

具有創新與創造力的程式設計開發模式



版本控管在教學上的應用

主講人：

臺北城市科技大學資管系
林慶昌博士

2017.10

具有創新與創造力的程式設計開發模式 - 版本控管



具有創新與創造力的程式設計開發模式 - 版本控管

- Git 版本控管介紹
- 如何建置版本控管服務
- 主題分支(branch)應用
- 版本控管的7大流程

電腦程式語言版本控管在教學上的應用



電腦程式語言版本控管在教學上的應用

- Github 介紹
- Bitbucket 介紹
- 版本控管服務導入電腦程式相關課程實例
- 畢業專題協作開發實例

具有創新與創造力的程式設計開發模式 - 版本控管

消失的設計力

- 程式設計課程的老師，經常面臨下列問題：
 - 上一次上課同學們的程式都在存放哪裡？USB 隨身碟？
 - 最近同學們的進度如何？
 - 分組之後 $1+1+1=1$ ，因為從頭到尾只有一個人做！沒有團隊默契！
 - 在多人的協同設計時，「版本」亂七八糟，經常出現多頭馬車，哪一份隨身碟的程式才是最新版的？

如何拯救「正在消失的設計力」？



答案可能在「版本控管」

為什麼要版本控管？

消失的設計力

- 程式設計課程的老師，經常面臨下列問題：
 - 上週同學們的程式都在存放哪裡？USB 隨身碟？
 - 最近他們的進度如何？
 - 分組之後 $1+1+1=1$ ，因為從頭到尾只有一個人做！沒有團隊默契！
 - 在多人的協同設計時，「版本」亂七八糟，經常出現多頭馬車，哪一份隨身碟的程式才是最新版的？

為什麼要版本控管？

因為

- 能讓老師們了解「我做過什麼！正在做什麼！請不要當我啊！」
- 凡走過必留下痕跡, 而且隨時可以還原任何版本
- 擺脫 $1+1+1+1+1=1$, 三個臭皮匠勝過一個諸葛亮
- 大家一起改, 能清楚知道對方改了什麼, 方便協調

為什麼要版本控管？



所以

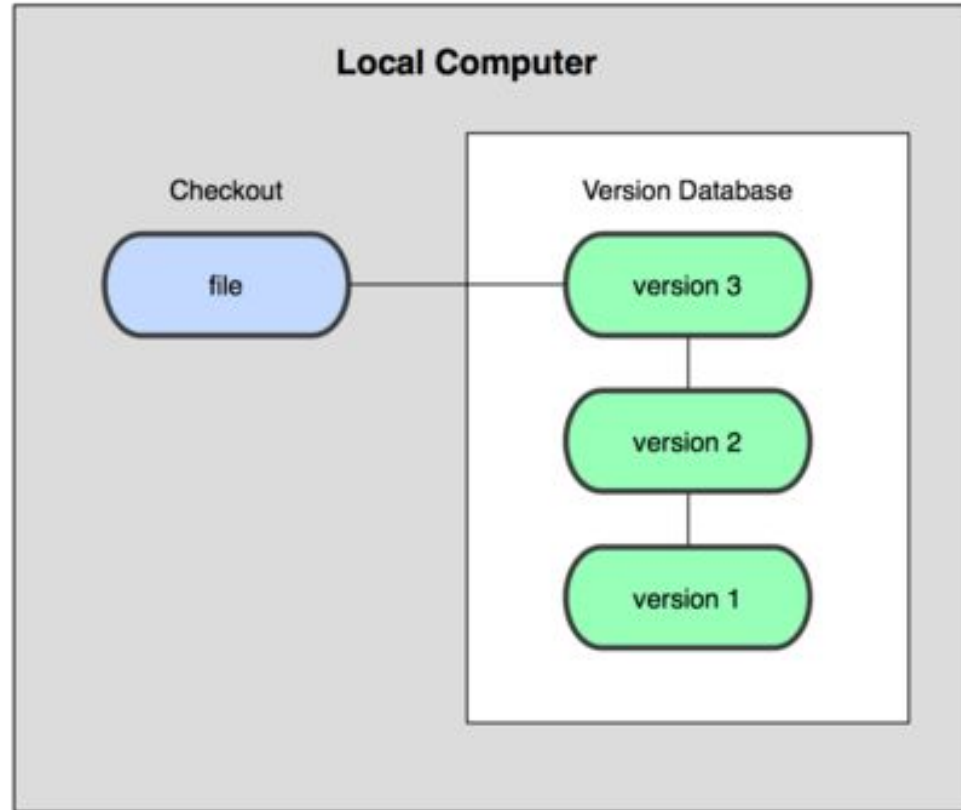
- 誰改！誰沒改！沒改的人就是米蟲，一目了然
- 發揮團隊精神，大家都能貢獻
- 設計團隊能隨時掌握專案進度，以及作品的成熟度

三種版本控管方式

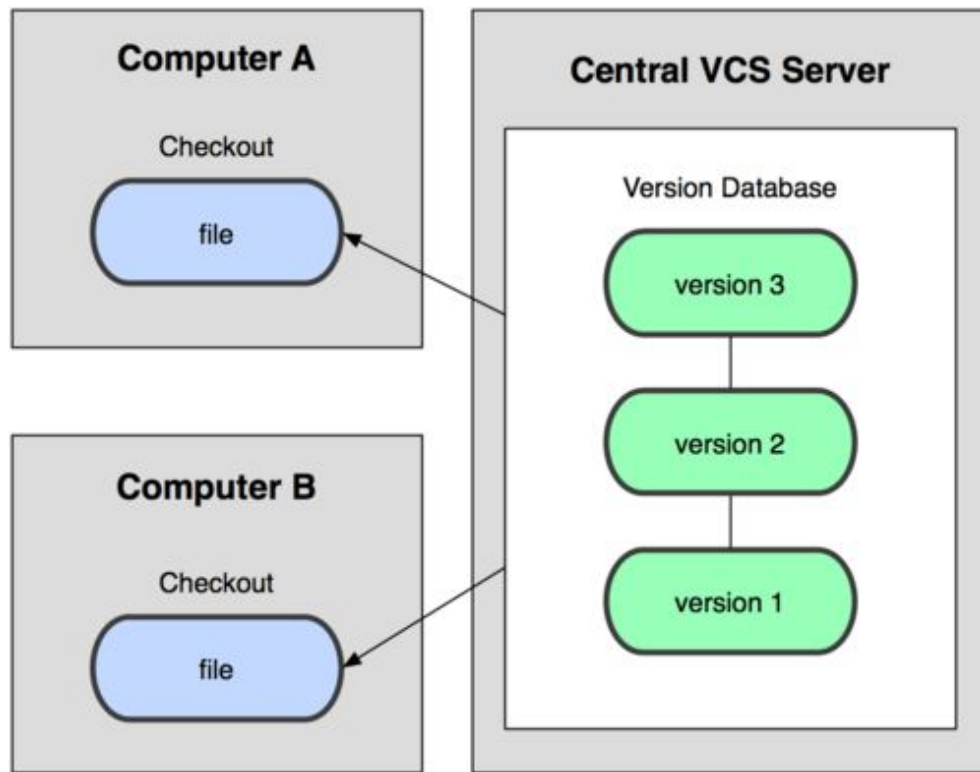


1. Local Version Control Systems (各自為政)
2. Centralized Version Control Systems (所有雞蛋放在同一個菜籃)
3. Distributed Version Control Systems (具有前兩項優點, 是目前主流)

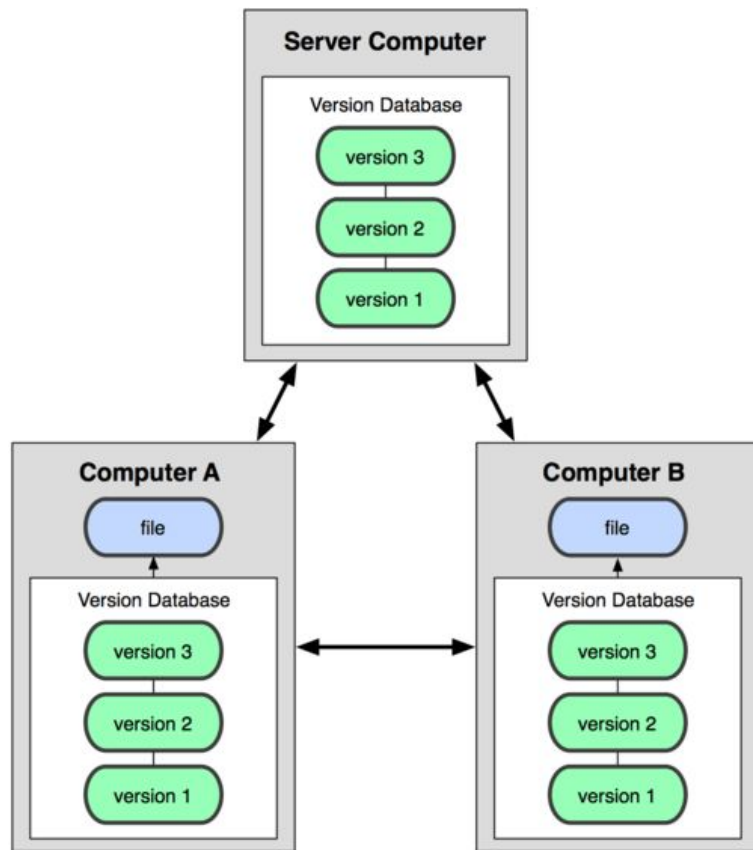
Local Version Control Systems



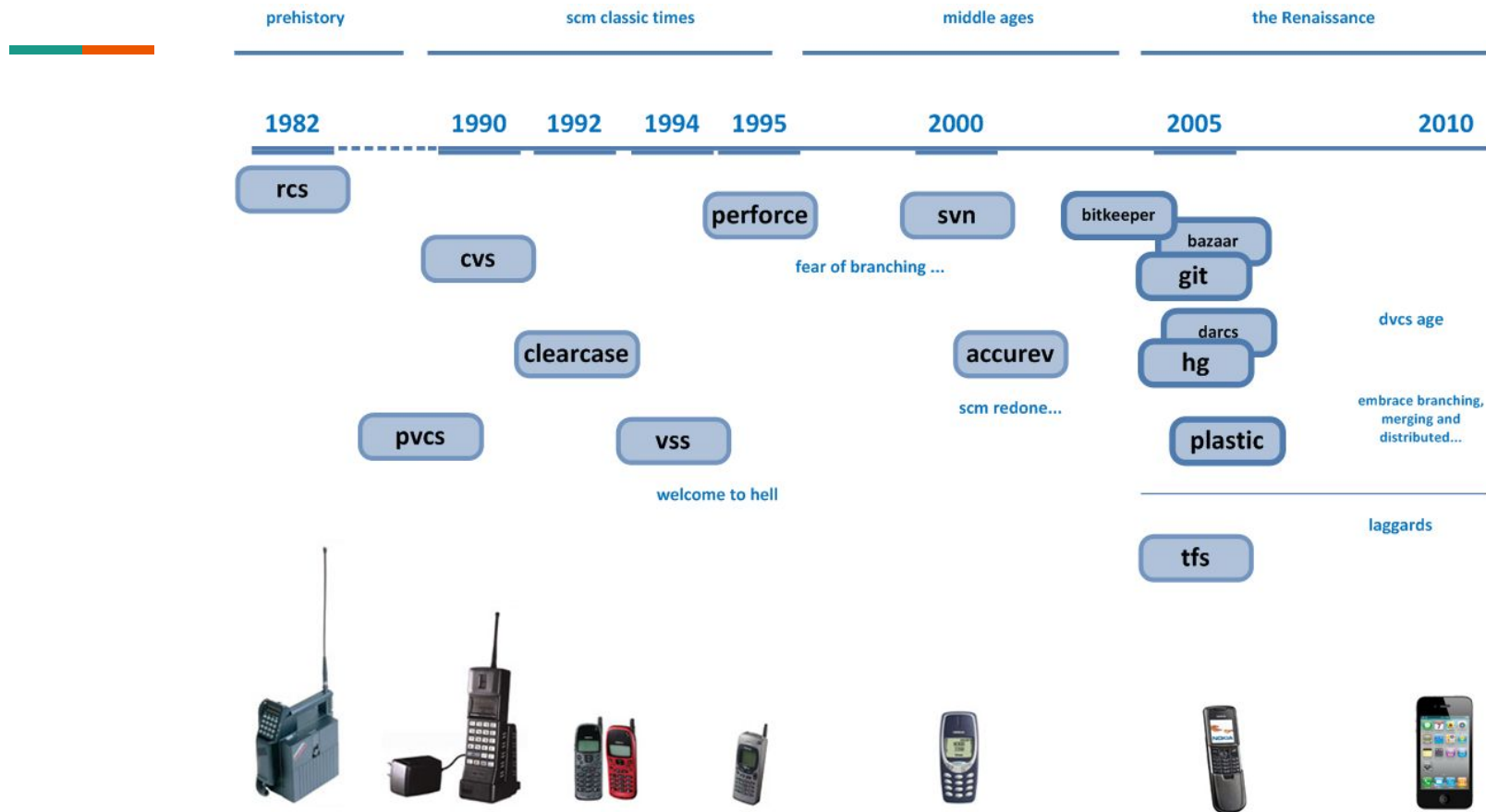
Centralized Version Control Systems 中央集權式



Distributed Version Control Systems 分散式



版本控管之前世今生



今天的主角 Git

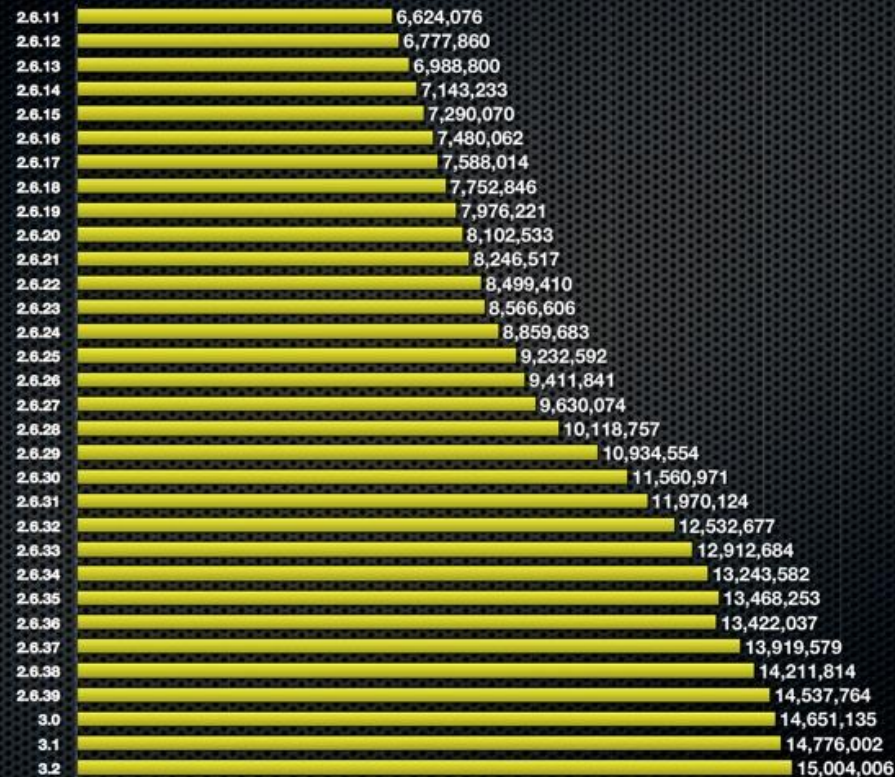


- Git 是一個分散式版本控管系統,原來是 Linux 核心開發者 Linus
- Torvalds 為了更好地管理 Linux 核心開發而創立 1。
- 從 2005 年 6 月 16 日發表了 Linux 2.6.12 核心之後,以後的 Linux 版本
- 全部採用 Git 進行發佈。
- Youtube
 - [Tech Talk: Linus Torvalds on git](#)

成功的案例: The Linux kernel is a big community

Number of lines of code in the Linux kernel

Linux kernel version

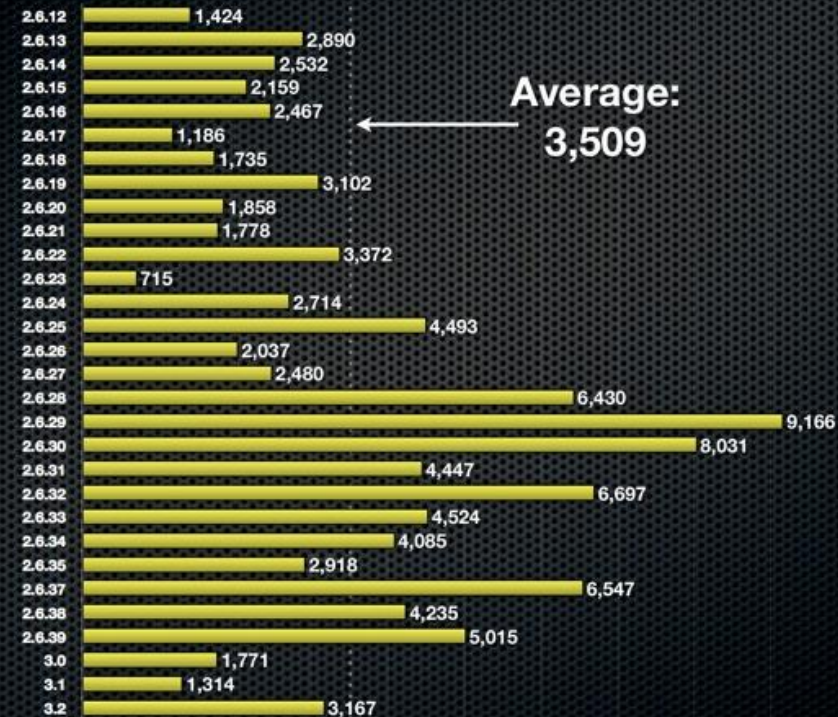


Data source: Linux Foundation

www.pingdom.com

Number of lines of code added to the Linux kernel per each day of development

Linux kernel version

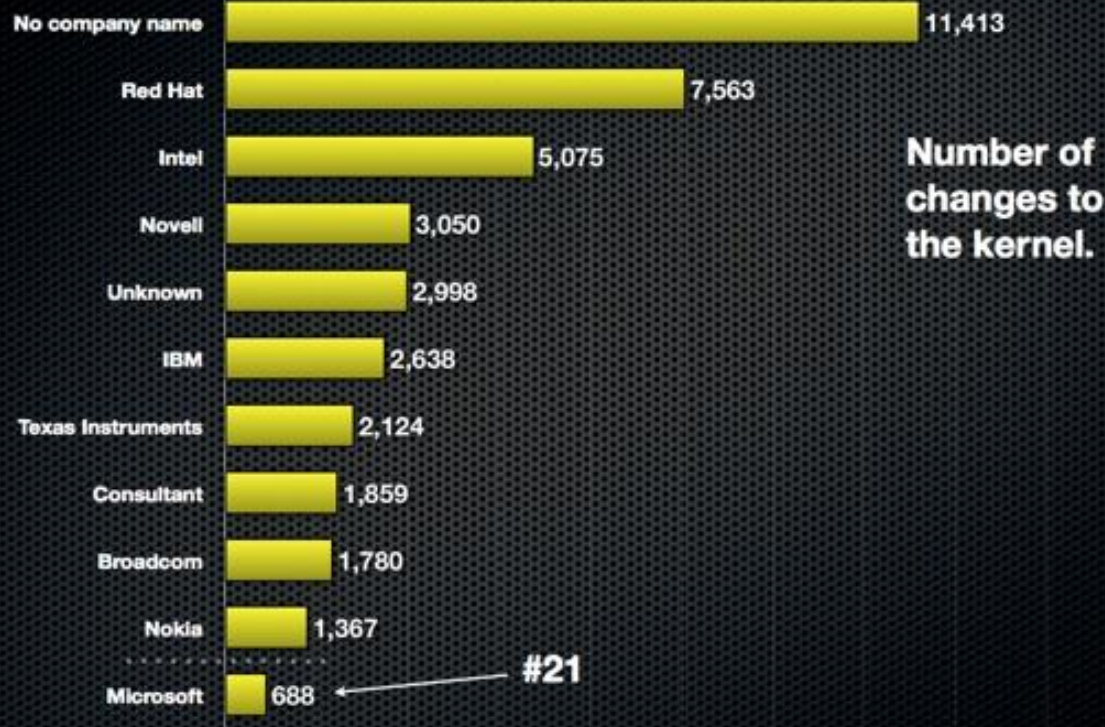


Data sources: Linux Foundation and Pingdom

www.pingdom.com

成功的案例: The Linux kernel is a big community

Top 10 contributors to the Linux kernel since version 2.6.36



Data source: Linux Foundation

www.pingdom.com

參與 Linux Kernel 開發者的身份:

”No company name” 表示開發者 (developer) 並不是替任何公司或營利單位工作,也沒有收取任何酬勞,完全是「義務志工」。

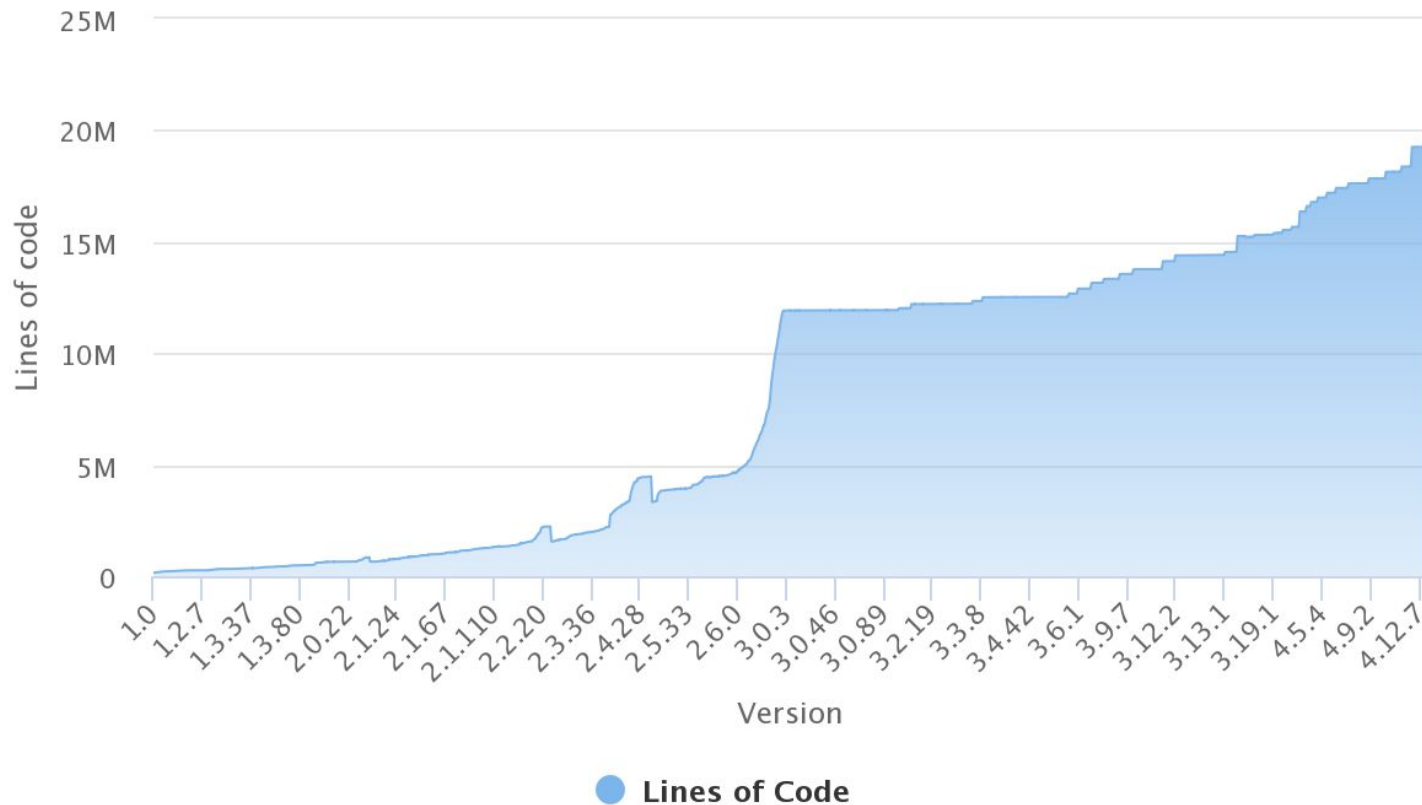
標示為”Unknow” 表示無法區別開發者 (developer) 的職業 (正職工作)

微軟公司也有上榜,排名為第 21 名。

成功的案例: The Linux kernel is a big community

Lines of code per Kernel version

Click and drag in the plot area to zoom in



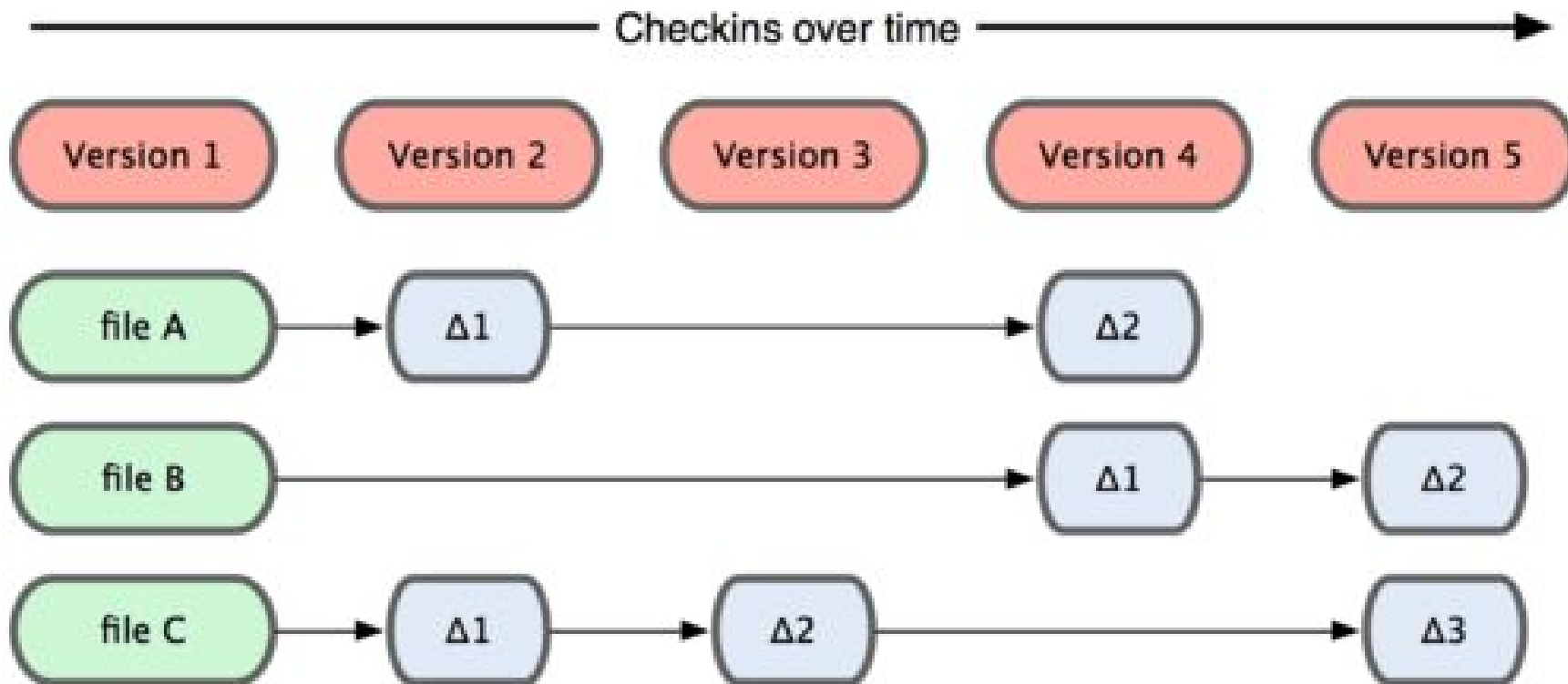
Git 的設計目標

- Speed (快速)、Simple design (簡單)
- Strong support for non-linear development (thousands of parallel branches) (支援非線性的開發)
- Fully distributed (在本機進行不需要網路連線也能單獨工作)
- Able to handle large projects like the Linux kernel efficiently (speed and data size) (能應付大型專案開發)
- Git是分散式版本控制，但每個人都有一份完整的本機儲存庫

Git 相關專業名詞

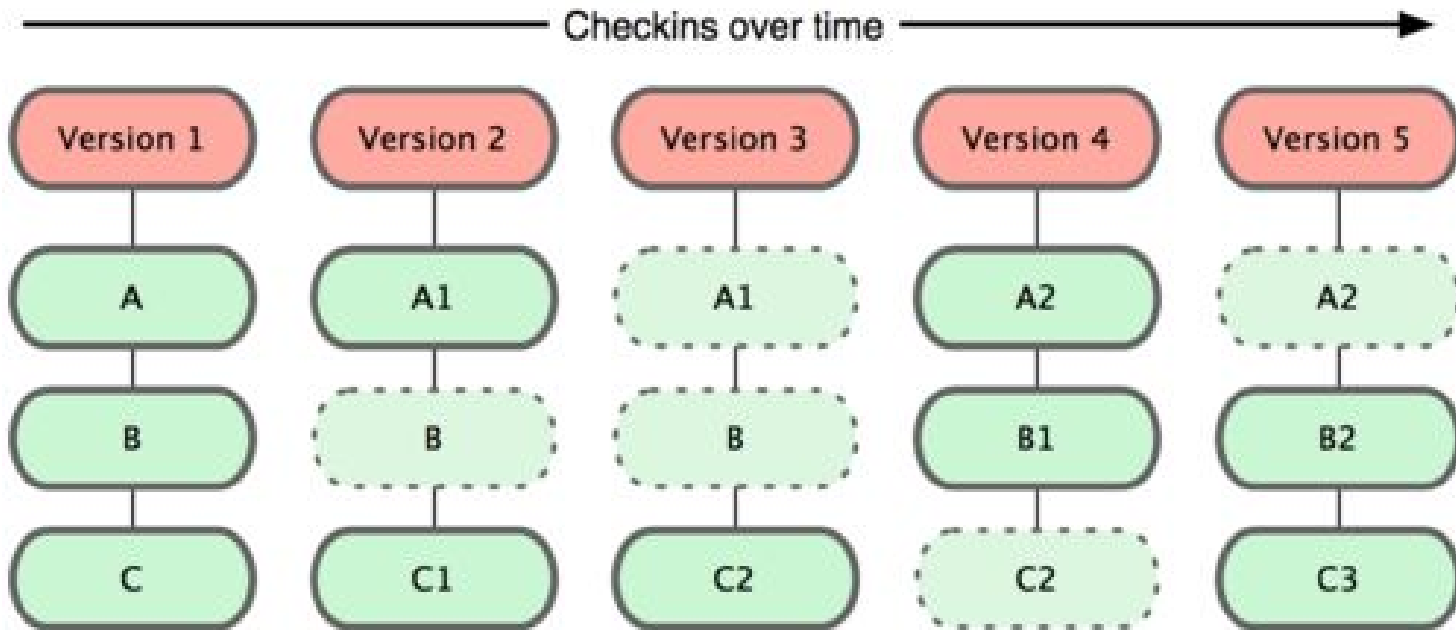
- 儲存庫 (repository)
- 提交 (commit)
- 工作目錄 (working directory)
- 暫存 (stage)
- 一個 Git 目錄裡，可以分成三種區域：
 - 目前工作目錄 (Working Directory)
 - 暫存準備遞交區 (Staging Area)
 - 儲存庫 (Repository)

傳統的版本控管作法：差異式比對儲存

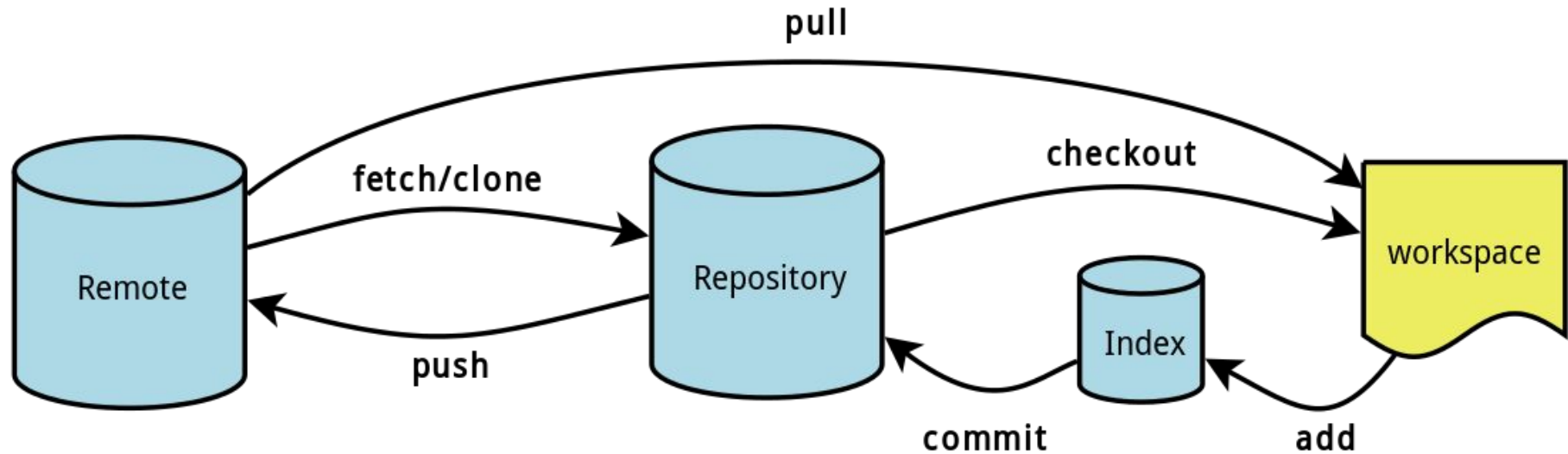


Git 的作法

- Snapshot 取代「差異式比對儲存」,稱為 Mini Filesystem
- Nearly Every Operation Is Local



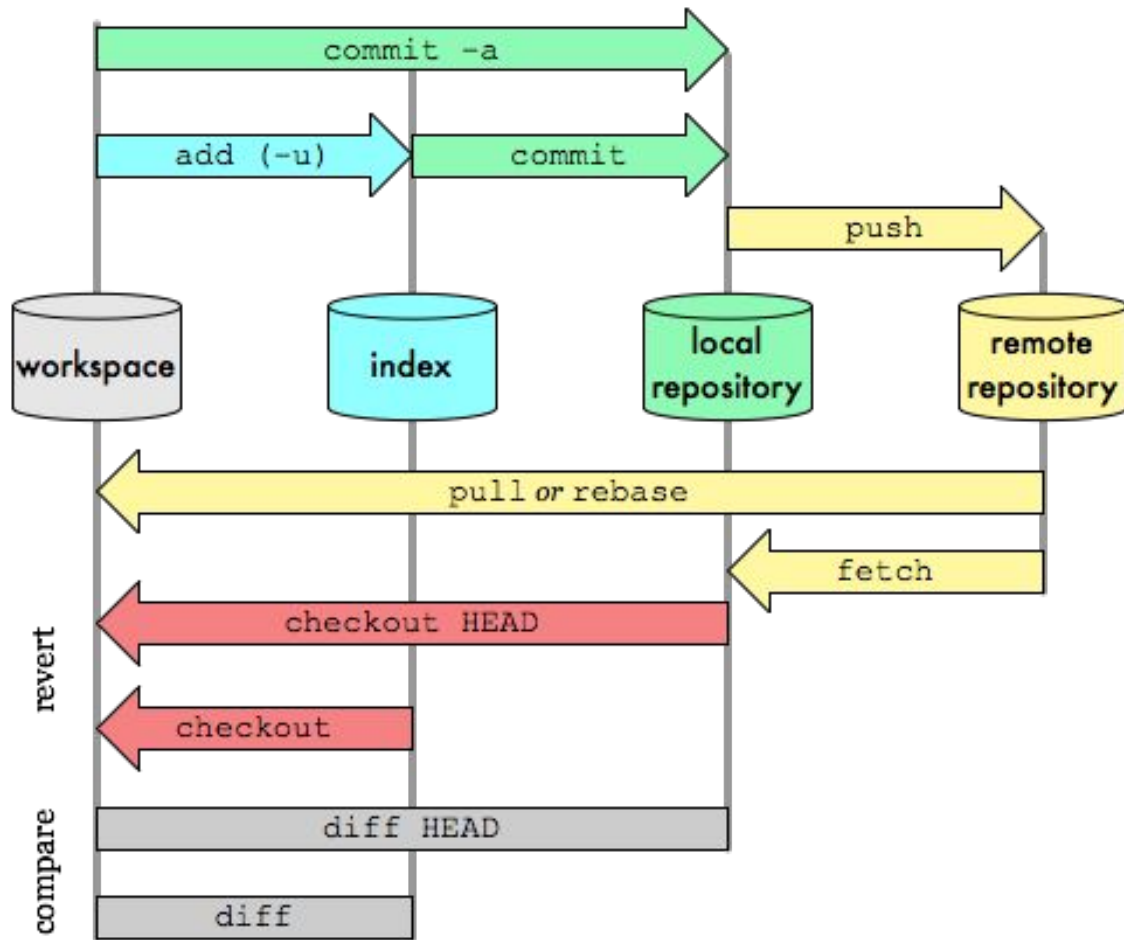
Working with remote repositories



Git Data Transport Commands

<http://osteele.com>

Git Basic



- `git init`
- `git add .`
- `git commit -am "hi"`
- `git push`
- `git pull`
- `git fetch`
- `git checkout HEAD`
- `git checkout`
- `git diff HEAD`
- `git diff`



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.



Documentation

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



Downloads

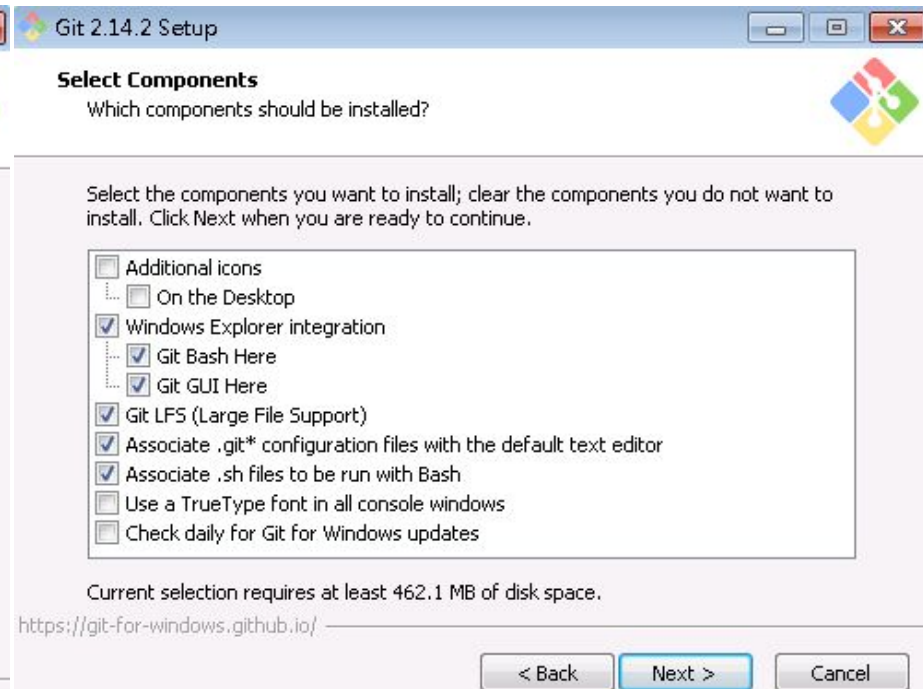
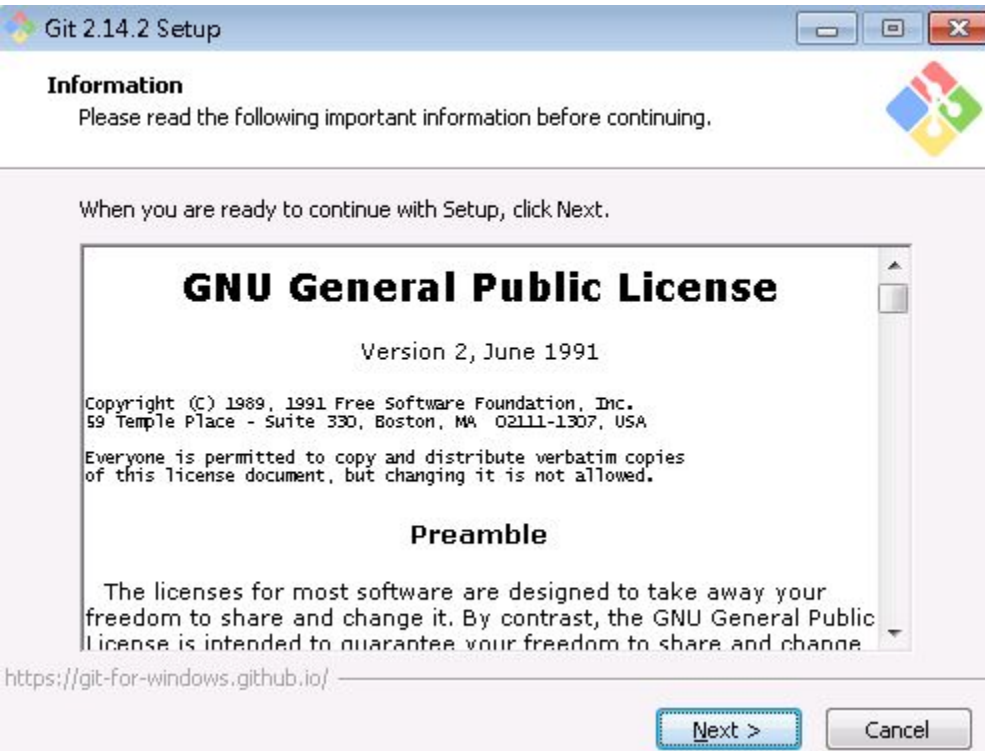
GUI clients and binary releases for all major platforms.



Community

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





Git 2.14.2 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

☐ Use Git from Git Bash only

This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

☒ Use Git from the Windows Command Prompt

This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

☐ Use Git and optional Unix tools from the Windows Command Prompt

Both Git and the optional Unix tools will be added to your PATH.

Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

<https://git-for-windows.github.io/>

< Back Next > Cancel

Git 2.14.2 Setup

Choosing HTTPS transport backend

Which SSL/TLS library would you like Git to use for HTTPS connections?

☒ Use the OpenSSL library

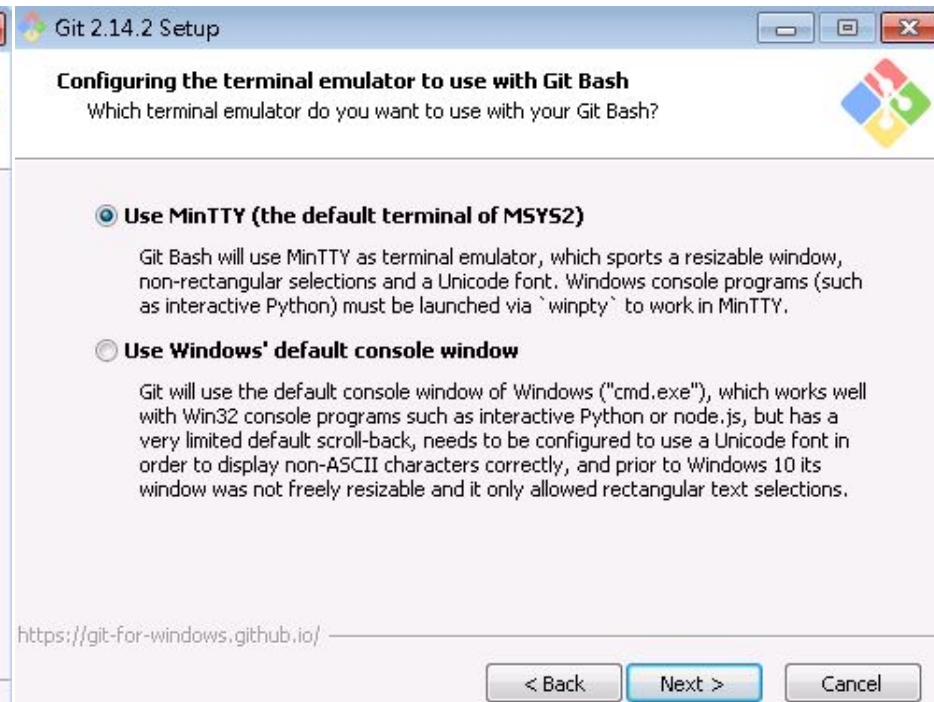
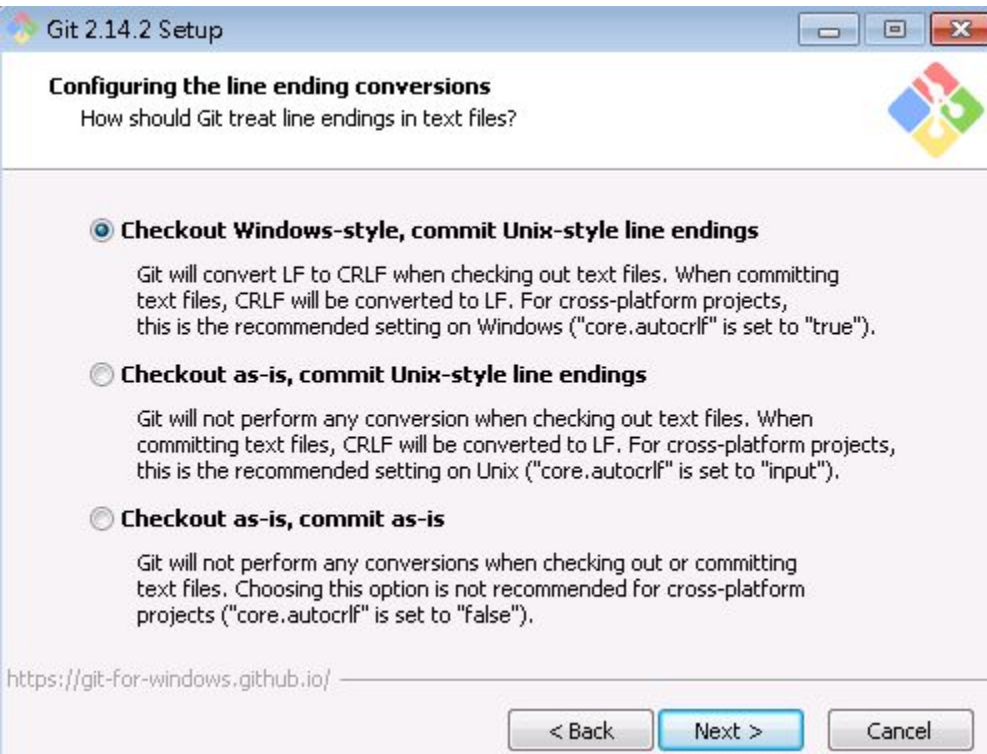
Server certificates will be validated using the ca-bundle.crt file.

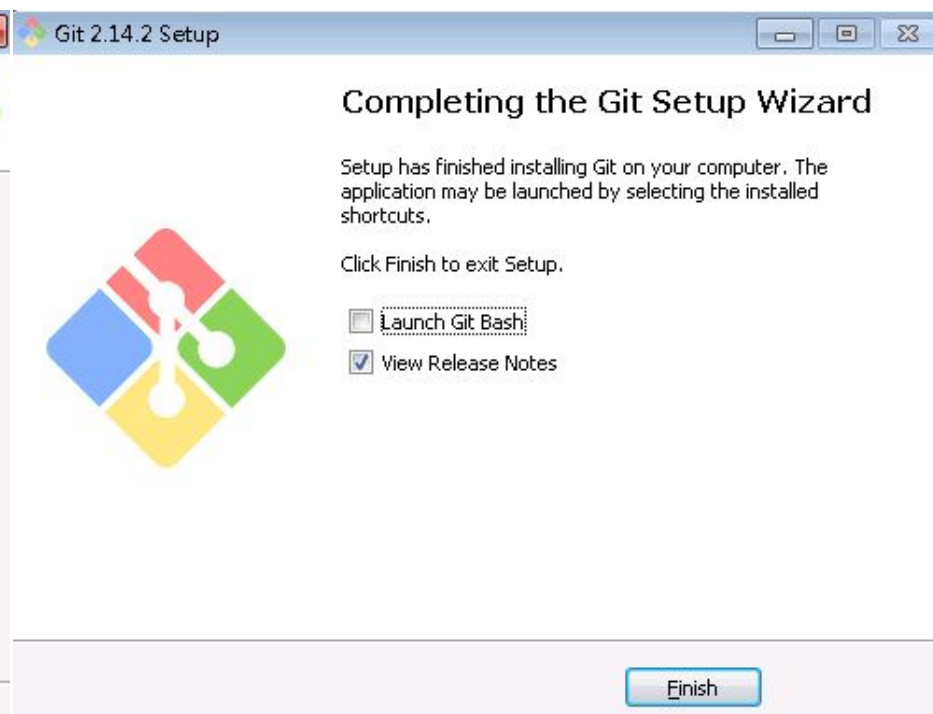
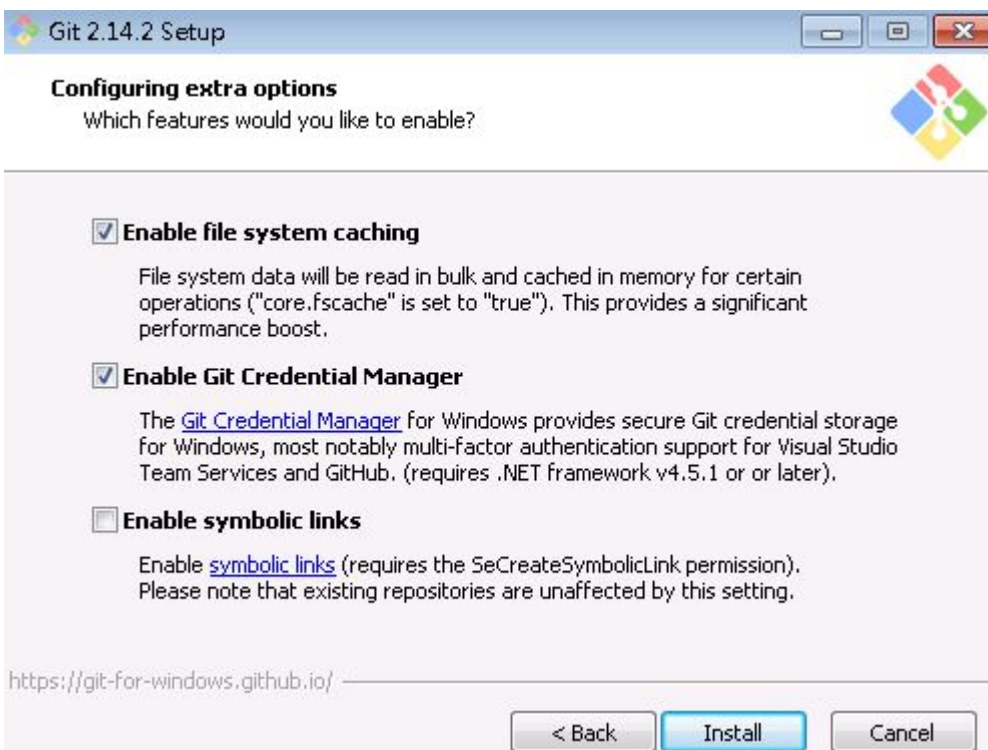
☐ Use the native Windows Secure Channel library

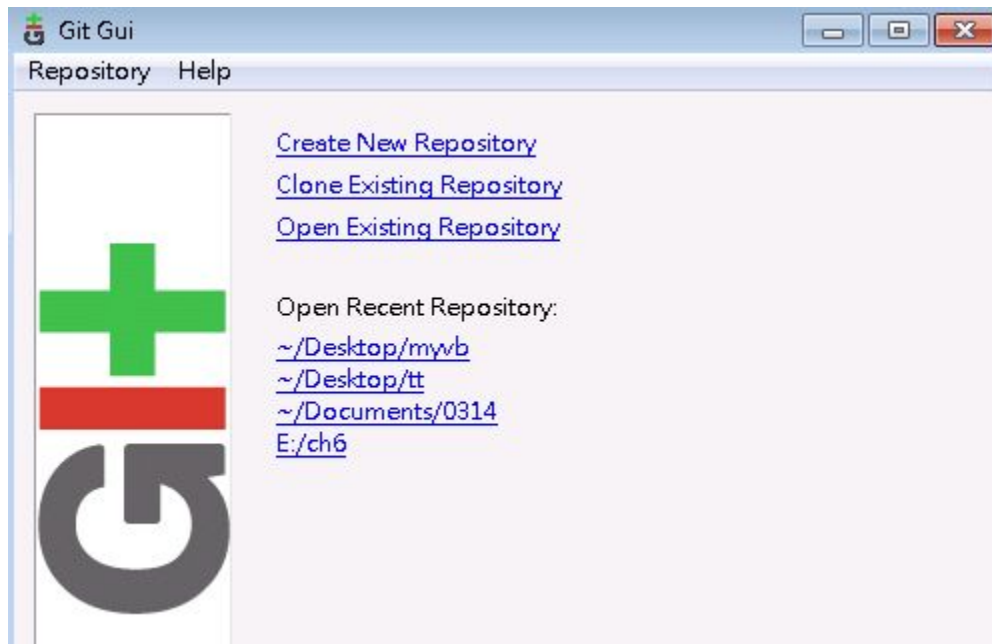
Server certificates will be validated using Windows Certificate Stores. This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.

<https://git-for-windows.github.io/>

< Back Next > Cancel







基本設定 Edit -> Options

Git Basic

The image shows the 'Git Gui (git): Options' dialog box, which is divided into two main sections: 'git Repository' and 'Global (All Repositories)'. Both sections contain identical settings for user information, merge options, and diff parameters. Blue arrows highlight the 'git Repository' and 'Global (All Repositories)' tabs, and the 'User Name' and 'Email Address' input fields in both sections.

git Repository

User Name: 輸入姓名

Email Address: 輸入電子郵件

☐ Summarize Merge Commits

Merge Verbosity: 2

☒ Show Diffstat After Merge

Use Merge Tool:

☐ Trust File Modification Timestamps

☐ Prune Tracking Branches During Fetch

☐ Match Tracking Branches

☒ Use Textconv For Diffs and Blames

☐ Blame Copy Only On Changed Files

Maximum Length of Recent Repositories List: 10

Minimum Letters To Blame Copy On: 40

Blame History Context Radius (days): 7

Number of Diff Context Lines: 5

Additional Diff Parameters:

Commit Message Text Width: 75

New Branch Name Template:

Default File Contents Encoding: utf-8 Change

☐ Warn before committing to a detached head

Staging of untracked files: ask

Global (All Repositories)

User Name: 輸入姓名

Email Address: 輸入電子郵件

☐ Summarize Merge Commits

Merge Verbosity: 2

☒ Show Diffstat After Merge

Use Merge Tool:

☐ Trust File Modification Timestamps

☐ Prune Tracking Branches During Fetch

☐ Match Tracking Branches

☒ Use Textconv For Diffs and Blames

☐ Blame Copy Only On Changed Files

Maximum Length of Recent Repositories List: 10

Minimum Letters To Blame Copy On: 40

Blame History Context Radius (days): 7

Number of Diff Context Lines: 5

Additional Diff Parameters:


Commit Message Text Width: 75

New Branch Name Template:

Default File Contents Encoding: utf-8 Change

☐ Warn before committing to a detached head

Staging of untracked files: ask

[Features](#) [Business](#) [Explore](#) [Marketplace](#) [Pricing](#)

[Sign in](#) or [Sign up](#)

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside millions of other developers.

Username

Email

Password

Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#)

[Start a project](#)

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name

Great repository names are short and memorable. Need inspiration? How about **symmetrical-octo-chainsaw**.

Description (optional)



Public

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



Private

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▼

Add a license: **None** ▼



Quick setup — if you've done this kind of thing before

or

HTTPS

SSH

`https://github.com/tpcucclin/myphp.git`



We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# myphp" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/tpcucclin/myphp.git
git push -u origin master
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/tpcucclin/myphp.git
git push -u origin master
```





END