

人生不能重來  
但 Git 可以

高見龍

投影片連結 <https://goo.gl/ZHavQ1>

# 高見龍 @eddiekao

a.k.a Eddie

- Ruby/Rails/iOS app 開發者、講師
- Ruby 技術推廣、教育、諮詢
- 台灣、日本 Ruby 技術研討會講者
- 目前於五倍紅寶石擔任紅寶石鑑定商職務
- 部落格：<http://kaochenlong.com>



# 高見龍 @eddiekao

a.k.a Eddie

- 日本 Ruby 認證開發者 (2012/1).

- Adobe 原廠認證 Flash 開發者 (2006/7).

- Linux LPI 國際認證 (2005/3).

- 五倍紅寶石共同創辦人

- 台灣 WebConf、PHPConf 發起人

- Rails Girls Taipei 發起人

- PTT Flash 版版主



5+/-



五倍紅寶石

j.tw



五倍紅寶石台北出礦坑

5xRuby Inc.

開始之前...

你可能每天會建立很多的檔案...

編輯、存檔、編輯、存檔.. × N

# 問題

你怎麼做檔案備份？

最簡便也最無敵的  
複製、貼上

Ctrl-C + Ctrl-V

名稱

- ▶  ProjectA
- ▶  ProjectA-20160822
- ▶  ProjectA-20160823-1
- ▶  ProjectA-20160823-2
- ▶  ProjectA-bak
- ▶  ProjectA-forEddie

怕本機硬碟壞掉？

備份到網路芳鄰就好啦

版本控制

Version Control

# 問題

什麼版本？要控制什麼？

# 備份

(就像玩遊戲的時候會儲存遊戲進度一樣)

# 歷史紀錄及證據

(出事的時候你會知道是從什麼時候開始就有問題，  
以及知道該找誰來罵)

Concurrent Versions System (CVS)  
since 1990

Subversion (SVN)

問題是...

如果沒有網路，或是 SVN 的伺服器故障的時候便無法使用

Git

什麼是 Git?

Git 也是一種版本控制系統

設計者 : Linus Torvalds

for managing Linux kernel source code in 2005

PB] : How did you approach it? Did you stay up all weekend to write it or was it just during regular hours?

Torvalds: ... It took about a day to get to be "self-hosting" so that I could start committing things into git using git itself ...

分散式

遠端、本地操作皆可  
(大部份的操作都是本地端的)

# 問題

git 我知道，現在工程師們  
都說在那個 Github 什麼的...

Git != GitHub

優點

開源、免費

速度快、檔案小

同時支援本地及遠端操作

容易與其它人共同協作

但是..

Git 是一種易學難精的工具

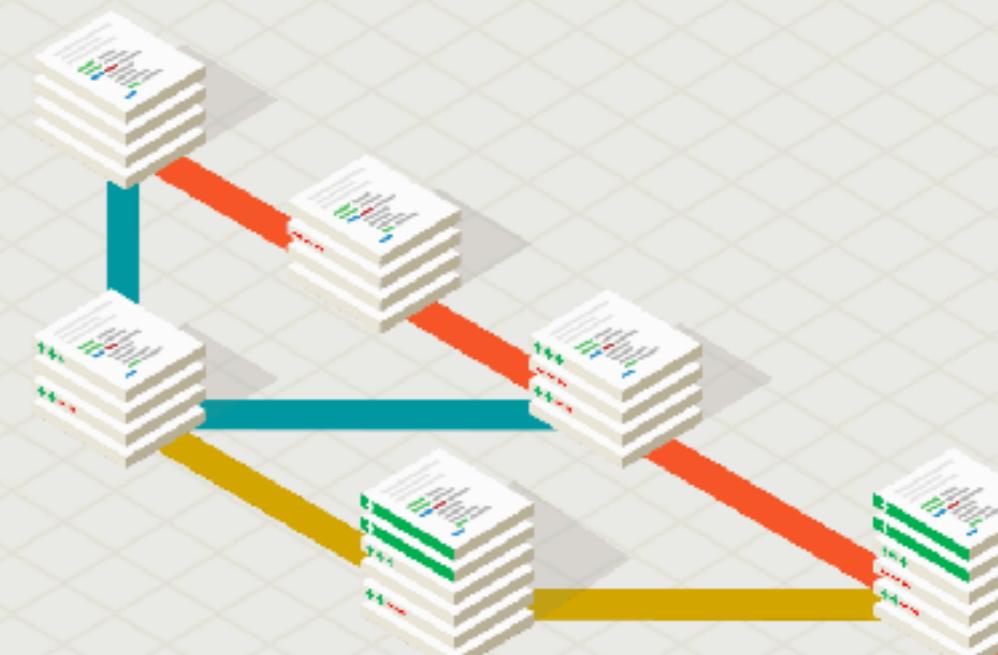
80 / 20 法則

20% 的 git 指令就足以應付平日 80% 的工作

安裝 Git

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.



Learn Git in your browser for free with **Try Git**.

## About

The advantages of Git compared to other source control systems.



## Downloads

GUI clients and binary releases for all major platforms.



## Documentation

Command reference pages, Pro Git book content, videos and other material.



## Community

Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.12.2**

Release Notes (2017-03-24)

Downloads for Mac



<https://git-scm.com/>

Mac:

```
$ brew install git
```

Ubuntu or some linux OS:

```
$ sudo apt-get install git-core
```

or

```
$ sudo apt-get install git
```

The screenshot shows the Codeanywhere web-based code editor. On the left is a file tree for a project named 'CA 6Editor1'. The 'index.html' tab is active, displaying HTML code with meta tags for description and keywords. The 'bundle.js' tab is also visible, showing a large block of JavaScript code. The interface includes a top navigation bar with File, Edit, Find, Goto, View, Preferences, and Help.

## Editor

With the amazing editor in Codeanywhere, you will forget you ever used any other code editor.

## Remote Connection

Edit your code remotely wherever it is; FTP, SFTP, even Dropbox or Google

## Terminal console

Use built-in terminal console to run any command on your Container or even remote servers (using SSH)

## Revisions

Revisions allow you to see differences between each and every one of your saves.

練習：  
請在你的電腦上安裝 Git

開始使用 Git

設定

# 檢視目前設定

```
$ git config --list
```

# 設定 username 及 email

```
$ git config --global user.name "eddie"  
$ git config --global user.email "eddie@5xruby.tw"
```

`~/.gitconfig`

**更方便的設定**

# 便利的設定

```
[alias]
  co = checkout
  br = branch
  aa = add --all
```

練習：

設定你的使用者名稱及 email.

不要害怕終端機或指令模式  
(仔細看它的輸出結果)

新增、初始 Repository

# 建立目錄

```
$ mkdir demo
```

# 初始化

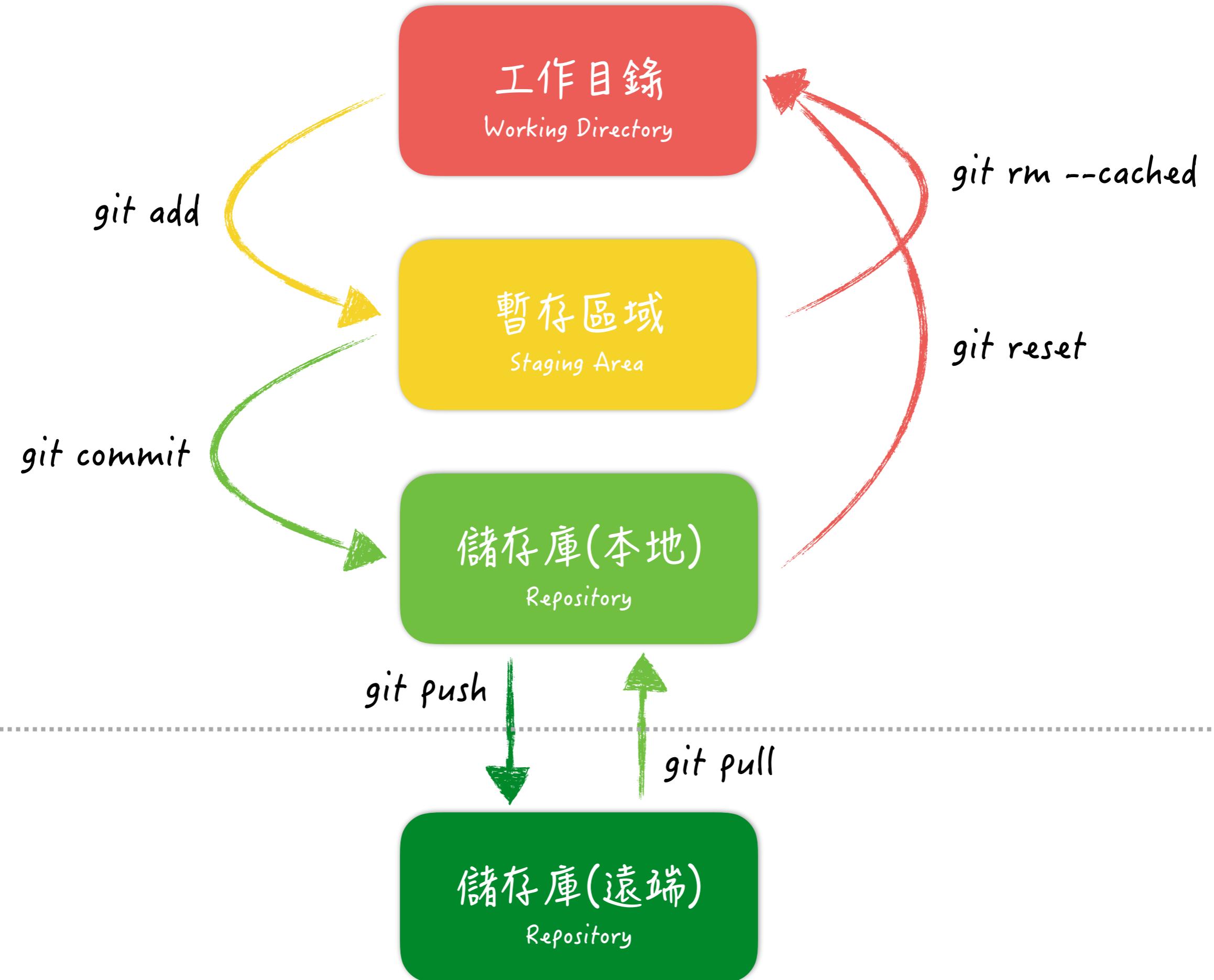
```
$ cd demo  
$ git init  
Initialized empty Git repository in /  
private/tmp/demo/.git/
```

練習：

建立一個新的目錄，並且 Git 進行  
初始化。

Working, Staging, and Repository

工作目錄, 暫存區域, 以及 儲存庫



# 狀況

我新增、修改了一些檔案，  
要怎麼用 Git 來管理？

把檔案放到 Git 裡

新增檔案 index.html

```
$ touch index.html
```

加到 staging

```
$ git add index.html
```

練習：

新增一個名為 “index.html”的檔案  
並增存放至 staging area

# 狀況

我要怎麼知道目前 Git 的管  
理狀態?

查看狀態

# 查看狀態

```
$ git status
```

```
Initial commit
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   index.html
```

練習：

試著把放在 staging area 的 “index.html” 改回原來的狀態

提示：git rm --cached

# 問題

git add 之後就算完成存檔了嗎？

提交 Commit

# 提交

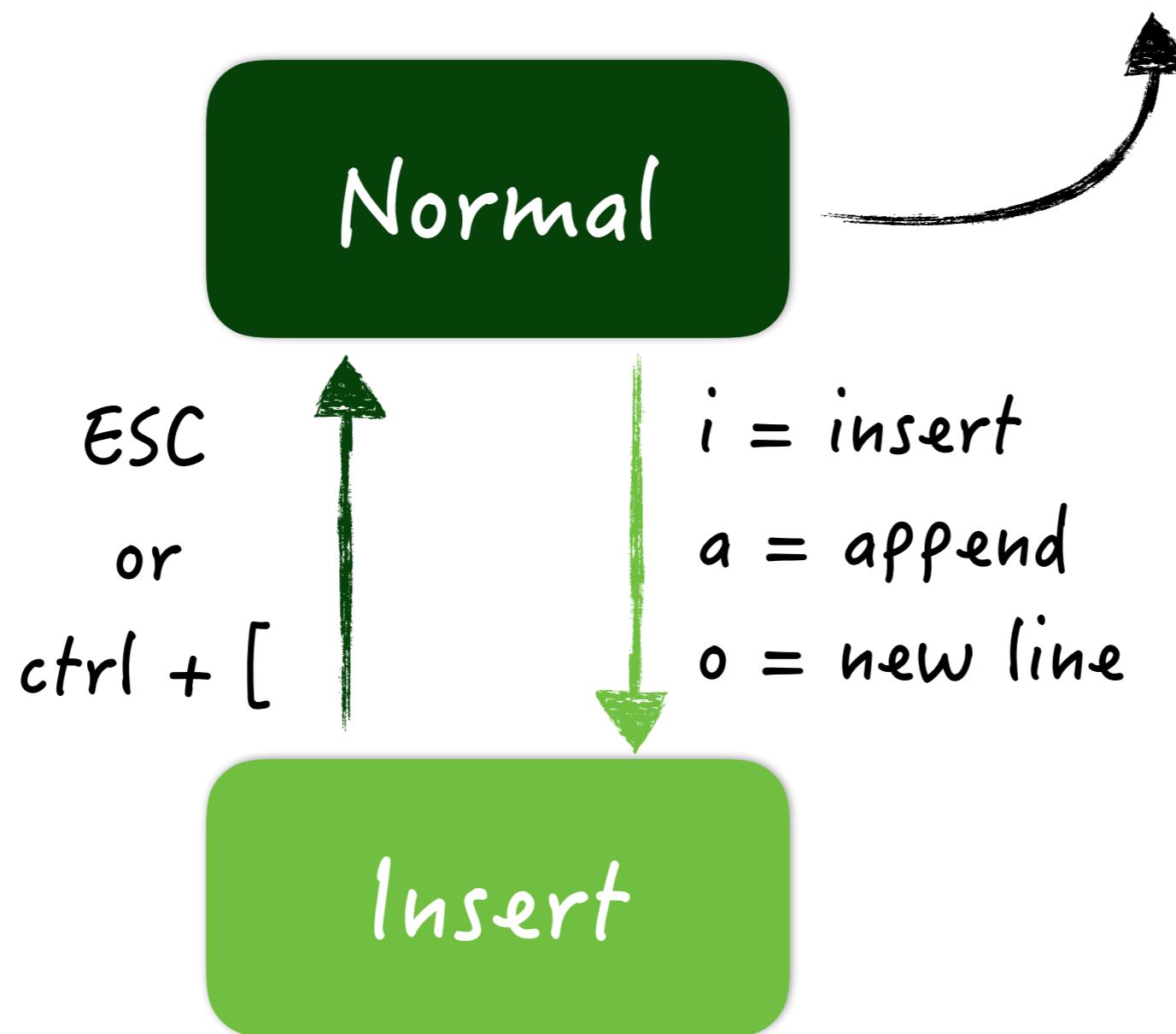
```
$ git commit -m "add index.html"  
  
[master (root-commit) cb96971] add index.html  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 index.html
```

# 超精簡 Vim 介紹

:w 存檔

:q 離開

:wq 存檔然後離開



# 不想用預設的編輯器？

emacs:

```
$ git config --global core.editor emacs
```

sublime text:

```
$ git config --global core.editor "subl -n -w"
```

atom:

```
$ git config --global core.editor "atom --wait"
```

練習：

*just commit it :)*

# 問題

什麼時候要 commit?

訊息很重要！

# 狀況

遇到澳客，做得很不爽，因為  
心情不好，不小心在 commit 訊  
息罵了客戶了，要怎麼消掉？

修改最後一次 commit

```
$ git commit --amend
```

練習：

你剛剛不小心在提交的訊息裡罵了  
客戶，想辦法把這個問題修正吧！

# 狀況

剛剛完成 commit，但發現有一個檔案忘了加到，又不想為了這個檔案重新再發一次 commit...

把 commit 併到上一次的 commit

```
$ git add hello.rb  
$ git commit --amend -m "update file  
and add hello.rb"
```

## 練習：

在最近一次的提交中，你漏了一個檔案，但你又不想再因此增加一次提交數，試著用 *--amend* 來把檔案加到上次的提交裡。

# 狀況

我剛剛新增了一個 images 目錄，但卻發現這個目錄好像沒辦法被加到 git 裡面？

注意事項：  
空的目錄無法被提交！

如果你還是想要提交一個空的目錄，只要隨便放個空的檔案在裡面就行了。慣例上會放個名為 ".keep" 或 ".gitkeep" 的空白檔案。

## 練習：

建立空的資料夾，並在裡面放一個  
空檔案，並完成提交。

# 狀況

我知道怎麼 commit 了，但要怎麼看之前 commit 的結果？

# 檢視提交紀錄

```
$ git log
```

```
commit cb96971765877246f4019e6771fd12d541c951e7  
Author: Eddie Kao <eddie@digik.com.tw>  
Date:   Sat Apr 16 07:34:21 2016 +0800
```

```
add index.html
```

# 檢視提交紀錄(一行精簡版)

```
$ git log --graph --oneline  
* cb96971 add index.html
```

IRON-RAILS

Stage

BRANCHES

master

REMOTES

origin

TAGS

SUBMODULES

OTHER

Short SHA	Subject
0e93768	update chapter 00
4479068	add chapter 29
f297835	fix typo
96fe1eb	Merge pull request #8 from sudoliyang/master
b1a4af1	fix markdown
3410b23	add chapter 28
ad90d88	typo
d5c3b87	add link
0b7b762	add chapter 27
6e3273c	add chapter 26
200c735	Merge branch 'master' of github.com:kaochenlong/learn-ruby-on-rails
f431771	Merge pull request #7 from sudoliyang/master
8051d86	fixed typo
f308018	update chapter 25
2734e5b	fix typo
d70fcff	add chapter 25

Subject: add chapter 27

Author:  Eddie Kao <[eddie@digik.com.tw](mailto:eddie@digik.com.tw)>

SHA: 0b7b762f5548a05

Date:  Thu Jan 12 2017 23:29:41 GMT+0800  
(CST)

Parent: 6e3273c045f00f0

add chapter 27

- README.md
- chapter27-order.md
- images/chapter27/fsm.png

練習：  
檢視一下你的提交紀錄

# 狀況

如果我只想要看某個特定檔案  
的歷史紀錄...?

# 檢視單一檔案的歷史紀錄

```
$ git log -p FILENAME
```

# 狀況

等等.. 什麼時候有這一行程式碼? 這一行誰寫的?

檢視每行程式碼的紀錄

```
$ git blame FILENAME
```

# 狀況

不小心把檔案或目錄刪掉了...

讓檔案或目錄回到最近一次的 commit 的狀態：

```
$ git checkout hello.rb
```

讓當前目錄回到最近一次的 commit 的狀態：

```
$ git checkout .
```

練習：

試著把某個檔案或目錄刪除，再用  
`git checkout` 指令救回來。

# 狀況

剛剛 commit 了，有點後悔，想要再改點東西再重新 commit...

取消最後一次提交 (soft)

```
$ git reset HEAD^
```

取消最後一次提交 (hard)

```
$ git reset HEAD^ --hard
```

讓目前的目錄回到上一次的 commit

```
$ git checkout .
```

清除 untrack 狀態的檔案

```
$ git clean -f  
$ git clean -df      # d = 清除目錄
```

練習：

試著把取消最後一次的提交

# 問題

我會 commit 了，那我就一路  
commit 到底就行了吧？

分支

對分支的誤解？

分支很便宜的

什麼時候要開分支？

新增分支 "cat"

```
$ git branch cat
```

刪除分支 "cat"

```
$ git branch -d cat  
Deleted branch cat (was cb96971).
```

檢視分支

```
$ git branch
```

```
  cat  
* master
```

切換到 "cat" 分支

```
$ git checkout cat
```

切換到 "dog" 分支，如果該分支不存在，會自動建立新的分支

```
$ git checkout -b dog
```

## 練習：

1. 建立分支名稱 "fruit"
2. 切換到 "fruit" 分支
3. 新增檔案 "banana.html" 並且完成提交

合併分支

合併分支 "cat"

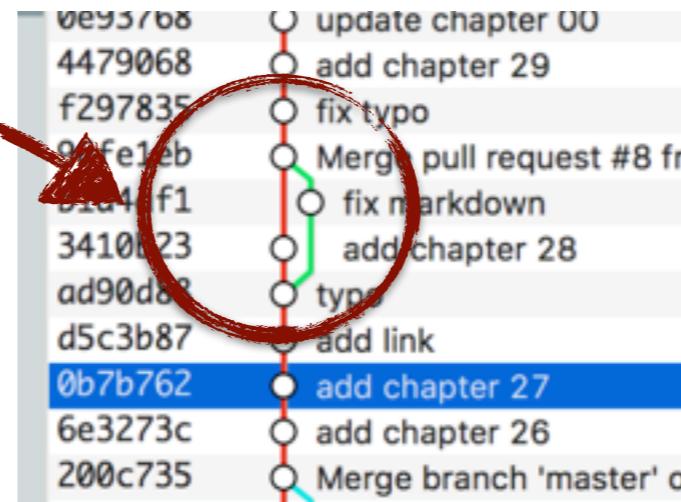
```
$ git checkout master  
$ git merge cat
```

## 練習：

1. 切換回 "master" 分支
2. 合併 "fruit" 分支到 "master"
3. 刪除 "fruit" 分支(如果不想留的話)

# 狀況

就是這種東西



我看有些人的分支都有看起來  
像分支的「小耳朵」線圖耶，  
怎麼我的分支都沒有...？

# 非快轉合併

```
$ git merge cat --no-ff
```

# 問題

合併過的分支要留著嗎？

# 檢視已經合併過的分支

```
$ git branch --merged
```

# 問題

歷史紀錄可以修改嗎？

修改提交

# 修改提交

```
$ git rebase -i cb96971

# 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
```

# 問題

發生衝突(Conflict)了，怎麼辦？

# 狀況

有看到別人的軟體會說什麼  
1.0、2.0 的，那個跟 git 有關係嗎？

標籤

里程碑

新增標籤 "1.0.0"

```
$ git tag 1.0.0
```

檢視標籤

```
$ git tag
```

練習：

請新增一個 "1.0.0-beta" 的標籤

# 狀況

我知道怎麼在自己電腦用 git 了，但要怎麼上傳到 GitHub 跟別人共用？

遠端操作

遠端節點

## 新增遠端節點

```
$ git remote add origin  
git@github.com:kaochenlong/dummy-  
git.git
```

## 刪除遠端節點

```
$ git remote rm origin
```

# 檢視遠端節點

```
$ git remote -v
```

```
origin git@github.com:kaochenlong/dummy-git.git (fetch)
origin git@github.com:kaochenlong/dummy-git.git (push)
```

上傳 Push

上傳

```
$ git push origin master
```

上傳 "cat" 分支到 origin 節點

```
$ git push origin cat
```

刪除遠端 "cat" 分支

```
$ git push origin :cat
```

下載更新 Pull

下載

```
$ git pull origin master
```

*git pull = git fetch + git merge*

從伺服器上取得 Repository

# 取得 Repository

```
$ git clone git://github.com/  
kaochenlong/eddie-vim.git
```

解決衝突 (conflict)

# 狀況

git 的 pull 是下載，clone 好像也是下載... 有不一樣嗎？

# 狀況

你手邊的工作做到一半...

老闆：「那個誰誰誰，網站掛了，你趕快先來修一下這個功能...」

把目前狀況存下來

```
$ git stash save
```

把最後一次的 *stash* 拿出來用

```
$ git stash apply
```

刪除最後一次的 *stash*

```
$ git stash drop
```

其它狀況

# 狀況

有些比較機密的檔案我不想放在 git 裡面一起備份...

*.gitignore*

<https://github.com/github/gitignore>

# 狀況

你不小心把帳號密碼寫在某個檔案裡，提交而且推出了...

針對每個節點刪除特定檔案

```
$ git filter-branch --tree-filter  
"rm -f config/password.txt"
```

# 狀況

你正在 develop 分支進行某個錯誤的修正，但你突然發現這個錯誤在某個分支的某個 commit 已經修過了...

合併某個特定節點

```
$ git cherry-pick 823520ed
```

合併某個特定節點前編輯訊息

```
$ git cherry-pick 823520ed --edit
```

取得某個特定節點內容但不進行合併

```
$ git cherry-pick 823520ed --no-commit
```

# 狀況

精神不好，不小心輸入了 *git*  
*reset HEAD^ --hard* 指令，檔案  
還救得回來嗎？

調閱 reflog

```
$ git reflog
```

回到 reset 之前的節點

```
$ git reset --hard 823520ed
```

# 狀況

前一天沒睡飽，不小心把還沒  
合併的分支刪掉了，救得回來  
嗎？！

調閱 reflog

```
$ git reflog
```

使用那個節點做出新的 branch

```
$ git checkout -b new_branch_name  
823520ed
```

GithHub

GitHub 是什麼？

Git Repository Server

開發者們的 facebook :)

[Overview](#)

Repositories 97

Stars 595

Followers 375

Following 11

## Popular repositories

[Customize your pinned repositories](#)

### [eddie-vim](#)

Yet another vimrc

● VimL ★ 329 ¥ 144

### [eddie-vim2](#)

Yet another vimrc

● VimL ★ 86 ¥ 23

### [learn-ruby-on-rails](#)

為你自己學 Ruby on Rails

★ 61 ¥ 9

### [programming-basic](#)

程式設計入門 - 使用 Ruby 及 Swift

★ 20 ¥ 3

### [rails\\_app\\_template](#)

● Ruby ★ 17 ¥ 13

### [ntub\\_homework](#)

● Ruby ★ 12 ¥ 47

## Eddie Kao kaochenlong

iOS App/Ruby/Rails Developer and Instructor, PHPConf / WebConf Taiwan Founder, Rails Girls Taipei Organizer. 5xRuby Founder.

[Developer Program Member](#)

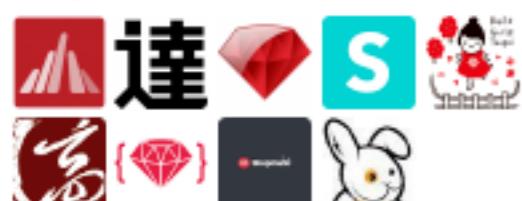
[5xRuby 五倍の紅宝石](#)

Taiwan, Taipei

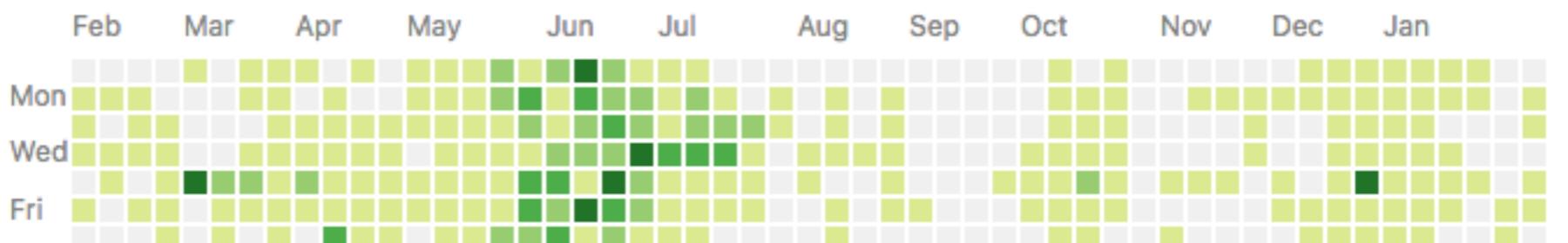
[eddie@digik.com.tw](mailto:eddie@digik.com.tw)

<http://kaochenlong.com/>

## Organizations



2,469 contributions in the last year

[Contribution settings ▾](#)

[Learn how we count contributions.](#)

Less More

跟厲害的開發者們交朋友 :)

開發者最好的履歷！

使用 GitHub

練習：

在 GitHub 上註冊帳號，新增一個 Repository，並上傳你目前的進度。

## 練習：

1. 複製一份你在 GitHub 上開的專案
2. 做些修改
3. 提交並上傳至原來的專案

## 練習：

1. 建立一個新的本地分支
2. 增加或修改一些檔案，然後把這個分支推上 GitHub

tag 預設不會被 push 上去

把標籤 "1.0.0" 推上去

```
$ git push origin 1.0.0
```

把所有的標籤推上去

```
$ git push origin --tags
```

與其它開發者互動

Fork & Pull Request (PR)

## 練習：

1. fork 一個坐在你旁邊的那位同學的專案.
2. 做些簡單的修改
3. 發送一個 pull request.

# 狀況

你從 GitHub 上 fork 了專案，該專案後來因為不斷的更新，但你的專案還是停在當初 fork 時候的狀態...

step 1: 新增原來 repo 的遠端節點

```
$ git remote add upstream ORIGIN_REPO
```

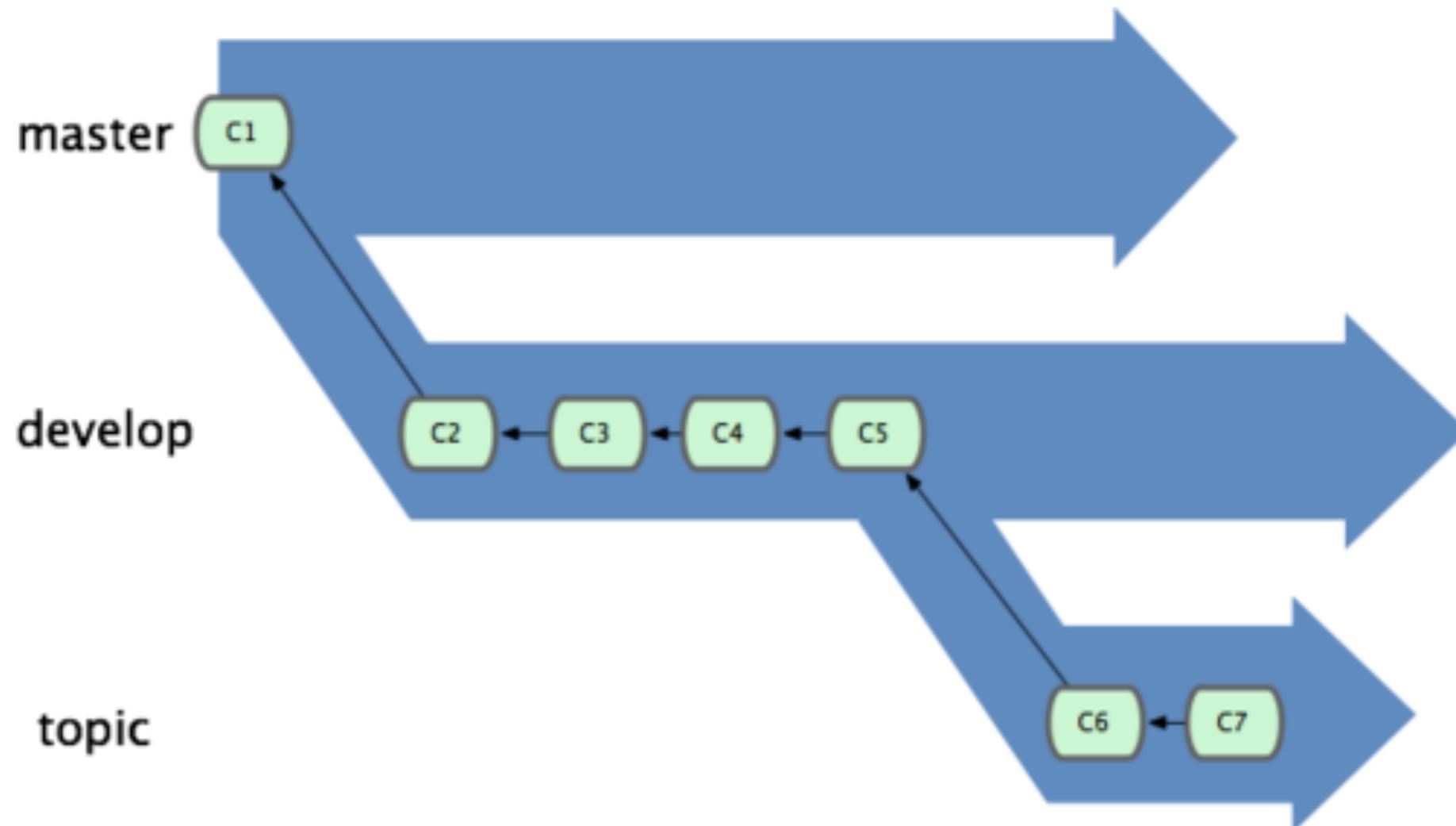
step 2: 取得該節點檔案

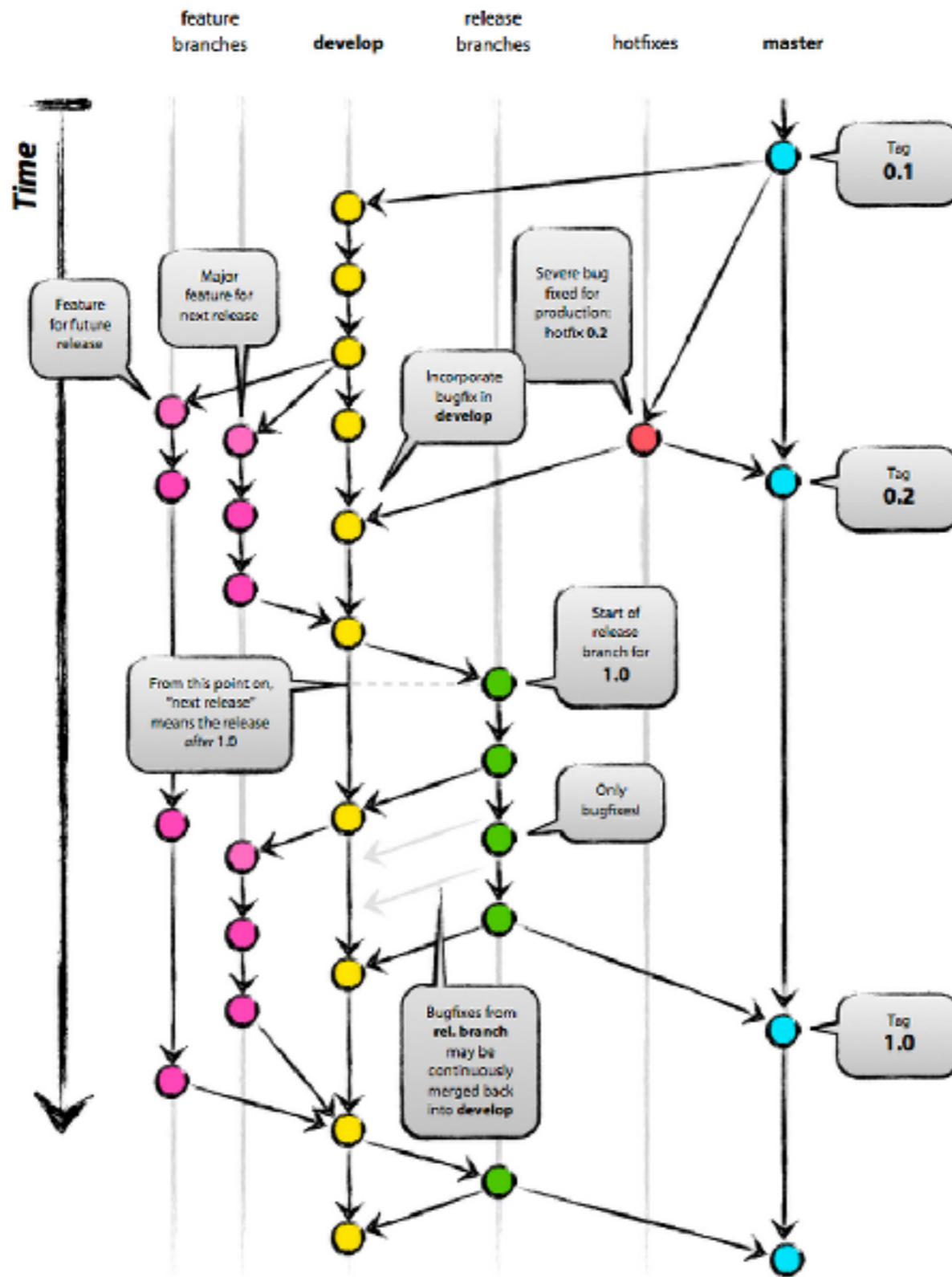
```
$ git fetch upstream
```

step 3: 合併

```
$ git merge upstream/master
```

Git Flow





# 分支

Master, Develop, Feature, Release, Hotfix

## 練習：

試著使用 Git flow 來新增一個  
feature 分支(member)，完成後合併  
回 develop 分支。

something else?

.git 目錄裡的東西？

## About

### Documentation

[Reference](#)[Book](#)[Videos](#)[External Links](#)

## Blog

## Downloads

## Community

Download this book in [PDF](#), [mobi](#), or [ePub](#) form for free.

This book is translated into [Deutsch](#), [简体中文](#), [正體中文](#), [Français](#), [日本語](#), [Nederlands](#), [Русский](#), [한국어](#), [Português \(Brasil\)](#) and [Čeština](#).

Partial translations available in [Arabic](#), [Español](#), [Indonesian](#), [Italiano](#), [Suomi](#), [Македонски](#), [Polski](#) and [Türkçe](#).

Translations started for [Azərbaycan dili](#), [Беларуская](#), [Català](#), [Esperanto](#), [Español \(Nicaragua\)](#), [فارسی](#), [हिन्दी](#),

# Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](#). Print versions of the book are available on [Amazon.com](#).



2nd Edition (2014)

[Switch to 1st Edition](#)

## 1. Getting Started

- [1.1 About Version Control](#)
- [1.2 A Short History of Git](#)
- [1.3 Git Basics](#)
- [1.4 The Command Line](#)
- [1.5 Installing Git](#)
- [1.6 First-Time Git Setup](#)
- [1.7 Getting Help](#)
- [1.8 Summary](#)

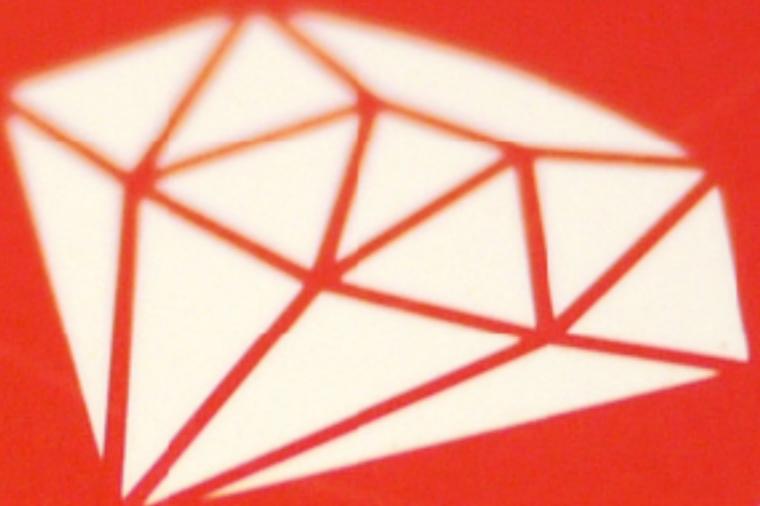
## 2. Git Basics

- [2.1 Getting a Git Repository](#)
- [2.2 Recording Changes to the Repository](#)
- [2.3 Viewing the Commit History](#)

## Download Ebook



5x



五倍紅寶石

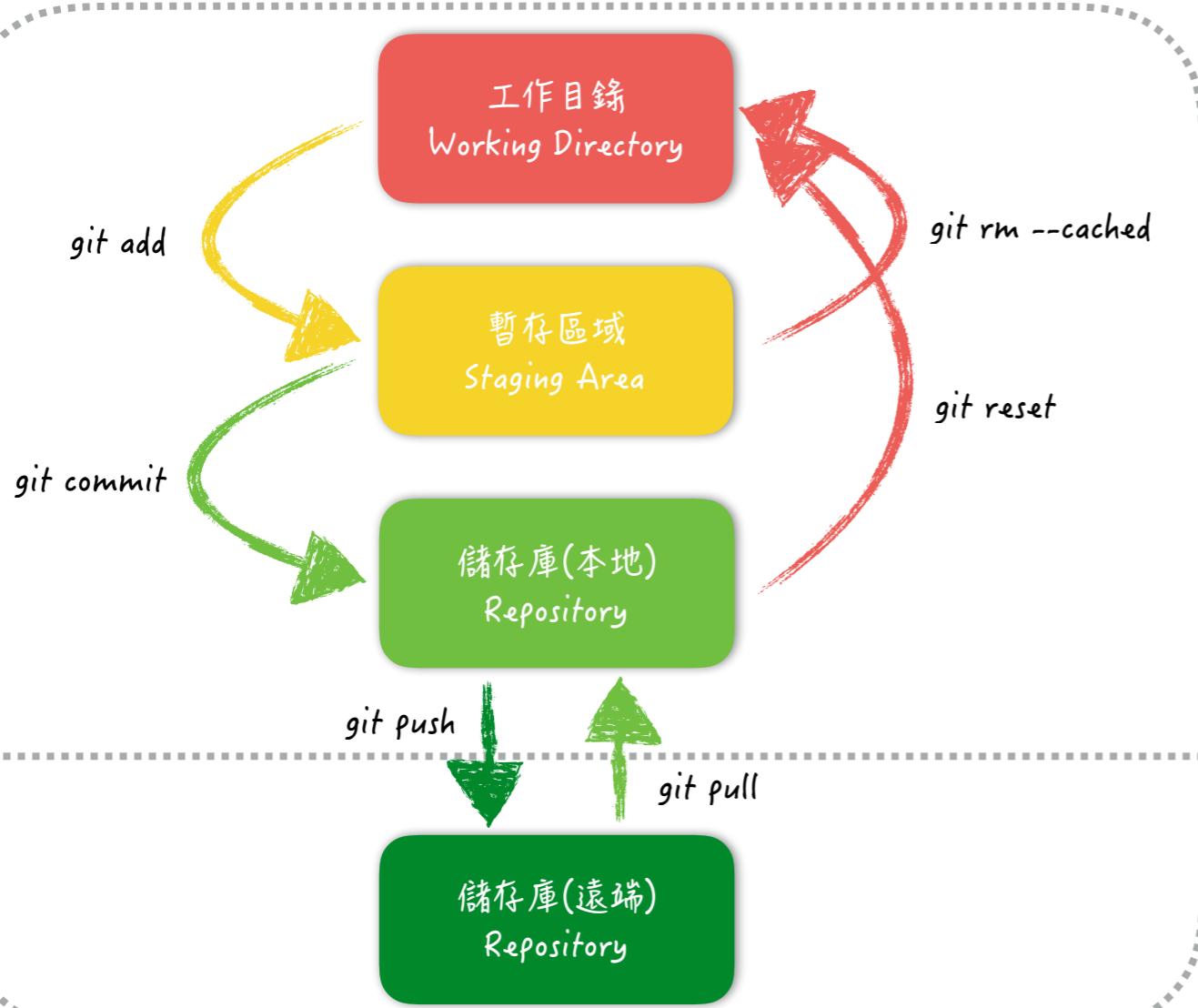
j.tw

高見龍

- Blog
- Facebook
- Twitter
- Email
- Mobile

<http://kaochenlong.com>  
<http://www.facebook.com/eddiekao>  
<https://twitter.com/eddiekao>  
[eddie@5xruby.tw](mailto:eddie@5xruby.tw)  
[+886-928-617-687](tel:+886-928-617-687)

Git 小抄



新增遠端節點 "origin" :

```
$ git remote add origin REMOTE_URL
```

刪除遠端節點 "origin" :

```
$ git remote rm origin
```

檢視遠端節點 :

```
$ git remote -v
```

把 "master" 分支內容推往 "origin" 節點 :

```
$ git push origin master
```

把遠端 "origin" 節點的 "master" 拉回本機並進行合併 :

```
$ git pull origin master
```

設定 :

```
$ git config --global user.name "5xruby"
$ git config --global user.email "hi@5xruby.tw"
```

初始化 :

```
$ git init
```

把檔案加到暫存區域 :

```
$ git add FILENAME
```

查看狀態 :

```
$ git status
```

提交 :

```
$ git commit -m "add index.html"
```

檢視紀錄 :

```
$ git log
```

取消最後一次提交 :

```
$ git reset HEAD^
```

檢視目前分支 :

```
$ git branch
```

新增分支 "5xruby" :

```
$ git branch 5xruby
```

切換分支到 "5xruby" :

```
$ git checkout 5xruby
```

合併分支 "5xruby" :

```
$ git merge 5xruby
```

刪除已合併分支 "5xruby" :

```
$ git branch -d 5xruby
```