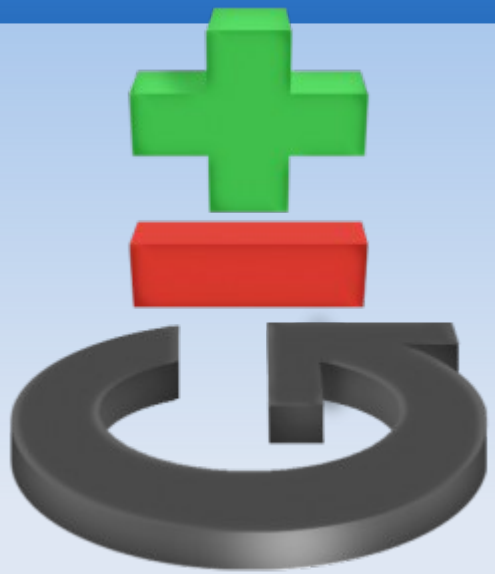


快速上手 Git 版本控制

Bo-Yi Wu
appleboy AT gmail.com

2012.02.05

我是誰？

- 現任台灣 CodeIgniter 站長
- 現任 CodeIgniter User Guide 翻譯
- About me:
  - 部落格 : http://blog.wu-boy.com
  - Twitter: https://twitter.com/#!/appleboy
  - Plurk: http://www.plurk.com/appleboy46
  - Github: https://github.com/appleboy
  - About me: http://about.me/appleboy

上課內容有任何疑問
請馬上打斷
也許您的問題就是大家的問題

版本控制需求

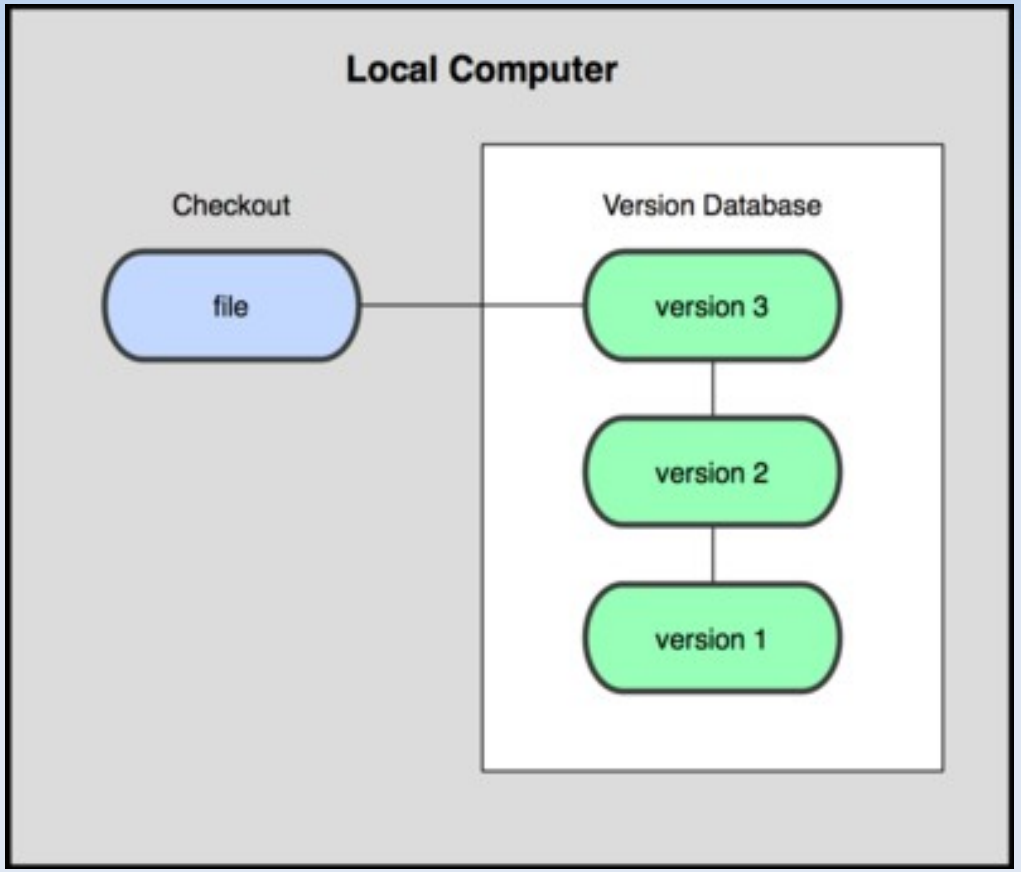
- 版本控制三大好處
  - 備份程式（美工設計師可以儲存各版本圖片）
  - 控管進度（團隊合作）
  - 任意復原（改爛了沒關係）

版本控制分類

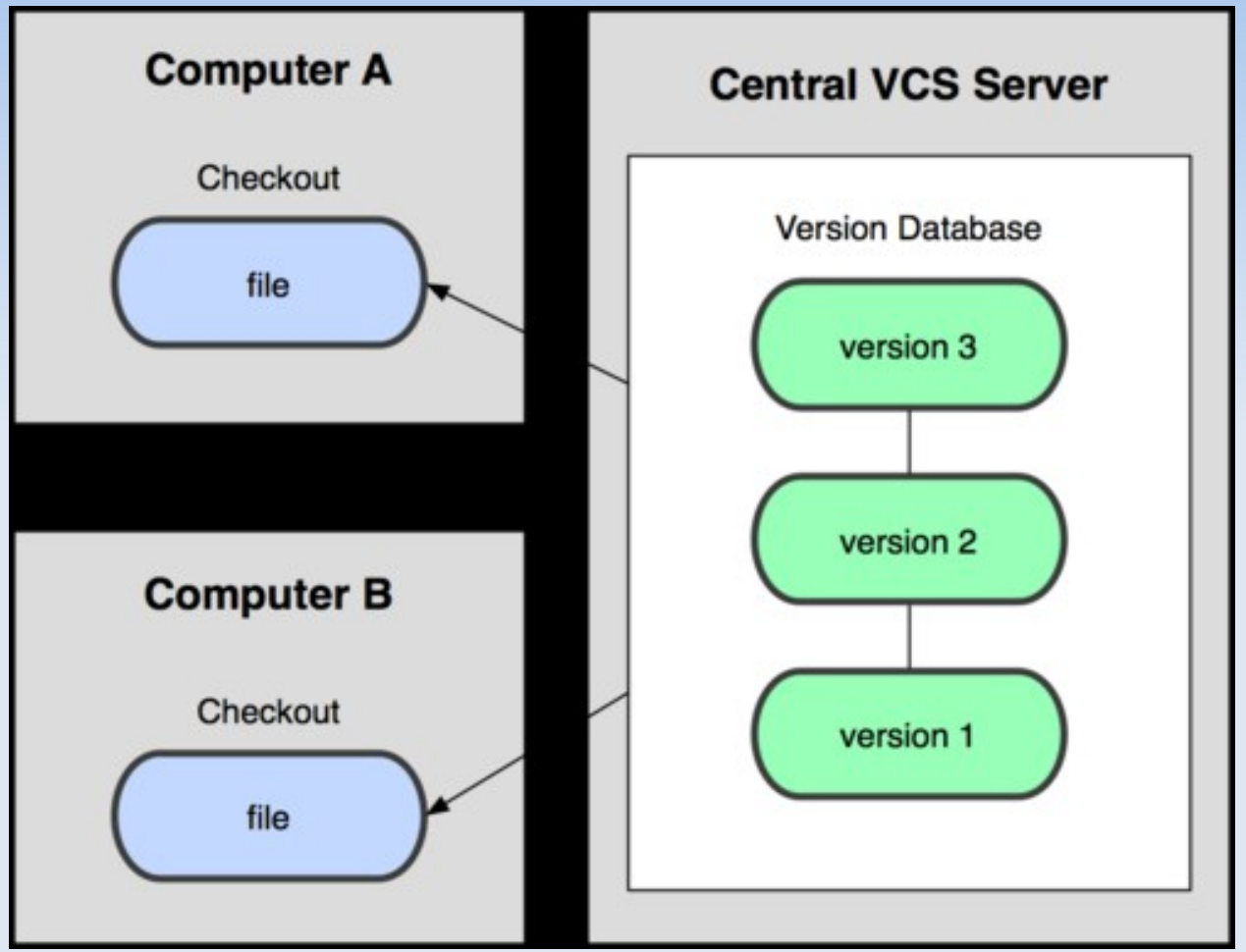
- 本地端版本控制 (Local Version Control)
- 集中式版本控制 (Centralized Version Control)
- 分散式版本控制 (Distributed Version Control)

本地端版本控制

1.改檔名？
2.用日期區別？
3.檔案備份？
4.擴充硬碟？
5.撰寫 Script?



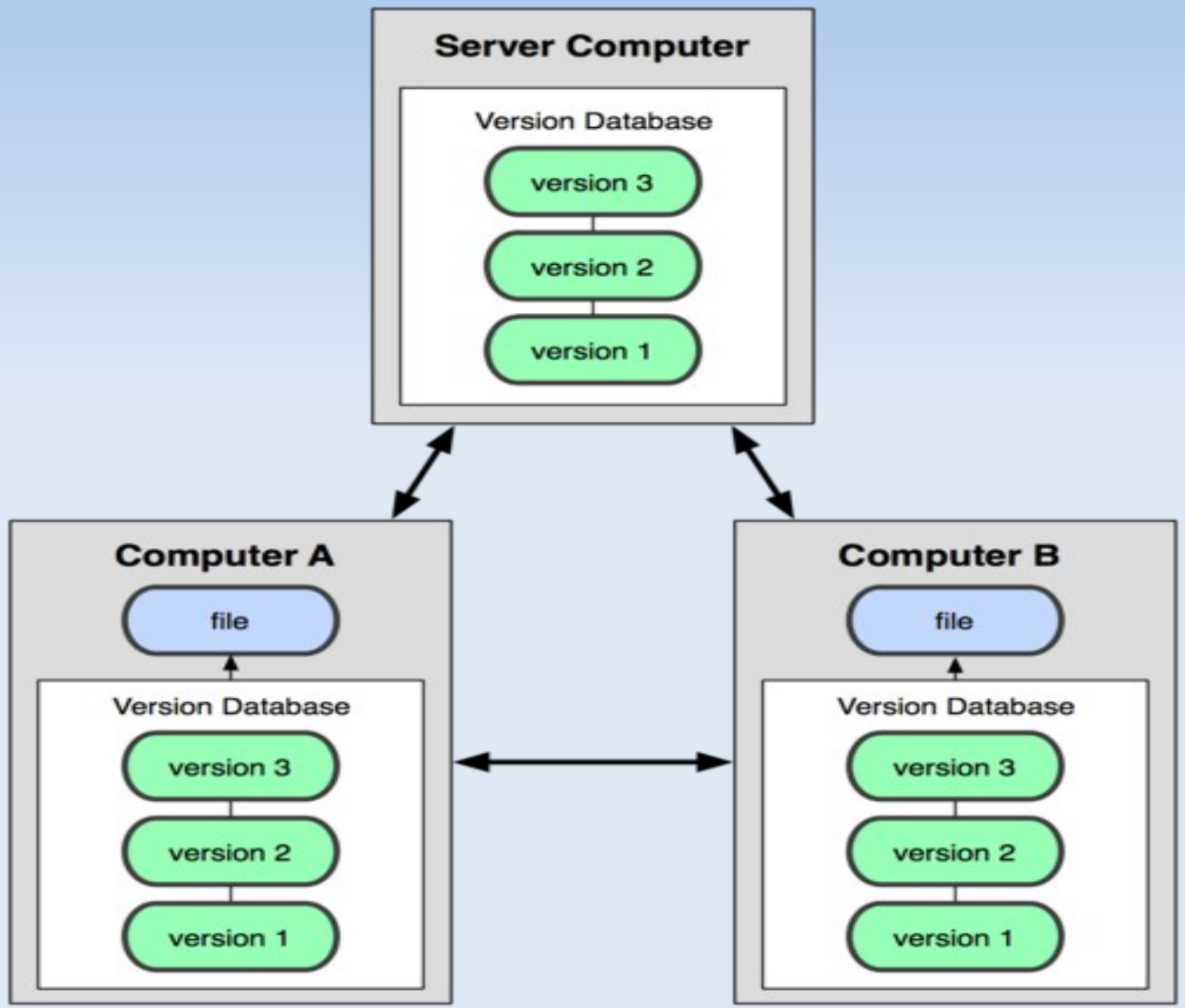
集中式版本控制 (SVN)



Server 掛掉該如何處理呢?

(員工：掛掉最好！！ XD)

分散式版本控制系統 (Git,Mercurial)



大家都是 Server 不必煩惱主機掛掉

Git 設計目標

- 簡單快速 (Speed)
- 設計簡單 (Simple design)
- 動態線性開發 (non-linear development)
  - 大家可以互相合併，不需要有固定主分支 (master)
  - 無限制平行處理 (隨意開 branch)
- 完全分散式處理 (Fully distributed)
- 處理極大量資料 (Linux Kernel)

Git Basics(基礎介紹)

- 直接記錄快照，而非比較差異
- 所有操作幾乎都在本機端執行 (分散式好處)
  - 不需要網路
  - 大家都有備份
- 資料完整性
  - 用 checksum SHA-1 hash 比對驗證檔案
- 檔案三種狀態
  - 增加 staging 觀念 (後面會講解)

安裝 Git

- Installing on Linux
  - yum install git-core
  - apt-get install git-core
  - http://help.github.com/linux-set-up-git/
- Installing on Windows (小烏龜)
  - Tortoisegit (Windwos Git GUI) 不推薦安裝
  - Msysgit (The official Git for Windows)
  - 安裝步驟 http://help.github.com/win-set-up-git/

動手操作

- 安裝步驟 http://help.github.com/win-set-up-git/
  - http://code.google.com/p/msysgit/downloads/list
  - 請下載 Git-1.7.9-preview20120201.exe
  - 設定 SSH Keys (參考上述文件操作)

Git 初始化設定

- 個人設定檔
  - Linux: ~/.gitconfig
  - Windows: C:\Documents and Settings\$USER
- 設定個人資訊
  - git config --global user.name "Bo-Yi Wu"
  - git config --global user.email appleboy.tw@xxx.com
  - 設定完成皆會寫到 .gitconfig 檔案

線上文件

- 透過三種方式查詢
  - git help **config**
  - git **config** help
  - man git-**config**

請先註冊 (擇一即可)

https://github.com/

https://bitbucket.org/

Github 跟 Bitbucket 差異

- Github
  - 功能強大
  - 私有 repository 需要收費
  - 真正 Social Coding
  - 許多 Open Source 專案都在此紮根 ...
- Bitbucket
  - 功能較少
  - 私有 repository 完全免費

開始操作 Git

兩種方式建立
Git Repository

建立新的 Git Repository

- 任何目錄下執行
  - $ git init
- 初始化 github 或 bitbucket host 專案
  - Format: git clone [url]
  - $ git clone git://github.com/appleboy/test.git
  - $ git clone git://github.com/appleboy/test.git abc
- 專案底下會出現 .git 目錄
  - .git 目錄記載所有版本資訊

git clone example

$ git clone git://github.com/phpbb/phpbb3.git

Cloning into phpbb3...

remote: Counting objects: 108296, done.

remote: Compressing objects: 100% (24699/24699), done.

remote: Total 108296 (delta 74585), reused 105904 (delta 72611)

Receiving objects: 100% (108296/108296), 23.78 MiB | 589 KiB/s, done.

Resolving deltas: 100% (74585/74585), done.

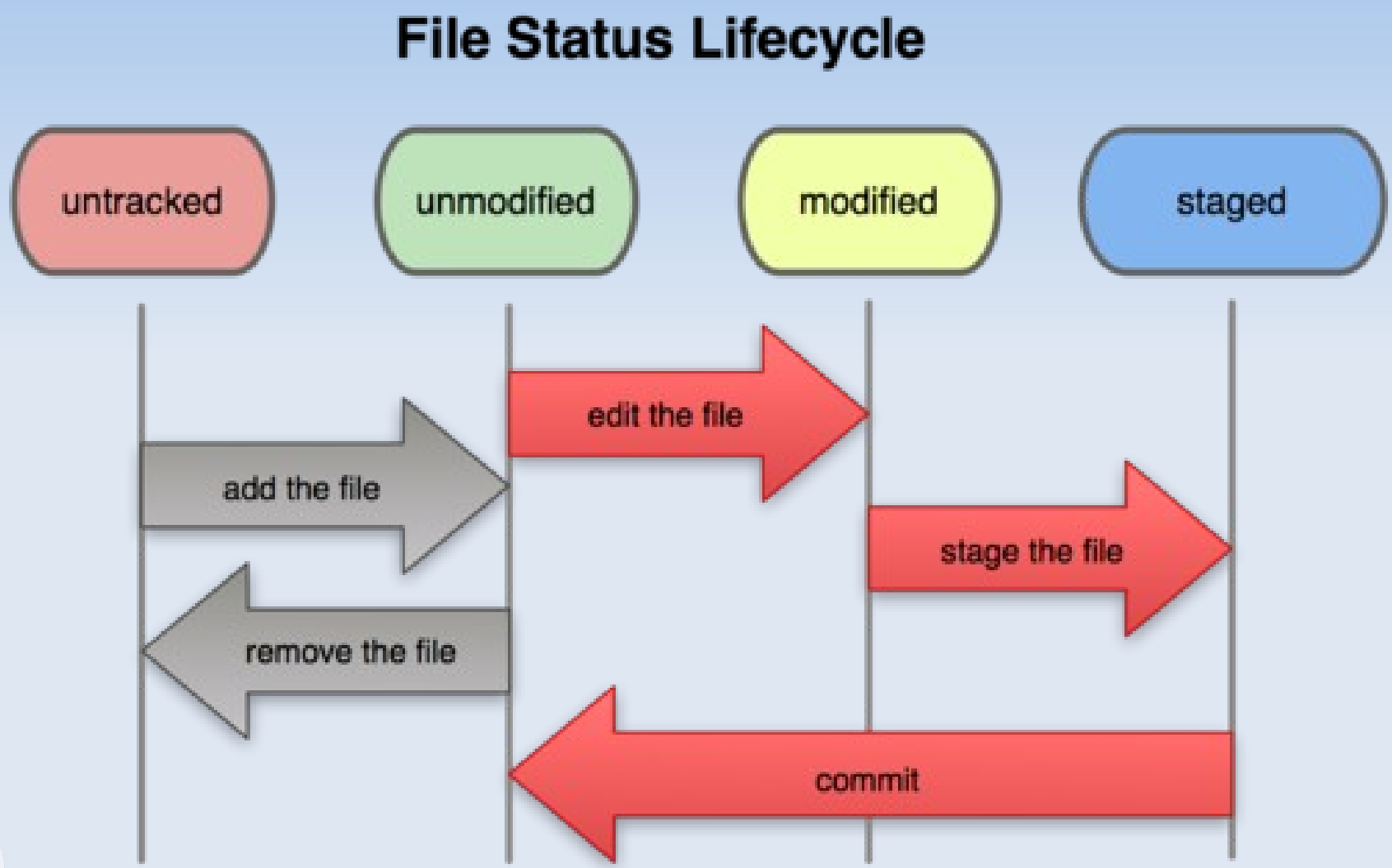
Git 四種 protocol

- file://
  - 本地端 (Local) 執行
- git://
  - Read-Only access
- https:// ssh://
  - Read+Write access
  - 建議用此方法 (避開公司防火牆)

第一次 commit

- 新增一個檔案
  - $ touch README
- 把檔案加入專案
  - $ git add README
- 提交變更
  - $ git commit

檔案狀態



用 git status 檢查檔案狀態

檔案操作

- untracked → staged (add the new file)
  - $ git add file_name
- unmodified → modified (edit the file)
  - $ echo 'test' > file_name
- modified → staged (stage the file)
  - $ git add file_name
- staged → unmodified (commit)
  - $ git commit
- unmodified → untracked (remove the file)
  - $ git rm file_name

忽略檔案 (Ignoring Files)

- 在專案目錄底下新增 .gitignore
  - 任意正規寫法都可以
  - *.o (不把 .o 檔加入 repository)
  - !lib.o (除了 lib.o 之外，其餘都略過)

小技巧 (密碼檔案)

- 可以新增 config.php.sample
- 略過 config.php
  - 將 config.php 寫到 .gitignore

觀看檔案差異 (git diff)

- 觀看目前跟上一版本差異
  - $ git diff
- 觀看 stage 跟上一版本差異 (已經 git add 過)
  - $ git diff --cached

更改檔名 (git mv)

- $ git mv README.txt README
  - 等同於
  - $ mv README.txt README
  - $ git rm README.txt
  - $ git add README

查看歷史紀錄 (git log)

- 列出修改檔案清單
  - $ git log --stat
  - $ git log --pretty=format:"%h - %an, %ar : %s"

```
Option  Description of Output
%H      Commit hash
%h      Abbreviated commit hash
%T      Tree hash
%t      Abbreviated tree hash
%P      Parent hashes
%p      Abbreviated parent hashes
%an     Author name
%ae     Author e-mail
%ad     Author date (format respects the -date= option)
%ar     Author date, relative
%cn     Committer name
%ce     Committer email
%cd     Committer date
%cr     Committer date, relative
%s      Subject
```

小技巧 (改爛還原就好)

- staged → modified (stage 狀態還原到 unstage)
  - $ git reset HEAD <file>
- modified → unmodified (改爛了沒關係)
  - $ git checkout -- <file>
- 修改最後一次 commit log
  - $ git commit --amend

上傳到 Remote Server

- 預設 branch 叫 master
  - $ git branch -a
- 預設 remote 叫 origin
  - $ git remote -v

上傳到 github

- 新增遠端 Server
  - Format: git remote add [short_name] [url]
  - $ git remote add origin git@xxxxxxx
- 上傳變更檔案到 Server
  - Format: git push [short_name] [branch_name]
  - $ git push -u origin master

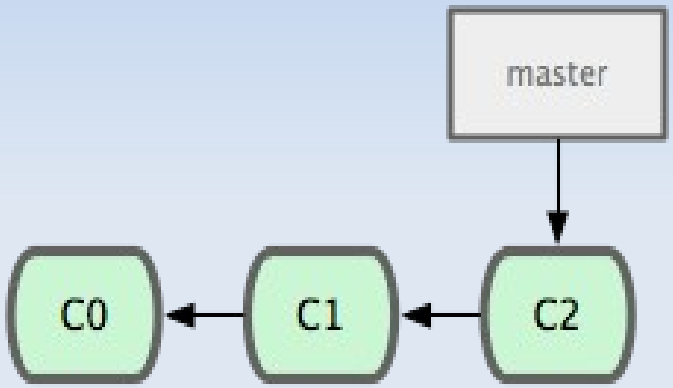
遠端更新及合併檔案

- 直接下載檔案
  - Format: git fetch [short_name]
  - $ git fetch origin
- 下載檔案並且 merge
  - $ git pull [short_name] [branch_name]
  - $ git pull origin master
- git pull = git fetch + git merge
  - git fetch origin + git merge origin/master

如何使用標籤 (Tag)

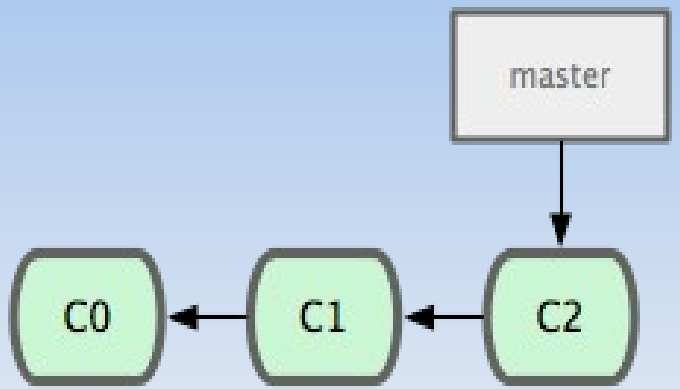
- 列出既有標籤
  - $ git tag -l
- 新增標籤
  - $ git tag -a v1.4 -m 'my version 1.4'
  - $ git tag -a v1.4 9fceb02
- 上傳標籤
  - $ git push origin v1.4
  - $ git push origin --tags (上傳所有標籤)
- 刪除標籤
  - $ git tag -d <tagname>
  - $ git push origin :refs/tags/v1.4

如何使用分支 **git branch**

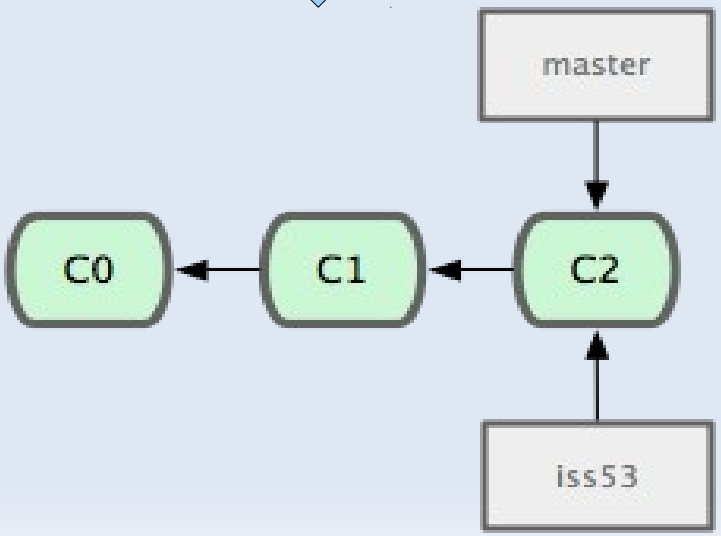


原來只有一個 master branch

新增 branch



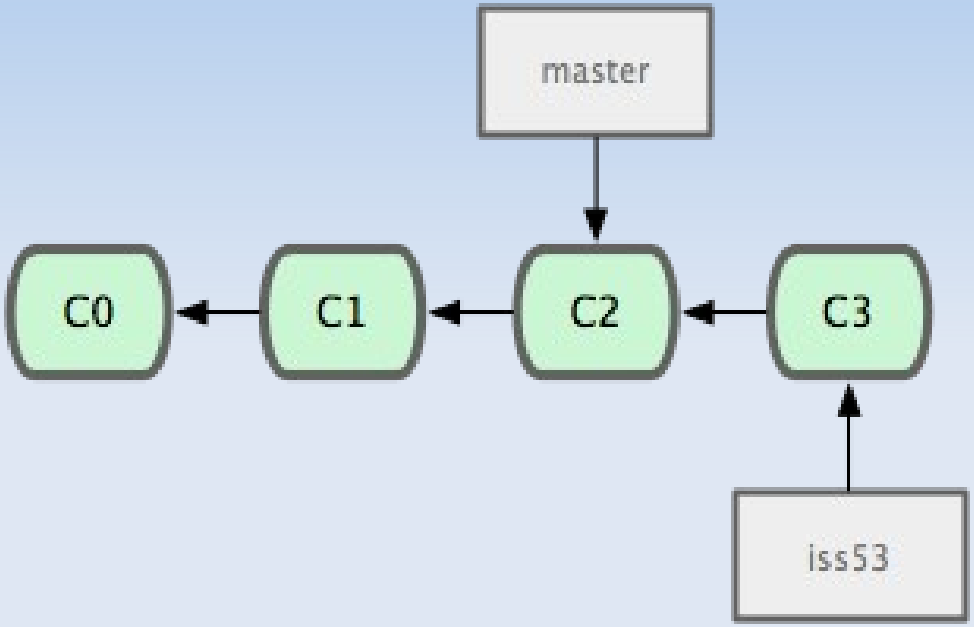
新增 iss53 branch



Git branch 指令

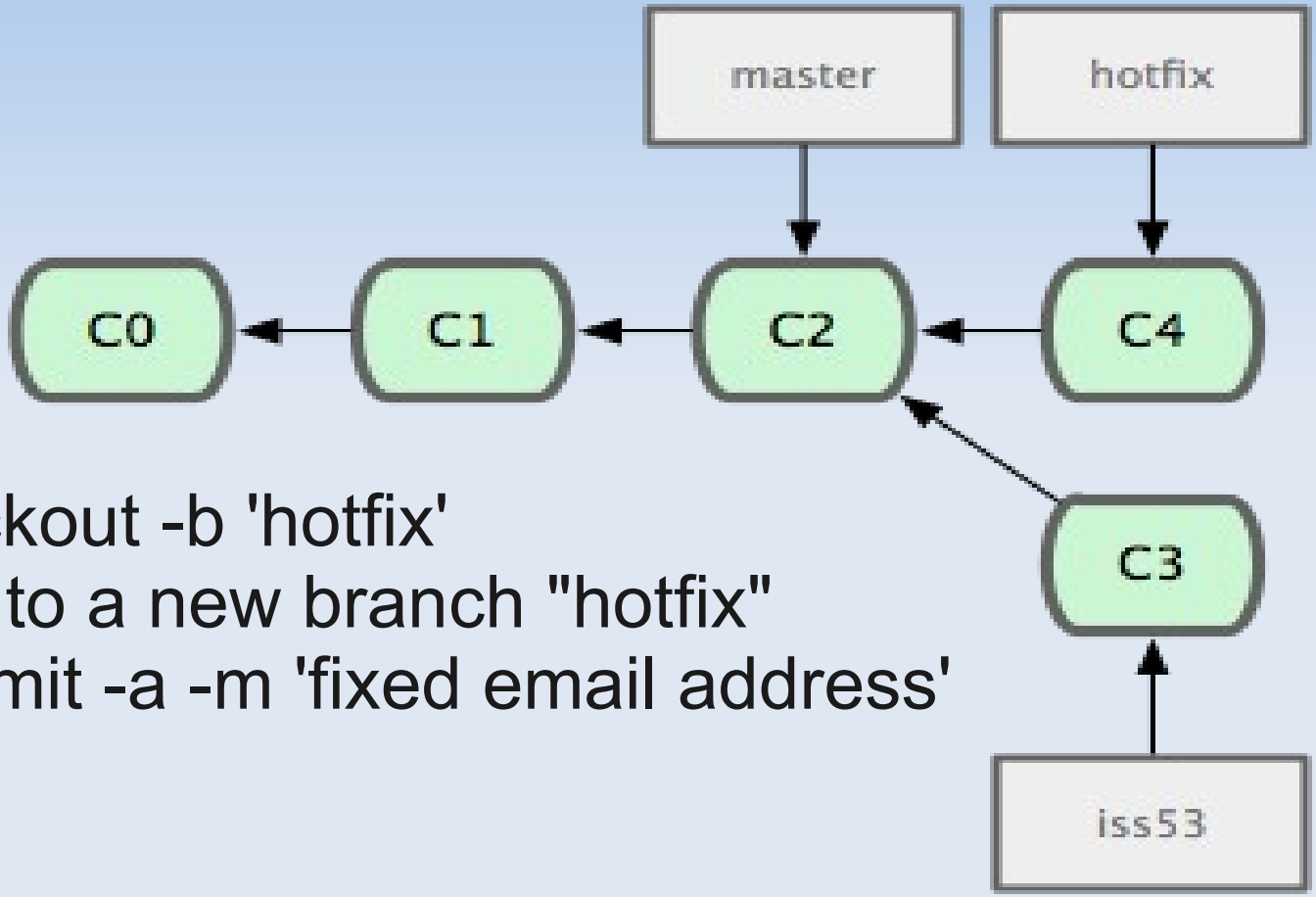
- 新增 branch
  - $ git branch [branch_name]
  - $ git branch iss53
- 刪除 branch(小心使用)
  - $ git branch -d iss53
  - $ git branch -D iss53 (强制删除)
- 切換 branch
  - $ git checkout [branch_name]
  - $ git checkout -b [branch_name] (新增且切換)
  - $ git checkout -b test
  - 等同於 git branch test && git checkout test

iss53 提交 commit



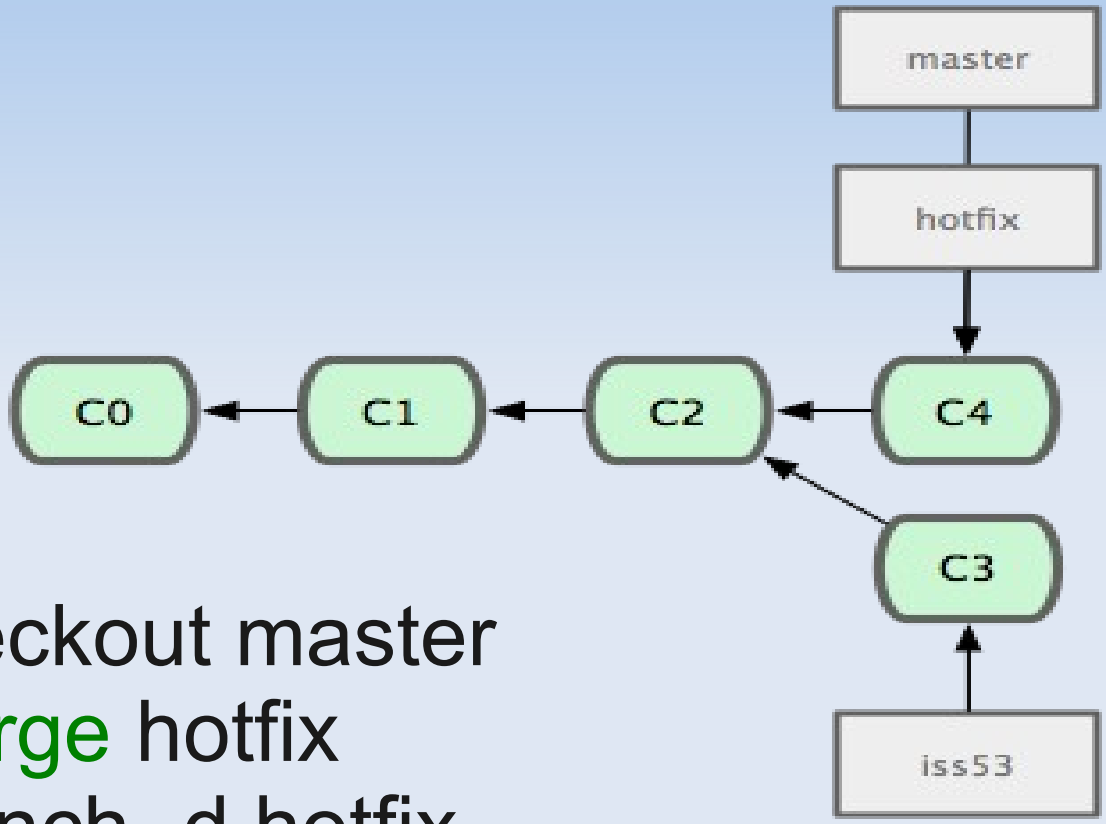

```
$ vim index.html
$ git commit -a -m 'new footer [issue 53]'
```

建立 **hotfix** 分支




```
$ git checkout -b 'hotfix'
Switched to a new branch "hotfix"
$ git commit -a -m 'fixed email address'
```

合併分支 master+hotfix




```
$ git checkout master
$ git merge hotfix
$ git branch -d hotfix
```

管理 **branch** 架構

- 列出全部 branch
  - $ git branch -a
- 詳細列出 branch
  - $ git branch -v
- 列出已經 merge 的 branch
  - $ git branch --merged
- 列出尚未 merge 的 branch
  - $ git branch --no-merged

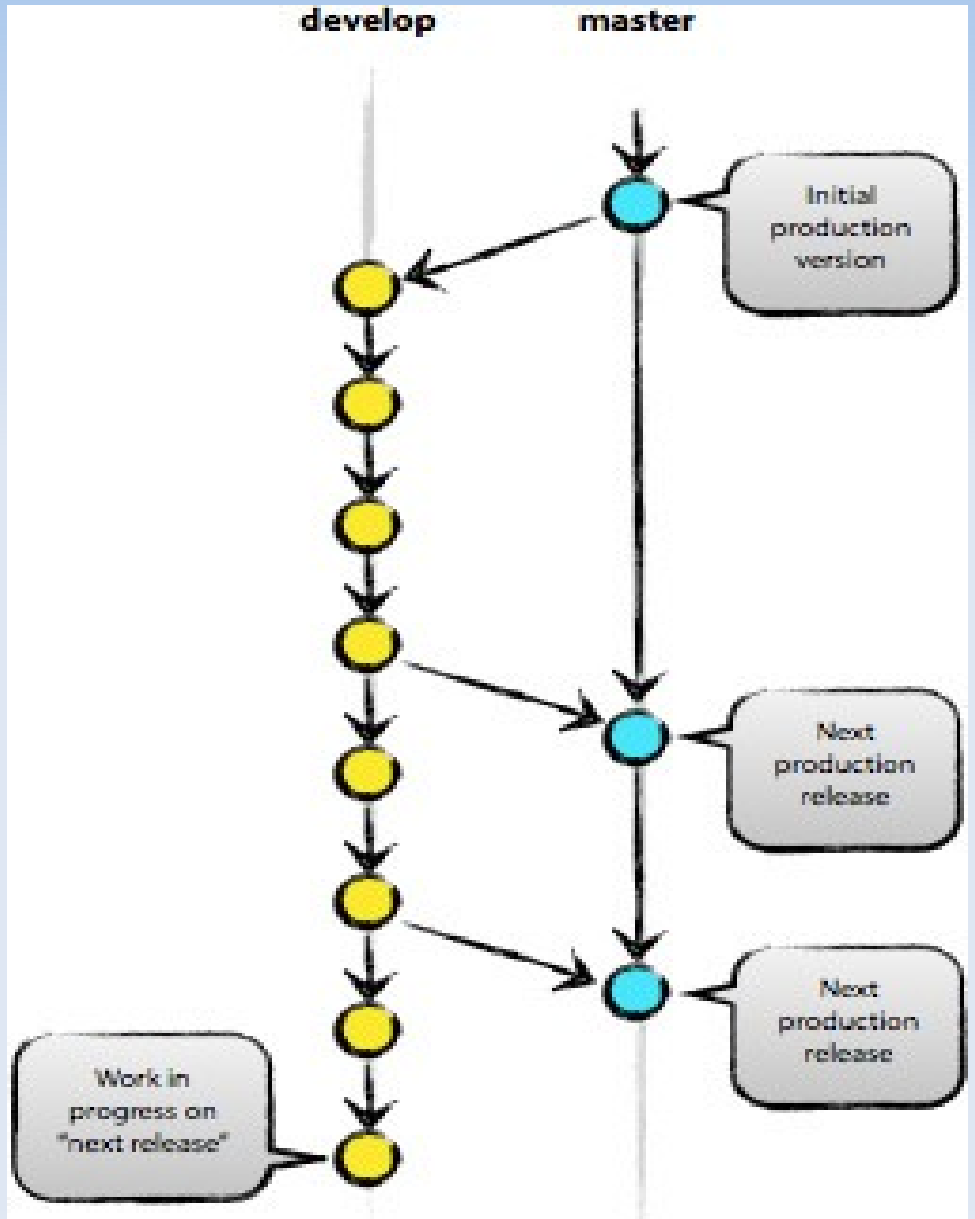
管理遠端 branch

- 上傳 branch
  - Format: git push origin [branch_name]
- 刪除 branch
  - Format: git push origin :[branch_name]
  - $ git push origin :hotfix

Git branch 設計

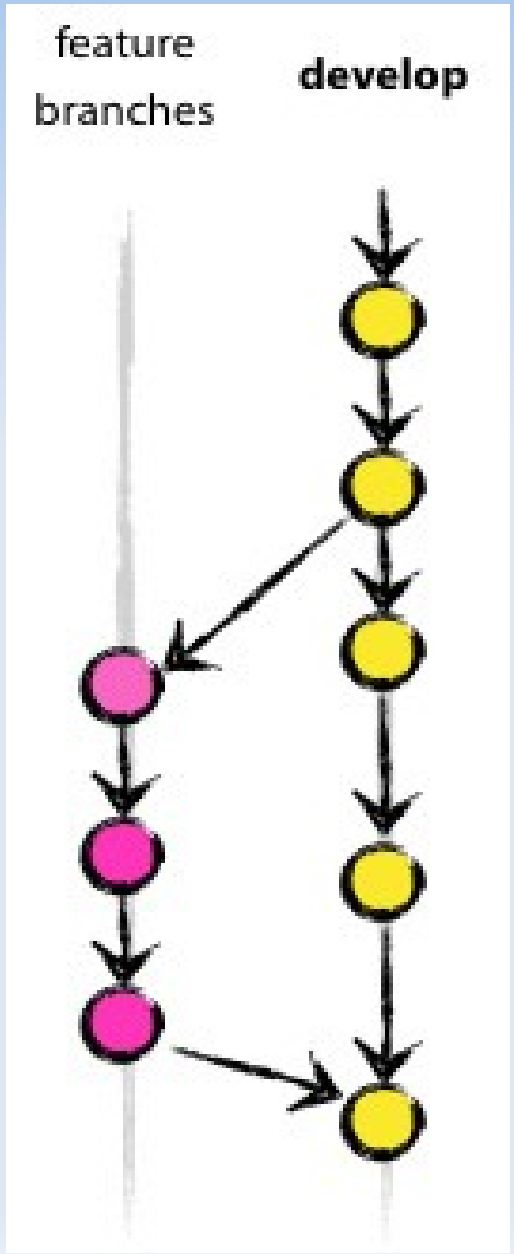
- 主要分支
  - master 主程式 (除非重大 bug，則會分出 hotfix 分支)
  - develop 開發分支 (用來在另外分支出 Release, feature)
- 次要分支
  - Hotfixes(由 master 分支，馬上修正 bug)
  - Feature(由 develop 分支，開發新功能)
  - Release(由 develop 分支，開發下一版 release)

主要分支

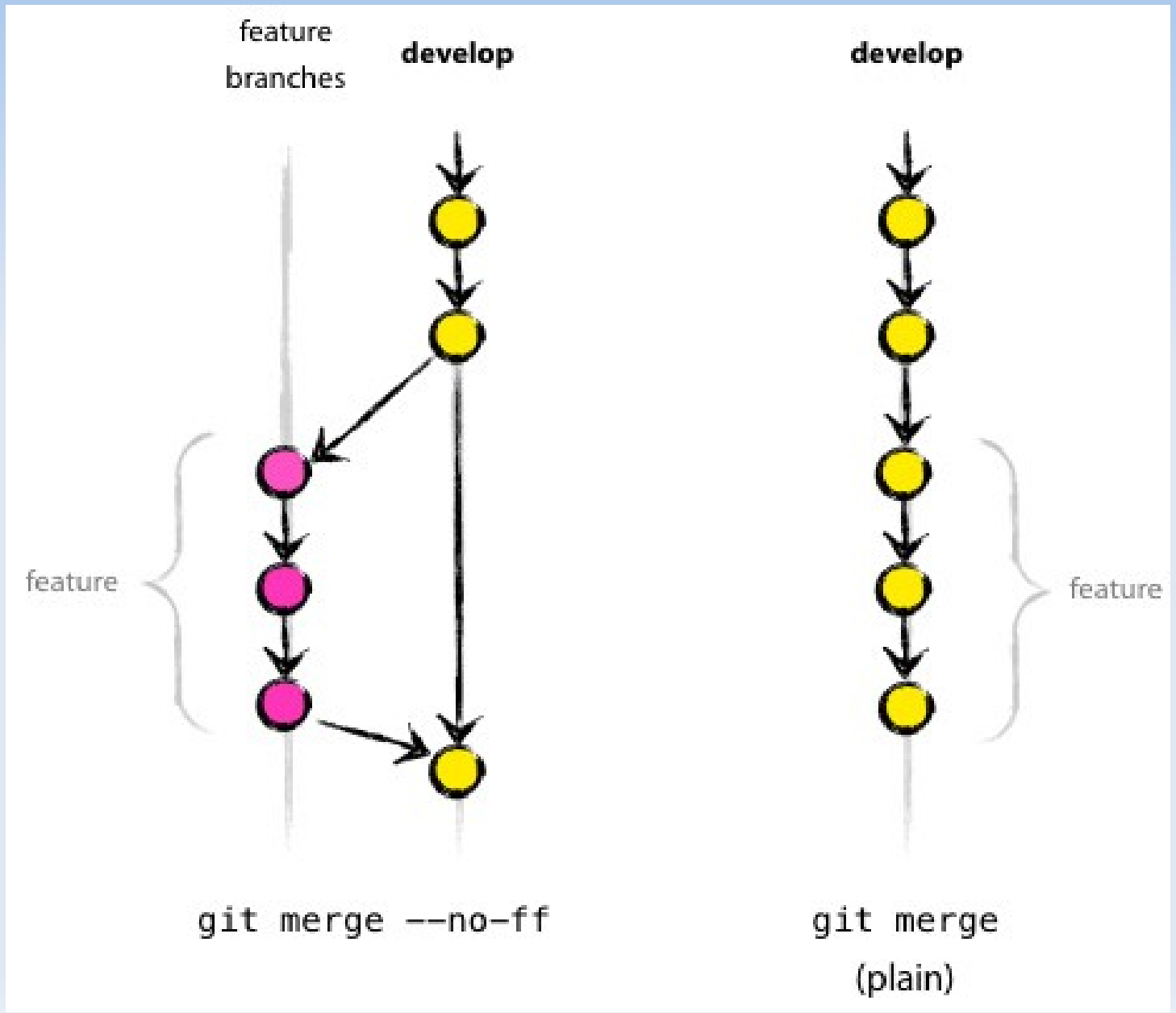

develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

新分支 Feature branches

- branch off from: develop
- merge back into: develop
  - $ git checkout -b feature develop
  - $ edit …..
  - $ git commit -a -m ”....”
  - $ git checkout develop
  - $ git merge --no-ff feature
  - $ git branch -d myfeature
  - $ git push origin develop



git merge --no-ff



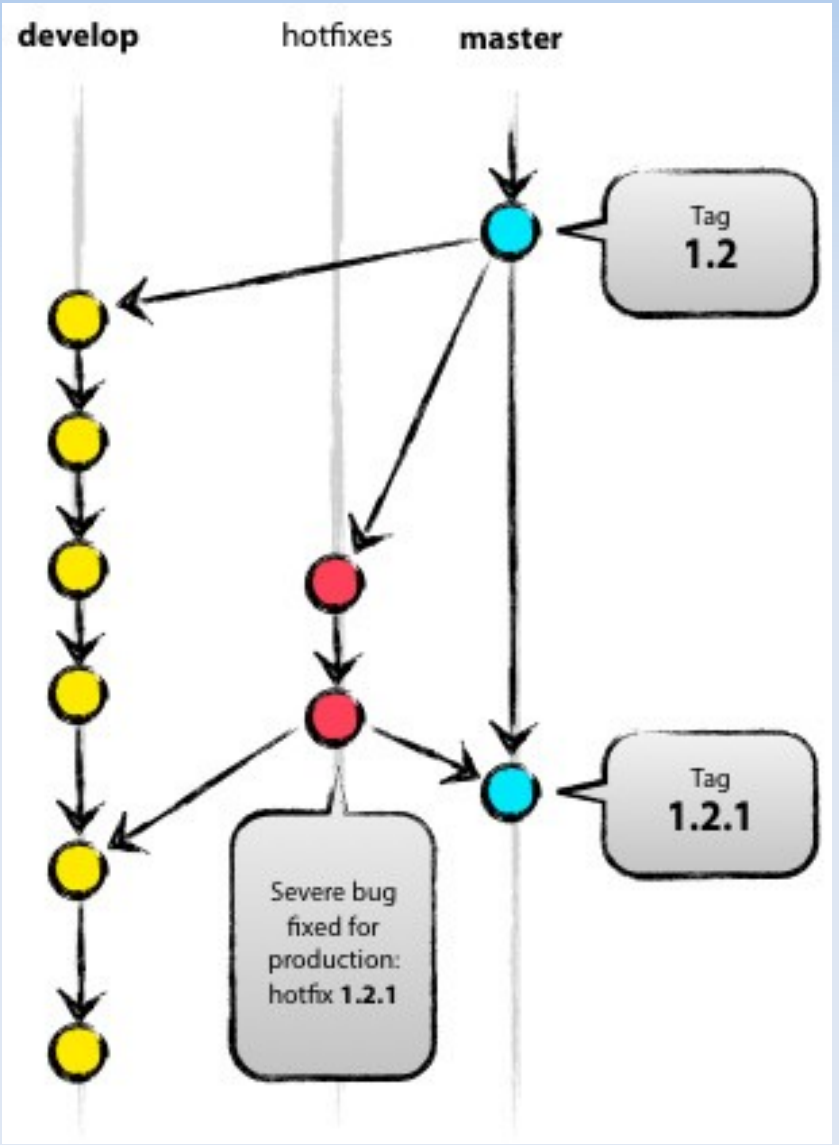
Release branches

- $ git checkout -b release-1.3 develop
- $ git commit -a -m "Update: release 1.3"
- $ git checkout master
- $ git merge --no-ff release-1.3
- $ git tag -a v1.3 -m "Release v1.3 Tag"
- $ git checkout develop
- $ git merge --no-ff release-1.3
- $ git push
- $ git push origin v1.3
- $ git branch -d release-1.3

重大 issue (Hotfix branches)

- branch off from: master
- merge back into: develop and master

```
$ git checkout -b hotfix-1.3.1 master
$ git commit -a -m "Hotfix: release 1.3.1"
$ git checkout master
$ git merge --no-ff hotfix-1.3.1
$ git tag -a v1.3.1 -m "Hotfix v1.3.1 Tag"
$ git checkout develop
$ git merge --no-ff hotfix-1.3.1
$ git branch -d hotfix-1.3.1
$ git push
$ git push origin v1.3.1
```



Git Submodule

專案用到很多
Open Source
(Blueprintcss, jQuery..)

Git Submodule

可以任意將他人的專案掛載在
任何目錄底下

建立 Git Submodule

- $ git submodule add <repository> [<path>]
  - 注意 path 部份，請勿先建立空目錄
  - $ git submodule add repository_url user_guide

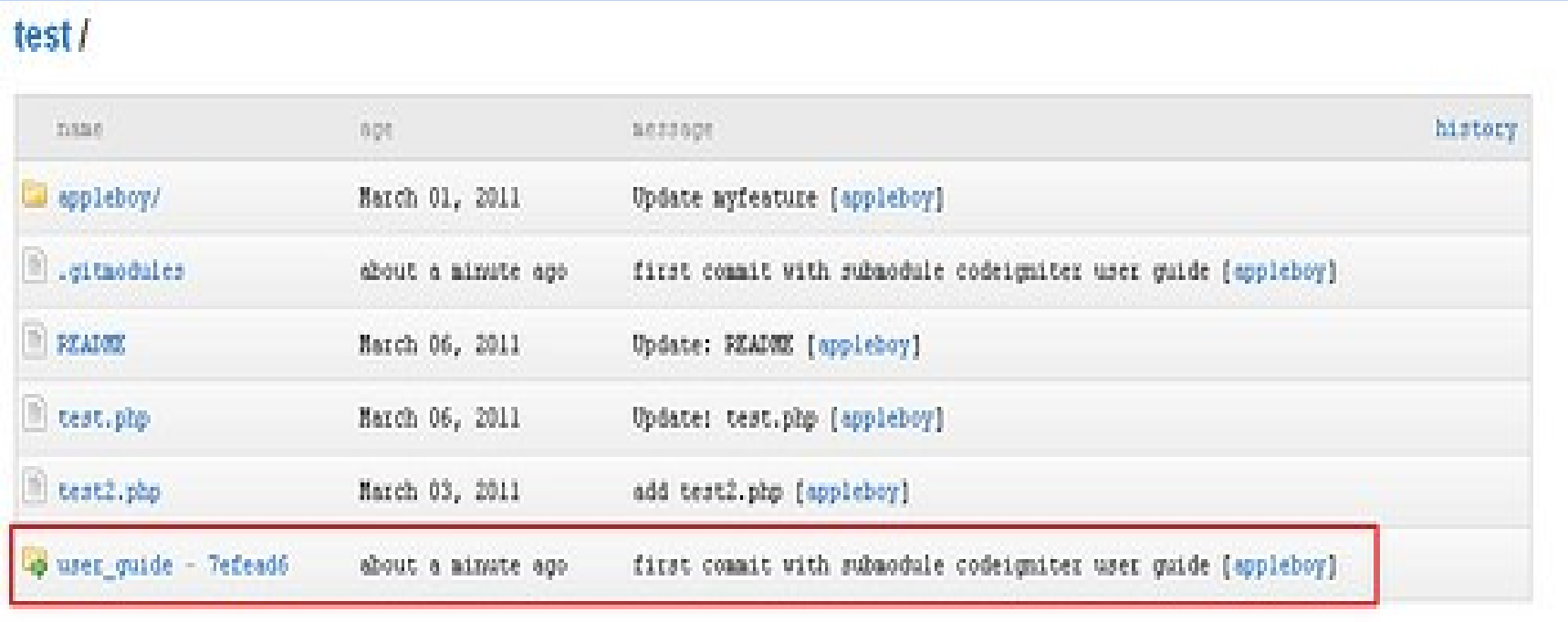
```
Cloning into user_guide...
remote: Counting objects: 32, done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 32 (delta 12), reused 32 (delta 12)
Unpacking objects: 100% (32/32), done.
```

Git Submodule

- git status 會發現多出兩個檔案
  - new file: .gitmodules
  - new file: user_guide
- open .gitmodules

```
[submodule "user_guide"]
    path = user_guide
    url = https://github.com/appleboy/CodeIgniter-TW-Language
```

Commit submodule



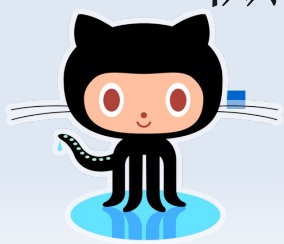
多出 git submodule 小圖示

clone project with Git Submodule

- $ git clone github_repository test
- 將 module 寫入 .git/config
  - $ git submodule init
- 下載 submodule 程式碼
  - $ git submodule update

更新已安裝 module

- 切換到 sub module 目錄
  - $ cd user_guide/
- 更新檔案
  - $ git pull origin master
- 回到專案目錄並且更新 submodule commit ID
  - $ cd /xxx/project
  - git commit -a -m ”update” && git push
- 檢查是否有相同的 commit ID
  - git submodule status



移除 Sub module

- 移除目錄
  - git rm --cached [目錄]
  - git rm [目錄]
- 修改 .gitmodules， 移除不需要的 module
- 修改 .git/config， 移除 submodule URL
- 執行 commit
  - git add . && git commit -m "Remove sub module"
- 最後 syn module 資料
  - git submodule sync

參考資料

- Pro Git http://progit.org/book/
- Git 文章系列 http://blog.wu-boy.com/tag/git/

謝謝大家

大家辛苦了