

Git Tutorial

<http://ihower.tw/git>

2014/10



Agenda

- 什麼是「版本控制系統」
- Git 簡介
- Git 內部原理
- Git 指令
- Git 開發模式及流程
- 更多 Git Tips
- GitHub 簡介



我的版本控制之路

- 2002 目錄管理法 (定時 copy 備份一次)
- 2005 SubVersion
- 2007 SVK
- 2008 Git (像 SVN 一樣只會 push/pull)
- 2009 Git (主要還是在 master 開發，有大功能才會開 feature branches)
- 2011 Git (根據 git flow，做任何事幾乎都開 topic branch。大玩 rebase)



1. 版本控制系統



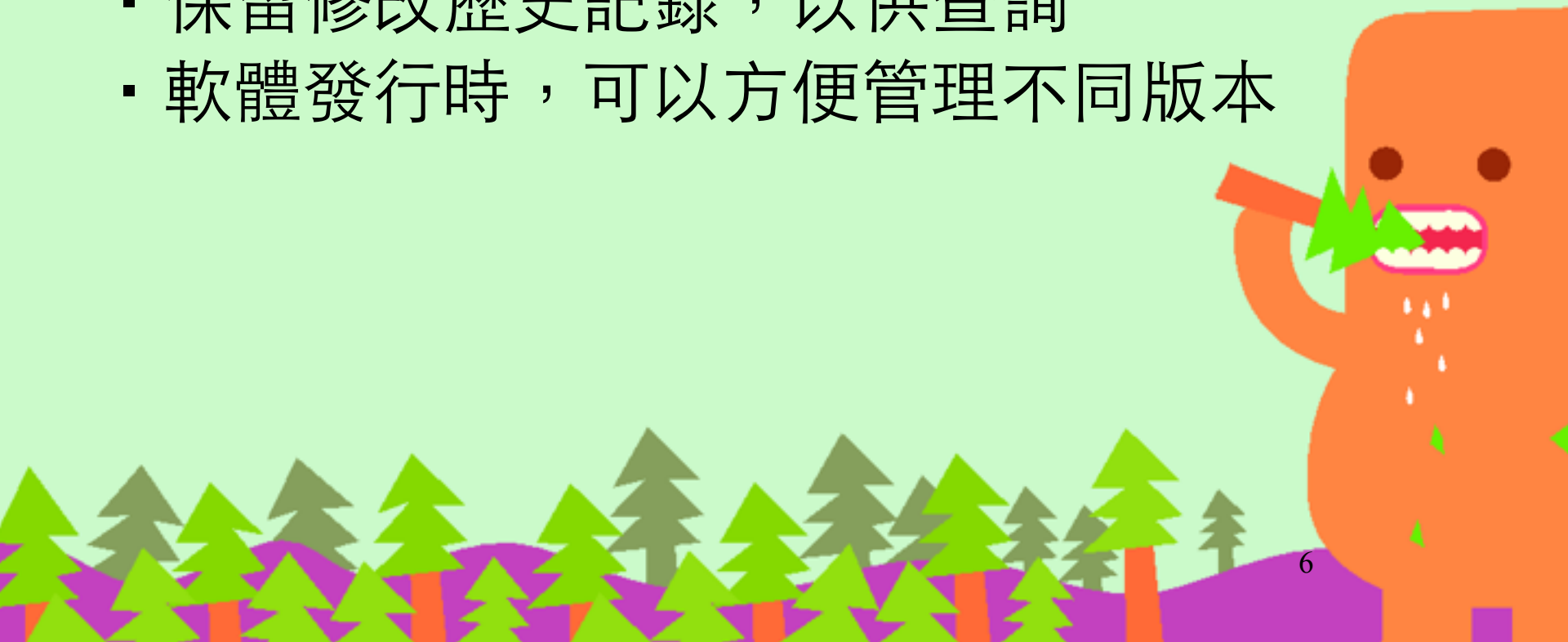
Why? 要解決的問題

- 檔案被別人或自己覆蓋，甚至遺失
- 想復原前幾天寫的版本
- 想知道跟昨天寫的差在哪裡
- 是誰改了這段程式碼，為什麼
- 軟體發行，需要分成維護版跟開發版



我們需要一種機制

- 可以隨時復原修改，回到之前的版本
- 多人協作時，不會把別人的東西蓋掉
- 保留修改歷史記錄，以供查詢
- 軟體發行時，可以方便管理不同版本



軟體開發的基本工具

- 版本控制系統不祇可以幫助妳追蹤修訂手上的工作進度，讓妳在千鈞一髮之際還能拾回過往辛苦的結晶，甚至能夠讓妳跟其他人協同工作、合作無間。

<http://jedi.org/blog/archives/004784.html>

- 那些台灣軟體產業所缺少的 – 版本控制系統

<http://blog.ez2learn.com/2011/10/20/taiwan-software-lacking-of-vcs/>

Version Control System (VCS)

- 建立 Repository (儲存庫)，用來保存程式碼
- 方便散佈程式給團隊，有效率協同開發
- 記錄誰改變什麼、在什麼時候、因為什麼原因
- Branch(分支)，可因不同情境分開開發
- Tag(標籤) 重要里程碑，以便參照



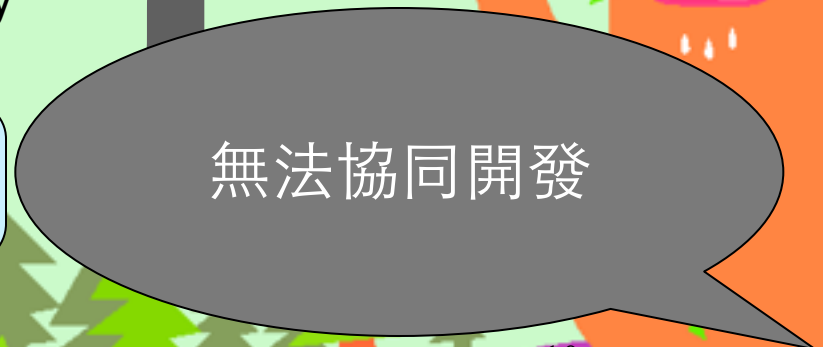
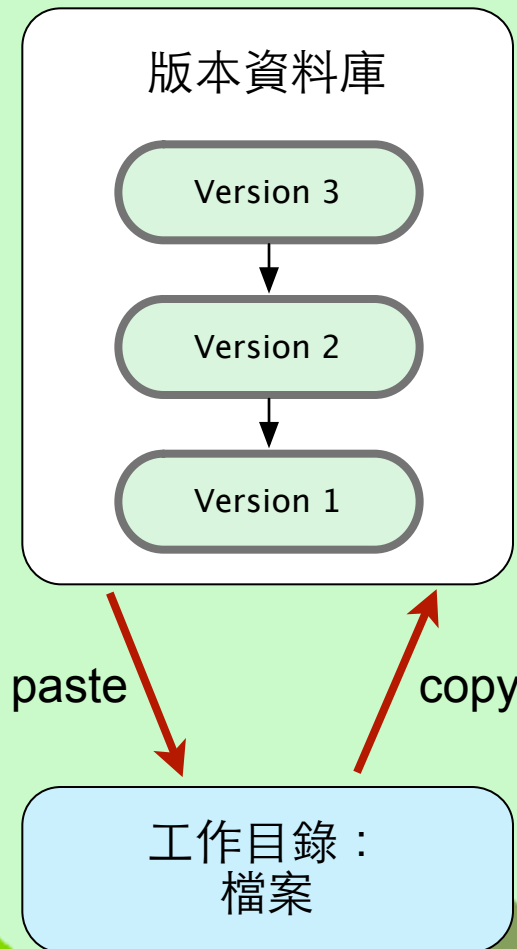
Repository 存什麼?

- 所有跑起來這個專案需要的東西
 - 所有原始碼、範例設計檔、文件等等
 - 暫存檔、log 檔案、build files 等編譯後的產物
則不需要存進 Repository

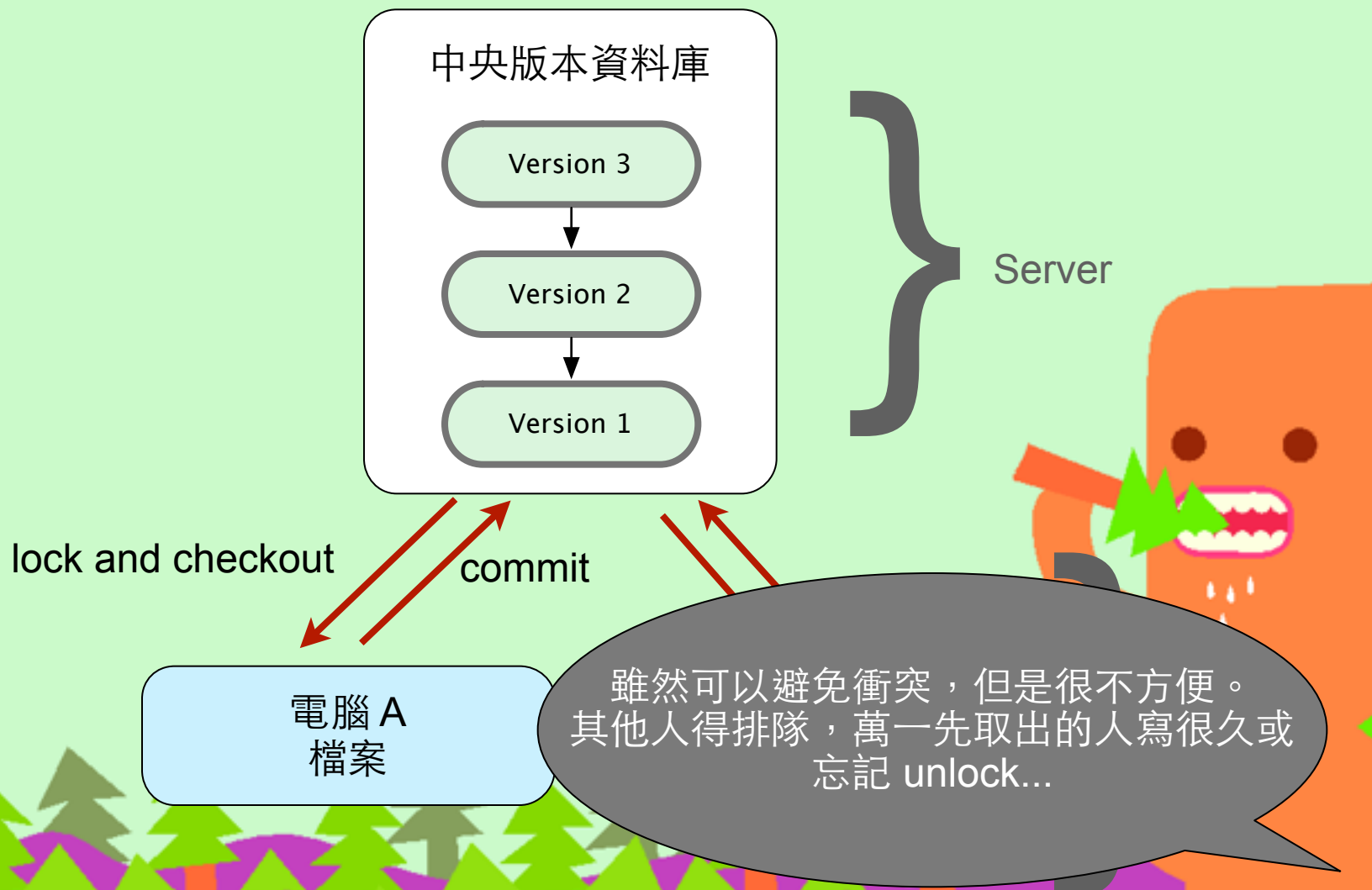


Local VCS

copy paste 資料夾管理, rcs

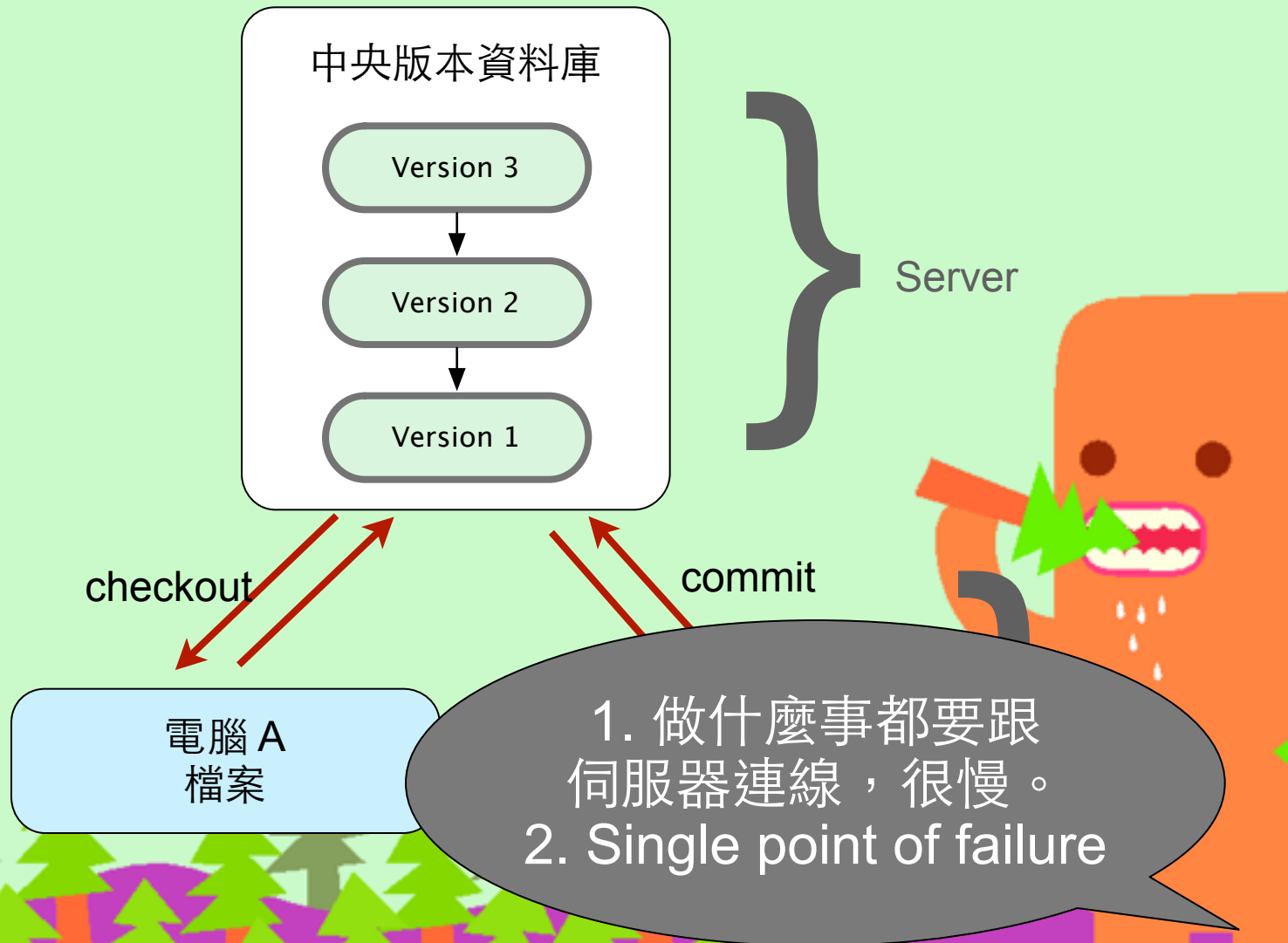


Centralized VCS (Lock型，悲觀鎖定)



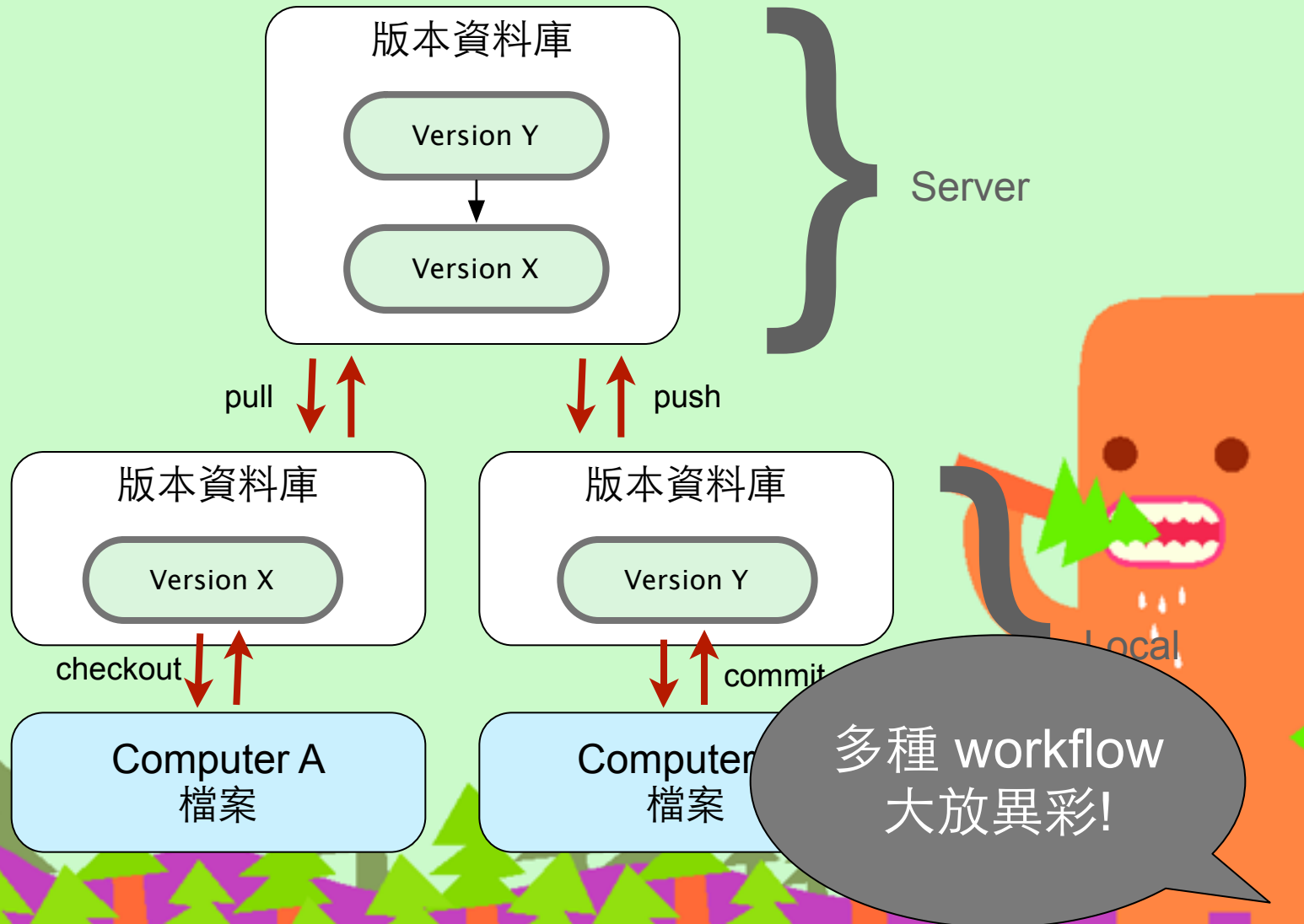
Centralized VCS (Merge型，樂觀鎖定)

CVS, Subversion, Perforce



Distributed VCS

Git, Mercurial(Hg), Bazaar



Local development

- 集中式的 VCS 系統，沒網路就不能開發，無法 commit，無法看 history log。
 - 坐高鐵，坐飛機的時候
 - 網路故障的時候
 - 咖啡店沒有無線網路的時候
- 分散式 CSV 系統即使沒網路，照常可以 commit 和看 history log。
- 不用擔心server備份



2. Git 簡介



Git

- 開放源碼的分散式版本管理系統(DVCS)
- 發明人 Linus Torvalds (Linux creator)
- 目的是管理 Linux Kernel 原始碼
- 2005/4 開始開發，2005/6 開始管理 Linux Kernel，2005/12 釋出 1.0 版



誰在用 Git?

- Git
- Linux Kernel
- Perl
- Eclipse
- Gnome
- KDE
- Qt
- Ruby on Rails
- Android
- PostgreSQL
- Debian
- X.org



Why Git is Better than X

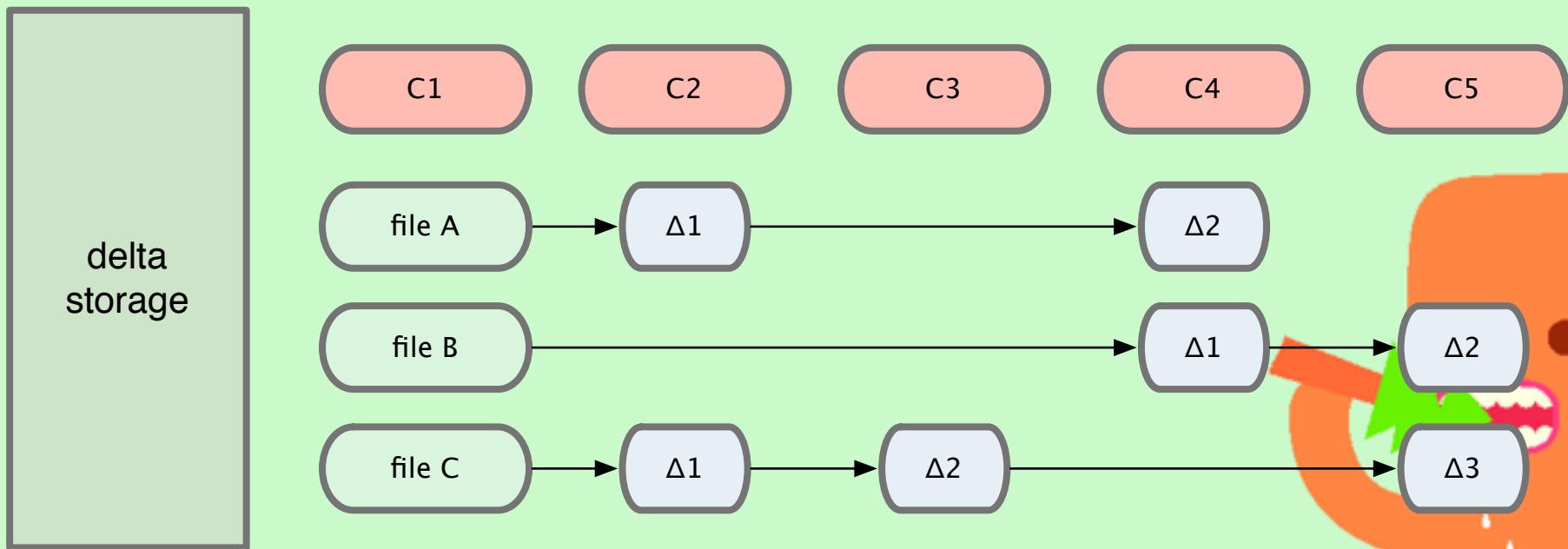
<http://thkoch2001.github.io/whygitisbetter/>

- 便宜快速的本地分支
- 所有內容都在本地端
- Git 很快
- Git 很省空間
- Staging 功能
- 它是分散式的
- 適用任何工作流程
- 我們有 GitHub!



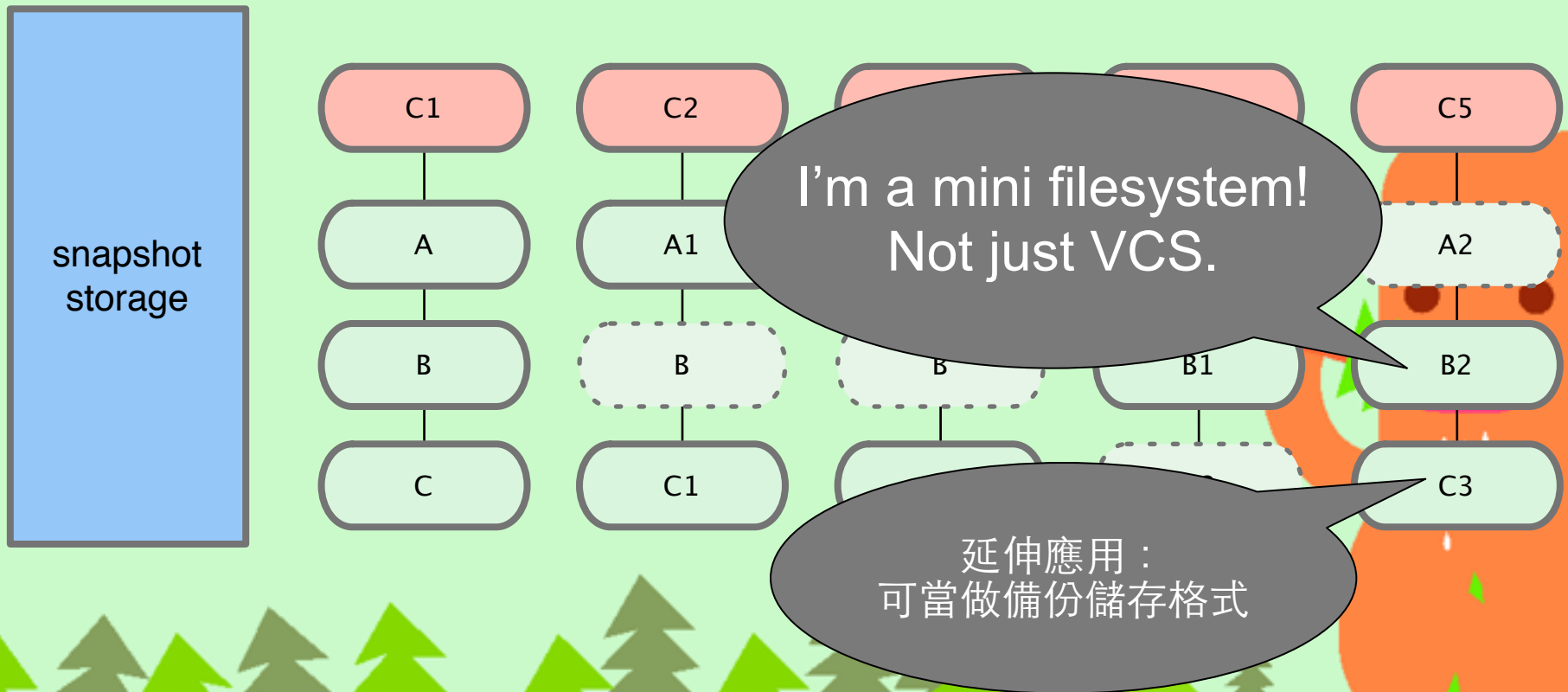
傳統 Delta 儲存方式

記錄每次檔案變更，開分支需要建立副本。



DAG 儲存方式 (DAG: Directed acyclic graph)

利用有向無環圖來記錄 metadata 建構出 snapshots 。
相同內容只會有一份記錄。



rails (branch: master)

Refresh

Terminal Reveal

RAILS

Stage

BRANCHES

2-3-stable

3-0-stable

3-1-stable

3-2-stable

fix_invalid_datetime_att...

master

patch_logger_encoding

REMOTES

origin

TAGS

OTHER

All Local "master"

Search Subject, Author, SHA

Subject	Date	Author
Update 4.0 Release Notes	July 4, 2012 7:23 PM	Rafael Mendonça F...
Kill not used constant since removal of runner method	July 4, 2012 8:02 AM	Carlos Antonio da...
remove duplicate requires of mocha.	July 4, 2012 5:28 AM	Aaron Patterson
remove the runner method copied from minitest	July 4, 2012 5:23 AM	Aaron Patterson
hook mocha in through m/t before_setup, after_teardown hooks	July 4, 2012 5:21 AM	Aaron Patterson
Merge pull request #6948 from frodsan/plugin_skip_ignore	July 4, 2012 1:00 AM	Rafael Mendonça F...
remove test/dummy from gitignore when "plugin new -T"	July 4, 2012 12:58 AM	Francesco Rodriguez
Merge branch 'lexmag-reference_statements'	July 4, 2012 12:48 AM	Rafael Mendonça F...
Don't need to use delete in the options hash	July 4, 2012 12:24 AM	Rafael Mendonça F...
Add changelog entry for references statements	July 3, 2012 3:44 AM	Aleksey Magusev
Refactor references schema definitions	July 3, 2012 3:43 AM	Aleksey Magusev
Make references statements reversible	July 3, 2012 3:42 AM	Aleksey Magusev
Add references schema statements	July 3, 2012 3:41 AM	Aleksey Magusev
Merge pull request #6946 from threedaymonk/queue-refs	July 3, 2012 11:44 PM	Aaron Patterson
Make TestQueueTest work with marshalling queue	July 3, 2012 8:03 PM	Paul Battley
Ensure jobs do not refer to the queue	July 3, 2012 6:59 PM	Paul Battley
Ensure test jobs are marshallable	July 3, 2012 6:55 PM	Paul Battley
Track queue threading with named classes	July 3, 2012 5:29 PM	Paul Battley
Merge pull request #6940 from acapilleri/observer_update	July 3, 2012 7:44 PM	Carlos Antonio da...

SHA: b5a2f24b6b2b466a5692f0d46b4a2b8154ee0e53

Author: Rafael Mendonça França <rafaelmfranca@gmail.com>

Date: Wed Jul 04 2012 19:23:51 GMT+0800 (CST)

Subject: Update 4.0 Release Notes

Refs: master origin/HEAD origin/master

Parent: f3353895a04dfd9a06cf1ca2e7bbfb80a38dbfad

Git it

Update 4.0 Release Notes

Related with 17d2115e3df6caf15d554b44e28177069fa3ed5b

guides/source/4_0_release_notes.textile

guides/source/4_0_release_notes.textile

... @@ -329,6 +329,20 @@ Moved into a separate gem <tt>sprockets-rails</tt>.

329 329

330 330 h3. Active Record

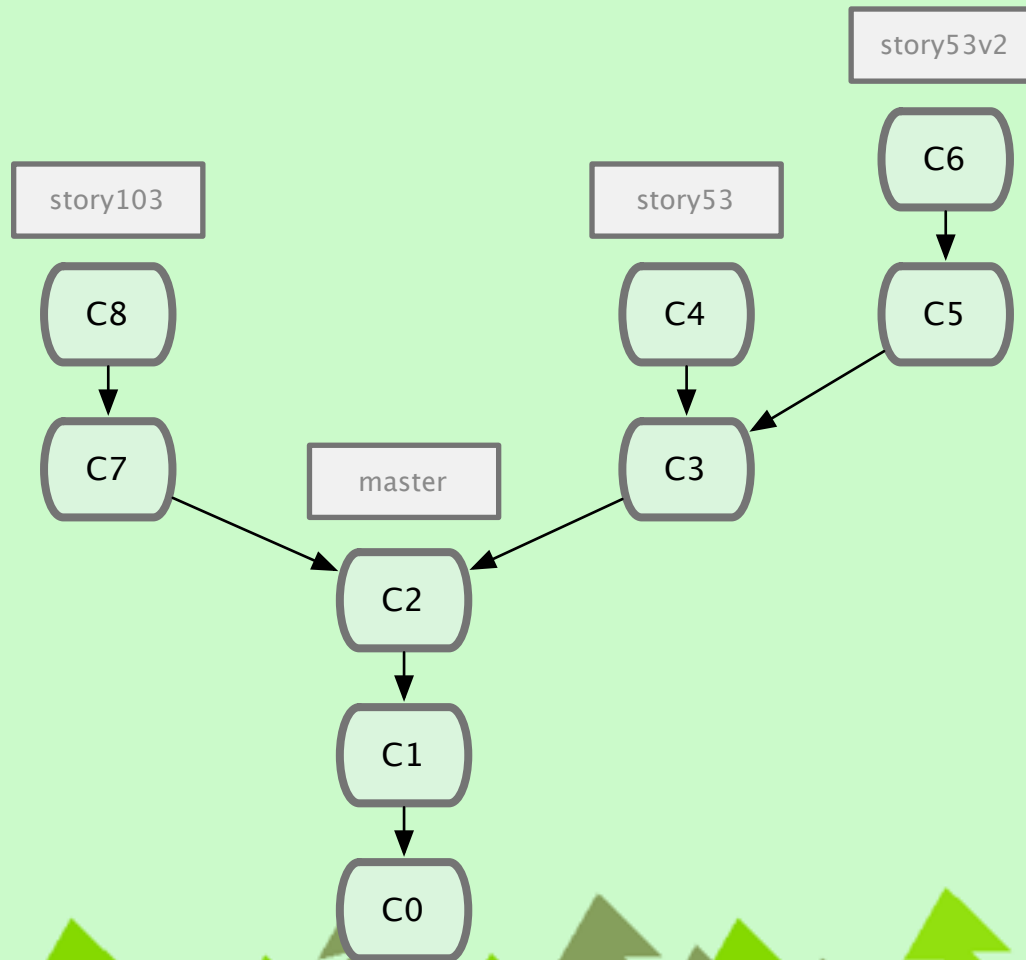
331 331

332 ** Add <tt>add_reference</tt> and <tt>remove_reference</tt> schema statements. Aliases, <tt>add_belongs_to</tt>

333 +

9286 commits loaded

Git graph example





@KentBeck

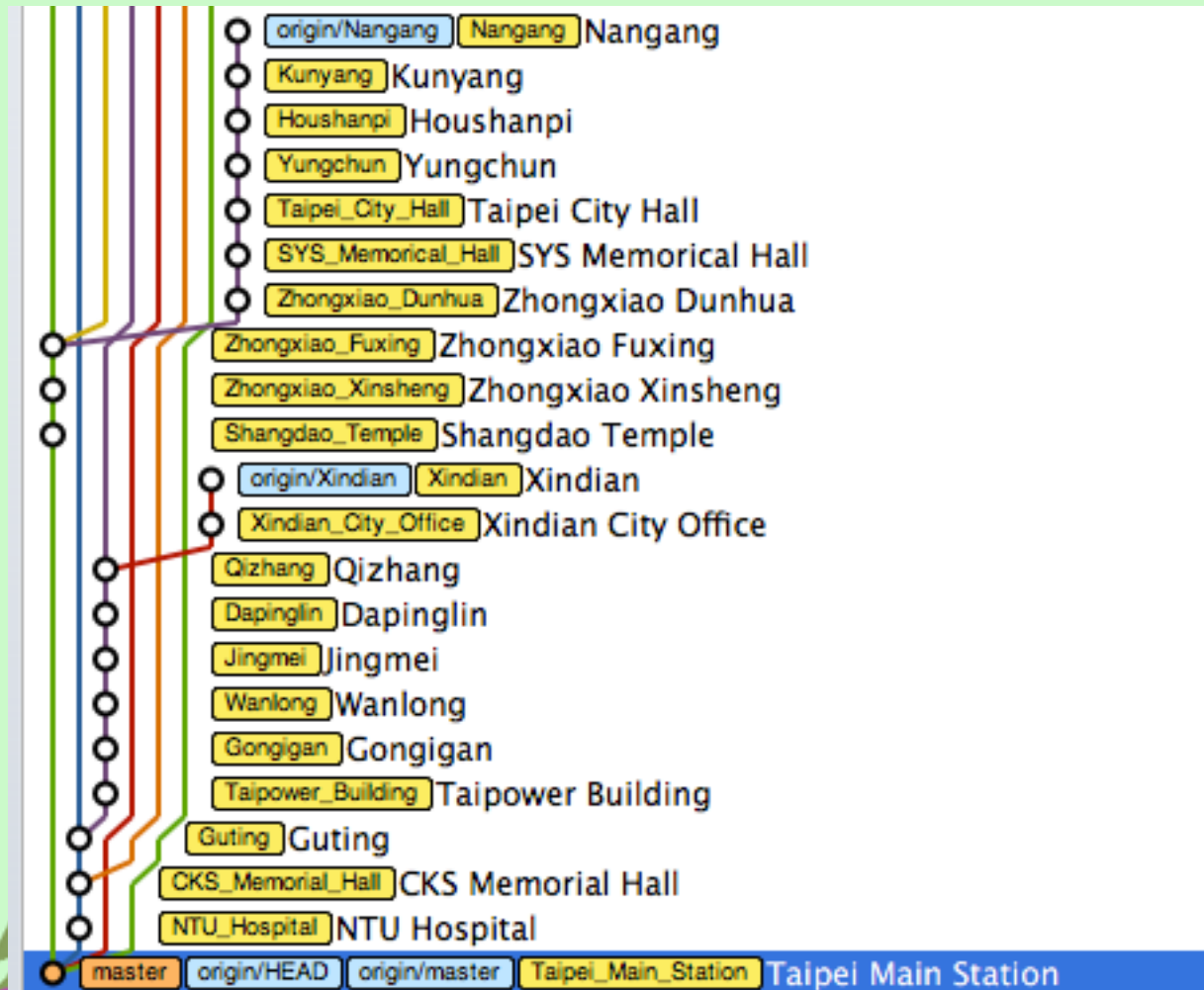
Kent Beck

finally figuring out that git commands are
strangely named graph manipulation
commands--creating/deleting nodes,
moving pointers around

1 Mar via [TweetDeck](#) ★ [Unfavorite](#) ↺ [Retweet](#) ↻ [Reply](#)

用 Git 表示台北捷運

<http://gugod.org/2009/12/git-graphing/>



3. Git 原理和內部運作



Source Control Made Easy

- <https://pragprog.com/screencasts/v-jwsceasy/source-control-made-easy>



Git SHA1 (demo)

- `echo sweet > sweet.txt`
- `git add .`
- `find .git/objects -type f`
- Git 內部儲存在 `.git/objects/aa/823728ea7d592acc69b36875a482cdf3fd5c8d`
- 這是 "blob" SP "6" NUL "sweet" LF 的 SHA1
- `printf "blob 6\000sweet\n" | shasum`
 - 或 `echo 'sweet' | git hash-object -w --stdin`
- `git cat-file -p aa823728ea7d592acc69b36875a482cdf3fd5c8d`



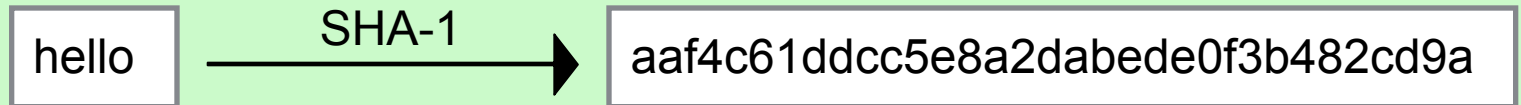
Git is Content Tracker

- Git is content-addressable filesystem
- SHA1 是根據內容產生的，跟檔名無關
 - 內容一樣的檔案，可以 reuse，例如 revert 的情況
 - 解耦合(decouple)了內容與當時的檔名
- Git 有四種 Objects 記錄內容：Blob、Tree、Commit、Tag



SHA-1

單向 Hash 雜湊加密演算法

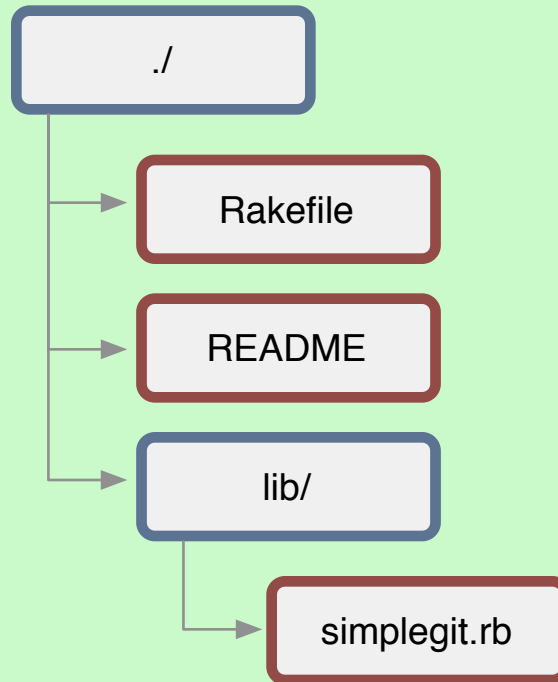


- 給定 message 容易計算出 hash
- 非常困難從 hash 推回 message
- 非常困難修改 message 而 hash 不變
- 非常困難有不同的 messages 而 hash 一樣
- hash 的分布很分散，跟 message 關聯不大



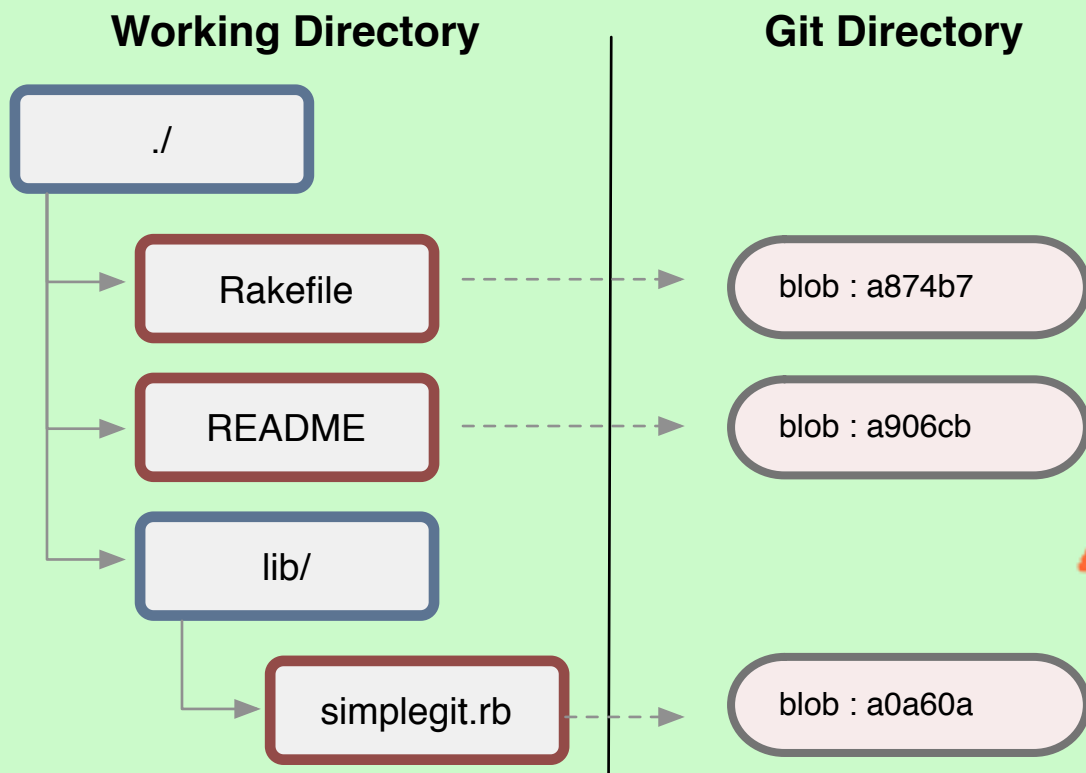
Git Object Database

Working Directory



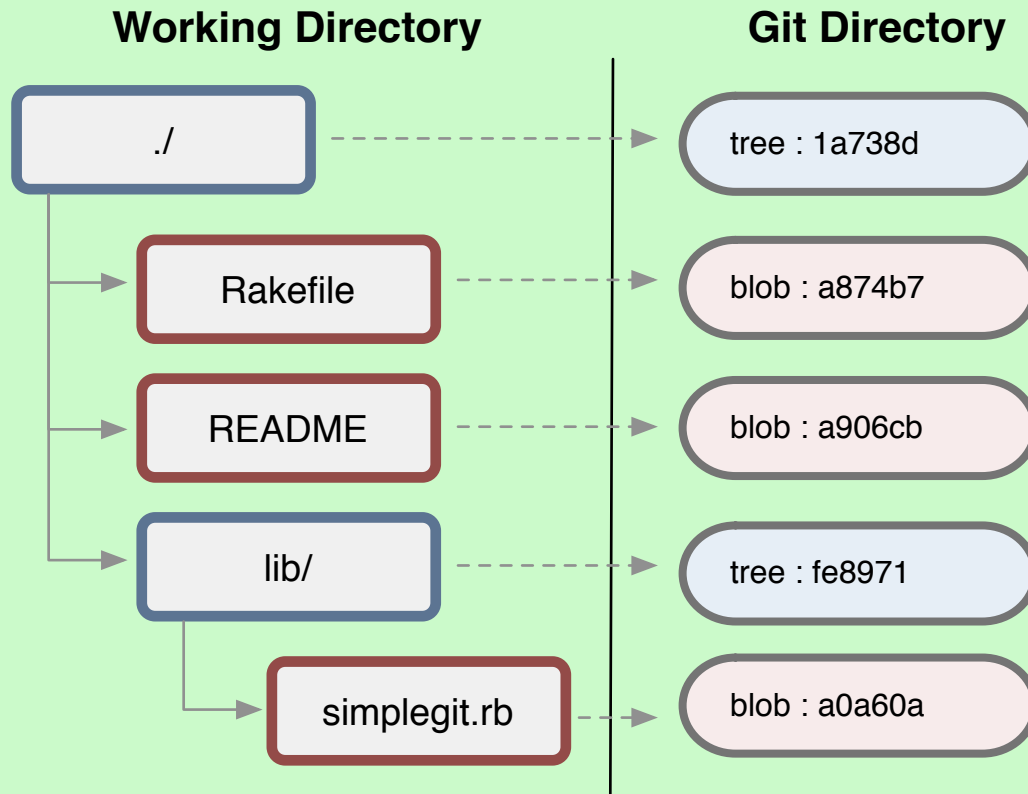
Blob

記錄檔案內容



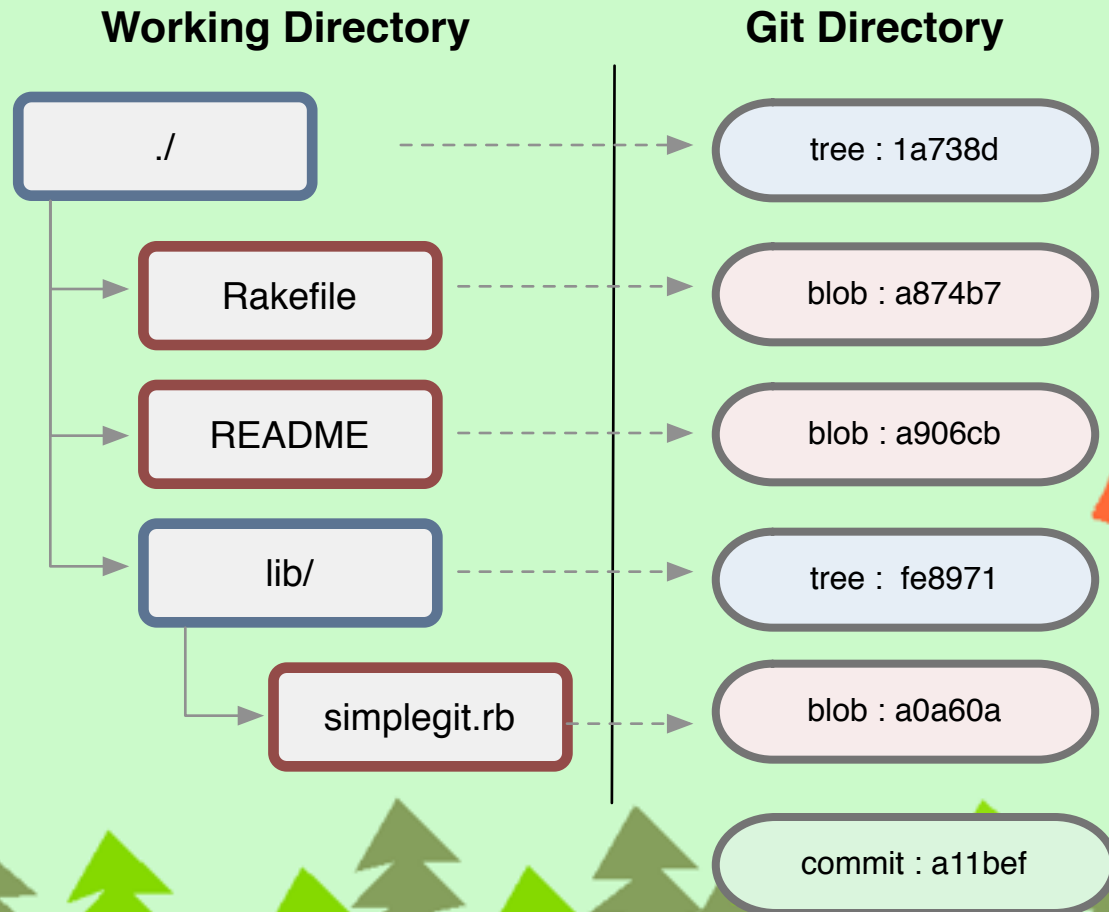
Tree

記錄該目錄下有哪些檔案(檔名、內容的SHA1)和Trees

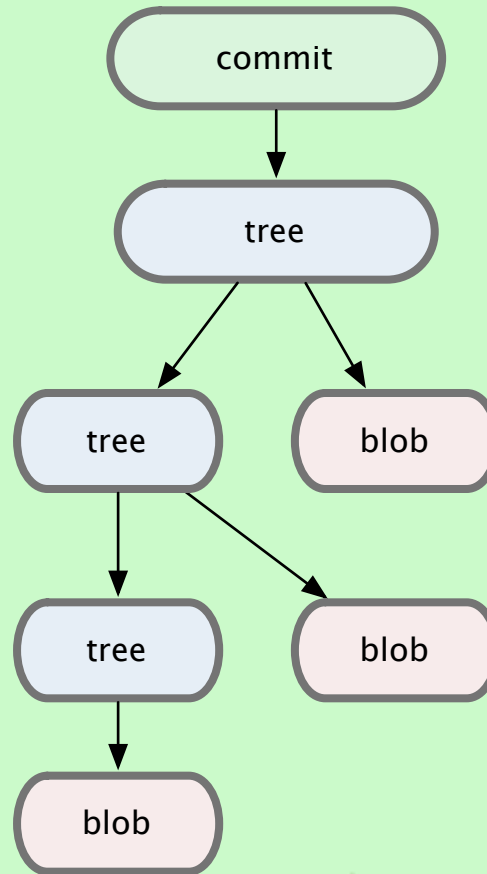


Commit

記錄 commit 訊息、Root tree 和 Parent commits 的 SHA1



Object DAG Example

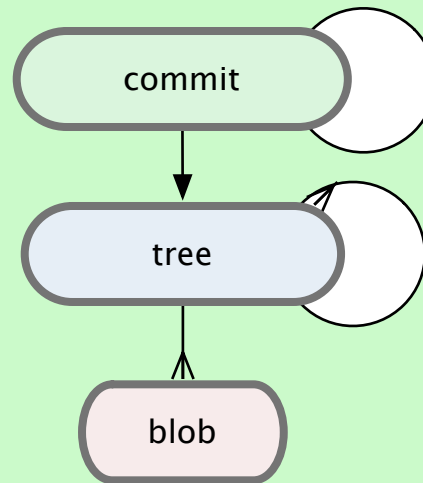


Trace Git commit (demo)

- 隨便抓一個 Commit 的 SHA1 開始：
 - `git cat-file -p a08181bf3`
(觀察這個 commit，找出 tree 位置)
 - `git cat-file -p ea44d629`
(觀察這個 tree，找出任一個 blob SHA1)
 - `git cat-file -p d9647d8a`
(觀察這個 blob 的內容)



Object DAG Model



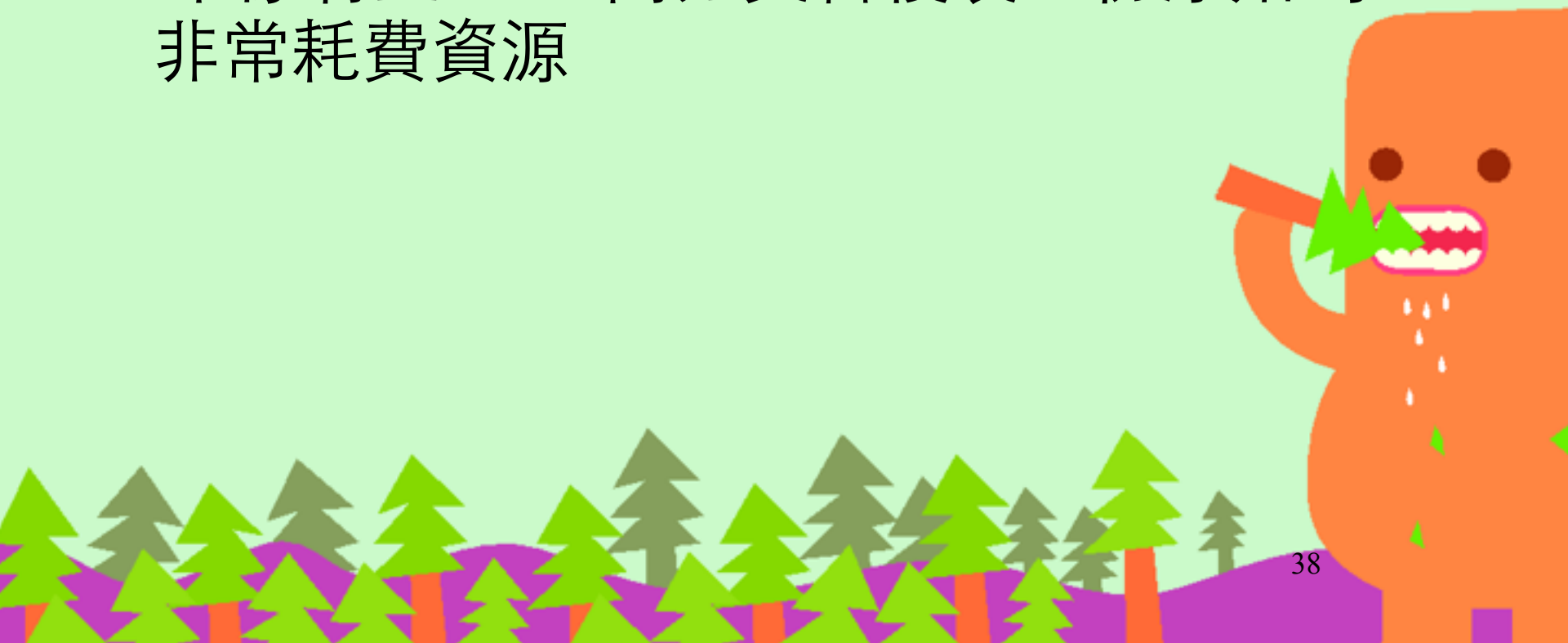
參照 refs

- Reference 會指向一個 Commit
- tag 不會移動，指向的 commit 都一樣
 - (帶有額外資訊的 tag 內部會用 Object 儲存)
- branch 指向該 branch 最新的 commit
- HEAD 指向 current branch



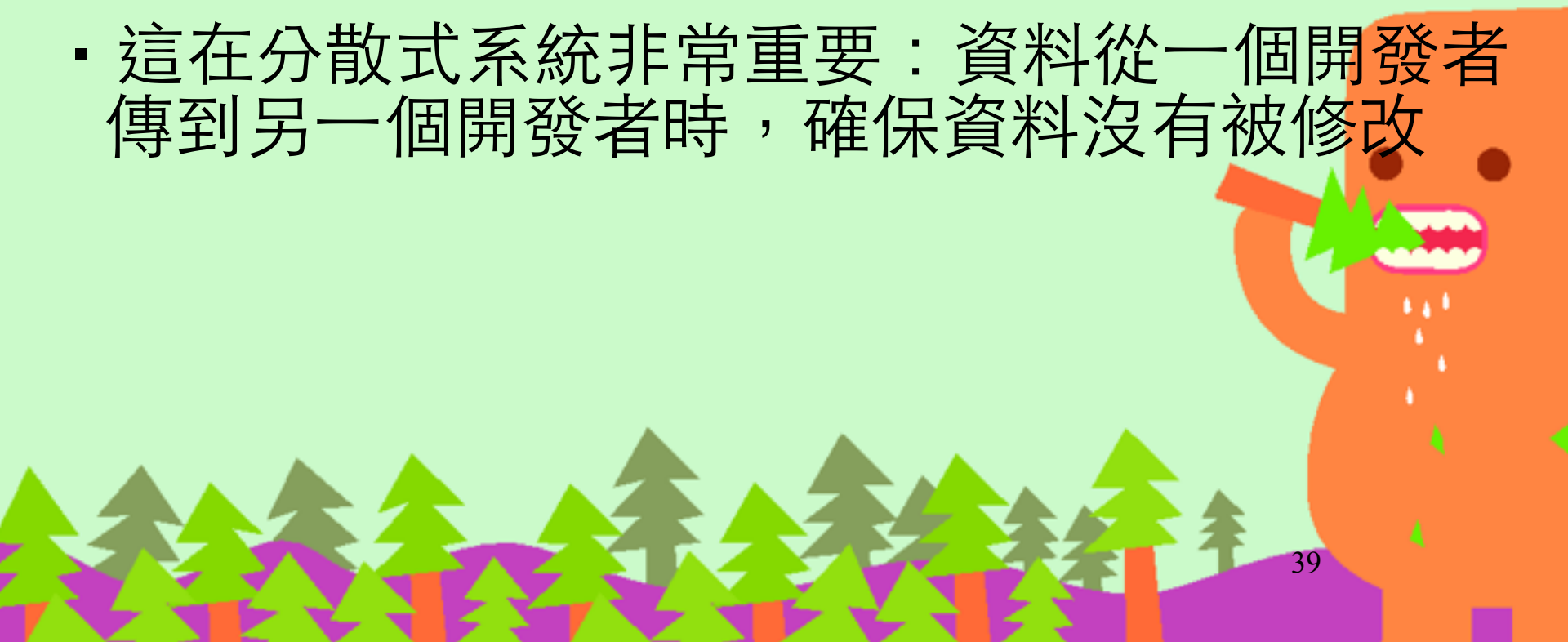
Reference is cheap!

- 開新 branch 和 tag 只不過是 refs 而已，直到真的有 commit 前都沒有什麼負擔
- 不像有些 CSV 開分支會複製一份原始碼，非常耗費資源

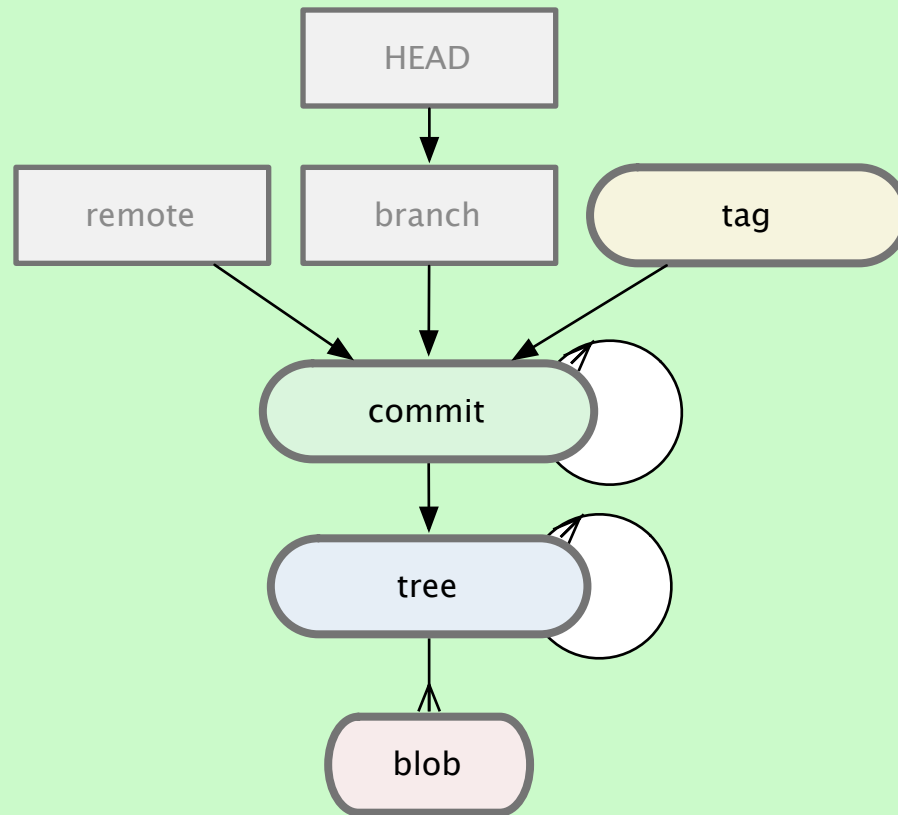


Integrity

- SHA1 是內容的 checksum
- Integrity: 如果檔案內容有損毀，就會發現跟 SHA1 不同。如果 tree 改檔名，也會被發現。
- 這在分散式系統非常重要：資料從一個開發者傳到另一個開發者時，確保資料沒有被修改



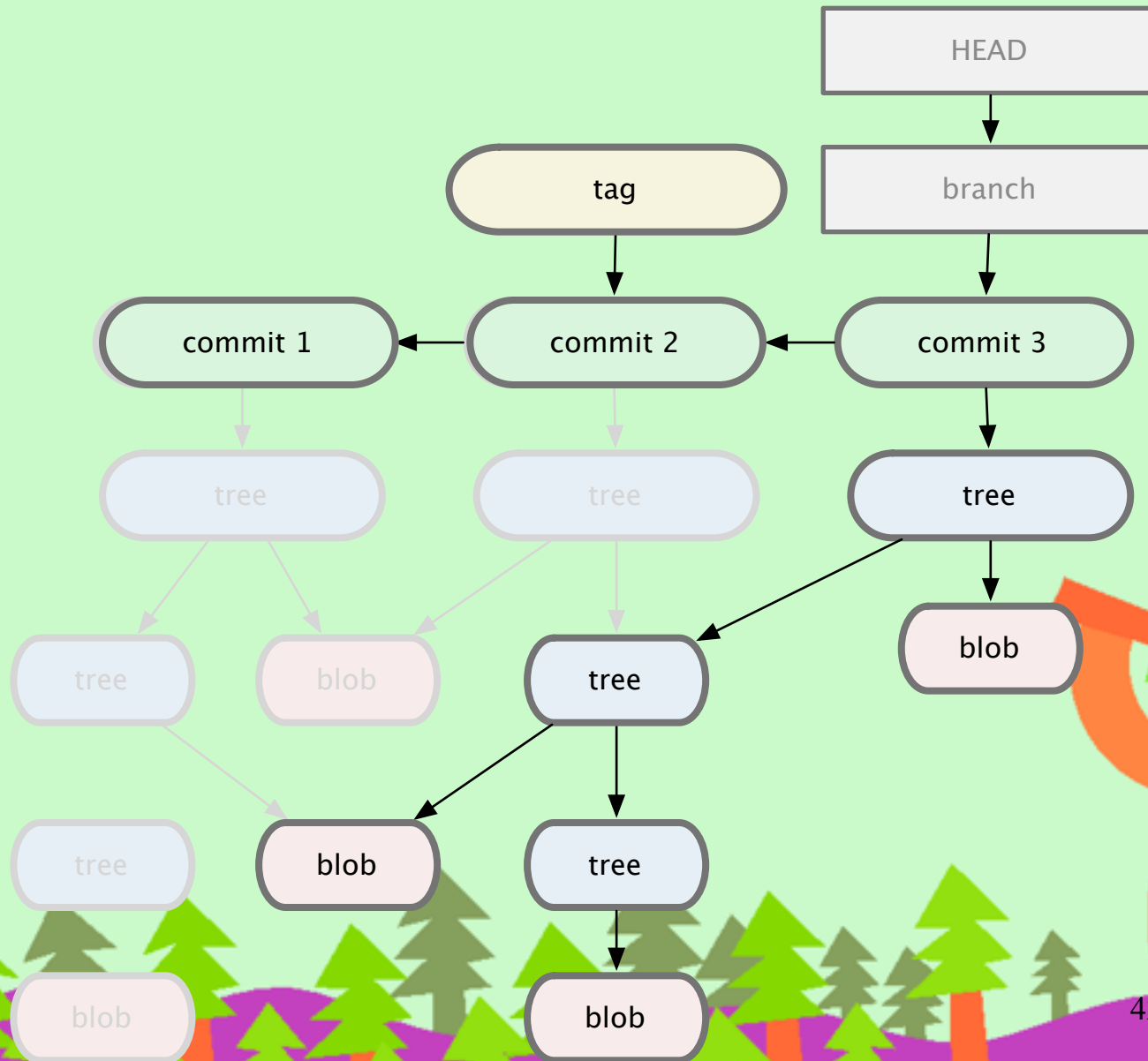
Object DAG Model



Trace Git branch (demo)

- `cat .git/HEAD` 拿到目前工作目錄 current branch 是指向哪一個 branch
- `cat .git/refs/heads/master` 拿到 master branch 指向的 commit
- `cat .git/refs/tags/foobar` 拿到 foobar tag 指向的 commit

Object DAG Full Example



自從用了 Git，就再也不會想用 SVN 了

- svn 開分支和 merge 超痛苦，能避免開 branch 就避免開 branch
- svn 看 log 超痛苦，需要等待網路連線
- svn commit 或 checkout 超慢，主機放國外常 commit 一半中斷



4-1. Git 指令基礎



Git is a collection of command-Line utilities

```
1. Shell
New Tab Info Close Execute
[~/sandbox] (master) ruby-1.9.3-p125$ git
add          fetch          relink
addall       filter-branch  remote
am          flow          repack
amend       format-patch  replace
annotate    fsck          request-pull
apply       g             reset
archive     gc            revert
bisect      get-tar-commit-id rm
blame       grep          rs
br          gui           rt
branch     help          send-email
bundle     imap-send    shortlog
checkout   init          show
cherry     instaweb     show-branch
cherry-pick l             smart-log
ci         log           smart-merge
citool     merge         smart-pull
clean      mergetool     st
clone      mv            stage
co         name-rev      stash
column     nb            status
commit     notes         submodule
config     p4            svn
cp         pair          tag
credential-cache pull          uncommit
credential-osxkeychain push        unstage
credential-store ra           whatchanged
describe   rc           who
diff       rebase
difftool   reflog
[~/sandbox] (master) ruby-1.9.3-p125$ git
```



Install Git

- 安裝 Git
 - <http://help.github.com/set-up-git-redirect>
- 安裝 Git GUI(optional)
 - GitX (Mac)
 - TortoiseGit/Git Extensions (Windows)
 - gitq, gitk (Linux)



Setup

- 編輯 ~/.gitconfig 或輸入以下指令
- `git config --global user.name "Your Name"`
- `git config --global user.email "your@email.com"`
- `git config --global color.ui true`
- `git config --global core.editor "vi"`



Setup (Windows only)

- 處理換行字元誤認為變更
- `git config --global core.autocrlf true`
- `git config --global core.safecrlf true`



從頭建立 Repository (demo)

- mkdir sandbox
- cd sandbox
- git init



第一次 commit (demo)

- touch README
- git add README
- git status
- git commit -m "First Commit"

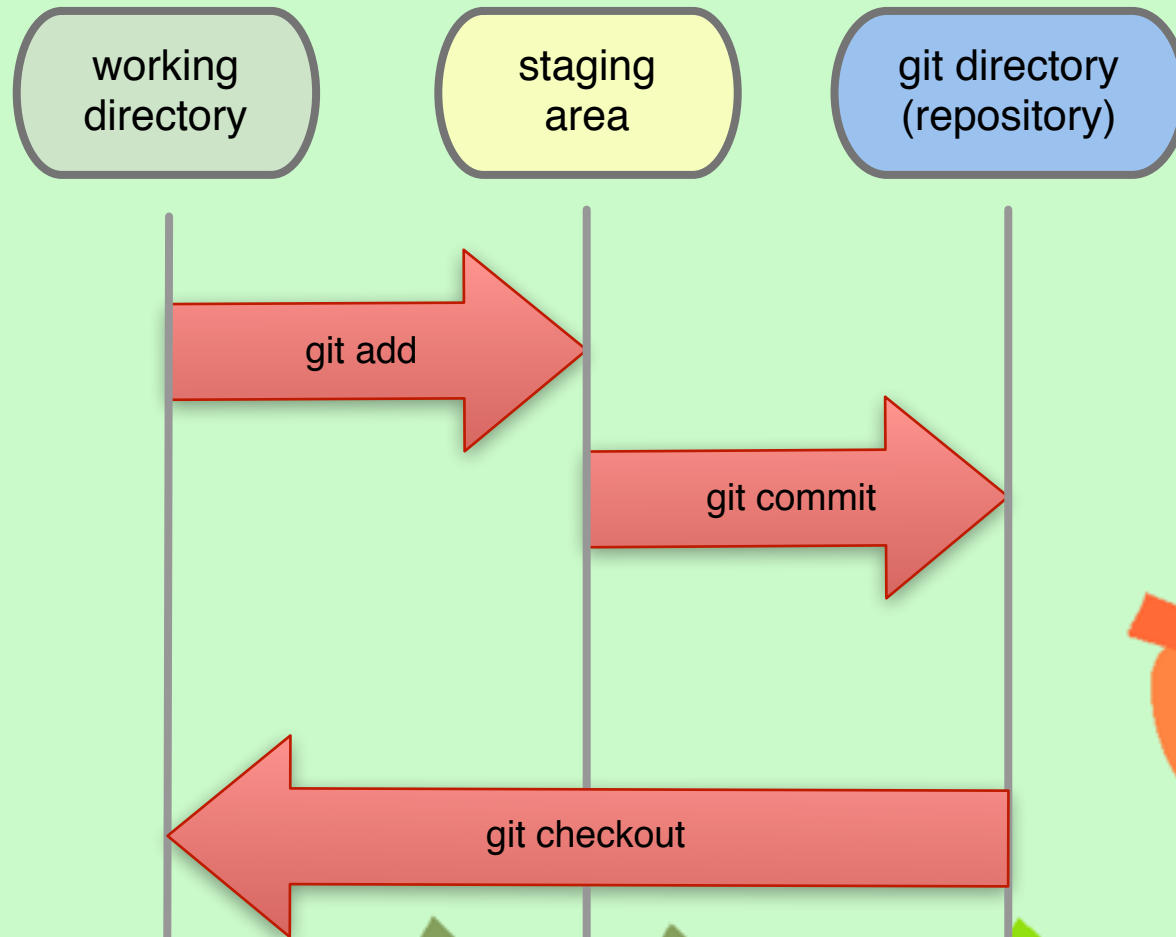


修改看看 (demo)

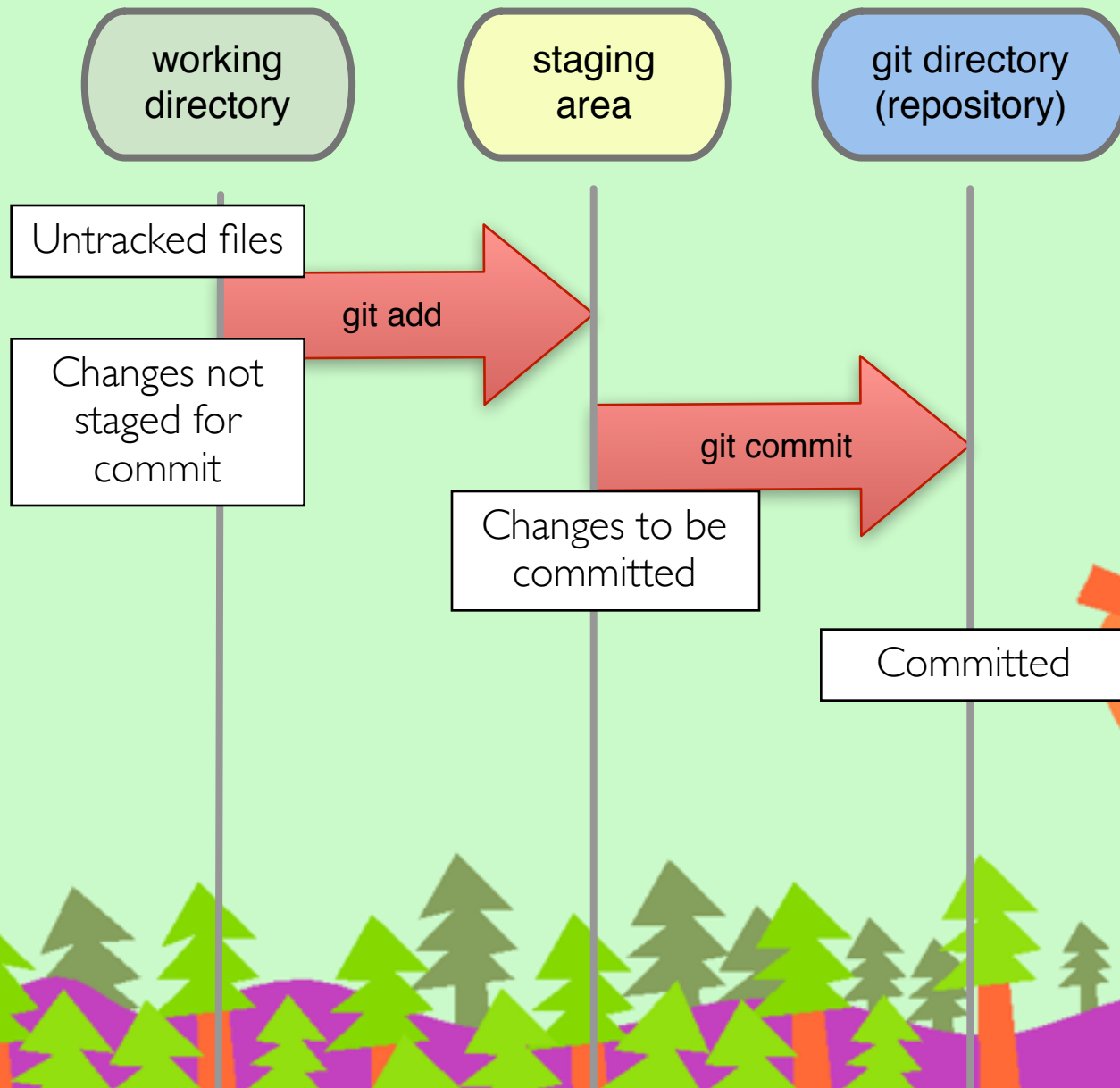
- 編輯 README 做些變更
- `git status`
- `git diff`
- `git add .`
(一次加入所有變更跟新增檔案，但不包括刪除的檔案!)
- `git status`
- `git diff --cached`
- `git commit -m "Update README"`



三種區域: Working tree/Staging Area/Repository



Working tree 裡的檔案有四種狀態



Why? Staging Area

- Working tree 可能很亂，包含了想要 commit 的內容和不相關的實驗修改等
- Staging area 的設計可以讓你只 commit 想要的檔案，甚至是想要的部份修改而已
- Staging Area 又叫作 Index



只 commit 部分檔案 (demo)

- touch a.rb
- touch b.rb
- git add a.rb
- git commit "Update a"
- git add b.rb
- git commit "Update b"



staging 之後又再修改內容 (demo)

- 修改 a.rb
- `git add a.rb`
- 再次修改 a.rb
- `git commit -m "commit a"`
 - 這時 commit 的內容是第一次修改當時的內容而已

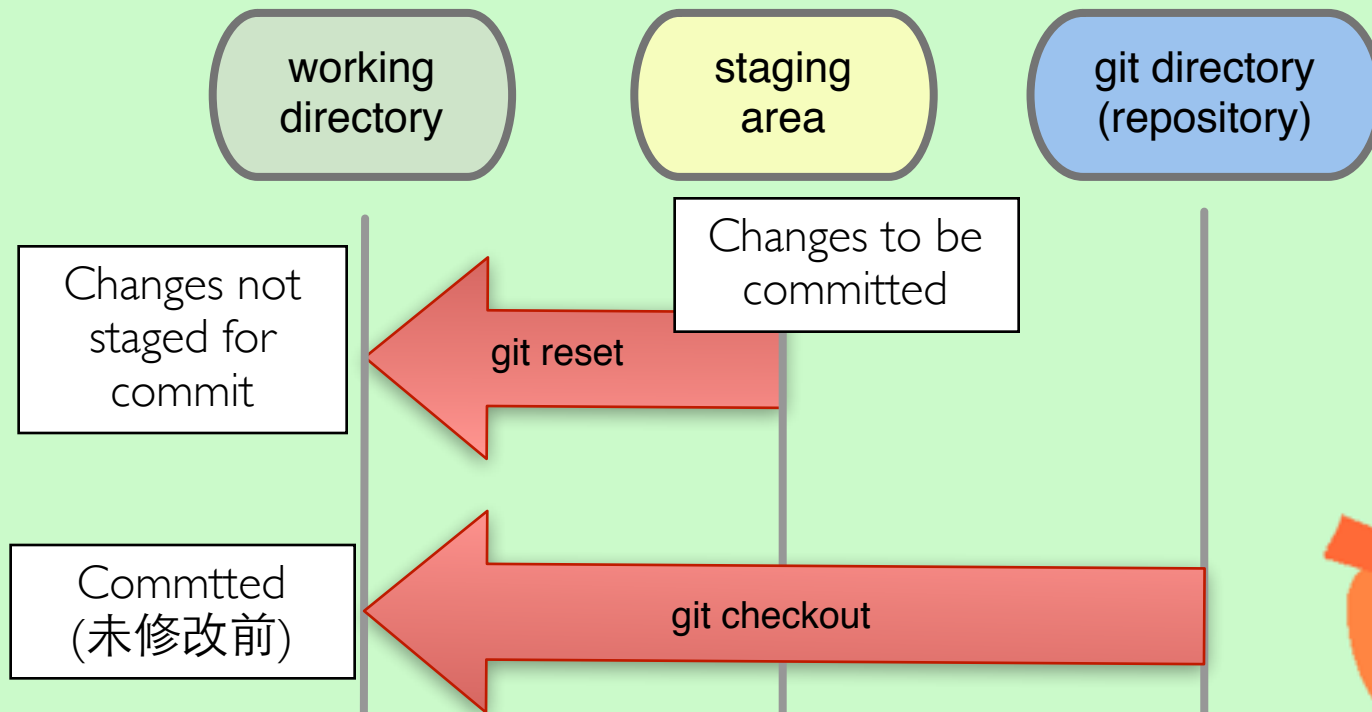


只 commit 同一檔案部分內容 (demo)

- `git add --patch`
 - y 加到 staging
 - n 不要加到 staging
 - s 可以拆小一點 hunk
- 或者用 GUI 例如 `gitx` 來選取



在 commit 之前如何復原修改? (demo)



刪除和搬移檔案 (demo)

- `git rm a.rb`
- `git mv b.rb c.rb`
- `git add .`
- `git commit "Remove a, Rename b to c"`
- 沒有 copy ? 因為 Git 是追蹤內容 (相同內容SHA1相同) , 你只要 cp 即可 , 不用擔心浪費空間。

revert (還原 commit 記錄)

- 新增一筆 commit 來做還原
 - 例如本來的 commit 是新增一行，那麼 revert commit 就會移除那一行
- `git revert e37c75787`
- `git revert HEAD^`



下標籤 (demo)

- `git tag foo`
- `git tag foo <SHA1>`
- `git tag bar -m "some message"`
- `git tag`
- `git tag -d foo`



歷史紀錄

- `git log`
- `git log --oneline`
- `git log --oneline --decorate --graph`
- `git log` 很多參數可以用，但是建議還是用 GUI 吧...



比較差異 Diff

- `git diff <SHA1>` 拿 Working Tree 比較
- `git diff <SHA1> <SHA1>`
- `git diff --stat <SHA1>`
- `git diff --cached` 或 `git diff --staged`
拿 Staging Area 來比較



砍掉 untracked 檔案 (demo)

- `git clean -n` 列出打算要清除的檔案
- `git clean -f` 真的清除
- `git clean -x` 連 `gitignore` 裡列的檔案也清掉



忽略不需要追蹤的檔案

- 編輯 `.gitignore` (屬於專案的設定，會 commit 出去)
- 編輯 `~/.gitignore` (你個人的 Global 設定)
- 空目錄不會 commit 出去，必要時需要放 `.gitkeep` 檔案
- `.gitignore` 大集合
<https://github.com/github/gitignore>



通常什麼檔案 不需要 commit 出去?

- tmp/*
- log/*
- 你個人環境的設定檔(例如你偏愛的編輯器設定檔)
- 可以自動產生的檔案
- build/* 等 compile 之後的檔案
- .DS_Store
- Thumbs.rb



Commit 基本原則

- 適當的粒度/相關性/獨立性
 - 以一個小功能、小改進或一個 bug fixed 為單位。
 - 對應的 unit test 程式在同一個 commit
 - 無相關的修改不在同一個 commit
 - 語法錯誤的半成品程式不能 commit
- git diff --check 可以檢查多餘的空白
- commit 訊息很重要
 - 第一行寫摘要
 - 有需要寫實作細節的話，放第二行之後



4-2. Git 指令：開分支

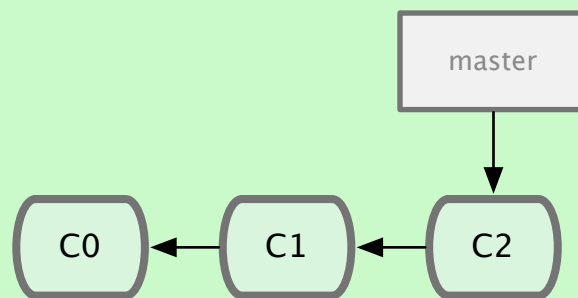


何時開 Branch ?

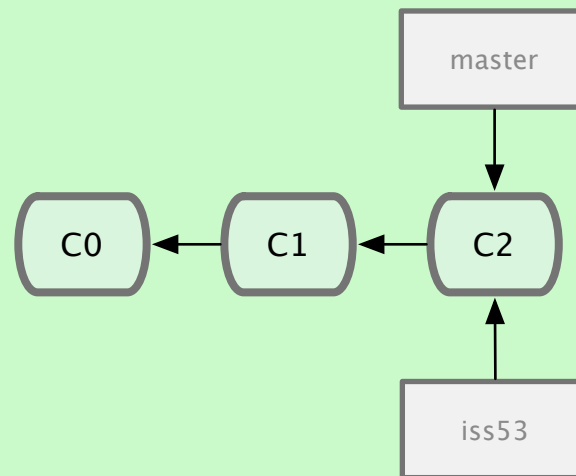
- Topic feature 開發新功能
- Bug fixes
- 重構 (refactor)
- 任何實驗



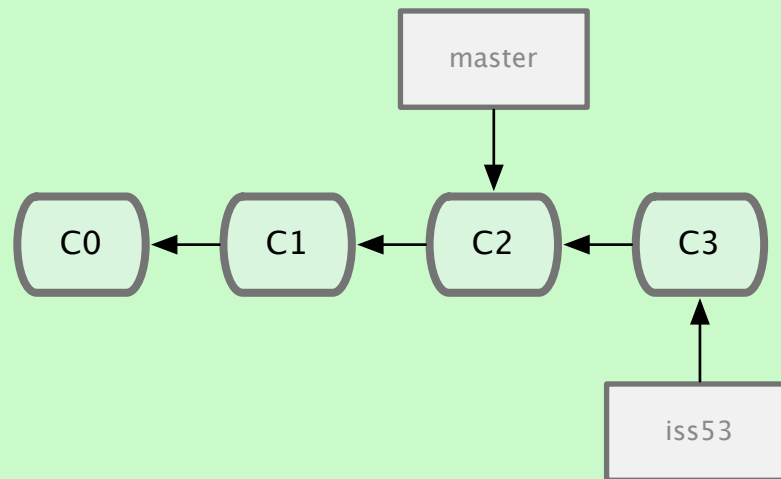
一個流程範例



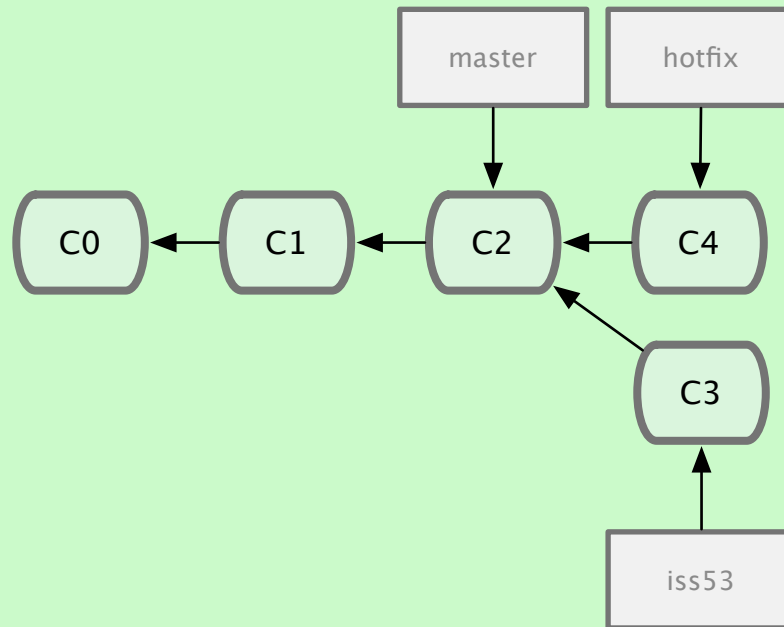
(master) git branch iss53



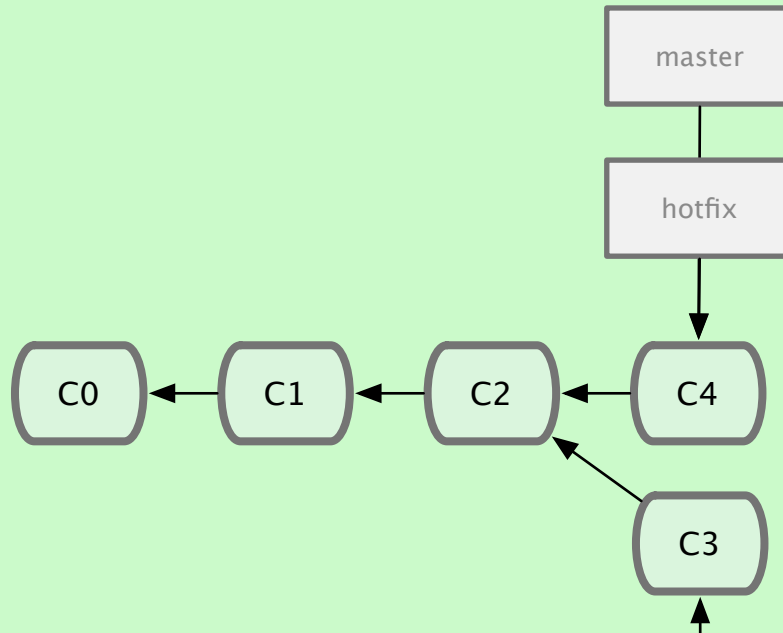
(iss53) git commit




```
(hotfix) git commit
```

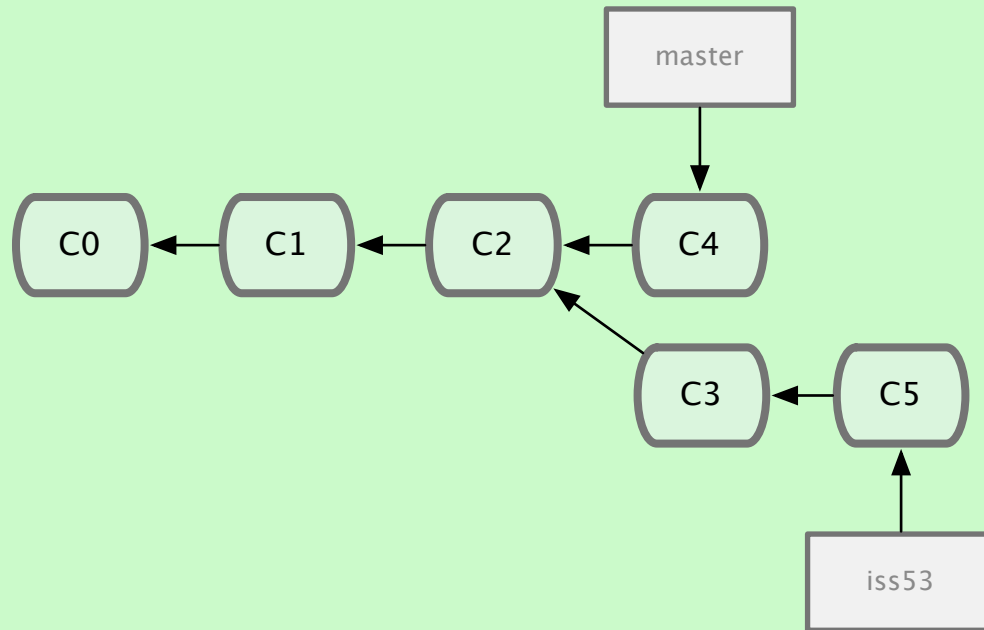


(master) git merge hotfix

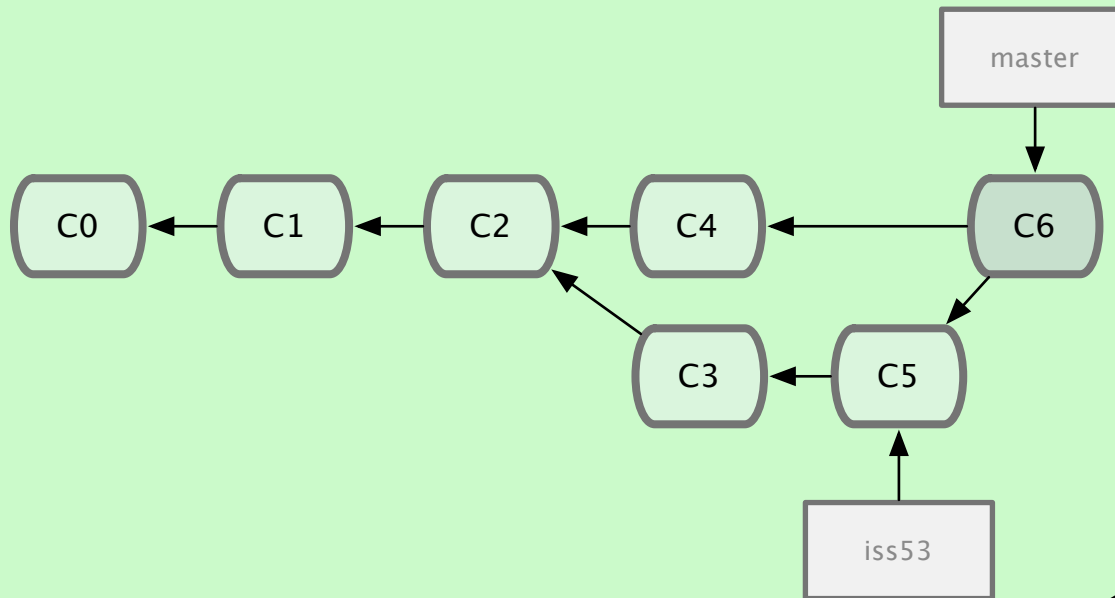


這種只需要改變參考的合併叫作
fast-forward

(iss53) git commit



(master) git merge iss53

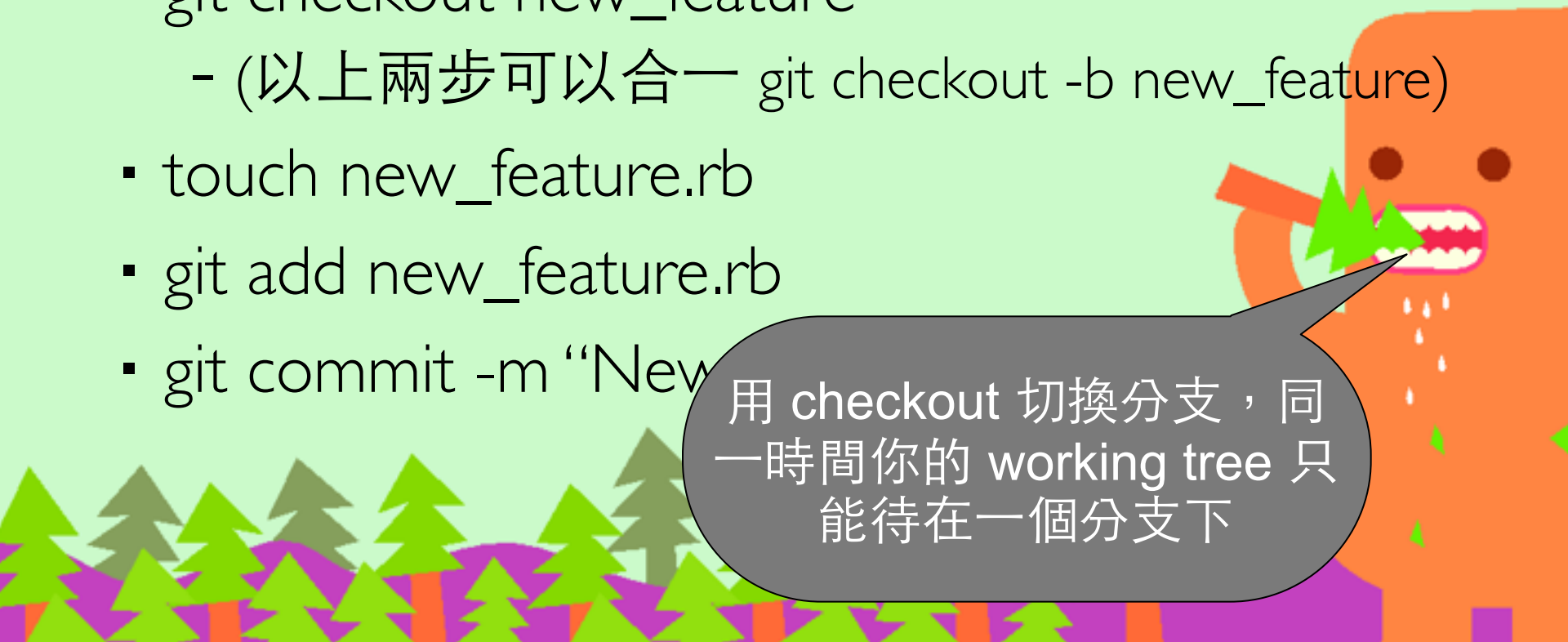


C6 是個 merge commit
(會有兩個 parent commits)

開 branch (demo)

- `git branch new_feature`
- `git branch` 或 `git branch <SHA1>`
- `git checkout new_feature`
 - (以上兩步可以合一 `git checkout -b new_feature`)
- `touch new_feature.rb`
- `git add new_feature.rb`
- `git commit -m "New"`

用 checkout 切換分支，同一時間你的 working tree 只能待在一個分支下



合併 branch 回來 (demo)

- `git checkout master`
- `git merge new_feature`



Merge CONFLICT 怎麼辦?

- `git status` 看哪些檔案有 conflict
- 手動編輯這些檔案，解決 <<<<< =====>>>>>> 有衝突的部分
- 或是 `git mergetool` 用工具解決
- `git reset HEAD --hard` 放棄合併



`git reset HEAD --hard`
超好用，可將 working tree
全部復原

Branch 更名和刪除

- `git branch -m old_name new_name`
- `git branch -M old_name new_name` (強制覆蓋)
- `git branch new_feature -d`
- `git branch new_feature -D` (強制刪除)



四種合併方式

- Straight merge 預設的合併模式
 - 保留被合併的 branch commits 記錄，並加上一個 merge commit，看線圖會有兩條 Parents 線。
 - 如果是 fast-forward，就不會有 merge commit。除非加上 --no-ff 參數。



四種合併方式 (cont.)

- Squashed commit
 - `git merge new_feature --squash`
 - 壓縮成只有一個 merge-commit，不會有被合併的 log。SVN 的 merge 即是如此。
- cherry-pick
 - `git cherry-pick 714cba`
 - 只合併指定的 commit
 - -x 參數會在 log 裡加注本來是哪個SHA1 (適合用在 backpointing 情境)



使用 branch 注意事項

- 切換 working tree 的 branch 時，如果有檔案在 staging area 或是 modified，會無法切換。
- 可以先 commit，反正只要不 push 出去，再 reset 回來即可
- 或是用 stash 指令暫存起來
 - git stash
 - git stash apply
 - git stash clear



Branches 當做存檔(savepoint)

- git branch is cheap!
- 建立 branch 即可保留當時的 Graph
- 就像遊戲存檔



安全合併法: Scout way

- 先建立一個 test_merge
- git checkout -b test_merge
- git merge XXX
- 不喜歡就重來
- 如果喜歡就回 git checkout master
- git merge test_merge 此時會 fast-forward



安全合併法: Savepoint way

- git branch savepoint
- git merge XXX
- git branch -d savepoint
- 如果不喜歡想重來
 - git reset savepoint --hard



無名 Branch 狀態

- `git checkout <branch_name>` 前往特定 Branch
- `git checkout <SHA1>` 則會前往特定節點，此時會是 detached HEAD 狀態，一種沒有名字的 current branch 狀態
- `git checkout -b new_branch` 就會給一個名字

什麼是 Three-way merge

- two-way merge 只用兩個檔案進行合併 (svn 預設即 two-way merge)
- three-way merge 除了要合併的兩個檔案，還加上兩個檔案的共同祖先。如此可以大大減少人為處理 conflict 的情況。



Git merge strategy

- recursive，當共同的祖先不只一個 commits 時，遞迴式的進行 three-way merge (此為預設)
 - `git merge foobar -s recursive -X ours`
 - `git merge foobar -s recursive -X theirs`
- resolve，直接挑一個共同的祖先進行 three-way merge
- octopus，當合併超過兩個以上的 branches 時
- ours，只用自己的 branch 的 commits
- subtree 合併成一個子目錄

Git 可以一次合併多個 Branchs

git merge A B C

```
recursive Merge branch 'c' into recursive
[c] from the c branch
Merge branch 'b' into recursive
[b] from the b branch
[a] from the a branch
from the main branch

SHA: 804dd695fffad2a04c3d75d72137
Author: Dustin Sallings <dustin@spy.net>
Date: Sun Dec 14 2008 13:01:54 GMT-
Subject: Merge branch 'c' into recursive
Refs: recursive

Merge branch 'c' into recursive

* c:
  from the c branch
```

```
Subject
[master] Merge branches 'a', 'b' and 'c'
[c] from the c branch
[b] from the b branch
[a] from the a branch
from the main branch

SHA: ae632e99ba0ccd0e9e06d09e864
Author: Dustin Sallings <dustin@spy.net>
Date: Sun Dec 14 2008 12:59:41 GMT-
Subject: Merge branches 'a', 'b' and 'c'
Refs: master

Merge branches 'a', 'b' and 'c'

* a:
  from the a branch

* b:
  from the b branch

* c:
  from the c branch
```

<http://stackoverflow.com/questions/366860/when-would-you-use-the-different-git-merge-strategies>

4-3. Git 指令：遠端操作



以 GitHub Repository 為例 建立一個專案

- 產生 SSH Key
 - `ssh-keygen -t rsa -C "your_email@youremail.com"`
- 至 Github 開一個專案
- `git remote add origin git@github.com:your_account/sandbox.git`
- `git push -u origin master`
- 之後只需要 `git push`
 - 出現 `![rejected]` 表示需要先做 `git pull`



SSH

Secure Shell

- 取代 telnet 通訊協定，提供加密通訊
- 使用 public-key cryptography 公開金鑰加密 (非對稱加密)



拷貝從一個已經存在的 Repository

- SSH 安全性最佳
 - `git clone git@github.com:ihower/sandbox.git`
- HTTP/HTTPS 速度最差，但能突破防火牆限制
 - `git clone https://ihower@github.com/ihower/sandbox.git`
- Git protocol 速度最快，但缺認證機制(因此只能做成唯讀)
 - `git clone git://github.com/ihower/sandbox.git`
- File 本機目錄 (有人用 Dropbox 分享 `git init --bare --shared` 目錄!! Crazy!!)
 - `git clone file://path/to/repo.git`

Clone 別人的專案

- 如果有讀寫權限，使用 SSH 協定
 - `git clone git@github.com:ihower/sandbox.git`
- 如果只有讀取權限，使用 Git 協定
 - `git clone git://github.com/ihower/sandbox.git`
- 如果有防火牆問題，用 HTTPS 協定
 - `git clone https://github.com/ihower/sandbox.git`



Pull 從遠端更新

- `git pull origin master` 或 `git pull`
 - 實際作用是先 `git fetch` 遠端的 branch
 - 然後與本地端的 branch 做 merge，產生一個 merge commit 節點



Push 將 Commit 送出去

- `git push` 或 `git push origin master`
 - 實際的作用是将本地端的 master branch 與遠端的 master branch 作 fast-forward 合併
- 如果出現 [rejected] 錯誤的話，表示你必須先作 pull。
- 預設不會 push tags 資訊
 - `git push --tags`



列出和取出遠端 branch

- `git branch -r`
- `git checkout -t origin/foobar`
- `git branch -r --merged` (列出已經被合併的)



刪除遠端 Branch

- `git push origin :foobar`
- `git pull` 和 `git fetch` 不會清除已經被刪掉的 branch
 - 請用 `git fetch -p`



4-4. Git 指令：還沒 push 前可以做的事



還沒 push 分享出去的 branch，你 可以...

- 不同於 SVN，Git 的 Commit 其實還在本地端，所以可以修改 commit logs!!
 - 修改 commit 訊息，甚至內容
 - 合併 commits
 - 打散 commit 成多個 commits
 - 調換 commits 順序



reset (回到特定節點)

- 跟 revert 不同，reset 是直接砍掉 commits 歷史記錄
- `git reset e37c75787`
- `git reset HEAD^` (留著修改在 working tree)
- `git reset HEAD^ --soft` (修改放到 staging area)
- `git reset HEAD^ --hard` (完全清除)



其實剛剛的安全合併法 savepoint way 不需要建立 branch

- 直接 `git reset <SHA1> --hard` 即可
- branch name 只是 reference



快速修正上一個 Commit

- 上一個commit有typo，想要重新commit
 - `git reset HEAD^`
 - 修正
 - `git commit -am "message"`
- 可以簡化成
 - 修正
 - `git commit -a --amend`



rebase (重新 commit 一遍)

- `git rebase -i e37c7578`

```
pick 048b59e first commit  
pick 995dbb3 change something  
pick aa3e16e changed
```

```
# Rebase 0072886..1b6475f onto 0072886
```

```
#
```

```
# Commands:
```

```
# p, pick = use commit
```

```
# r, reword = use commit, but edit the commit message
```

```
# e, edit = use commit, but stop for amending
```

```
# s, squash = use commit, but meld into previous commit
```

```
# f, fixup = like "squash", but discard this commit's log message
```

```
# x, exec = run command (the rest of the line) using shell
```

```
#
```

```
# If you remove a line here THAT COMMIT WILL BE LOST.
```

```
# However, if you remove everything, the rebase will be aborted.
```



rebase is dangerous!!
It rewrite history!

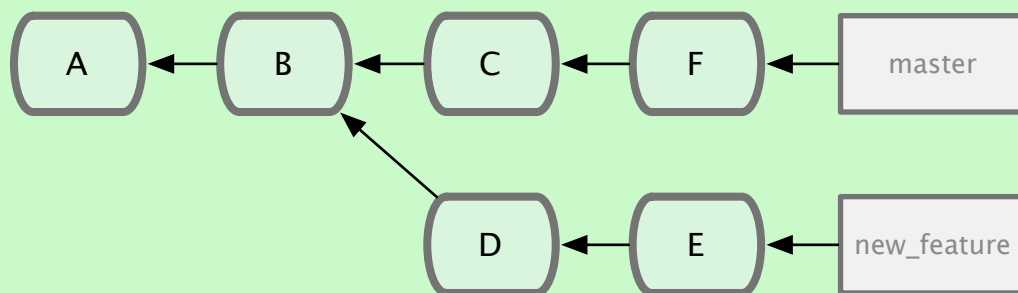


But I like it!

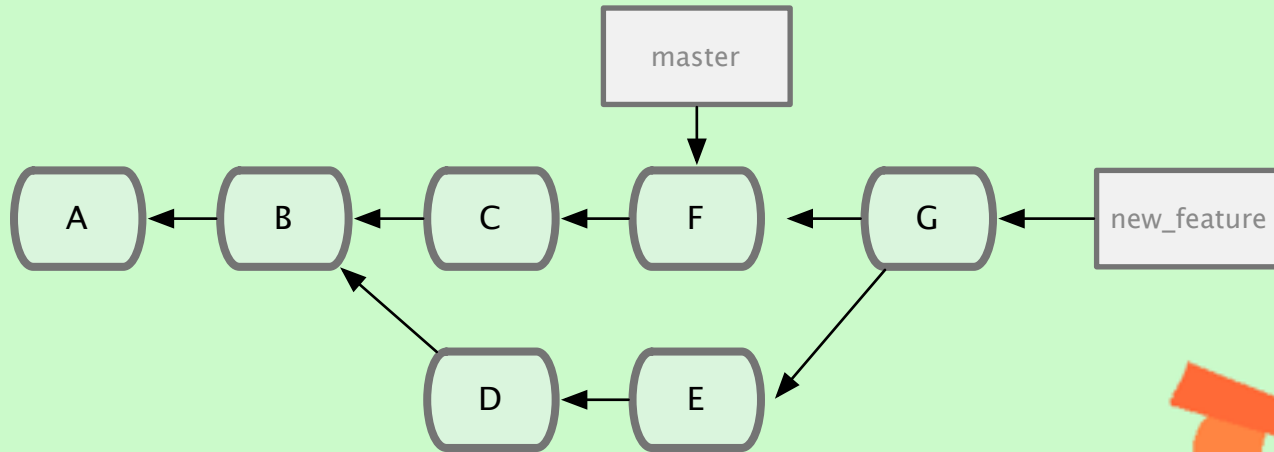
第四種合併方式: rebase

- rebase 調整分支點 (僅適用於 local branch)
 - 1. 從要被合併的 branch 頭重新分支
 - 2. 將目前 branch 的 commits 記錄一筆一筆重新 apply/patch 上去
 - 等於砍掉重練 commit log，僅適合還沒分享給別人的 local branch。

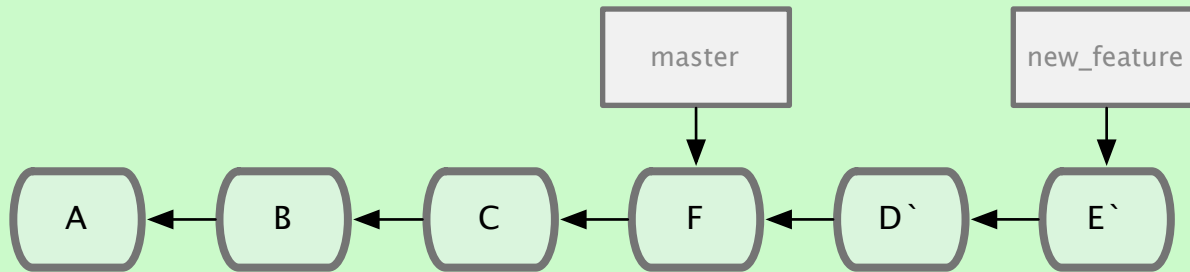
一個範例: 開發中的 new_feature 想要合併 master 主幹的變更



如果是 merge
(new_feature) git merge master



如果用 rebase，就沒有 G 點!
(new_feature) git rebase master



用這招可以不產生分支線和 merge commit

rebase notes

- 注意到 D` 和 E` 和本來的 D 和 E commits 已經被視為不同 commit，SHA1 也不同了
- rebase 過程中，每一次 apply 都有可能發生 conflict，跟 merge 只發生一次不同。
- 發生 conflict 怎麼辦? 可以選擇修好後 continue, 忽略 skip 或放棄 abort
 - 記得一定要離開 rebase 的 context 情境



已經 push 分享出去的 commits
記錄，請千萬不要再修改記錄
push 出去!!



集大成： rebase + merge 的 龜毛 合併法

(假設我們要將 feature branch 合併回主幹 develop)

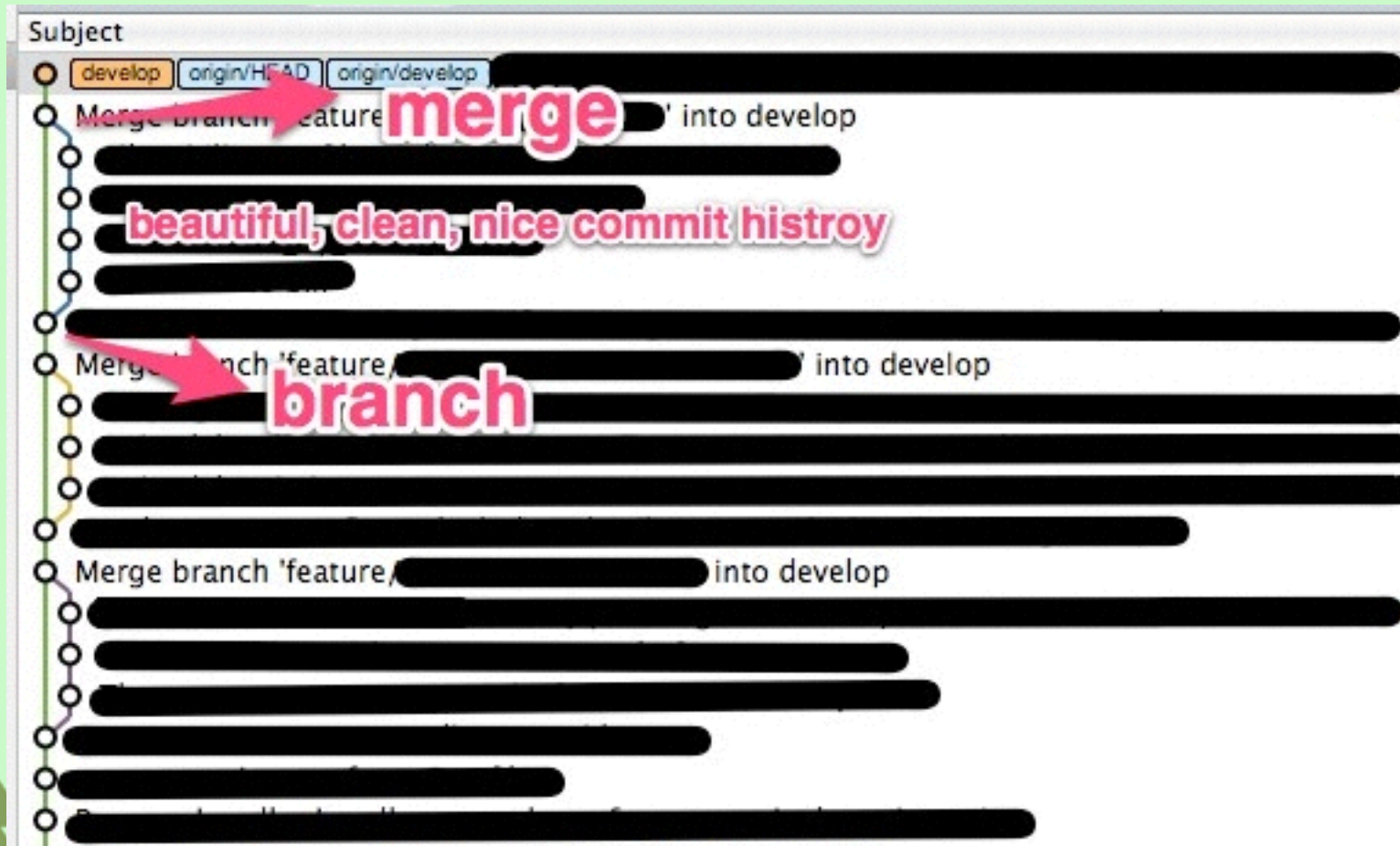
▪ 原因

- feature branch 很亂，不時 merge 與主幹同步
- feature branch 有 typo，commit 訊息想改
- feature branch 有些 commits 想合併或拆開

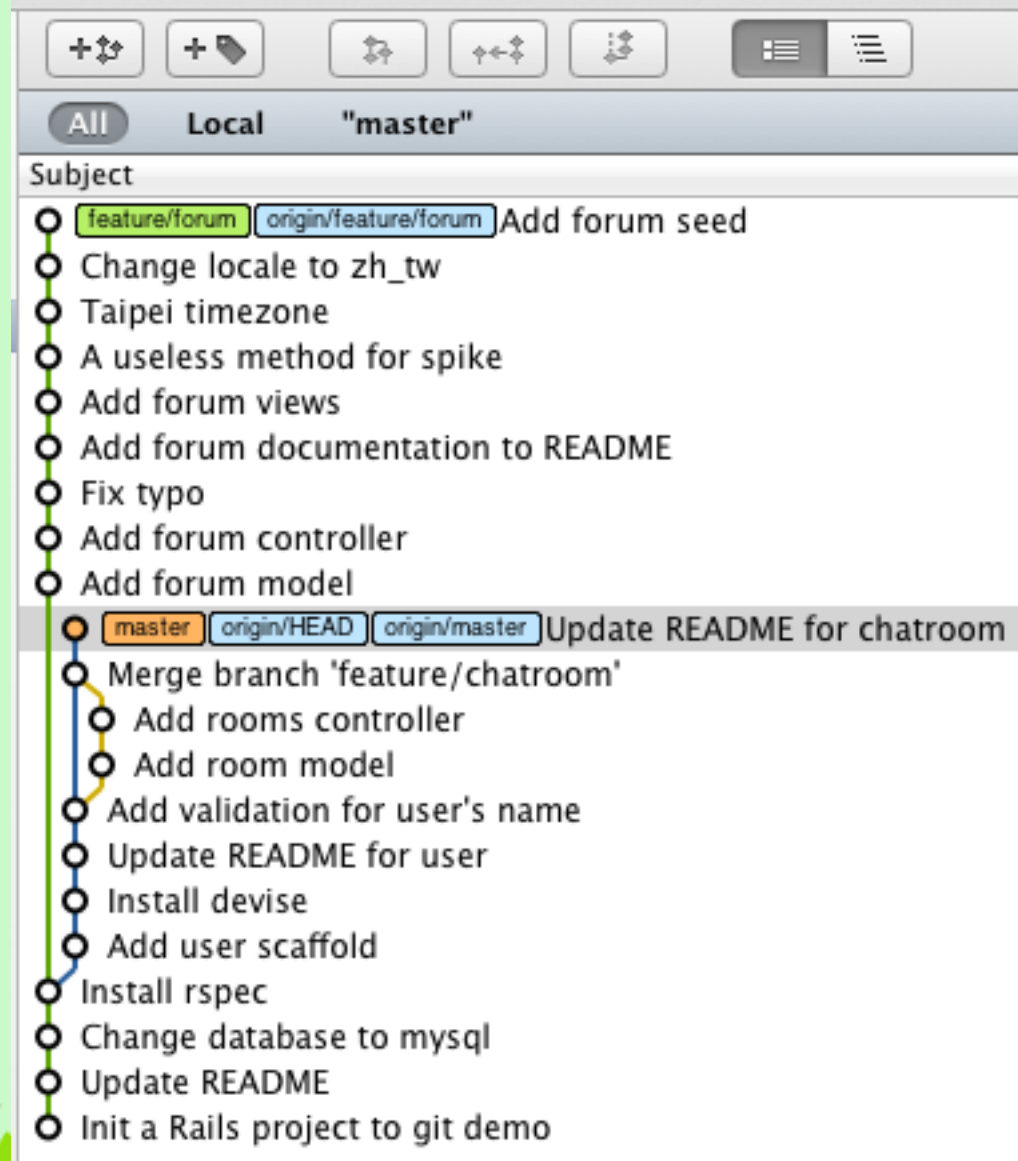
▪ 作法

- 先在 feature branch 做 git rebase develop -i
- (反覆整理直到滿意) git rebase 分岔點 -i
- 在從 develop branch 做 git merge feature --no-ff

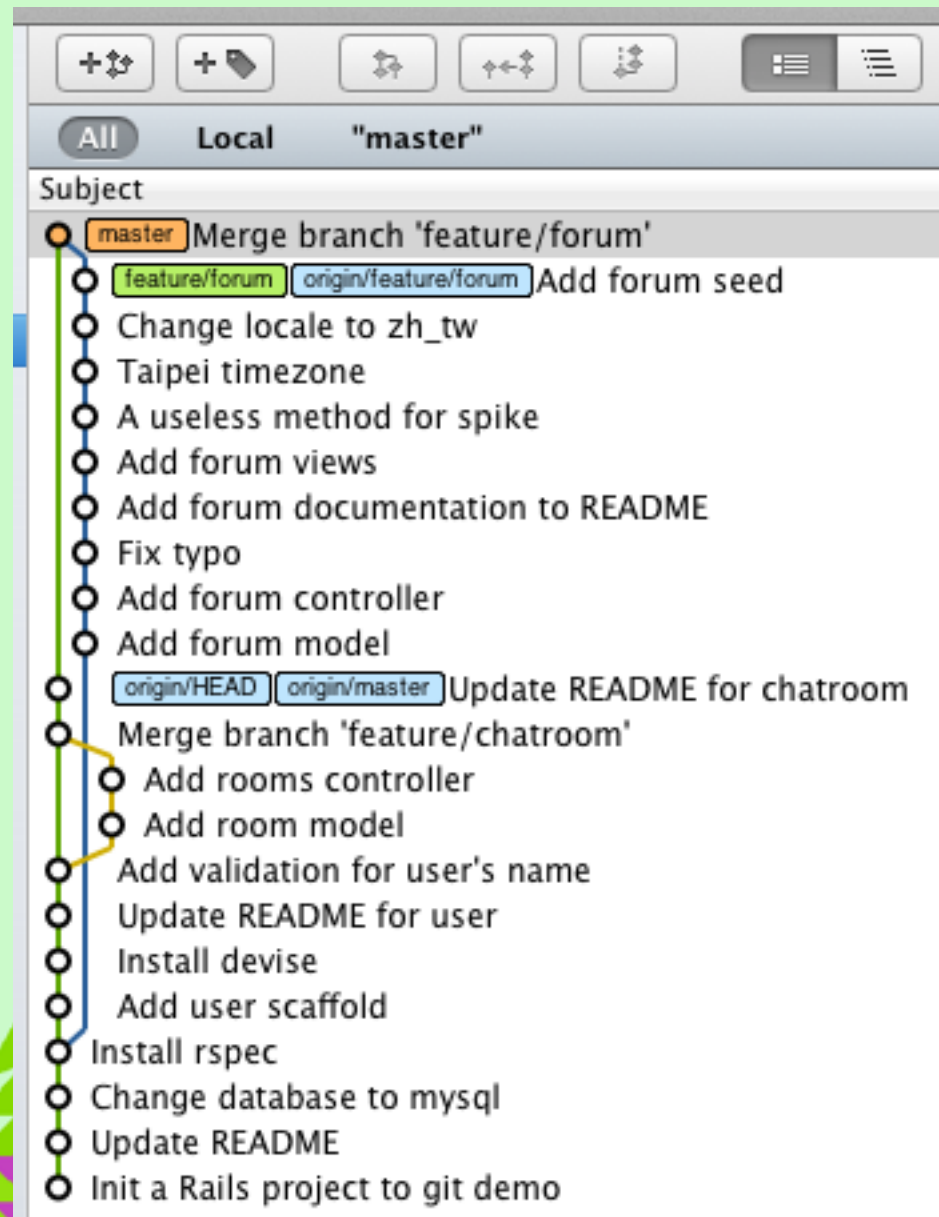
超級乾淨，每一次的 merge
commit 就代表一個功能完成



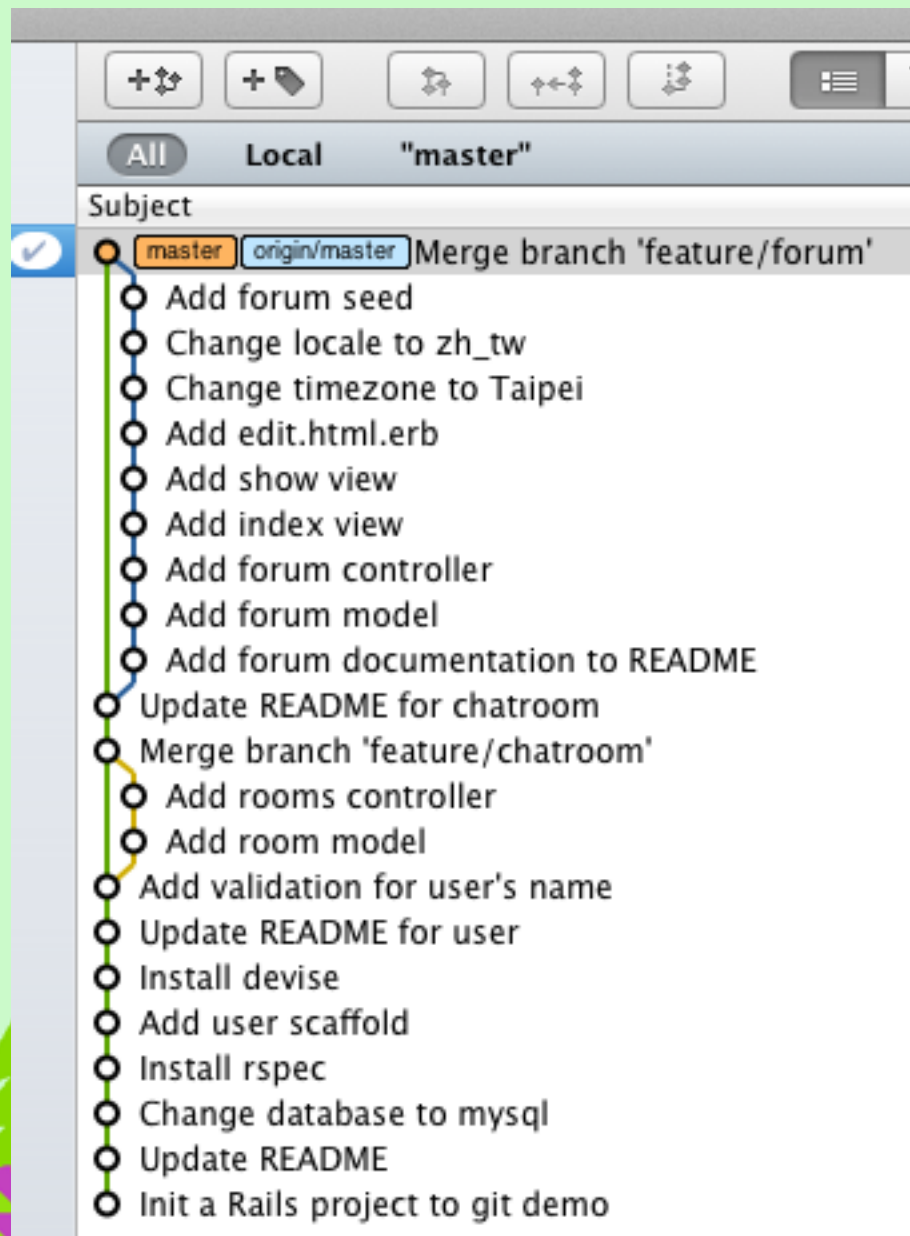
Demo (before merge)



Demo (normal merge)



Demo (rebase + merge)



Rebase demo screencast

- <http://ihower.tw/blog/archives/6704>



注意事項 (I)

- 必須要加 `--no-ff` 才會有 merge commit。不然會是 fast-forward。
- rebase 之後的 feature branch 就不要再 push 出去了
- 如果有遠端的 feature branch，合併完也砍掉

注意事項 (2)

- 不求一次 rebase 到完美，不然中間的 conflict 會搞混。
- 可以一次改點東西就 rebase 一次，然後開個臨時的 branch 存檔起來，再繼續 rebase 自己直到滿意為止。



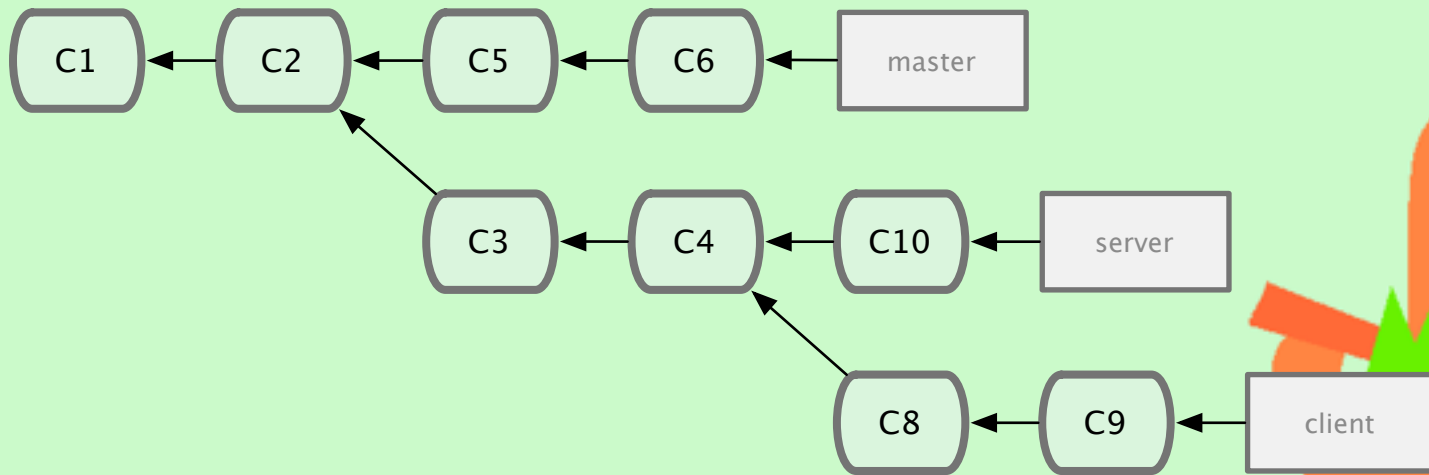
rebase --onto 指定從哪裡開始rebase

- <http://blog.yorkxin.org/2011/07/29/git-rebase>

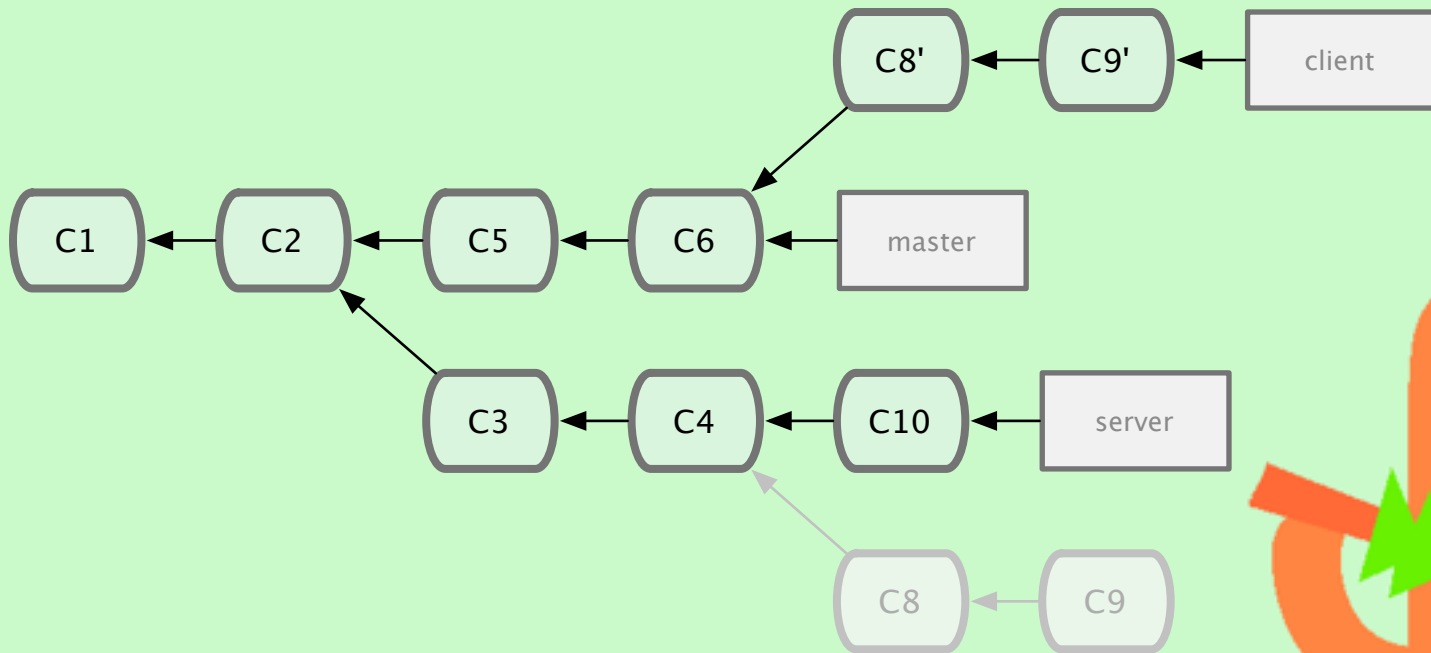


rebase --onto example

想把 client branch 的 C8, C9 搬到 master



(在client branch中) git rebase --onto C6 C4



其實你用 cherry-pick
慢慢搬也可以

5. Git 開發模式及流程

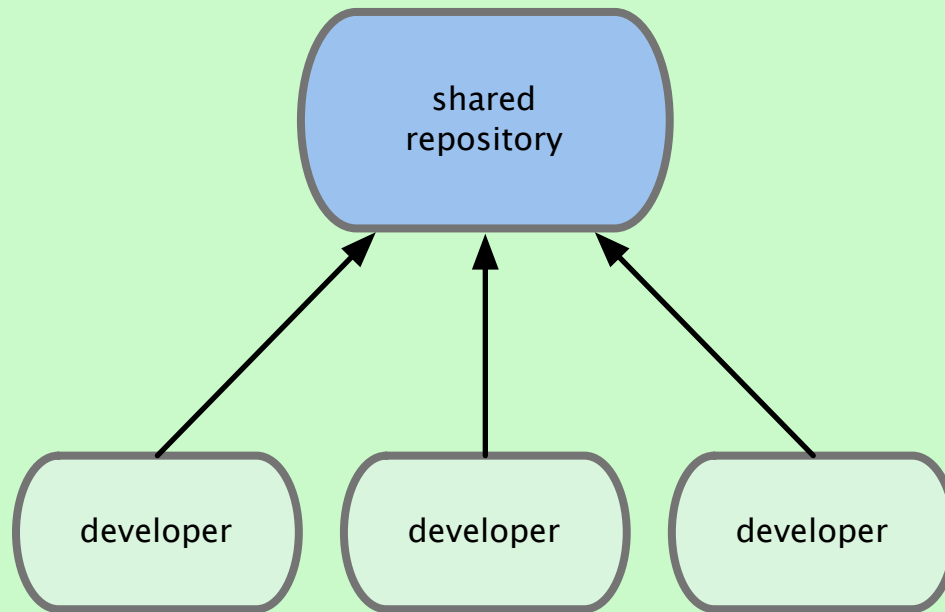


1. 常見的 Remote Repositories 的管理方式



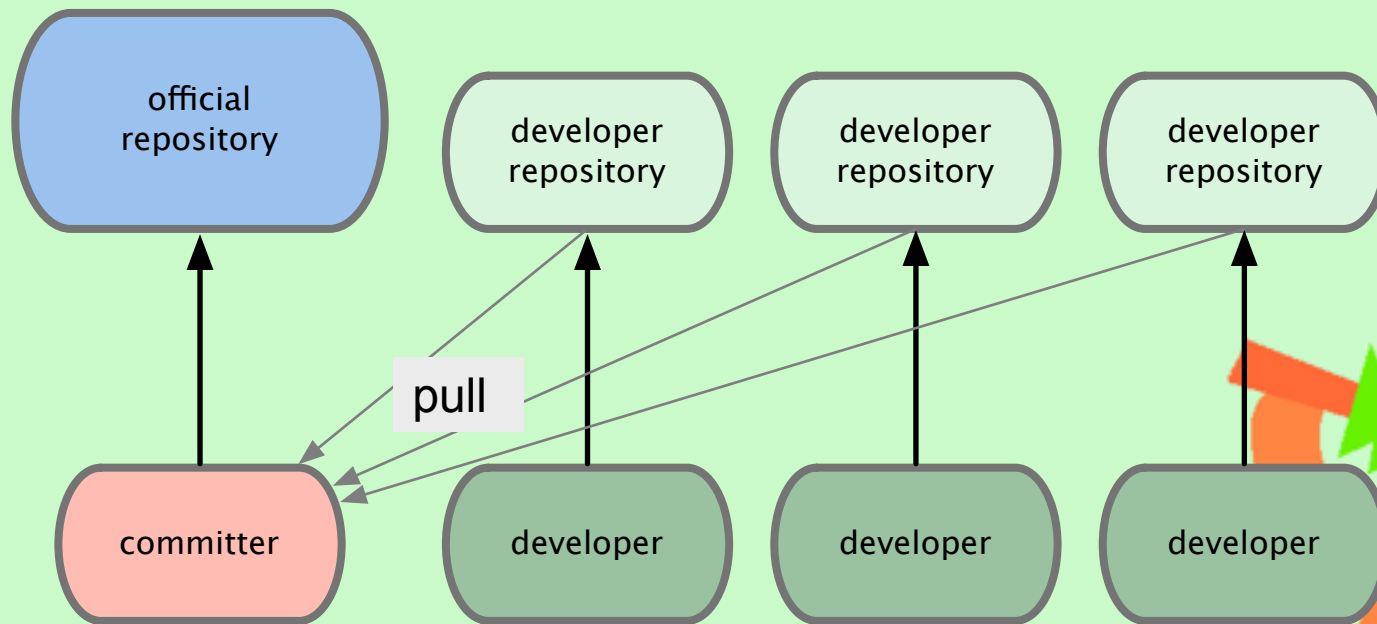
Centralized Workflow

團隊內部私有專案，大家都有權限 Push 到共用的 Repository。
使用 Branches 來管理流程(下述)



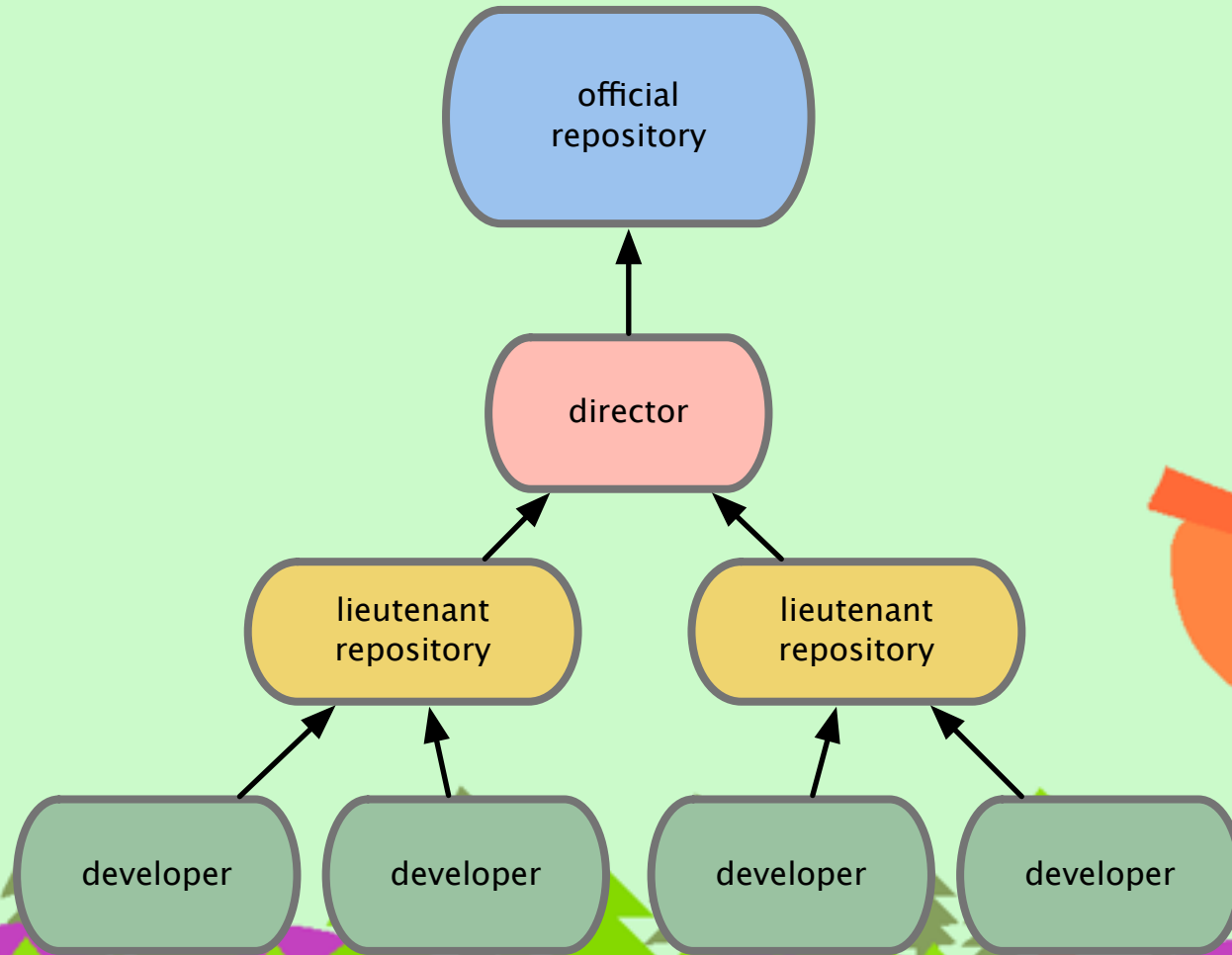
Integration-Manager Workflow

適合一般 Open Source 專案，只有少部分人有權限可以 Push 到 Repository，其他開發者用 request pull 請求合併。
例如 GitHub 提供的 Fork 和 Pull Request 功能



Dictator and Lieutenants Workflow

多層權限控管，適合大型 Open Source 專案，例如 Linux Kernel
又分成 patch 型或 pull request 型



2. 團隊如何管理 Branches 分支?



分支模式

by Continuous Delivery ch.14

- Develop on Mainline
- Branch for Release
- Branch by Feature
- Branch by Team



Develop on Mainline

- 以唯一的 Mainline 作為開發用途
 - 還是可以開以不合併為前提的 branch，例如 release branch 或 spike 實驗
- Continuous Integration 最好做，程式碼總是 integrated 的狀態
- 開發者總是拿到最新的 code
- 避免開發後期 merge hell 和 integration hell 的問題
- 缺點：releasable 程度可能降低，必須在架構設計上有增量式開發的能力和技巧。

在單一Branch 上做增量式開發的技巧?

- Feature Toggle

<http://martinfowler.com/bliki/FeatureToggle.html>

- Branch By Abstraction

<http://continuousdelivery.com/2011/05/make-large-scale-changes-incrementally-with-branch-by-abstraction/>



Branch for Release

- 功能開發完成，準備釋出時建立這個 release branch。而同時還會在主幹繼續開發新功能。
- 在 release branch 上只 commit bugfixes (這個 bugfixes 同時 commit 進主幹)
- 不要在 release branch 上又開 branch，避免階梯狀的 branch 結構，會很難知道哪些 code 在哪些 branch 有。

以 Ruby on Rails 為例

- 例如：Rails 目前的版本是 4.1
- master 是開發版本，開發下一次的主要版本 4.2
- stable branches，用 cherry-pick 做 backporting
 - 2-3-stable 已停止維護
 - 3-2-stable security fixes
 - 4-0-stable bugs fixes
 - 4-1-stable bugs fixes

Branch by Feature

- 根據 Feature 或 User Story 來建立 Branch 開發，直到 Branch 驗收完成才合併回主幹
- 可讓主幹總是 releasable
- 缺點是與主幹的分歧造成合併問題和不利於 CI
 - 主幹如果有更新，Feature Branch 必須每日合併回來
 - Feature Branch 週期越短越好，幾天到不超過一個開發週期 (Iteration)
 - 不要同時開太多 Feature Branch (避免半成品)
 - 需要一個 Tech Lead 來負責主幹的合併

Feature Branch 的問題

- 對 Open Source Project 來說非常有效率
 - Small core team 可以主導要不要接受 patch
 - Release date 不固定，可以慢慢考慮 patch
- 但對大型商業團隊來說，容易變成 Anti-pattern (超長 branch 合併不回來)。團隊需要良好紀律：
 - codebase 需要良好模組化
 - 大家都乖乖定期更新主幹的 code，並且經常 commit 程式到主幹
 - Delivery team 不會因為時程壓力而胡亂 merge

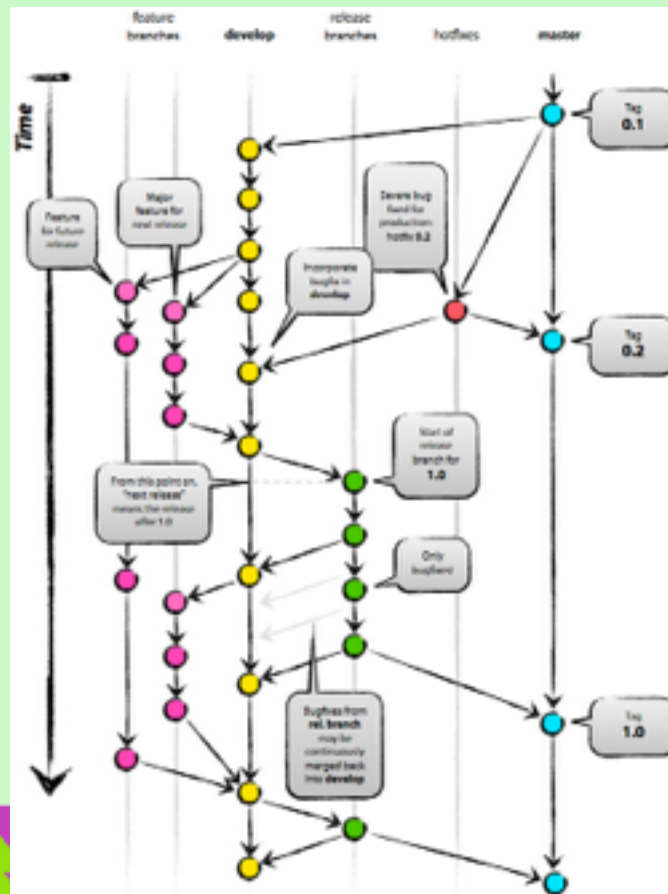
Branch by Team

- 適合多團隊同時開發相對獨立的功能
- 根據 Team 來切分
- 主幹總是 releasable
- 當 Team 的 Branch 穩定時合併回主幹
- 主幹有更新時要馬上要合併回來

Git flow: 一套如何管理 branches 的操作實務

<http://nvie.com/posts/a-successful-git-branching-model/>

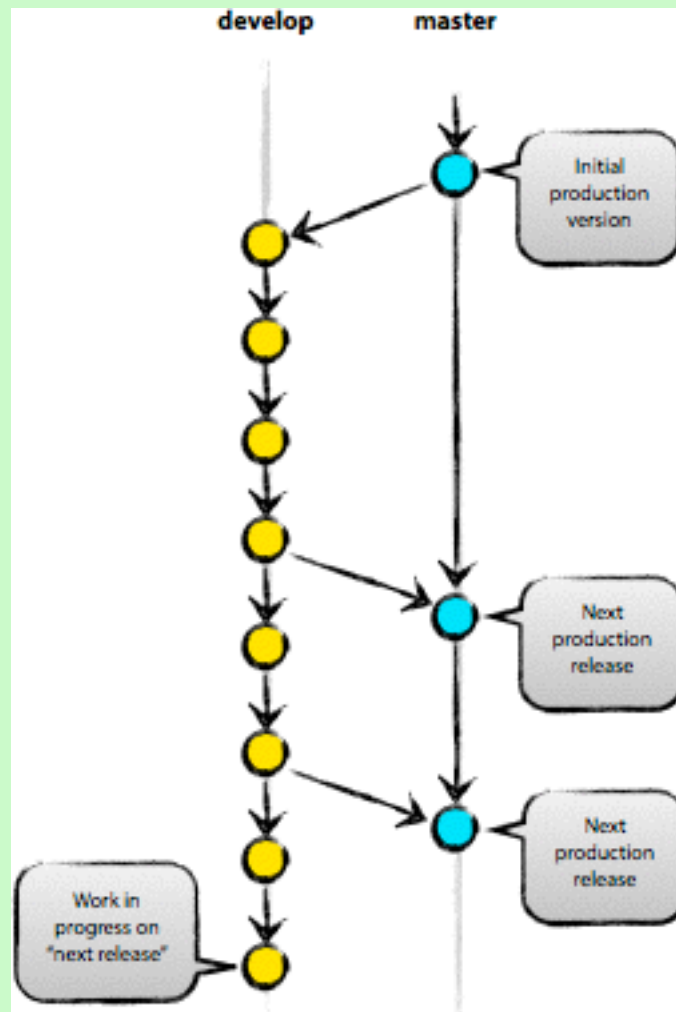
<http://ihower.tw/blog/archives/5140>



兩個主要分支

- master: 永遠處在 production-ready 穩定狀態
- develop: 最新的下次發佈開發狀態

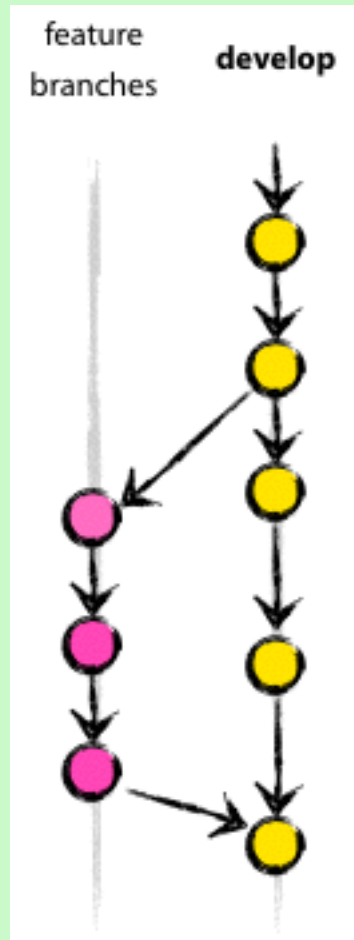


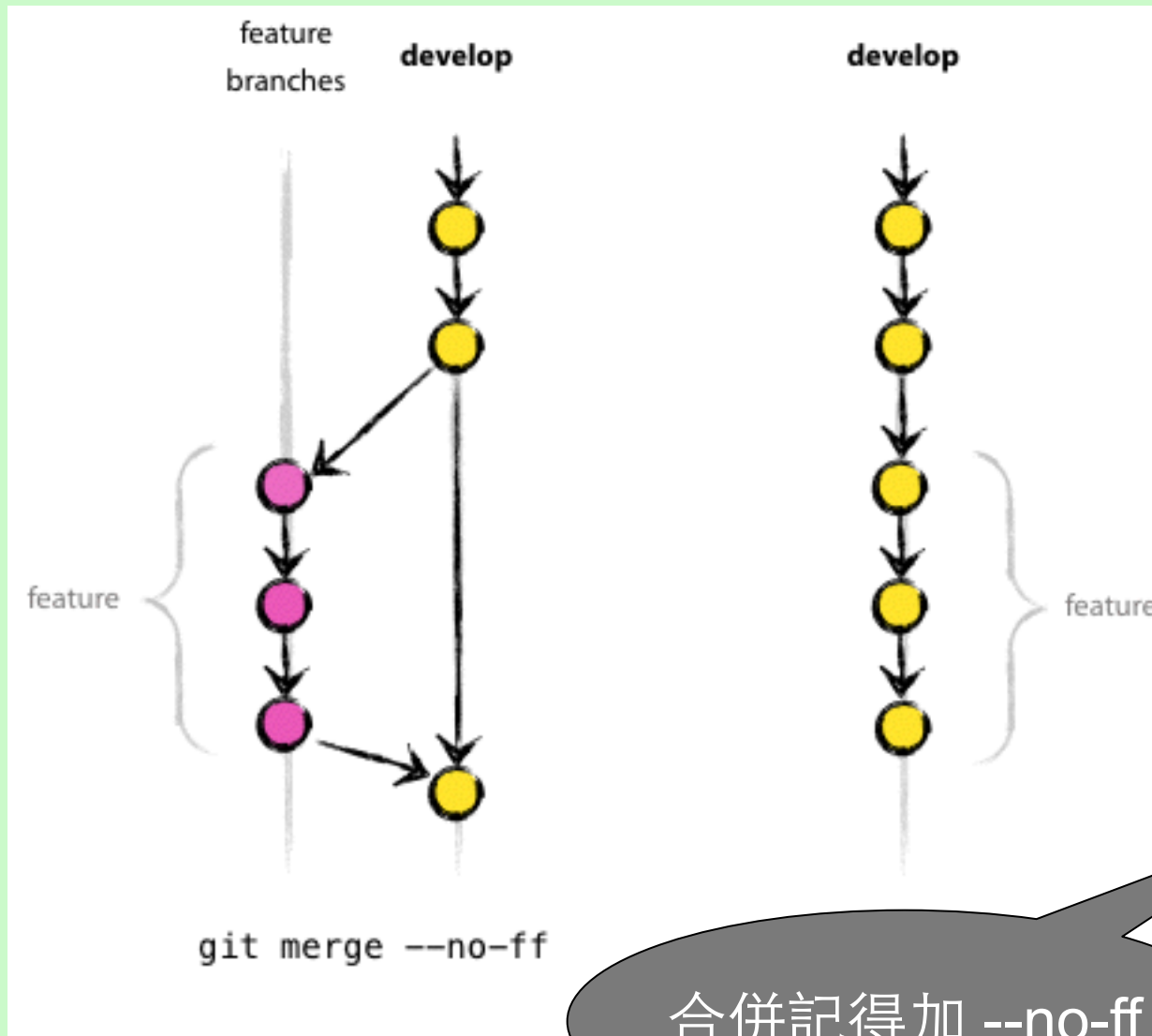


三種支援性分支(I)

- Feature branches
 - 開發新功能或修 bugs
 - 從 develop 分支出來
 - 完成後 merge 回 develop
 - 如果開發時間較長，則需定期同步 develop 主幹的程式(初學可用 merge，建議改用 rebase)，不然最後會合併不回去。

做任何開發幾乎都是先開
branch!!





合併記得加 --no-ff

三種支援性分支(2)

- Release branches
 - 準備要 release 的版本，只修 bugs
 - 從 develop 分支出來
 - 完成後 merge 回 master 和 develop

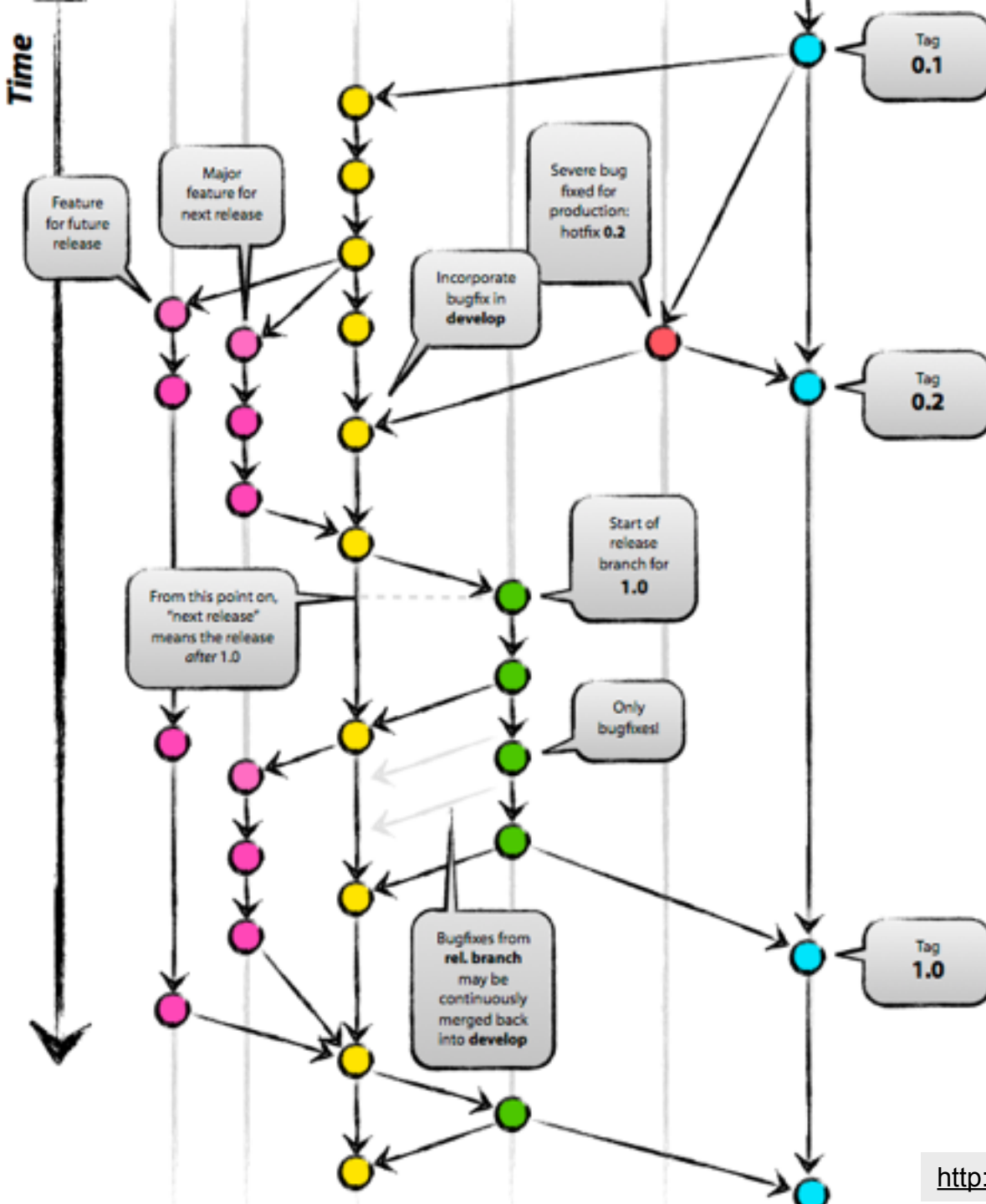


三種支援性分支(3)

- Hotfix branches
 - 等不及 release 版本就必須馬上修 master 趕著上線
 - 會從 master 分支出來
 - 完成後 merge 回 master 和 develop







粉紅色: Feature branches
黃色: develop
綠色: release branches
紅色: hotfix branches
藍色: master



git-flow 工具

- `git flow init`
- `git flow feature finish feature_name`
- `git flow feature publish feature_name`
- `git flow feature track feature_name`
- `git flow feature finish feature_name`



Github flow

<http://scottchacon.com/2011/08/31/github-flow.html>

- master 是 stable/production 可佈署的版本
- 任何開發從 master branch 分支出 feature/* branch
- 送 pull request 進行 code review 和測試
- 合併回 master 代表可以釋出



Branch-Owner Policy

- 規定只有 project leaders 可以 commit/merge 進 develop branch 。
- 規定只有 release team 可以管理 master branch 。
- 其他開發者都開 topic branches，完成後發 pull request，大家做 code review 後，沒問題才 merge 進 develop 。
- GitHub 的 pull request 功能推薦

Body attributes and some new Tag Helpers

https://github.com/rails/rails/pull/30

github SOCIAL CODING

ihower 10 | Dashboard | Inbox 0 | Account Settings | Log Out

Explore GitHub | Gist | Blog | Help | Search...

rails / rails

Unwatch | Fork | 7,740 | 1,322

Source | Commits | Network | Pull Requests (18) | Graphs | Branch: master

Open chriseppstein wants someone to merge 3 commits into rails:master from chriseppstein:body_attributes #30

Discussion | Commits <> 3 | Diff >> 6

chriseppstein opened this pull request September 19, 2010

Body attributes and some new Tag Helpers

Having a consistent convention for body classes makes styling easier :) This approach also let's the body live in the layout but still be modified by the template if necessary.

Open

+ 239 additions
- 2 deletions

All Pull Requests

chriseppstein and jeremy are participating in this pull request.

+ chriseppstein added some commits

- 739108c New helper methods for working with tag attributes
- d299f22 Make it easier to manage the body tag in the layout
- d337d1b Convert the default scaffold to use the body_tag helper

jeremy commented January 10, 2011

I like this idea (and I use a similar low-tech helper) but I'm concerned it introduces a broad API for such a simple task. And the API is all at the view level despite being request-wide state, so you can't add attributes in your controller.

chriseppstein commented January 10, 2011

Code Review 討論

Source

Commits

Network

Pull Requests (22)

Graphs

Branch: master

Open

rolftimmermans wants someone to merge 5 commits into rails:master from rolftimmermans:desc_tracker

#225

Discussion @

Commits <> 5

Diff >> 7

Showing 7 changed files with 64 additions and 34 deletions.

•	activesupport/lib/active_support/descendants_tracker.rb	16	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
•	activesupport/test/core_ext/duration_test.rb	1	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
•	activesupport/test/core_ext/string_ext_test.rb	1	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
→	activesupport/test/{descendants_tracker_test.rb + descendants_tracker_test_cases.rb}	36	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
+	activesupport/test/descendants_tracker_with_autoloading_test.rb	35	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
+	activesupport/test/descendants_tracker_without_autoloading_test.rb	8	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
•	activesupport/test/multibyte_chars_test.rb	1	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■

activesupport/lib/active_support/descendants_tracker.rb

```
... @@ -1,5 +1,3 @@
1 -require 'active_support/dependencies'
2 -
3 1 module ActiveSupport
4 2   # This module provides an internal implementation to track
5 3   # which is faster than iterating through ObjectSpace.
... @@ -18,12 +16,16 @@ module ActiveSupport
18 16   end
19 17
20 18   def self.clear
21 -   @@direct_descendants.each do |klass, descendants|
22 -     if ActiveSupport::Dependencies.autoloaded?(klass)
23 -       @@direct_descendants.delete(klass)
24 -     else
25 -       descendants.reject! { |v| ActiveSupport::Dependencies.autoloaded?(v) }
19 +   if defined? ActiveSupport::Dependencies
20 +     @@direct_descendants.each do |klass, descendants|
21 +       if ActiveSupport::Dependencies.autoloaded?(klass)
22 +         @@direct_descendants.delete(klass)
23 +       else
24 +         descendants.reject! { |v| ActiveSupport::Dependencies.autoloaded?(v) }
25 +       end
26 26   end
27 +   else
28 +     @@direct_descendants.clear
27 29   end
28 30   end
```

對整個 Branch 做 Diff

開發與佈署流程

<http://ihower.tw/blog/archives/7798>

- 分支流程不只與開發相關，也與測試和軟體部署(釋出)流程相關
- 什麼時候，哪個分支跑 CI 自動測試?
- 什麼時候，哪個分支佈署到 Staging Server 進行人工測試?

6. 更多 Git Tips



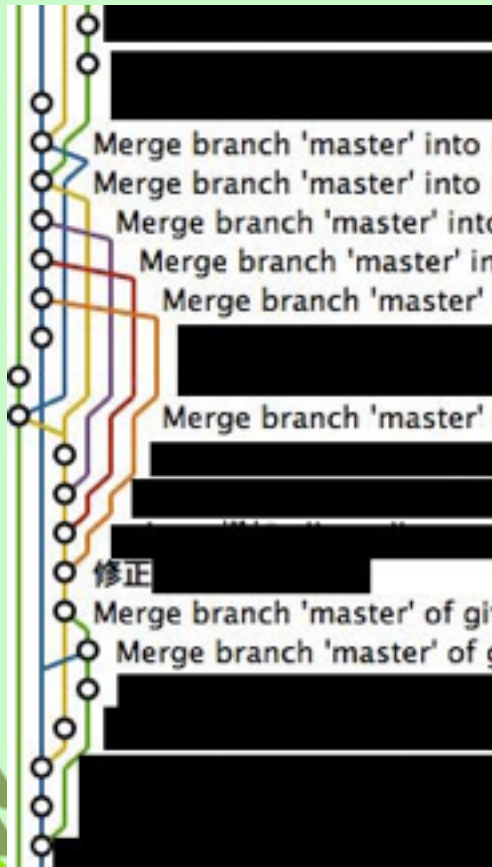
其他更多指令

- git archive
- git bisect
- git blame
- git grep
- git show
- git gc
- git format-patch, send-email, am



使用 git pull --rebase 可以避免 無謂的 merge 節點，讓線圖變乾淨

<http://ihower.tw/blog/archives/3843>



特別是很多人在同一
個 branch 同時開發，例如
主幹 master，容易發生這
種情況！

git pull 何時用 merge? 何時用 rebase?

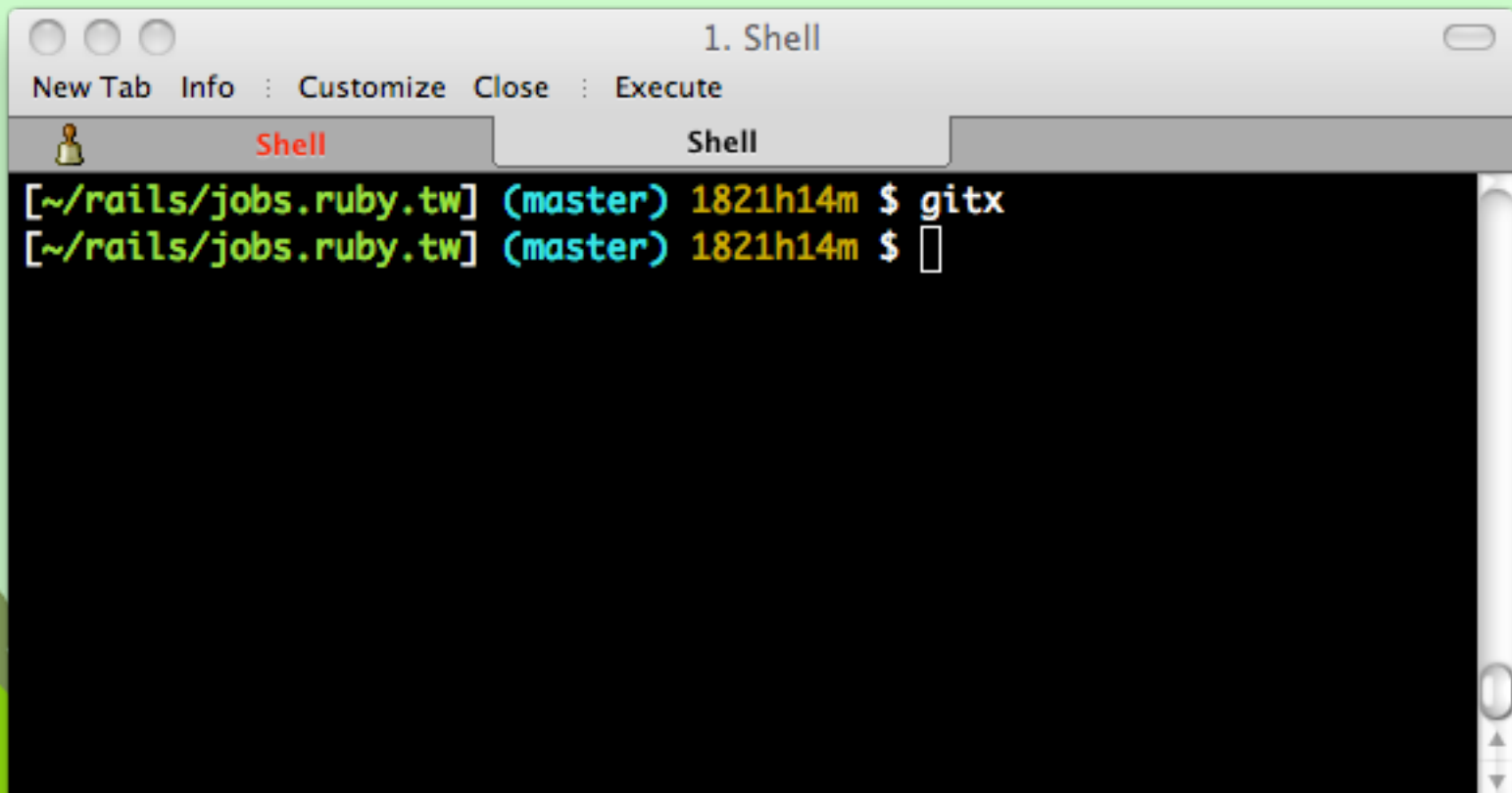
- 修改比較多，預期會有 conflict，用 merge
 - 是不是一開始就應該開 feature branch 來做呢?
- 修改範圍較小，不太預期有 conflict，建議可以用 --rebase

但是如果採用 git flow 利用 branches，就可以大幅降低無謂 merge 情形!

Bash Prompt

<http://ihower.tw/blog/archives/5436/>

- 顯示目前在哪一個 branch
- 顯示距離上次 commit 有多久了



The screenshot shows a terminal window titled "1. Shell" with a menu bar containing "New Tab", "Info", "Customize", "Close", and "Execute". The terminal has two tabs, both labeled "Shell". The prompt is "[~/rails/jobs.ruby.tw] (master) 1821h14m \$". The user has entered the command "gitx", and the output is "[~/rails/jobs.ruby.tw] (master) 1821h14m \$".

```
[~/rails/jobs.ruby.tw] (master) 1821h14m $ gitx
[~/rails/jobs.ruby.tw] (master) 1821h14m $
```

bash-completion

- <http://bash-completion.alioth.debian.org/>
- Install via Homebrew (Mac only)
 - brew install bash-completion
 - cp /usr/local/etc/bash_completion.d/git-completion.bash ~/.git-bash-completion.sh
 - 編輯 ~/.bash_profile 加入
[-f ~/.git-bash-completion.sh] && . ~/.git-bash-completion.sh



Useful alias

編輯 ~/.gitconfig

[alias]

```
co = checkout
ci = commit
st = status
br = branch -v
rt = reset --hard
unstage = reset HEAD
uncommit = reset --soft HEAD^
l = log --pretty=oneline --abbrev-commit --graph --
decorate
amend = commit --amend
who = shortlog -n -s --no-merges
g = grep -n --color -E
cp = cherry-pick -x
nb = checkout -b
```


Useful alias

編輯 ~/.gitconfig

[alias]

```
# 'git add -u' 只包含刪除的檔案，不包含新增檔案  
# 'git add .' 包含修改和新增的檔案，但是不包含刪除  
# 'git addall' 通通加到 staging
```

```
addall = !sh -c 'git add . && git add -u'
```

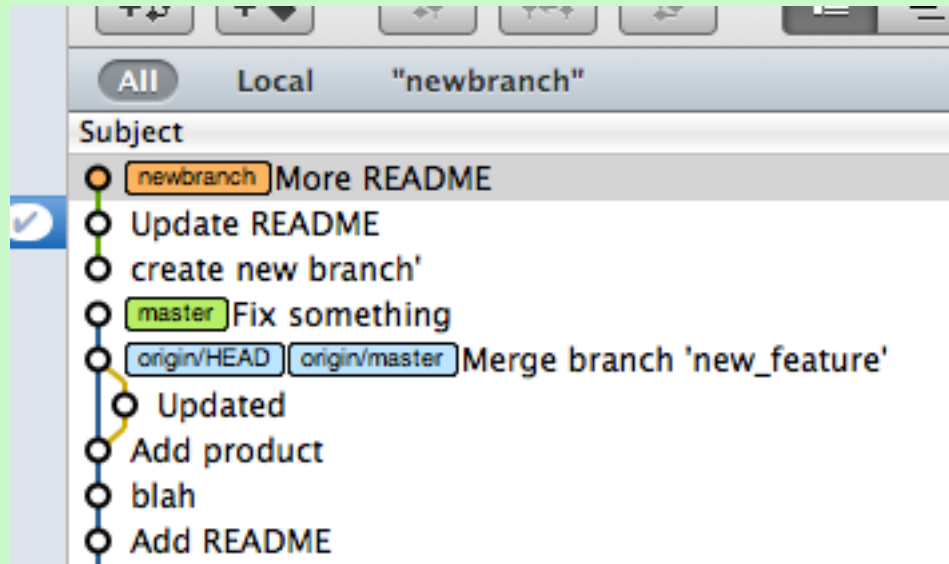


建立沒有 parent 的獨立 branch

- 建立毫不相關的分支，用來保存其他資訊，就像目錄一樣。不需要被 merge 回主幹。
- 用途範例：Github page
- `git checkout --orphan new_branch`
 - `git rm -rf .` # 砍掉重練
 - 加新檔案
 - `git add .`
 - `git commit -m "new branch"`



Orphan branch example



弄丟 commit 和 branch 怎麼辦?

<http://www.programblings.com/2008/06/07/the-illustrated-guide-to-recovering-lost-commits-with-git/>

- 例如搞砸了 `git reset`, `rebase` 或誤刪 branch
- 在 `git gc` 前都還有機會救回來 (預設會保留孤兒 commits 90天)
- `git reflog` 或 `git fsck --lost-found`
- `git cherry-pick <SHA1>`
- 或 `git merge <SHA1>`



git-filter-branch 過濾

- 只取特定目錄

<http://gugod.org/2012/07/split-a-git-repository/>

- 刪除特定敏感檔案

<https://help.github.com/articles/remove-sensitive-data>



Git Submodule 或 Subtree

- Git 不像 SVN 可以只取出其中的子目錄
- 如何將第三方程式碼加進來? 像 SVN Externals?
- Submodule
 - <http://josephjiang.com/entry.php?id=342>
 - <http://josephjiang.com/entry.php?id=357>
- Subtree
 - <http://git-scm.com/book/ch6-7.html>
- 我沒用到
 - 因為 Ruby/Rails Develops 用不太到 XD
 - Ruby 用 Bundler 工具來管理 dependencies，可以確保大家都跑相同的套件版本，又可以解決套件相依性問題



git svn

- git 可以吃 SVN repository
- git svn clone <http://url/svn/trunk/> project_name
- (normal git operations)
- git svn dcommit



自行架設 Git Server

- git --bare via SSH
- Gitis
- Gitolite
- Gitlab 基於 Gitolite 的 Web 介面
 - <http://gitlabhq.com/>



7. Github 簡介





= **github**
SOCIAL CODE HOSTING

誰在用 GitHub?

- twitter
- facebook
- rackspace
- digg
- Yahoo!
- shopify
- EMI
- six apart
- jQuery
- YUI 3
- mootools
- Ruby on Rails
- node.js
- symfony
- mongodb
- Erlang



rails / rails

Unwatch

Fork

7,862

1,352

Source

Commits

Network

Pull Requests (22)

Graphs

Switch Branches (12)

Switch Tags (83)

Branch List

Ruby on Rails — Read more

<http://rubyonrails.org>

Downloads

HTTP Git Read-Only

<https://github.com/rails/rails.git>

This URL has Read-Only access

define_attr_method correctly defines methods with invalid identifiers



spastorino (author)
about 16 hours ago

commit c834a751d2acbd55b580
tree 423e181f132e2977ac12
parent fda45f4fc493f5596375

rails /

name	age	message	history
actionmailer/	March 10, 2011	Fix typo [spastorino]	
actionpack/	about 22 hours ago	fixes an issue with number_to_human when conver... [joshk]	
activemodel/	about 16 hours ago	define_attr_method correctly defines methods wi... [spastorino]	
activerecord/	2 days ago	Merge branch 'master' of git://github.com/lifo/... [fxn]	
activeresource/	March 05, 2011	Active Resource typos. [rtlechow]	
activesupport/	4 days ago	Revert "It should be possible to use ActiveSupp... [josevalim]	
bin/	June 19, 2010	add missing shebang to rails bin. LH [#4885 sta... [smtlaissezfaire]	

Ruby is the #2 most popular language on GitHub

Explore Repositories **Languages** Timeline Search Tips

Ruby Recently Created Recently Updated

Most Watched Today

-  wbailey / **kata**
-  injekt / **slop**
-  apneadiving / **Google-Maps-for-Rails**
-  codegram / **resort**
-  shuber / **attr_encrypted**



Most Watched This Week

-  injekt / **slop**
-  mxcl / **homebrew**
-  rails / **rails**
-  NoamB / **sorcery**
-  diaspora / **diaspora**

Most Watched This Month

-  postrank-labs / **goliath**
-  mxcl / **homebrew**
-  rails / **rails**
-  meskyanichi / **backup**
-  diaspora / **diaspora**






Most Watched Overall

-  rails / **rails**
-  mxcl / **homebrew**
-  diaspora / **diaspora**
-  joshuaclayton / **blueprint-css**
-  plataformatec / **devise**






Most Forked Today

-  mxcl / **homebrew**
-  thoughtbot / **capybara-webkit**
-  oprcode / **cookbooks**
-  edavis10 / **redmine**
-  drbrain / **meme**






Most Forked This Week

-  mxcl / **homebrew**
-  rails / **rails**
-  diaspora / **diaspora**
-  chicagoruby / **booksiebot7000**
-  drbrain / **meme**

Most Forked This Month

-  mxcl / **homebrew**
-  rails / **rails**
-  diaspora / **diaspora**
-  mojombo / **jekyll**
-  plataformatec / **devise**

Most Forked Overall

-  mxcl / **homebrew**
-  rails / **rails**
-  diaspora / **diaspora**
-  thoughtbot / **paperclip**
-  Shopify / **active_merchant**

All Languages

- ActionScript
- Ada
- Arc
- ASP
- Assembly
- Boo
- C
- C#
- C++
- Clojure
- CoffeeScript
- ColdFusion
- Common Lisp
- D
- Delphi
- Duby
- Eiffel
- Emacs Lisp
- Erlang
- F#
- Factor
- FORTRAN
- Go
- Groovy
- Haskell
- HaXe
- Io
- Java
- JavaScript
- Java

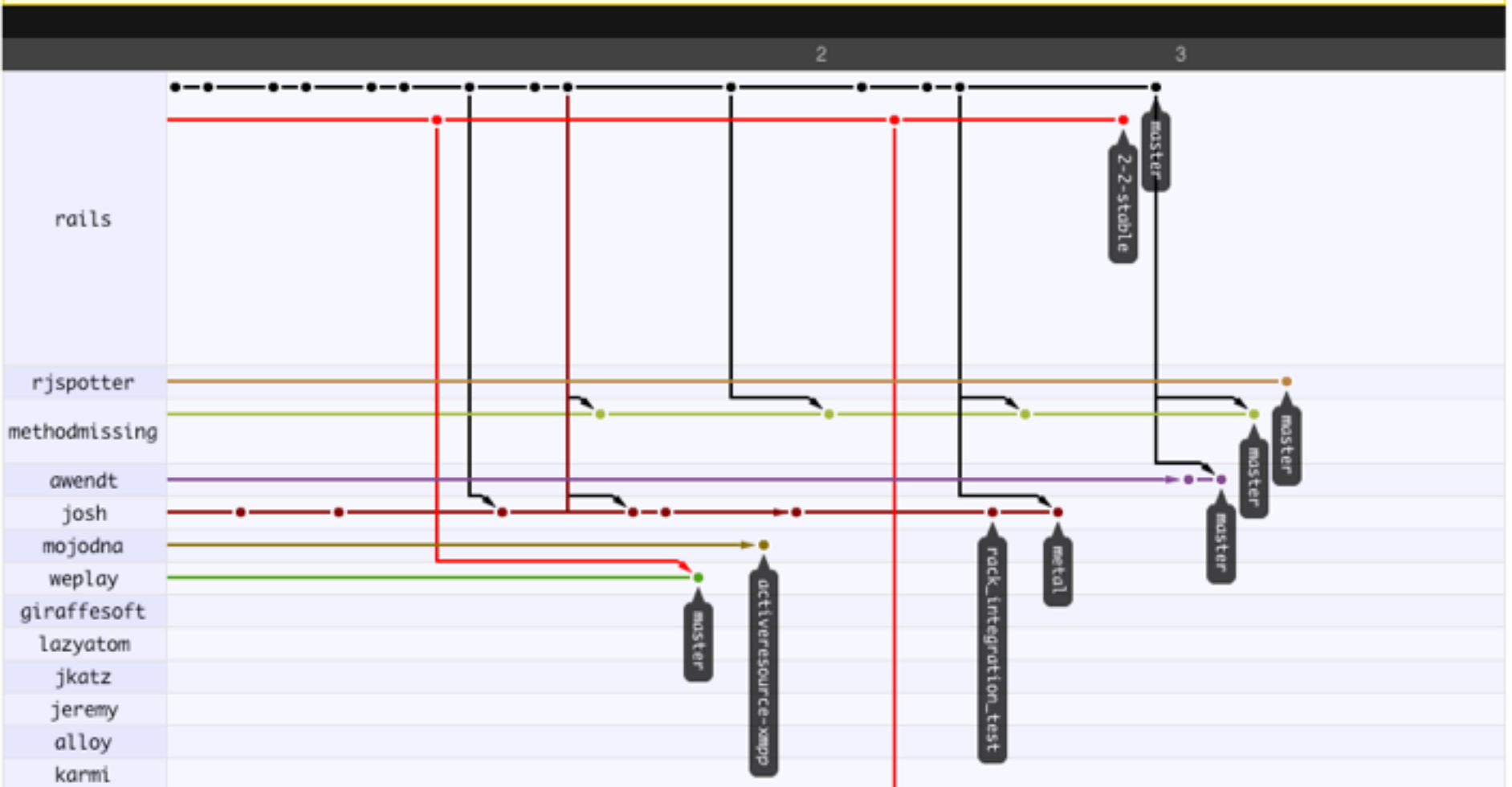
Fork network

The rails network graph

All branches in the network using **rails/rails** as the reference point. [Read our blog post about how it works.](#)

[Show Help](#)

This graph is out of date. We are showing you a cached version while we bring it up to date.



KGT III L

如何送 Patch 給開源專案?

- Fork 成我的遠端 Repository
- Clone
- Fix bugs 或 Enhance features
- Push
- 送 Pull request 給原作者，請他合併



Fork & Pull Requests

有 75% 的被 fork 開源專案，有 outside merge 回來的記錄!!

被 watch 超過一人以上的開源專案，有 25% 有 outside contributions 紀錄!!

Pull Requests - gugod/App-perlbrew - GitHub

https://github.com/gugod/App-perlbrew/pulls

gugod / App-perlbrew

Source Commits Network Pull Requests (1) Issues (15) Wiki (2) Graphs Branch: master

All Requests 1

Yours

Find a user...

rafi 1

trcj 0

tokuhim 0

melo 0

dagolden 0

book 0

avar 0

francuncy 0

abh 0

doy 0

gix 0

poletix 0

yibe 0

dalinaum 0

1 open request

Read Open Closed Submitted Updated Popularity

Use Devel::PatchPerl to make old perls build

Old perl releases have the tendency to not build anymore on modern platforms and require some pat...

rafi submitted to gugod/App-perlbrew 2 days ago Updated about 3 hours ago 2 comments

documents -j and enables parallel testing

parallel testing needs TEST_JOBS=N for "make test_harness"

dagolden submitted to gugod/App-perlbrew March 12, 2011 Updated March 13, 2011 1 comment

Fix switch and use

With version 0.16 of 'perlbrew', the 'switch' command doesn't work as specified. The bashrc code ...

melo submitted to gugod/App-perlbrew March 08, 2011 Updated March 10, 2011 9 comments

trivial spelling fix

No description available

abh submitted to gugod/App-perlbrew March 06, 2011 Updated March 06, 2011 1 comment

Topic/exec brew only

Hi I've added an option to exec: --brew-only This way, I can install a module using 'perlb...

francuncy submitted to gugod/App-perlbrew February 27, 2011 Updated March 01, 2011 3 comments

Use gtar on Solaris



ihower 53

[Dashboard](#)[Inbox 0](#)[Account Settings](#)[Log Out](#)[Explore GitHub](#)[Gist](#)[Blog](#)[Help](#)**wycats** (Yehuda Katz)[Message](#)[Unfollow](#)

Name Yehuda Katz
Email wycats@gmail.com
Website/Blog <http://www.yehudakatz.com>
Company Engine Yard
Location San Francisco
Member Since Jan 11, 2008

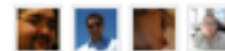
91

public repos

1,597

followers

Following 4 coders and watching 199 repositories



Public Repositories (91)

 net-http

Ruby 2 1

Last updated 2 days ago

52 week participation

jquery-governance

Ruby 18 11

Last updated 2 days ago

52 week participation

handlebars-site

Ruby 3 1

Last updated March 07, 2011

52 week participation

Public Activity

wycats pushed to master at [sproutcore/abbot](#) about 13 hours ago
[91bce4a](#) Update for 1.5.0.pre.5 **wycats pushed to master at [sproutcore/sproutcore](#) about 13 hours ago**
[8d23ff1](#) Update changelog for SproutCore 1.5.0.pre.5 **wycats pushed to master at [sproutcore/Todos-Example](#) about 14 hours ago**
[62c4b57](#) Use itemClassBinding for todos list. **wycats pushed to master at [sproutcore/abbot](#) about 14 hours ago**
[790fcd0](#) Remove cloneable requirement, since we deleted this file.
[af6a1c3](#) Fix whitespace in Chance so we don't emit warnings.
[4007475](#) Remove trailing comma from unit test Buildfile.
[4 more commits »](#)**Follow!**



ihower 53

Dashboard

Inbox 0

Account Settings

Log Out

Explore GitHub

Gist

Blog

Help

Search...

carlhuda / bundler

Watch

Fork

1,064

196

Source

Commits

Network

Pull Requests (19)

Issues (147)

Wiki (14)

Graphs

Branch: 1-0-stable

Home

Pages

Wiki History

Git Access

New Page

Edit Page

Page History

Gem.refresh to reload the gemfile

From [issue 739](#):

Hi guys,

For working unicorn you introduced Gem.refresh (back in f0a65fea). And it looks like this:

```
gem_class.send(:define_method, :refresh) { }
```

The problem is it's in the unicorn for a reason. Unicorn offers zero-downtime restarts. When you send it USR2 it forks and reloads rails/rack and then starts responding to users requests.

When Gemfile is changed it doesn't pick up changes. For example, I have working unicorn instance, then I add a gem, I deploy, send USR2, and unicorn couldn't restart. Because it loaded old Gemfile and have no idea about new gem. Practically it dumps this backtrace (<https://gist.github.com/5c27d25d134e1cfa0eae>) to the unicorn.stderr.log and continues to run old instance with the new current directory.

I propose Gem.refresh should reload Gemfile. What do you guys think?

elucid says:

I don't think this is actually a problem with Bundler. I've put up a simple repo at <https://github.com/elucid/unicorn-reload> to demonstrate how USR2 reloads with Unicorn work. If the old Gemfile is being reloaded after an upgrade, I think this probably has to do with the way that Unicorn was started in the first place. In particular, new Unicorn master processes will be started in the exact same directory that the original was. If you update your app code in place and nothing else changes, that should be fine. However if each deploy has its own directory you could be pointing at an old one. You can manually set this to something stable (e.g. a symlink that gets updated on each release) by calling `working_directory somedir` in your Unicorn config file.

Wiki

Source

Commits

Network

Pull Requests (22)

Graphs

Tree: c834a75

Switch Branches (12) +

Switch Tags (83) +

Comments

Contributors

Ruby on Rails

<http://rubyonrails.org>

Downloads

HTTP

Git Read-Only

<https://github.com/rails/rails.git>

This URL has Read-Only access

define_attr_method correctly defines methods with invalid identifiers



spastorino (author)

about 16 hours ago

commit c834a751d2acbd55b580

tree 423e181f132e2977ac12

parent fda45f4fc493f5596375

Showing 2 changed files with 6 additions and 5 deletions.

activemodel/lib/active_model/attribute_methods.rb 5

activemodel/test/cases/attribute_methods_test.rb 6

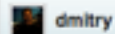
activemodel/lib/active_model/attribute_methods.rb

show inline notes

View file @ c834a75

```
... @@ -108,9 +108,8 @@ module ActiveModel
108 108     else
109 109     # use eval instead of a block to work around a memory leak in dev
```

2



dmitry

about 3 hours ago

Forgot to cleanup comment?



spastorino

repo collaborator

dmitry, hey thanks for pointing me to this. I saw this yeah I have to try a few more things and do another com

Add a line note

```
110 110     # mode in fogi
111 111     sing.class_eval <<-eorb, __FILE__, __LINE__ + 1
112 112     def #{name}; #{value.nil? ? 'nil' : value.to_s.inspect}; end
113 113     eorb
111 111     value = value.nil? ? 'nil' : value.to_s
112 112     sing.send(:define_method, name) { value }
114 113     end
115 114     end
116 115
```

Code Review

Diff

Github pages

<http://pages.github.com/>

- 叫做 `your.github.com` 的 repository，Github 會自動建立 `http://your.github.com` 的靜態網頁。
- 擁有 `gh-pages` 的分支的 `your_project`，Github 會自動建立 `http://your.github.com/your_project` 靜態網頁。

在 Git，你可以建立毫不相關的分支，用來保存其他資訊，就像目錄一樣。
(開 Branch 的用途不一定就是要 merge)

GitHub is your code portfolio



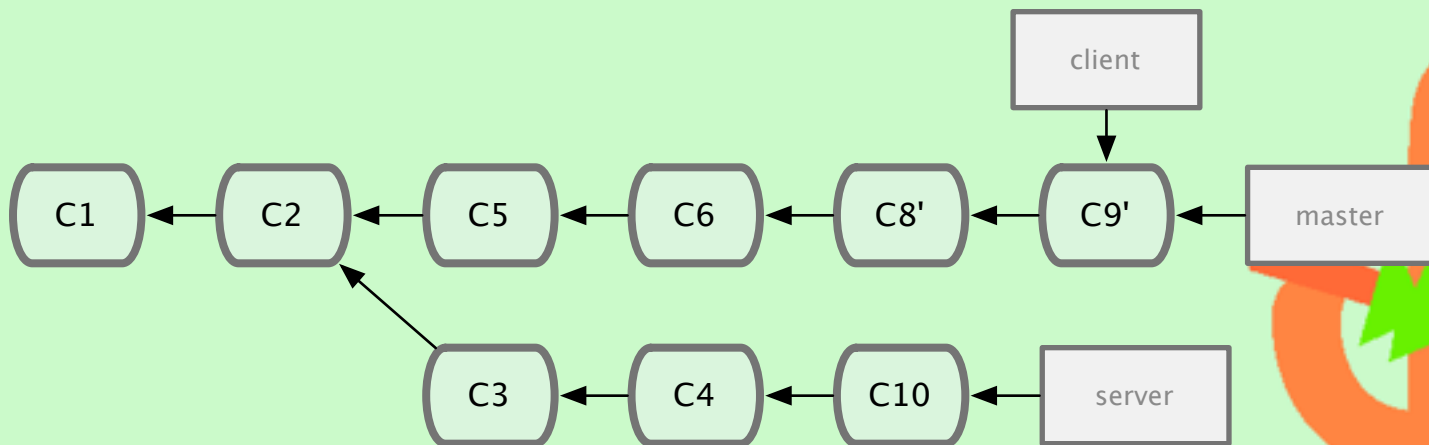
Thank you.

請 google “ihower” 就可以找到我及這份投影片。



特別感謝

- Scott Chacon 公開的 OmniGraffle diagram 圖檔
<https://github.com/schacon/git-presentations>



參考書籍和影片

- <http://pragprog.com/screencasts/v-jwsceasy/source-control-made-easy>
- <https://peepcode.com/products/git>
- <https://peepcode.com/products/advanced-git>
- Git Internals, Peepcode
- Pragmatic Version Control Using Git, Pragmatic
- Pragmatic Guide to Git, Pragmatic
- Continuous Delivery Ch.14
- Version Control with Git, O'Reilly

參考資料

- <http://ihower.tw/blog/category/git>
- <http://nfarina.com/post/9868516270/git-is-simpler>
- <http://tomayko.com/writings/the-thing-about-git>
- <http://think-like-a-git.net>
- <http://yanpritzker.com/git-book/>
- <http://progit.org/book/>
- <http://git-scm.com>
- <http://help.github.com/>
- <http://gitimmersion.com>
- <http://www.gitready.com/>
- <http://gitref.org>
- <http://try.github.com>

