict in Role.java
Automatic merge failed; fix conflicts and then
commit the result.
```

衝突發生了!

```
public class Role {
    private String name; // it is name
    private int hp;
    private int offense;
    private int defense;
```

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

```
<<<<<< HEAD (Current Change)
```

```
    private int status;
    private String dsecription;
```

```
=====
```

```
    private String description;
```

```
>>>>>> hotfix (Incoming Change)
```

Git會幫忙在程式碼中標示衝突的部分

合併發生衝突該如何解決₂

35

- 解決衝突後(通常需要雙方討論)重新add與commit

```
public class Role {  
    private String name; // it is name  
    private int hp;  
    private int offense;  
    private int defense;  
    private int status;  
    private String description;
```

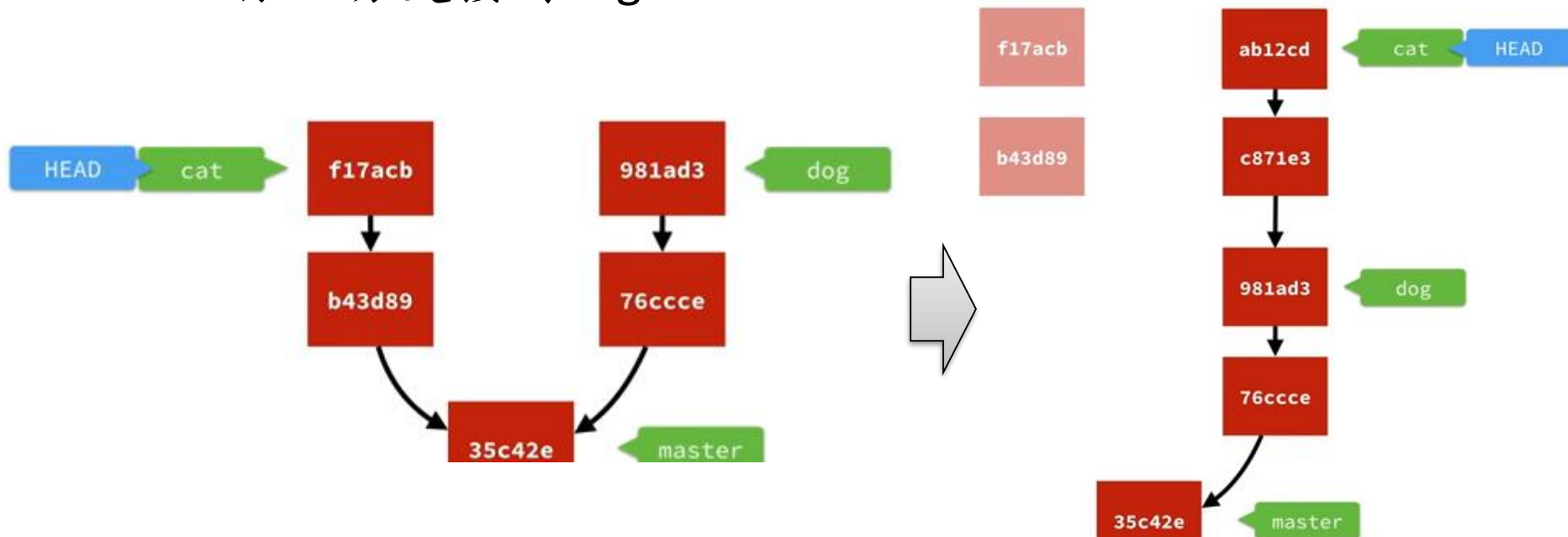
```
Albert@Albert-HP MINGW64 ~/simplegame  
(master|MERGING)  
$ git add *.java
```

```
Albert@Albert-HP MINGW64 ~/simplegame  
(master|MERGING)  
$ git commit -m "conflict solved"  
[master ed43ad7] conflict solved
```

另一種合併分支方式: rebase

36

- git rebase dog (目前在cat分支)
 - ▣ cat分支使用 dog 分支當做我新的參考基準
 - ▣ 將cat分支接到dog之上



標籤(tag)是什麼？

37

- 在 Git，標籤（tag）是一個指向某一個 Commit 的指標。
- 通常在開發軟體有完成特定的里程碑，例如軟體版號 0.1.0 或是 beta-release，可以使用標籤註記。
 - ▣ 可參考語意化標籤：<https://semver.org/lang/zh-TW/>

有附註(Annotated)的標籤

38

- `git tag`指令可以新增標籤
- `git show`指令可顯示某個標籤的詳細資訊

```
Albert@Albert-HP MINGW64  
~/simplegame (master)
```

```
$ git tag beta -a -m "it is  
able to be tested"
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)  
$ git show beta  
tag beta
```

```
Tagger: Albert Ma <albert@ntou.edu.tw>  
Date:   Wed Dec 18 18:05:14 2019 +0800
```

```
it is able to be tested
```

```
commit  
ed43ad75d3109c12a0b5aae5212795aa7d375e52  
(HEAD -> master, tag: beta)
```

```
Merge: 17a7970 d2285b6  
Author: Albert Ma <albert@ntou.edu.tw>  
Date:   Wed Dec 18 17:59:47 2019 +0800
```

```
...
```

GitHub 是什麼？

39

- <https://github.com/>
- 目前全球最大的 Git Server。
- 可以幫忙貢獻其它人的專案，其它人也可以幫忙我們的專案。
- 是開發者最好的履歷。

創建GitHub Repository

40

- 請先註冊好帳號，然後選擇[Start a Project]
- 填完Repository name，按下[Create Repository]即可。

Create a new repository

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

[Import a repository.](#)

Owner



albertma2020 ▾

Repository name *

/ simplerpg



Great repository names are short and memorable. Need inspiration? How about **cautious-waddle**?

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

設定SSH (Secure Shell)

41

- Generating a new SSH key
 - ▣ Open Git Bash
 - `$ ssh-keygen -t ed25519 -C your_email@example.com`
- Copying the SSH public key to your clipboard
 - ▣ 使用者目錄\ssh，以編輯器打開.pub檔，再複製全部內容
- Adding a new SSH key to your GitHub account
 - ▣ <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

Git Push₁

42

□ *git remote* 指令

```
Albert@Albert-HP MINGW64 ~/simplerpg (master)
$ git remote add origin
https://GitHub.com/albertma2020/simplerpg.git
```

(用HTTPS、帳密登入GitHub，目前已被禁止)

```
Albert@Albert-HP MINGW64 ~/simplerpg (master)
$ git remote add origin
git@github.com:albertma2020/simplerpg
```

(用SSH的方式連結GitHub)

若需修改網址或名稱：<https://backlog.com/git-tutorial/tw/reference/remote.html>

Git Push₁

43

□ *git push* 指令

```
Albert@Albert-HP MINGW64 ~/simplerpg (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 223 bytes | 111.00
KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://GitHub.com/albertma2020/simplerpg.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch
'master' from 'origin'.
```

GitHub Login

GitHub
Login

albertma2020

.....



Login



Cancel

Don't have an account? [Sign up](#)
[Forgot your password?](#)

Git Push₂

44

□ git remote add origin

- add 指令是指要加入一個遠端節點 (若改成set-url即可修改既有的設定)
- origin是遠端節點位置的代號

□ git push -u origin master

- 把master這個分支的內容，推向origin的位置。
- 在origin遠端 Server 上，如果master分支不存在，就建立一個。
- 但如果本來Server上就有master分支，便會讓master分支更新到最新的進度上。
- -u參數：
 - 它會指向並追蹤某個分支。
 - 通常 upstream 會是遠端 Server 上的某個分支。
 - 當下回執行 git push 指令而不加任何參數時，它就會預設推往 origin，並且把 master 這個分支推上去。

加入協作者

45

- 你可以於“Settings”->“Manage access” 加入其他開發者，才能協同開發。

Git Clone

46

- 另一位開發者可以先把專案`git clone`下來。(第一次拉專案請用`clone`，而非`pull`)

```
albert@Albert-Office MINGW64 ~/workspace
$ git clone
https://GitHub.com/albertma2020/simplerpg.git
Cloning into 'simplerpg'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 6 (delta 0), reused 6 (delta 0),
pack-reused 0
Unpacking objects: 100% (6/6), done.
```

- 接著一樣可以`git add`、`git commit`、`git push`
 - ▣ 若要用SSH進行push，記得改remote網址

Git Pull

47

□ *git pull* = *git fetch* + *git merge*

- ▣ Pull 指令會上線抓東西下來(Fetch)，並且更新本機的進度(Merge)。(發生在多人開發的情況下)

```
Albert@Albert-HP MINGW64 ~/simplerpg (master)
```

```
$ git pull
```

```
remote: Enumerating objects: 4, done.
```

```
remote: Counting objects: 100% (4/4), done.
```

```
remote: Compressing objects: 100% (2/2), done.
```

```
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

```
Unpacking objects: 100% (3/3), done.
```

```
From https://GitHub.com/albertma2020/simplerpg
```

```
8215dc9..6454750 master -> origin/master
```

```
Updating 8215dc9..6454750
```

```
Fast-forward
```

```
test.txt | 1 +
```

```
1 file changed, 1 insertion(+)
```

```
create mode 100644 test.txt
```

為何有時無法Push?₁

48

- 通常這個狀況會發生在多人一起開發的時候
 - ▣ 別人在你之前Push新的Commit上去了。

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git push
To https://GitHub.com/albertma2020/simplerpg.git
! [rejected]          master -> master (fetch first)
error: failed to push some refs to
'https://GitHub.com/albertma2020/simplerpg.git'
hint: Updates were rejected because the remote contains work
that you do
hint: not have locally. This is usually caused by another
repository pushing
hint: to the same ref. You may want to first integrate the
remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --
help' for details.
```


為何有時無法Push?

49

□ 你必須先Pull再Push：

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://GitHub.com/albertma2020/simplerpg2
   908075f..8f8d69b  master    -> origin/master
Merge made by the 'recursive' strategy.
 GameGUI.java | 135 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 135 insertions(+)
 create mode 100644 GameGUI.java
```

```
Albert@Albert-HP MINGW64 ~/simplegame (master)
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.31 KiB | 267.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://GitHub.com/albertma2020/simplerpg2.git
   8f8d69b..bec4b5c  master -> master
```

Pull Request (PR)

50

- 如何提供貢獻到GitHub開源專案？
 - ▣ 先以分岔(Fork)指令複製原始的專案到自己的 GitHub 帳號底下。
 - ▣ 對複製回來的專案進行修改(你會有完整的權限)與 Commit，並記得Push。
 - ▣ 接下來發Pull Request申請，讓原專案作者知道你有新開發版本。
 - ▣ 原作者核可此申請，接著他就會你做的這些新版本修改合併(Merge)到原始的專案裡。
- 一般團隊開發亦可採用此模式。

GitHub Fork₁

51

- 貢獻者連結原始專案，選擇複製(Fork)專案

The screenshot shows the GitHub interface for the repository 'albertma2020 / PullRequestDemo'. The repository is public. The 'Fork' button is circled in red, indicating the action to be taken. The repository has 0 stars, 1 watch, and 0 forks. The 'About' section states 'No description, website, or topics provided.' The 'Releases' section is visible at the bottom.

Search or jump to... Pulls Issues Marketplace Explore

albertma2020 / PullRequestDemo Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main Go to file Add file Code About

albertma2020 feat: first version 12 minutes ago 1

demo_querystring.js feat: first version 12 minutes ago

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

No description, website, or topics provided.

0 stars 1 watching 0 forks

Releases

GitHub Fork₂


52

□ 執行Create fork

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner *

 soselab2020 ▾

Repository name *


/ PullRequestDemo ✓

By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

☒ Copy the `main` branch only

Contribute back to albertma2020/PullRequestDemo by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

Create fork

GitHub Fork₃

53

□ Fork創建成功後，即可再進行後續開發

The screenshot shows the GitHub interface for a repository named 'PullRequestDemo' by user 'soselab2020'. The repository is public and is a fork of 'albertma2020 / PullRequestDemo', which is highlighted with a red circle. The repository has 0 stars, 0 watchers, and 1 fork. The main branch is 'main'. A recent commit by 'albertma2020' is shown with the message 'feat: first version'. The repository also has a file named 'demo_querystring.js'. A banner at the bottom encourages adding a README.

Search or jump to... / Pull requests Issues Marketplace Explore

soselab2020 / PullRequestDemo Public

Forked from albertma2020/PullRequestDemo

<> Code Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

This branch is up to date with albertma2020/PullRequestDemo:main.

Contribute Sync fork

albertma2020 feat: first version ffa375a 34 minutes ago 1 commit

demo_querystring.js feat: first version 34 minutes ago

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

About

No description, website, or topics provided.

0 stars

0 watching

1 fork

Releases

No releases published

Create a new release

Open Pull Request₁

54

- 貢獻者於Fork專案追加Commit後，可申請Compare&Pull Request

Search or jump to... / Pulls Issues Marketplace Explore

soselab2020 / PullRequestDemo Public

forked from albertma2020/PullRequestDemo

<> Code Pull requests Actions Projects Wiki Security Insights Settings

patch-1 had recent pushes 5 minutes ago

Compare & pull request

main Go to file Add file Code

This branch is 1 commit ahead of albertma2020:main. Contribute Sync fork

soselab2020 fix: add message 1 minute ago 2

demo_querystring.js fix: add message 1 minute ago

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

About No description, website, or topics provided. 0 stars 0 watching 1 fork

Releases No releases published Create a new release

Packages

Open Pull Request₂


55

- 貢獻者可填入相關說明，實際建立 Pull Request。

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

 base repository: albertma2020/PullRequestDemo


base: main

←

head repository: soselab2020/PullRequestDemo

compare: patch-1









✓ **Able to merge.** These branches can be automatically merged.



fix: add log


Write

Preview

H B I       @  

我追加了log機制，請考慮合併。

Attach files by dragging & dropping, selecting or pasting them.

☒ Allow edits by maintainers 

Create pull request

Helpful resources
[GitHub Community Guidelines](#)

Merge Pull Request₁

56

- 原作者可看到貢獻者提出之Pull Request。

The screenshot shows the GitHub interface for the repository `albertma2020 / PullRequestDemo`. The `Pull requests` tab is selected and highlighted with a red circle. Below the repository header, a notification banner states: "Label issues and pull requests for new contributors. Now, GitHub will help potential first-time contributors discover issues labeled with `good first issue`". Below this, the search bar shows filters `is:open is:pr`. A green button labeled "New pull request" is visible. The pull request list shows 1 Open and 0 Closed requests. The first pull request, titled `fix: add log`, is highlighted with a red circle. It was opened 3 minutes ago by `soselab2020`.

Search or jump to... / Pull requests Issues Marketplace Explore

albertma2020 / PullRequestDemo Public

Pin Unwatch 1 Fork 1 Star 0

Code Issues **Pull requests 1** Actions Projects Wiki Security Insights Settings

Label issues and pull requests for new contributors [Dismiss](#)

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with `good first issue`

Filters is:open is:pr Labels 9 Milestones 0 [New pull request](#)

Clear current search query, filters, and sorts

1 Open 0 Closed Author Label Projects Milestones Reviews Assignee Sort

fix: add log

#1 opened 3 minutes ago by soselab2020

Merge Pull Request₂

57

□ 原作者可選擇合併Pull Request。

albertma2020 / PullRequestDemo Public

<> Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

fix: add log #1 Edit <> Code

Open soselab2020 wants to merge 1 commit into albertma2020:main from soselab2020:patch-1

Conversation 0 Commits 1 Checks 0 Files changed 1 +2 -1

soselab2020 commented 5 minutes ago First-time contributor

我追加了log機制，請考慮合併。

fix: add log Verified 3f4308a

Add more commits by pushing to the patch-1 branch on soselab2020/PullRequestDemo.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

Reviewers Suggestions albertma2020 Request

Still in progress? Convert to draft

Assignees No one—assign yourself

Labels None yet

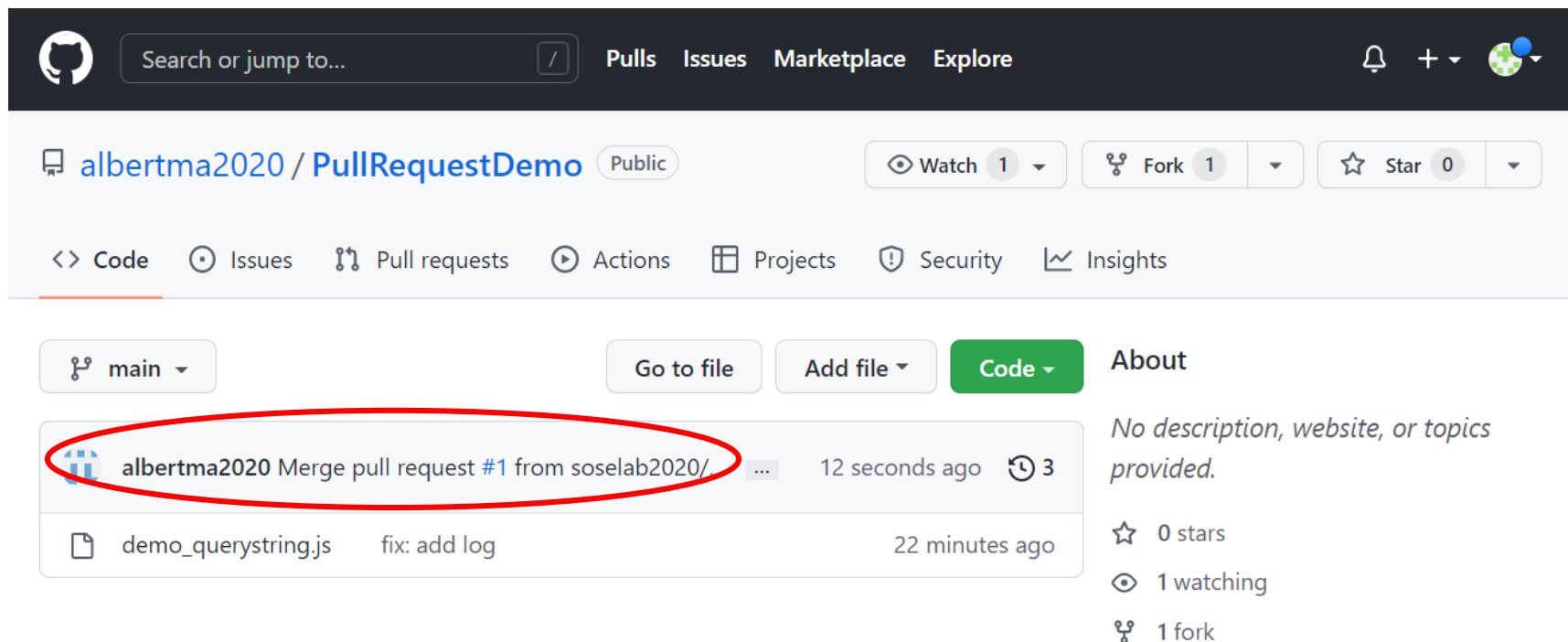
Projects None yet

Milestone No milestone

Merge Pull Request₃

58

- 最後原作者即可看到Pull Request內容已被成功合併至主專案。



The screenshot shows the GitHub interface for the repository `albertma2020 / PullRequestDemo`. The repository is public. The navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Security, and Insights. The main content area shows a list of pull requests. The first pull request, titled `albertma2020 Merge pull request #1 from soselab2020/...`, is circled in red. This pull request was merged 12 seconds ago and has 3 commits. Below the pull request list, the file `demo_querystring.js` is shown with the commit message `fix: add log` and a timestamp of 22 minutes ago. On the right side, the 'About' section indicates that no description, website, or topics were provided. The repository has 0 stars, 1 watching, and 1 fork.

Search or jump to... Pulls Issues Marketplace Explore

albertma2020 / PullRequestDemo Public

Watch 1 Fork 1 Star 0

Code Issues Pull requests Actions Projects Security Insights

main Go to file Add file Code About

albertma2020 Merge pull request #1 from soselab2020/... 12 seconds ago 3

demo_querystring.js fix: add log 22 minutes ago

No description, website, or topics provided.

0 stars 1 watching 1 fork

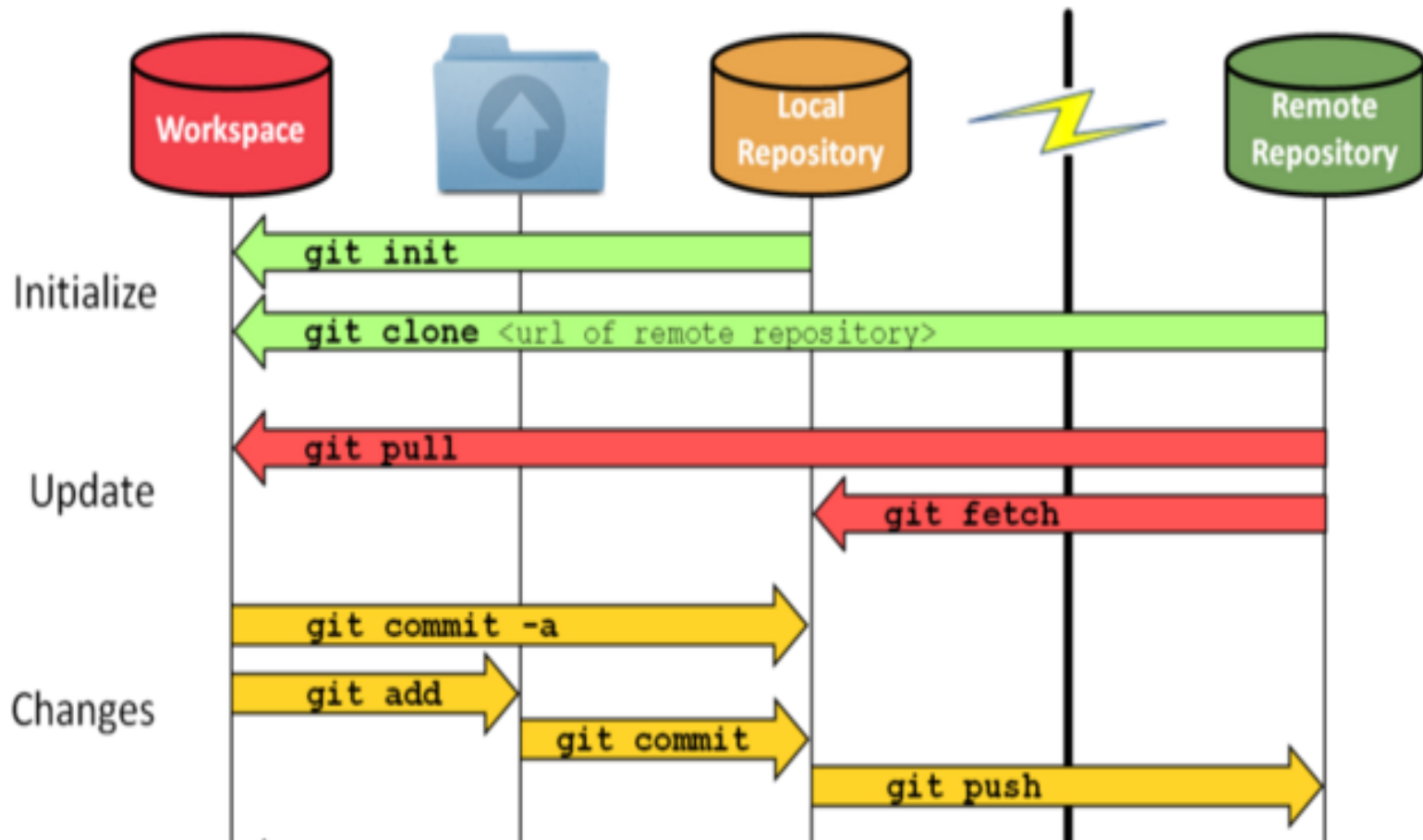
Git Ignore

59

- 專案目錄下若有些檔案不想被Git控管，只要在專案目錄裡放一個 `.gitignore` 檔案，並且設定想要忽略的規則即可。
- 常用設定規則：
 - ▣ 忽略專案各目錄中的X.txt檔案：`X.txt`
 - ▣ 忽略整個Y目錄：`Y`
 - ▣ 只忽略Y目錄中的X.txt檔案：`Y/X.txt`
 - ▣ 忽略所有目錄中附檔名是.txt 的檔案：`*.txt`
 - ▣ 忽略Y目錄中所有附檔名是.txt 的檔案：`Y/*.txt`

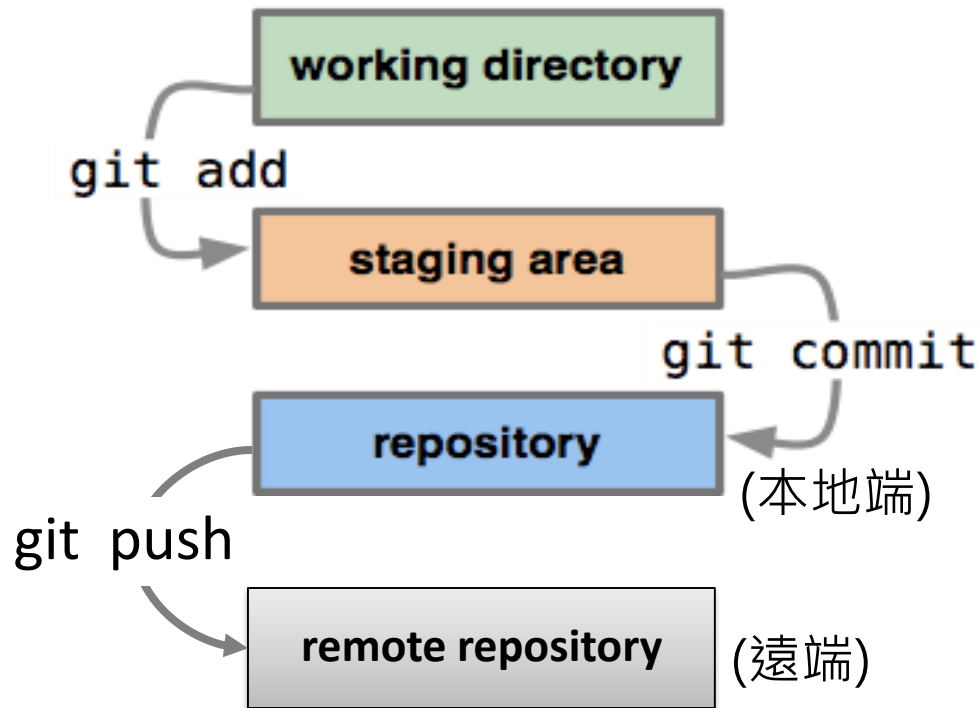
Git 整體運作流程與重要指令

60



Git & GitHub Overview

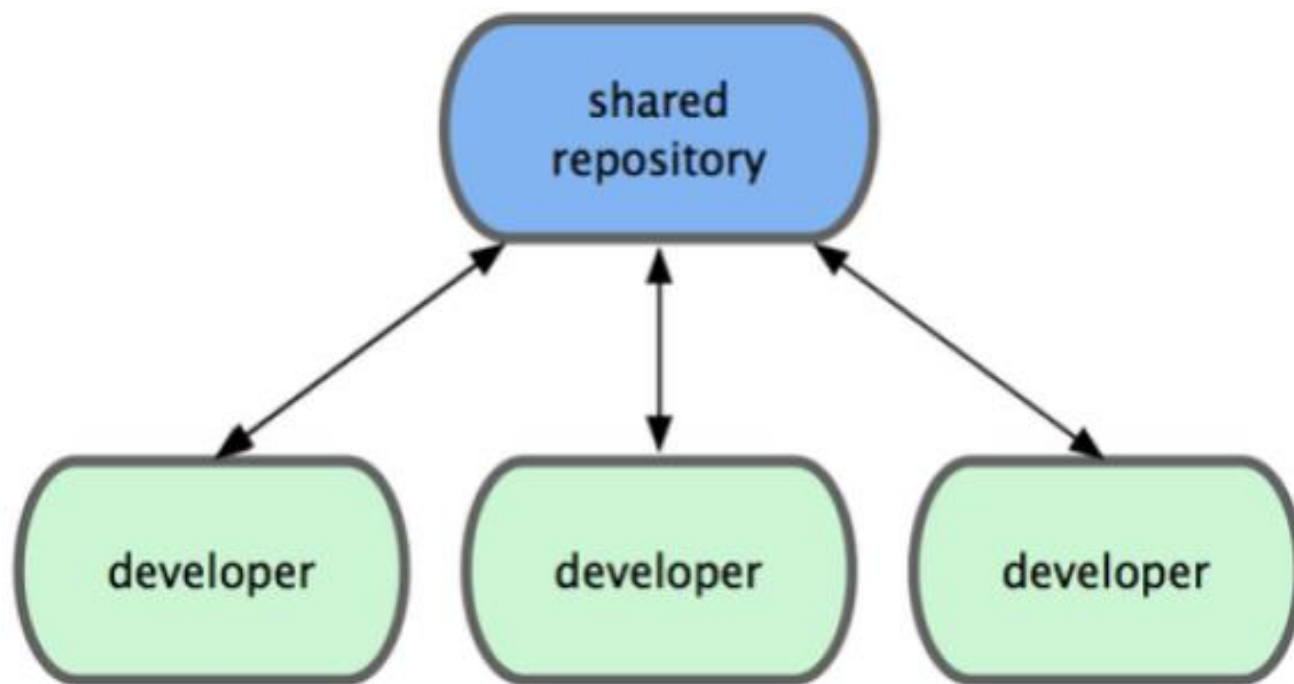
61



<https://stackoverflow.com/questions/676450/eclipse-git-what-does-staged-mean>

集中式版本控制₁

62



集中式版本控制₂

63

- 共用儲存庫
- 流程容易理解
- 類似 SVN 流程
- 多人同時進行版控，不同開發習慣的成員間容易產生衝突

整合管理版本控制¹

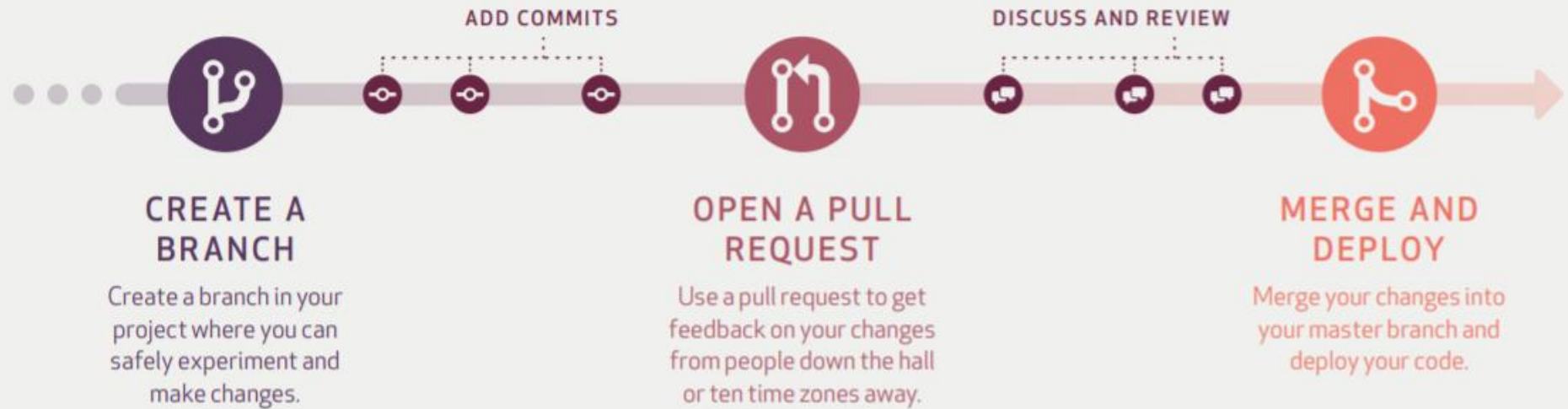
整合管理版本控制₂

65

- 專案維護人員先推送一個版本到主要儲存庫
- 專案開發成員各自複製(Fork and Clone)該儲存庫回去開發
- 專案開發成員推送變更到自己的儲存庫
- 專案開發成員向專案維護人員提出要合併的請求(Pull Request)
- 專案維護人員進行審核，再將變更進行合併整合
- 專案維護人員將合併的變更推送回主要儲存庫

GitHub Flow₁

66



<https://docs.github.com/en/get-started/quickstart/github-flow>

GitHub Flow₂

67

- 負責特定Feature/Fix之成員建立Branch
- 成員修改程式碼
- 成員建立Pull Request
- 其他成員進行審查
- 主責Merge的成員核可Pull Request (併到main分支)
- 建立分支的成員將其分支刪除

其他的GIT雲端服務

68

- GitLab: <https://about.gitlab.com/>
- BitBucket: <https://bitbucket.org/>

Git/GitHub 圖形化介面

69

- ❑ GitHub Desktop: <https://desktop.github.com/>
- ❑ SourceTree: <https://sourcetreeapp.com>
- ❑ TortoiseGit: <https://tortoisegit.org/>

Any Question?

70

