



# 認識 Git 版本控制與 GitHub 協作平台



多奇數位創意有限公司

技術總監 黃保翕 ( Will 保哥 )

部落格：<http://blog.miniasp.com/>

Understanding Git Version Control

# 認識 GIT 版本控制



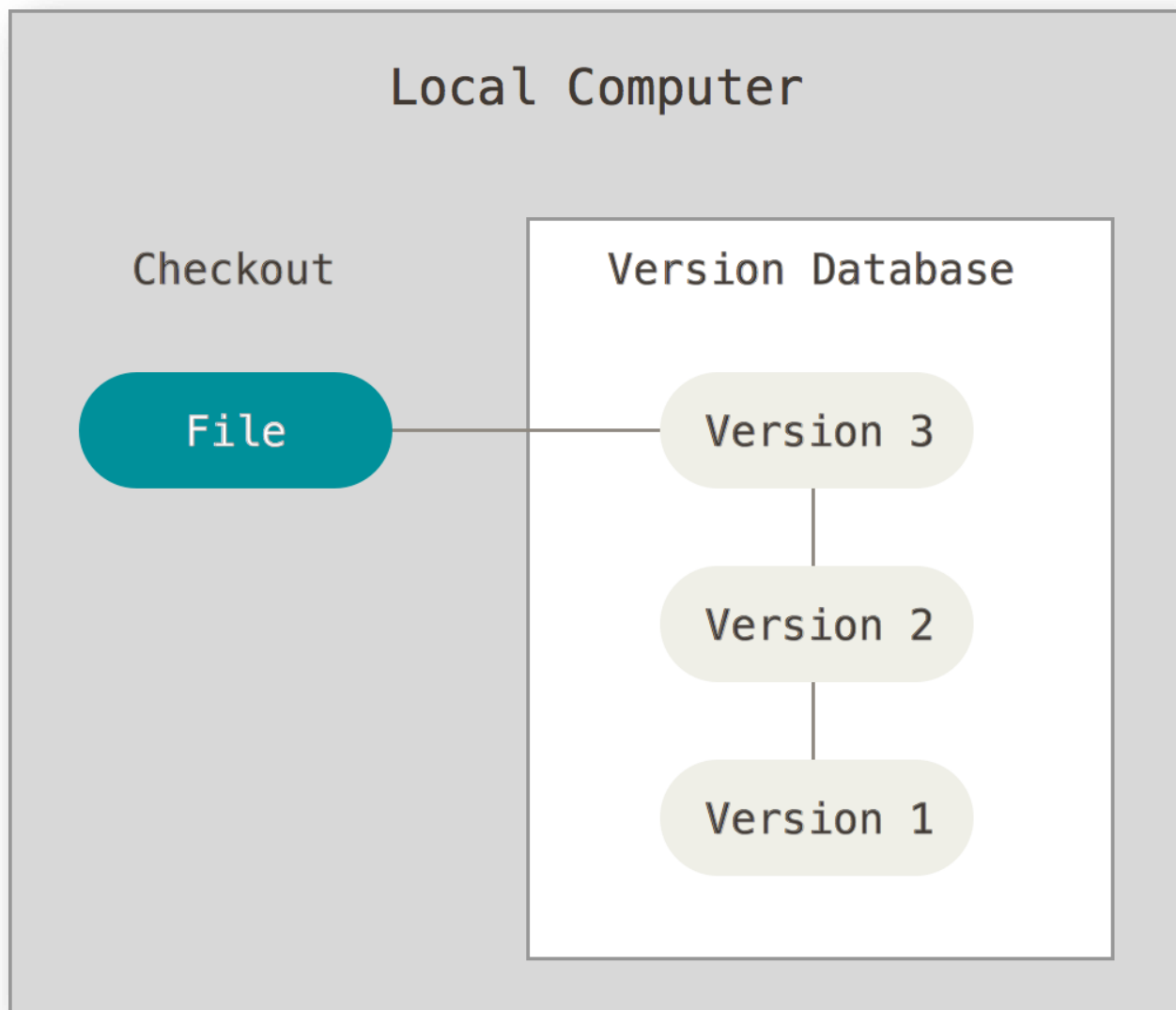
# 重要名詞解釋

- 版本控管 ([Version Control](#))
  - 完整記錄軟體**變化**的過程 (人、事、時、地、物)。
  - 為了紀錄版本變化而衍生出查詢歷史紀錄、復原變更、比對差異、標記版本、變更追蹤等需求。
  - 為了多人同時進行版控進一步衍生出協同作業、分支合併、版控流程、發行管理等進階應用。
- 工作區 (Workspace)
  - 頻繁異動的開發目錄 (資料夾) (內含版控資訊)
- 儲存庫 (Repository)
  - 本地儲存庫、遠端儲存庫、共用儲存庫
- 分支合併 (Branching & Merging)
  - 有效管理多人同步 "**變更**" 的利器

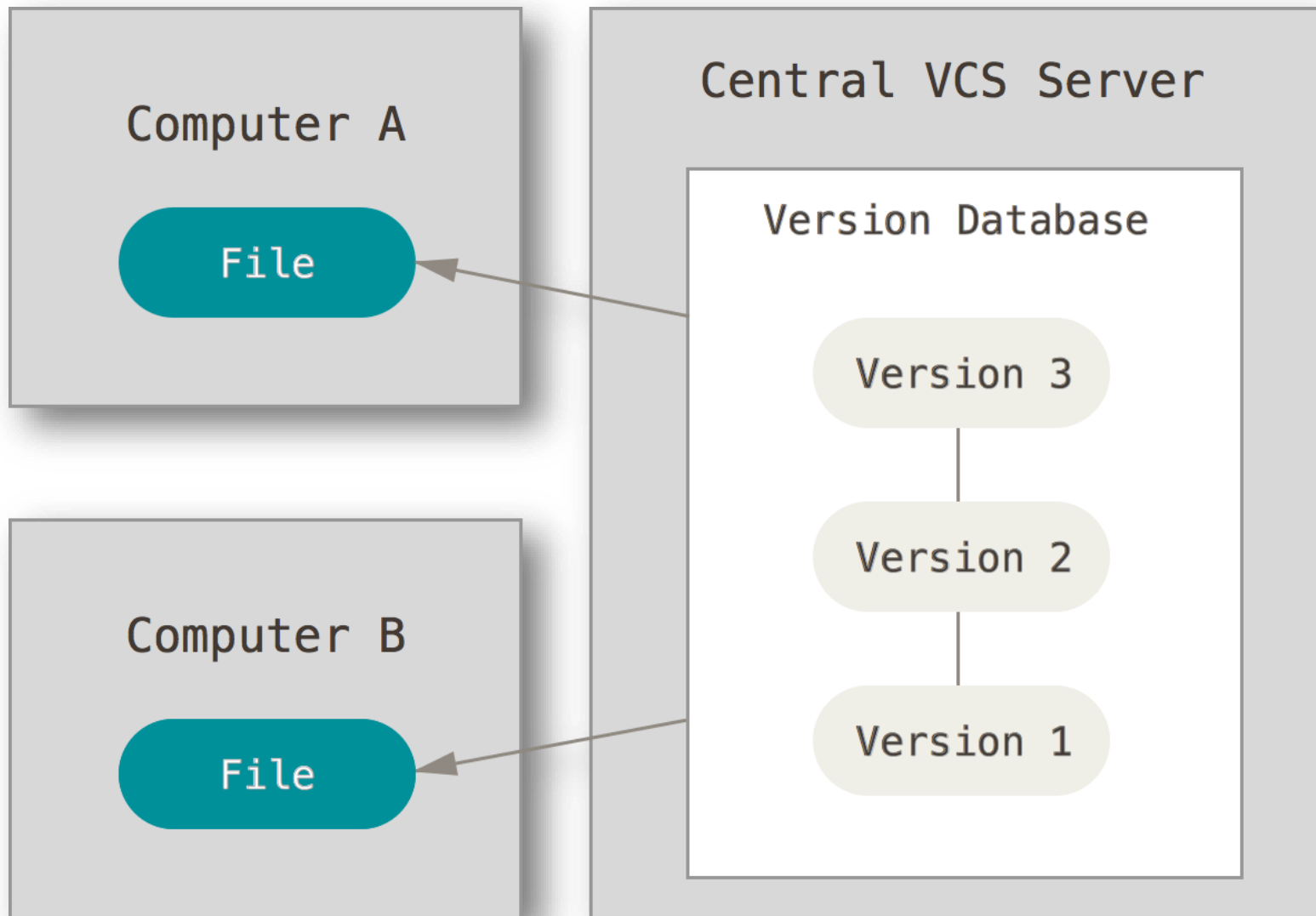
# 版本控管的種類

- 僅本地
  - 資料夾控管
    - 使用 Copy / Paste 作為變更的儲存依據 (完整保存)
  - RCS ( [Revision Control System](#) )
    - 使用 diff 工具進行內容變更的儲存依據 (差異保存)
- 集中式版本控管
  - [CVS](#) ( Concurrent Versions System )
  - [SVN](#) ( Subversion )
  - [TFVC](#) ( Team Foundation Version Control )
- 分散式版本控管
  - [Git](#)
  - [Mercurial](#)
  - [Bazaar](#)

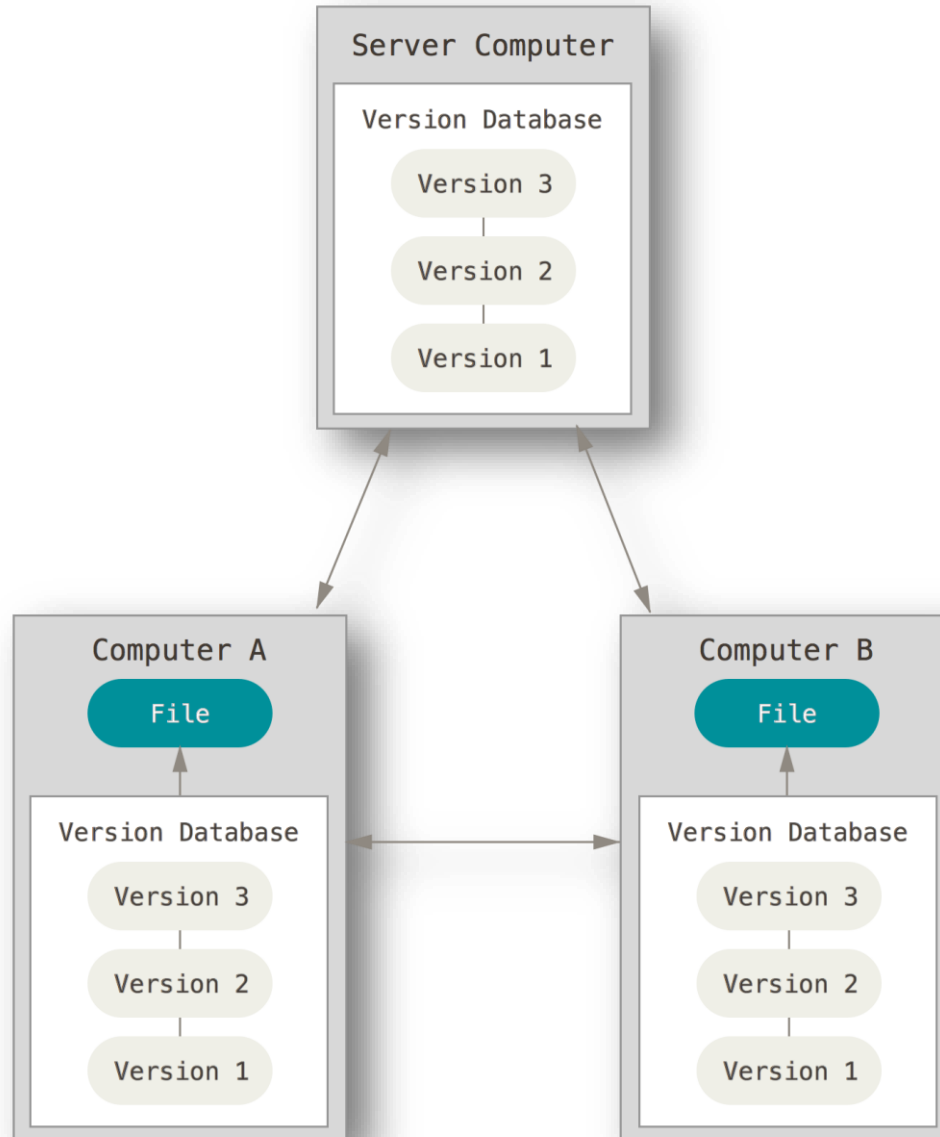
# 版本控管的種類 – 本地版控系統



# 版本控管的種類 – 集中式版控系統



# 版本控管的種類 – 分散式版控系統



# 集中式 vs. 分散式

- 集中式版本控管

- 優點

- 可選用**鎖定**或**合併**等版控策略
    - 較為精細的權限控管（針對**目錄**或**檔案**）

- 缺點

- **沒網路**就無法進行版本控管，其中包含：
      - 依然可以進行本地開發，但**無法提交新版本**
      - 無法查詢歷史紀錄（或取出先前提交的歷史版本）

- 分散式版本控管

- 優點

- **本地端**的**工作區**會保有**完整的儲存庫**
      - 等同於每個人都擁有一份**完整的儲存庫備份**
    - 可以在**本地端**建立**離線的版本與歷史紀錄**

- 缺點

- 無法採用**鎖定**版控策略（僅能使用**合併**策略）
    - 無法針對專案進行**精細的權限控管**



# 什麼是 Git ？

- 由 Linus Torvalds 發明
- 初期主要用來控管 Linux Kernel 的原始碼
- Linus Torvalds 非常擅長**檔案系統**設計
- 特色
  - 強力支援非線性開發模式 (分散式開發模式)
  - 相容於現有作業系統，易於發布儲存庫
  - 有效率的處理大型專案 (檔案多、參與人多)
  - 極佳的儲存庫完整性檢查，保護歷史紀錄
  - 多種彈性的合併策略可供選擇 (複雜)

# 關於 Git 的分散式版控系統

- 每個人都有一份完整的儲存庫副本
  - 完全不需要伺服器端的支援就可以運作版本控制
  - 每次提交版本變更時，都僅提交到本地的儲存庫
  - 提交速度非常快，也不用網路連線，可大幅節省開發時間
- 使用 Git 版本控管時，沒有所謂的**權限控管**這件事
  - 每個成員都能把儲存庫複製(clone)回來
  - 也可以在本地提交變更，沒有任何權限可以限制
  - 使用 Git 時，唯一能設定的權限是：
    - 能否存取**遠端儲存庫** (remote repository 或 upstream repository)
    - TFS (Git) 可以支援分支權限控管
- Git 擁有非常強悍的**合併追蹤**能力
  - 取得他人變更後的版本後，隨時可透過**合併**方式進行整合
  - 合併多人的版本只要有存取**共用儲存庫**的權限或管道即可。
    - 同一台伺服器上可以透過**共用資料夾**進行取得儲存庫
    - 也可透過 SSH 或 HTTP/HTTPS 遠端存取另一台伺服器的 Git 儲存庫

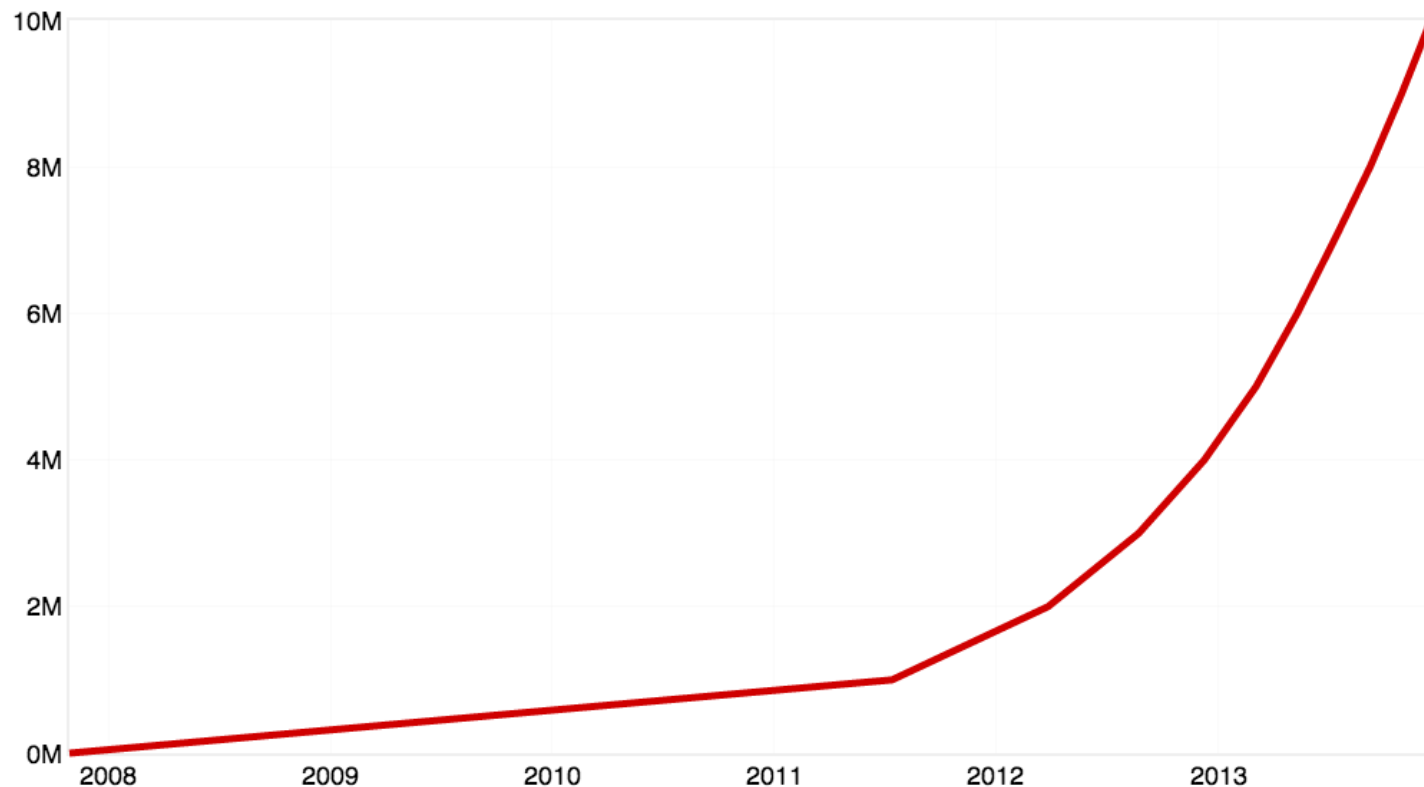
The GitHub Platform

# 認識 GITHUB 協作平台



# 超過 1 千萬個專案

- [10 Million Repositories](#) (December 24, 2013)



# GitHub 帶來的效益

- 開放原始碼
  - 包含**源碼控管**、**專案管理**、**議題管理**、**文件管理**、...
- 開放式資料
  - 強化資料開放性與協同作業
- 開放式政府
  - 透過開放促進民眾參與
    - [http://taipeicity.github.io/tpe\\_dome/](http://taipeicity.github.io/tpe_dome/)
    - <http://taipeicity.github.io/SmartCityMap/>
    - [http://taipeicity.github.io/traffic\\_realtime/](http://taipeicity.github.io/traffic_realtime/)
    - <http://taipeicity.github.io/foodtracer/>
    - [http://taipeicity.github.io/eoc\\_119/](http://taipeicity.github.io/eoc_119/)

# 選用 GitHub 的理由

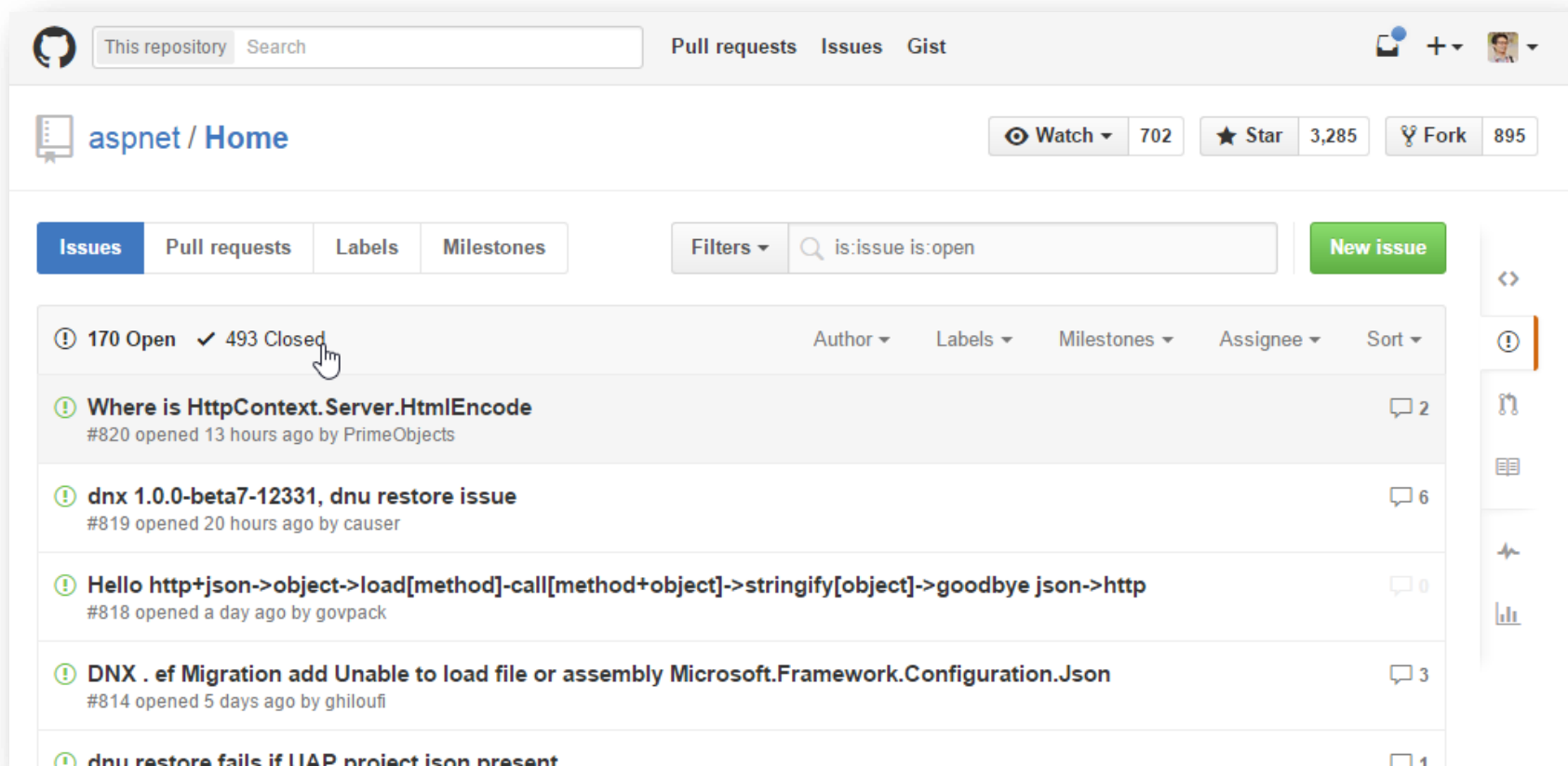
- 完整且彈性的協作機制
  - 原始碼版本控管
  - 完整的原始碼協作/審核機制
  - 議題追蹤與專案管理
  - Wiki 文件協作
  - 免費網頁空間
- 世界最多人用的開放平台，高知名度

# 原始碼控管 (Code)

- 可使用任何 Git 工具進行版控
  - 分支、標籤、發行、線上版本比對、線上合併分支
- 可套用任意 Git 版控流程 (架構十分彈性)
- GitHub 自創一套簡化版的 GitHub 版控流程
  - <https://guides.github.com/introduction/flow/>

# 專案管理 / 議題管理 (Issues)

- 微軟 ASP.NET 5 專案 - Issues



The screenshot shows the GitHub interface for the repository `aspnet / Home`. The `Issues` tab is selected, displaying a list of 170 open issues and 493 closed issues. The first issue is titled "Where is HttpContext.Server.HtmlEncode" and was opened 13 hours ago by `PrimeObjects`. The second issue is titled "dnx 1.0.0-beta7-12331, dnu restore issue" and was opened 20 hours ago by `causer`. The third issue is titled "Hello http+json->object->load[method]-call[method+object]->stringify[object]->goodbye json->http" and was opened a day ago by `govpack`. The fourth issue is titled "DNX . ef Migration add Unable to load file or assembly Microsoft.Framework.Configuration.Json" and was opened 5 days ago by `ghiloufi`. The fifth issue is titled "dnu restore fails if UAP project ison present".

Repository: `aspnet / Home`

Watch: 702, Star: 3,285, Fork: 895

Issues: 170 Open, 493 Closed

Filters: `is:issue is:open`

New issue

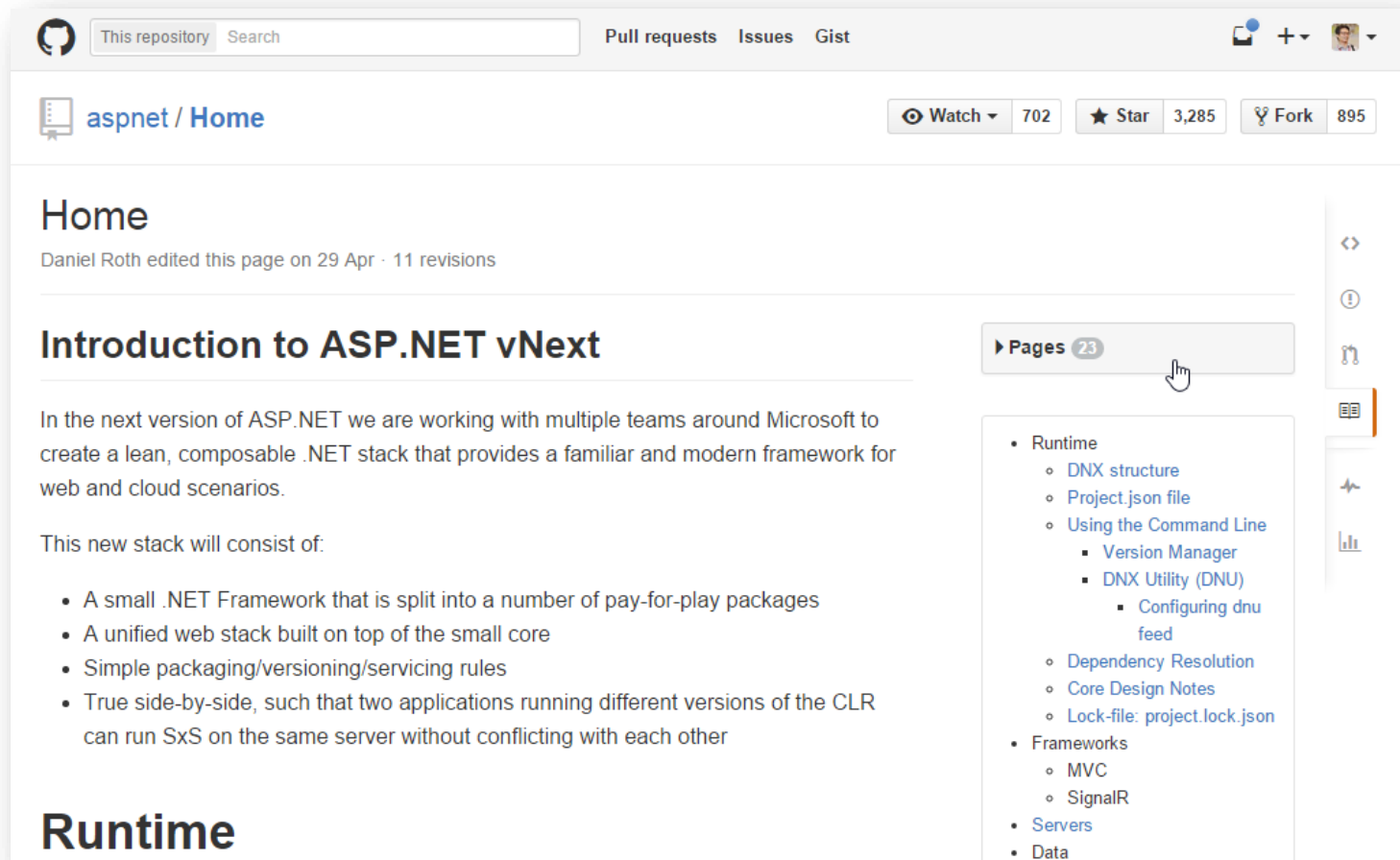
Issues list:

- Where is `HttpContext.Server.HtmlEncode`  
#820 opened 13 hours ago by `PrimeObjects`
- `dnx 1.0.0-beta7-12331, dnu restore issue`  
#819 opened 20 hours ago by `causer`
- Hello `http+json->object->load[method]-call[method+object]->stringify[object]->goodbye json->http`  
#818 opened a day ago by `govpack`
- `DNX . ef Migration add Unable to load file or assembly Microsoft.Framework.Configuration.Json`  
#814 opened 5 days ago by `ghiloufi`
- `dnu restore fails if UAP project ison present`



# 文件管理 (Wiki)

- 採用 Wiki 格式進行文件管理 (彈性十足)



The screenshot shows the GitHub repository for ASP.NET vNext. The main heading is "Home", with a subtext indicating it was edited by Daniel Roth on 29 Apr with 11 revisions. Below this is the title "Introduction to ASP.NET vNext". The text describes the next version of ASP.NET as a lean, composable .NET stack. A list of features is provided, including a small .NET Framework, a unified web stack, simple packaging rules, and side-by-side execution of different CLR versions. On the right side, there is a "Pages" sidebar with 23 items, including Runtime, Frameworks, Servers, and Data. The "Runtime" section is expanded, showing sub-items like DNX structure, Project.json file, Using the Command Line, Dependency Resolution, Core Design Notes, and Lock-file: project.lock.json.

This repository Search Pull requests Issues Gist

aspnet / Home Watch 702 Star 3,285 Fork 895

## Home

Daniel Roth edited this page on 29 Apr · 11 revisions

### Introduction to ASP.NET vNext

In the next version of ASP.NET we are working with multiple teams around Microsoft to create a lean, composable .NET stack that provides a familiar and modern framework for web and cloud scenarios.

This new stack will consist of:

- A small .NET Framework that is split into a number of pay-for-play packages
- A unified web stack built on top of the small core
- Simple packaging/versioning/servicing rules
- True side-by-side, such that two applications running different versions of the CLR can run SxS on the same server without conflicting with each other

## Runtime

Pages 23

- Runtime
  - DNX structure
  - Project.json file
  - Using the Command Line
    - Version Manager
    - DNX Utility (DNU)
      - Configuring dnu feed
  - Dependency Resolution
  - Core Design Notes
  - Lock-file: project.lock.json
- Frameworks
  - MVC
  - SignalR
- Servers
- Data

# 免費網頁 (GitHub Pages)

- 免費的網頁空間
- 免費的 *username.github.io* 域名
- 免費的自訂網址 (custom domain)
- 無限網頁流量、支援全球 CDN 與負載平衡

Basic Git

# 基礎 GIT 入門



# 安裝 Git 版控工具

- <https://www.git-scm.com/>
- Windows
  - [Git for Windows](#)
  - [TortoiseGit](#)
  - [GitHub Desktop](#)
- Mac
  - [Git for Mac](#)
  - [GitHub Desktop](#)
- Linux / Unix
  - [Git for Linux and Unix](#)

# 初始化設定 Git 工具

- 設定必要 Git 參數
  - `git config --global user.name "你的姓名"`
  - `git config --global user.email "你的Email"`

# 建立工作區與本地儲存庫

- 練習 1：建立工作區（順便建立儲存庫）
  - mkdir p1
  - cd p1
  - git init
  - echo test > test.txt
  - git status
  - git add .
  - git status
  - git commit -m "Initial commit"
  - git log
- 練習 2：複製遠端儲存庫（也會建立工作區）
  - <https://github.com/doggy8088/Learn-Git-in-30-days>
  - git clone https://github.com/github/gitignore.git
  - git log

# 建立 GitHub 遠端儲存庫

- 練習 3：從 GitHub 建立新的遠端儲存庫
  - <https://github.com/new>
  - 建立名稱為 **git-p3** 的專案，並授權給你隔壁的同學
- 練習 4：複製遠端儲存庫
  - **git clone** <GIT\_URL\_HTTPS\_or\_SSH>
  - **cd** git-p3
  - <建立 *n* 個版本>
  - **git push**
    - 輸入 GitHub 帳號、密碼 （透過 **Git Shell** 可不用輸入帳密）
  - 到 GitHub 查看 Git 版本資訊

# 發行 Git 本地儲存庫

- 練習 5：將本地儲存庫發佈到遠端儲存庫
  - `git remote add origin <GIT_URL_HTTPS_or_SSH>`
  - `git push -u origin master`

注意：本地與遠端都有版本的狀態，需透過 `git pull` 先合併版本！

- 練習 6：將本地儲存庫發佈到不同的遠端儲存庫
  - `git remote add bitbucket <GIT_URL_HTTPS_or_SSH>`
  - `git push -u bitbucket master`

注意：本地與遠端都有版本的狀態，需透過 `git pull` 先合併版本！



# 常用的 Git 版本控管指令

- 新增檔案
  - `git add .`
  - `git add <dir>/<filename>`
  - `git add -u .`
  - `git add --all`
- 刪除/更名檔案
  - `git rm 'Gruntfile.js'`
  - `git mv t1.txt t2.txt`
- 提交變更 / 建立新版本
  - `git commit`
  - `git commit -m "版本紀錄"`
  - `git commit --amend`
- 顯示工作目錄的索引狀態
  - `git status` (完整輸出)
  - `git status -s` (精簡輸出)
- 重置工作目錄的索引狀態
  - `git reset` (復原狀態、保留變更)
  - `git reset --hard` (復原狀態、復原變更)
  - `git checkout -- test.txt`
- 查詢歷史紀錄
  - `git log`
  - `git log -10 --abbrev-commit`
  - `git log --oneline`
  - `git whatchanged`
- 建立分支 / 切換工作目錄
  - `git branch <branch_name>`
  - `git checkout <branch_name>`
  - `git checkout -b <branch_name>`

# 常用的 Git 分支與合併指令

- 列出分支 ( 預設分支：**master** )
  - `git branch` (本地分支)
  - `git branch -r` (遠端分支)
  - `git branch -a` (所有分支)
- 建立分支
  - `git branch <branch_name> [<ref>]`
- 切換分支
  - `git checkout <branch_name>`
- 建立並切換分支
  - `git checkout -b <branch_name> [<ref>]`
- 刪除分支
  - `git branch -d <branch_name>`
  - `git branch -D <branch_name>`
- 更名分支
  - `git checkout master`
  - `git branch -m <old> <new>`
  - `git branch -M <old> <new>`
- 顯示目前工作目錄所在分支
  - `git status`
- 合併分支
  - `git checkout master`
  - `git merge <branch_name>`
- 合併衝突處理
  - 手動編輯衝突檔案
  - 使用 Git 工具解決衝突檔案
- 放棄合併
  - `git merge --abort`
  - `git reset --hard`

# Git 與 GitHub 議題追蹤 (Issues)

- 使用 GitHub Issues 管理工作
  - [Mastering Markdown](#)
- 介紹 GitHub Flow
  - [Understanding the GitHub Flow](#)
  - [GitHub Flow in the Browser](#)



# 使用 GitHub Pages

- <https://pages.github.com/>
- 設定步驟
  - 建立儲存庫: `username.github.io`
  - 複製儲存庫: `git clone`
  - 建立頁面檔: `index.html`
  - 推送到遠端 GitHub
    - `git add --all`
    - `git commit -m "Initial commit"`
    - `git push -u origin master`
  - 連結網站
    - `http://username.github.io`

# 相關連結

- [Git](#)
- [GitHub Guides](#)
- [GitHub Cheat Sheet](#)
- [GitHub Pages](#)
- [30 天精通 Git 版本控管](#)

# 聯絡資訊

- The Will Will Web

記載著 Will 在網路世界的學習心得與技術分享

- <http://blog.miniasp.com/>

- Will 保哥的技術交流中心 (臉書粉絲專頁)

- <http://www.facebook.com/will.fans>

- Will 保哥的噗浪

- <http://www.plurk.com/willh/invite>

- Will 保哥的推特

- [https://twitter.com/Will\\_Huang](https://twitter.com/Will_Huang)