國 立 虎 尾 科 技 大 學
National Formosa University

機 械 設 計 工 程 系 暨 精 密 機 械 工 程 科
Department of Mechanical Design Engineering & Department of Junior Precision Mechanical Engineering

專 題 製 作 報 告
Project Report

網際內容管理系統
Application of Web-based Content Management Systems
在精密機械工程教學與研究上的應用
in Teaching and Research of Precision Mechanical Engineering

指導教授： 嚴 家 銘 老 師
班 級： 五 精 四 甲
學 生： 郭 樺 (50733105)
高 沁 安 (50733144)
林 冠 澔 (50733146)
林 侑 昌 (50733152)

中華民國 110 年 6 月

國立虎尾科技大學
機械設計工程系暨精密機械工程科
學生專題製作合格認可證明

專題製作學生：五精四甲 50733105 郭　樺
五精四甲 50733144 高沁安
五精四甲 50733146 林冠澔
五精四甲 50733152 林侑昌

專題製作題目：網際內容管理系統在精密機械工程教學與研究上的應用

評 審 委 員：_____

_____

_____

_____

指 導 老 師：_____

系　　主　　任：_____

中 華 民 國　　年　　月　　日

第一章　前言

1.1 研究動機

我們是以教學與研究歷程的重要性作為研究動機，因為我們發現很多學術上的研究流程不夠詳細，導致後者無法完全得知這些專案的過程是如何進行，造成他們無法順利地接手，甚至重頭開始進行，所以我們希望不只提供學生在學習與研究流程能夠不只留下具體成果，也能有效呈現更細部的歷程與資訊，以作為學習與研究更有力的佐證資料。

1.2 製作目的

目的分為兩部分:

1. 探討如何利用 Fossil SCM 虛擬與實體伺服器，讓五專精密機械工程科所有相關師生包含已經畢業的校友，得以透過 @gmail 帳號登入，並在網際內容管理系統中進行知識管理與互動，擬藉此提升課程教學與專題研究效益。
2. 允許使用者透過學校配發的 @gm 登入後，有權限在伺服器上自行建立獨立的倉儲系統並且自行管理。

1.3 未來展望

此專題希望用戶能利用架設的分散式版次管理系統在各別的倉儲或社群進行社會化共同項目開發與資訊交流等，包括允許使用者追蹤其他使用者、組織、軟體庫的相關資訊，對開發項目進行評論且改動等。

第二章　分散式系統

要合作就必須溝通與協調，俗話說合作就是力量大，但是溝通與協調不僅僅是一種本能，更是一種能力。這就是分散式運算的問題來源，在分散式環境中，合作對象是分散各地的電腦，在這些電腦透過網路連接在一起，每台電腦都能讀立運行，且同時也能藉由分散式系統來進行合作，若以一個基層人員而言，需要將上級、同級、下級，甚至客戶，的每一件事項在對應的部分做串聯，甚至獨立解決，而產生一加一大於二的成效。

2.1 分散式系統的基本架構

分散式系統則是將硬體設備或軟體元件，分布在不同的網路主機上，藉著彼此之間的網路進行通訊與協調的系統。可以想像成一群獨立的電腦系統集中起來對外提供服務，但對於使用者而言，就像是在使用一台強大的主機，就是透過網路將許多台電腦資源連結起來。當有一個大型任務需要執行時，會先將任務分割成許多小型運算工作，再分派給所有的電腦執，最後再將所有執行的結果彙整。網格運算與雲端運算即是分散式系統常見的應用。而通常更複雜的分散式系統也是由以下三個系統架構 (圖.2.1) 組何而成。

- 中心化網路 (CENTRALIZED)
- 去中心化網路 (DECENTRALIZED)
- 分散式系網路 (DISTRIBUTED)

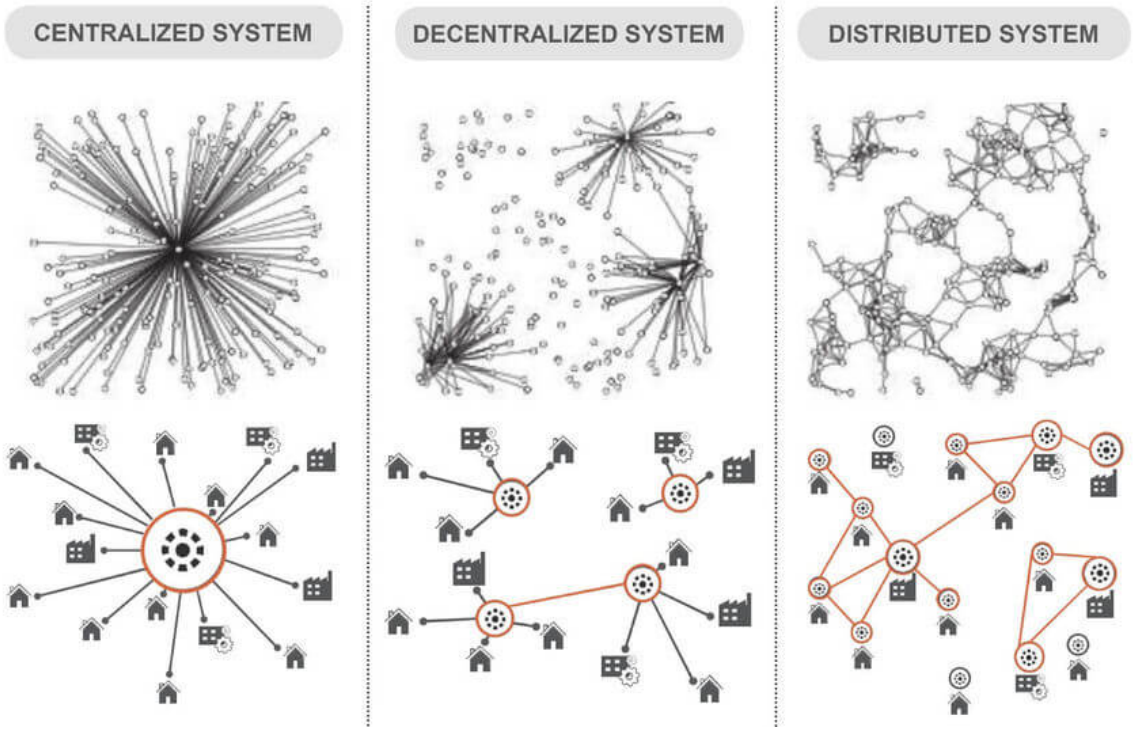

圖 2.1: 基本架構

2.2 版本控制

版本控制是一種紀錄資料變化或內容改變的完整歷程，並且每次改動將賦予一個新代碼，當版本控制系統開始使用時，是所有開發者合作的一種方式，共用一個儲存庫 (Repository) 的概念，儲存庫可以視為一種資料庫，開發者通常會利用版本控制來追蹤、維護原始碼、檔案以及設定檔等等的改動。而通常版本控制分為 (圖 2.2) 以下兩種形式。

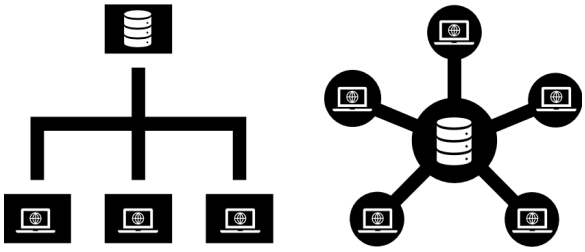

圖 2.2: 集中式版本控制 (左) vs. 分散式版本控制 (右)

而集中式版本控制與分散式版本控制系統的功能都包含了，同步、追溯、以及備份，相差不大，最主要的差別在於集中式的資料庫是集中控管，因此需要透過網路連結主機，開發者需要提交檔案或要對資料庫做一些其他的操作，都必須要在能夠連接上網路的環境下進行做操作，因此在網路環境或速度不佳的地區，會影像到開發效率。而分散式的資料庫允許不只一份，事實上，每個開發者都可以在自己的主機上建立儲存庫，因此版本代碼會記錄在儲存庫上，因此可以在無需網路環境下進行操作，需要進行遠端同步時，才需要網路連線，而開發者才能進行推送 (push) 的操作到其他儲存庫上，而其他開發者需要在網路連線下，才能進行拉取 (pull) 的操作，在各自進行開發的工作。

2.3 事件的排序問題

分散式系統在事件的排序上是一個很重要的部分。在單一主機系統下，我們可以由系統時鐘來判斷兩個事件發生的前後順序，例如處理元需要使用某個資源之前，需要擁有使用權，所以使用資源的事件一定得發生在擁有資源使用權的事件後。在分散式系統中沒有時鐘，所以在理論上必須對基本的觀念與邏輯有一定的能力，才不製造資料的衝突。

當本地儲存庫推送遠端時，因為遠端儲存庫資料以經被其他協同者更新，所以本地儲存庫推送時，會造成衝突，而無法成功推送如 (圖 2.3)。

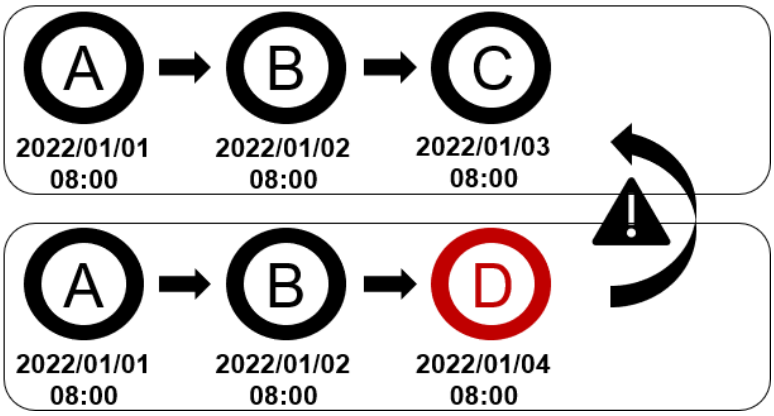

圖 2.3: 遠端儲存庫 (上)、本地端儲存庫 (下)

這時候，需要遠端儲存庫進行合併，若直接覆蓋，則會造成 C 直接消失，然後在進行推送的操作，而遠端就進行同步的動作如 (圖 2.3)。

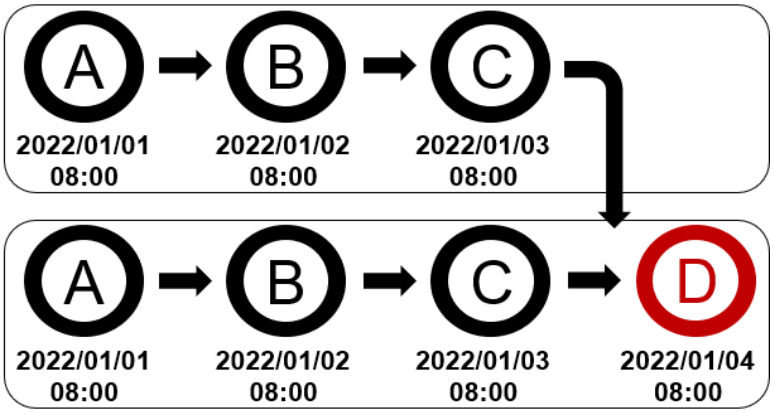

圖 2.4: 遠端儲存庫 (上)、本地端儲存庫 (下)

執行合併會自動合併修改或更新資料，但也有不能自動合併的時候。若遠端儲存庫和本地儲存庫的同一個地方都發生修改的情況下，這時，因為不能自動判斷要導入哪一個修改內容，於是就發生錯誤。而發生衝突的部分，需要以手動或其他方式修改內容，在執行合併的操作，就可以推送資料，而遠端就能進行同步動作。

第三章　伺服器架構

此專題採用 Windows 10 版本作為我們的架設所使用作業系統，Windows 作業系統在作業系統中的市佔率高達七成五以上，且 Windows 10 作業系統在 Windows 作業系統中的使用率高達八成以上。

以下為 Windows 作業系統與 Linux 作業系統的比較。

比較	Windows	Linux
介面	介面統一，所有 Windows 程式選單幾乎一致。	圖形介面風格依發行版不同而不同，可能互不相容。
使用	使用比較簡單，容易入門。圖形化介面對沒有電腦背景知識的使用者使用十分有利	命令列介面，需要學習才能掌握，但也有圖形介面使用上，也較令另列易上手。
學習	系統構造複雜、變化頻繁，且知識、技能淘汰快，深入學習困難。	系統構造簡單、穩定，且知識、技能傳承性好，深入學習相對容易。
軟體	商業軟體為主。有很多軟體只能在 windows 裏運行，無法相容 Linux。	自由軟體為主。Linux 相容的軟體有需多處於開發中，選擇性相對 Windows 較少。
使用者	七成五以上。	不到一成。

3.1 Windows 環境配置

在 Window 系統安裝完成後，將所需要的軟體安裝到系統上，之後將安裝的軟體在系統上設定環境變數。

環境變數是一個動態命名的值，可以影響電腦上行程的行為方式，使用者通過設定環境變數，來更好的執行程序。

在作業系統搜尋如 (圖 3.1)

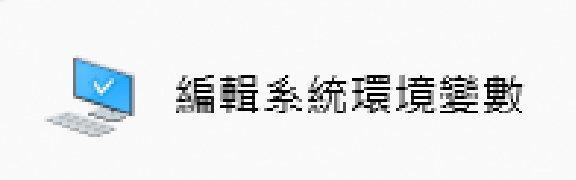

圖 3.1: 關鍵字

選擇環境變數如 (圖 3.2)

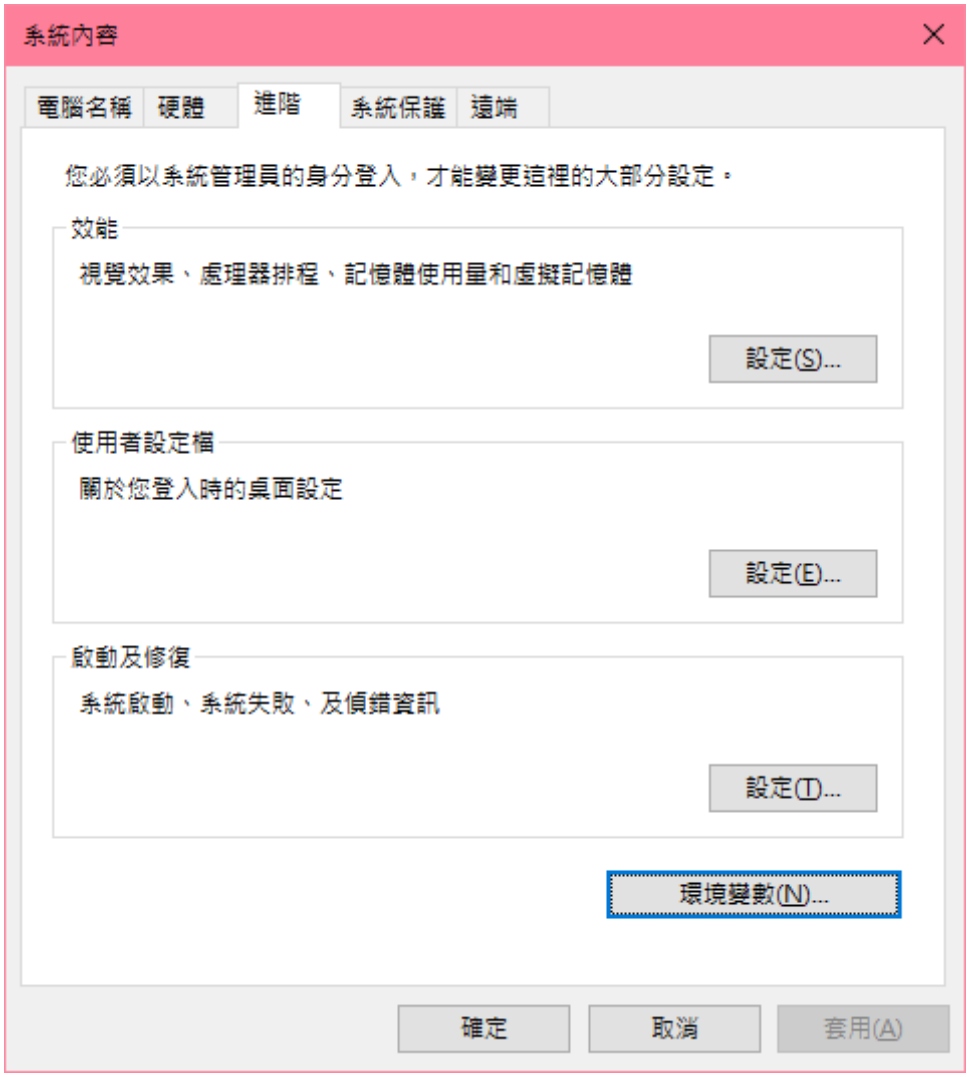

圖 3.2: 環境變數

選擇系統變數中的 Path 如 (圖 3.3)

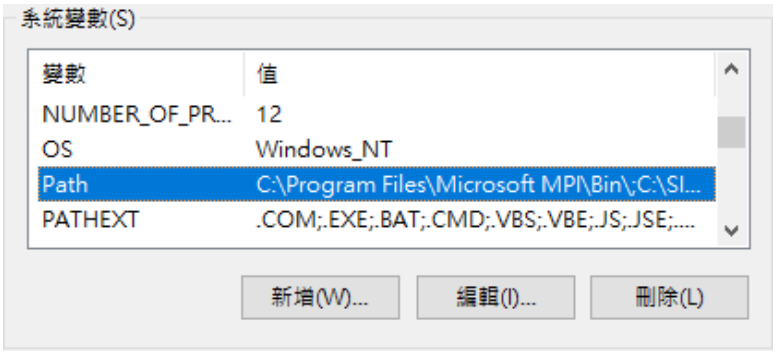

圖 3.3: Path

新增變數並且指定路徑如 (圖 3.4)

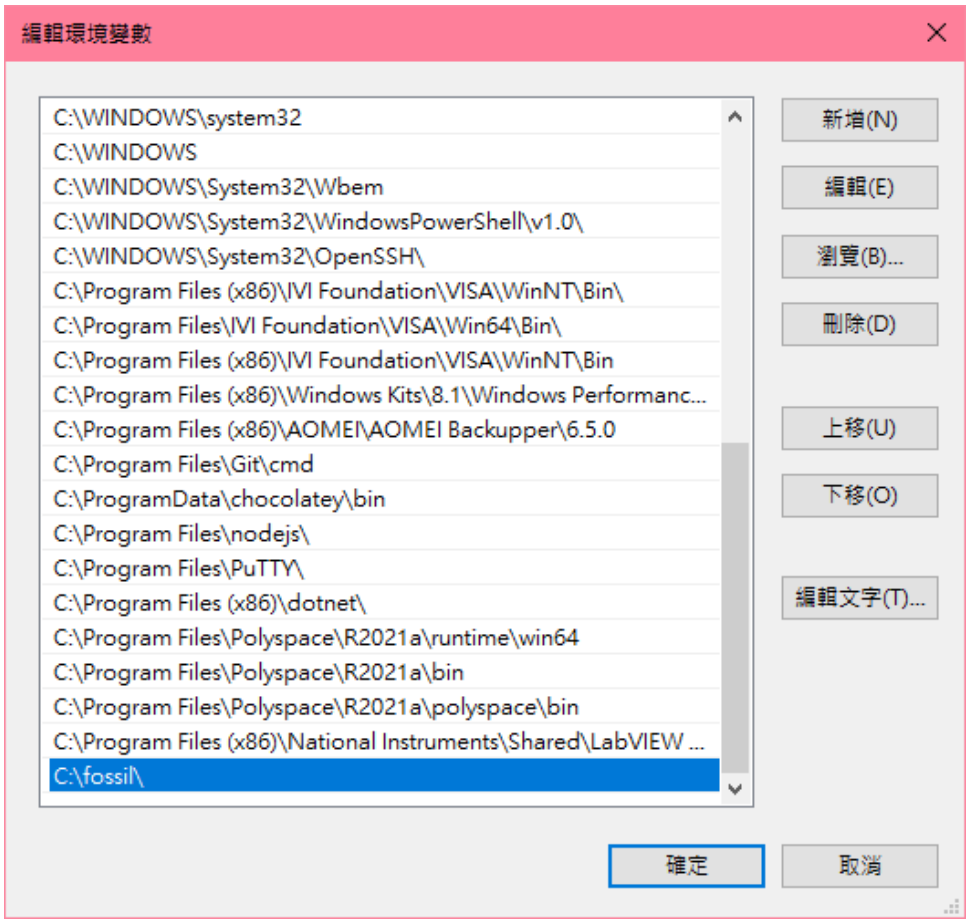

圖 3.4: 編輯環境變數

3.2 Fossil SCM

對於研究或專案計畫而言，工作中經常使用的具是電腦或其他相關電子產品，而對於研究過程或開發的流程、錯誤及解果的每個步驟都相刀的重要，而對負責軟體開發工作的軟體團隊成員來說，版本控制系統是一套相當重要的軟體工具。如果沒有版本控制系統，開發團隊成員將難以有效控制研究或開發過程，並可能導致效率的增加。

目前已經存在許多成熟的版本控制系統，例如較為知名的 Git、Subversion 或 CVS 等等。若以架設方式加以細分，則有分散式與 Client-Server 二種不同的系統分類。除了這些系統以外，網路上還可以找到許多其他各具特色的版本控制系統。雖然這些系統的知名度較低，但如果仔細檢視其優點與特色，仍然可以找到一些頗為出色的版本控制系統。

因此本專題所使用的是 Fossil，是一套採用分散式處理方式的版本控制系統。

圖 3.5: Fossil

因此本專題所使用的是 Fossil，是一套採用分散式處理方式的版本控制系統。

3.2.1 特色

一般人對於軟體本身的使用需求，多半是希望越容易操作越好，並且有相當程度的穩定性與可靠性。而操作簡單與系統本身穩定性高，正是 Fossil 所強調的二大重點。

在穩定性方面，Fossil 採用 SQLite 資料庫作為儲存的平台，再加上永久性的檔案結構，即使遭遇斷電或是元件 (不包括硬碟) 損毀等問題，資料也不會損失。

在可靠性方面，Fossil 目前也正在研發新的保護機制，在未來的版本中將會提供自動檢查機制，只要資料提交，系統便會自動進行資料內容驗證，以確保資料內容無誤，以提升可靠性。

Fossil 之所以可以作為官方網站的平台，是因為除了版本控制系統相關的功能以外，亦可利用 Fossil 作為 Blog 平台的架設方案。所以無論使用者需要的是單純的版本控制，或是希望架設網站作為資訊分享的平台，都能利用 Fossil 一併解決。所以在取得 Fossil 之後，不只獲得 Fossil 本身的原始碼，還取得了一整個網站系統的架設方案。

3.2.2 功能

1. 項目管理**:** 除了像 Git 和 Mercurial 那樣做分佈式版本控制之外，Fossil 還支持錯誤追蹤、wiki、論壇、電子郵件警報、聊天和技術說明。
2. **Web** 介面**:** Fossil 具有內置、主題化、可擴展和直觀的 Web 界面，其中包含豐富多樣的信息頁面。
3. 一體式**:** Fossil 是一個獨立的、獨立的可執行文件。要安裝，只需下載適用於 Linux、Mac 或 Windows 的預編譯二進製文件，並將其放在 PATH 上。還提供易於編譯的源代碼。
4. 自託管**:** 使用各種技術在幾分鐘內建立一個項目網站。Fossil 具有 CPU 和內存效率。大多數項目可以舒適地託管在每月 5 美元的 VPS 或 Raspberry Pi 上。還可以設置自動 GitHub 鏡像。
5. 簡單網路系統**:** Fossil 使用普通的 HTTPS 或 SSH 進行網絡通信，因此它可以在防火牆和代理後面正常工作。該協議帶寬效率很高，以至於 Fossil 可以通過撥號、弱 3G 或客機 Wifi 輕鬆使用。

6. 自動同步**:** Fossil 支持自動同步模式，通過減少與分佈式項目相 關的不必要的分叉和合併數量，有助於保持項目向前發展。

7. 開源**:** BSD 授權條款。

3.2.3 Fossil 操作簡介

Fossil 與其他版本控制系統相同，主要使用的都是儲存庫 (Repository) 的概念。儲存庫可以視為一種資料庫，其中存放了專案相關的檔案與資料等等。使用者處理這些檔案時，需要先將資料取出 (Check Out) 並存放至本地端中，也就是使用者的工作目錄。等到完成了新增、修改等各種工作，再將本地端的資料回存 (Check In) 至儲存庫之中，即可完成整個處理作業。也就是說，在使用 Fossil 時，主要的動作有三項：新增或複製儲藏庫、取出資料、進行資料處理與存取等工作。

專乁案開始執行時，需要新增一個儲藏庫，此時可以使用下列指令，建立一個新的儲存庫。儲存庫名稱未限制，因此可以使用任何命名方式，也可以不用使用任何副檔名。

fossil init *repository-filename*

儲存庫是以檔案方式存在於系統之中，所以執行此指令之後，便可在檔案系統中找到以儲存庫名稱命名的檔案。大多數 Fossil 的操作都針對本地的儲存庫進行處理，如果需要存取遠端系統中的儲存庫，以下列指令將遠端儲存庫複製到本地中，再進行後續處理。

fossil clone *https://your-id-num@your-domain-name/ your-db.fossil*

建立或複製完儲存庫之後，此時可以先建立一個新目錄，切換到此目錄並且以下列指令展開取得 _ FOSSIL_ 檔案。

fossil open *repository-filename*

此時可以將所需提交的檔案放置此目錄，並且以下列指令將指定的檔案加入到儲存庫。

fossil add *file...*

若需要儲存庫的資料刪除，以下列指令將指定的檔案刪除於儲存庫。

fossil rm *file...*

如果覺得依序加入或移除檔案的方式有些麻煩，則可以以下列指令，讓儲存庫與本地目錄中的檔案清單自動的做對比，如果有檔案存在於本地目錄，但並未在儲存庫之中出現，則會自動將此檔案加入儲存庫之中。相反的，如果檔案只出現在儲存庫，但本地目錄中找不到該檔案，則會自動將此檔案自儲存庫之中移除。

fossil addremove

當新增或刪除的部分完成時，需要以下列指令做最後的提交，所新增或刪除的檔案才會正式生效。

fossil commit

以上僅是 Fossil 最基本的操作方式，並未完整涵蓋 Fossil 所有的指令與其參數。以下為 Fossil 經常使用的指令。

add	cat	diff	ls	remote	tag
addremove	changes	extras	merge	revert	timeline
all	chat	finfo	mv	rm	ui
amend	clean	gdiff	open	settings	undo
annotate	clone	grep	patch	sql	unversioned
bisect	commit	help	pull	stash	update
blame	dbstat	info	push	status	version
branch	delete	init	rebuild	sync	xdiff

3.3 Stunnel

Stunnel 是一種代理，旨在將 TLS 加密功能添加到現有客戶端和服務器，而無需對程序代碼進行任何更改。它的架構針對安全性、可移植性和可擴展性（包括負載平衡）進行了優化，使其適用於大型部署。

圖 3.6: Stunnel

3.3.1 功能

- 可移植性（線程模型）
- 性能和可擴展性
- 支持 OpenSSL 安全功能
- 其他跨平台功能
- Unix 特性
- Windows 功能

3.3.2 為何要使用 Stunnel?

資料在網路傳輸下，如 (圖 3.7) 若沒有經過任何保護等機制，當網路遭受駭客或其他網路攻擊時，容易造成資料被竊取，因此為了使網路在傳輸下是安全的，因此使用 Stunnel。

Stunnel 是一個可以用 SSL 對任意 TCP 連接加密的程式，並可工作在 Unix 和 Windows 平臺上。它採用 Client/Server 模式，將 Client 端的網路資料採用 SSL 加密後，安全的傳輸到指定的 Server 端再進行解密還原，然後再發送到訪問的伺服器。

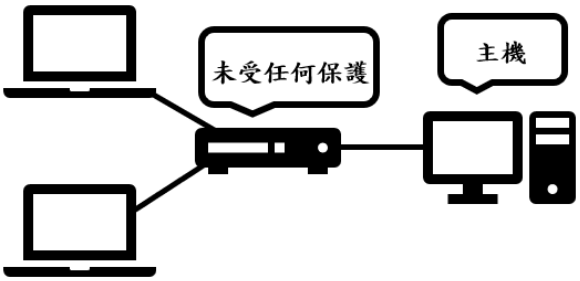

圖 3.7: 普通網路

在加密傳輸過程中，可充分確保資料的安全性，我們只要把 Server 端程式安裝在局域網外的一台伺服器上，即可保證傳輸的資料在局域網內是安全的，如 (圖 3.8)。

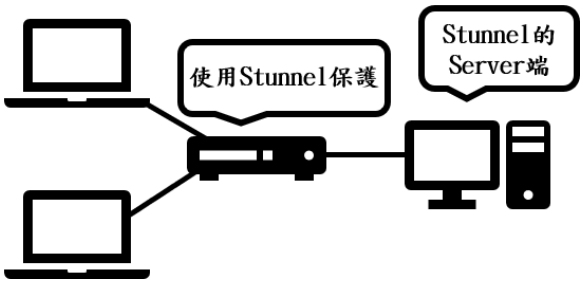

圖 3.8: 加密網路

3.3.3 設定操作

因為要將 Fossil 採用 HTTPS 內容與 Stunnel 互動。因此與 Stunnel HTTPS 結合，此外需要將 stunnel.conf 設定如下。

```
[https]
accept = pj5073.cycu.org:443
connect = 9000
cert = fullchain.pem
key = privkey.pem
TIMEOUTclose = 0
```

原本使用的 cert 與 key 是使用 localhost.crt 與 localhost.key 為自簽章憑證，自簽章憑證一樣可以用來加密，但是由於該憑證不屬於公開金鑰基礎建設簽發憑證，其他沒信任該憑證的 HTTPS 存取者的網頁瀏覽器會顯示一個警告，說這個憑證不被他們的電腦或瀏覽器信任。因此必須接受數位簽章的 public key，因此 cert 與 key 目前已經設定為 public key，此部分由後續的 3.6 Let's Encrypt 章節所探討。

3.3.4 負載平衡

用來在多個電腦、網路連接、CPU、磁碟驅動器或其他資源中分配負載，以達到最佳化資源使用、最大化吞吐率、最小化回應時間、同時避免過載的目的。使用帶有負載平衡的多個伺服器組件，取代單一的組件，可以通過冗餘提高可靠性。

利用 Stunnel 將本地服務器的 IP 地址與端口 port 9000、5000、5001、9001 的資訊，分別轉給外部的客戶端連接端口 443、8443、9443、5443 的 HTTPS 協定連線，設定如下。

```
[https]
accept = pj5073.cycu.org:443
connect = 9000
cert = fullchain.pem
key = privkey.pem
TIMEOUTclose = 0
https>https

accept = pj5073.cycu.org:8443
connect = 5000
cert = fullchain.pem
key = privkey.pem
TIMEOUTclose = 0
https>https

accept = pj5073.cycu.org:9443
connect = 5001
cert = fullchain.pem
key = privkey.pem
TIMEOUTclose = 0
https>https

accept = pj5073.cycu.org:5443
connect = 9001
cert = fullchain.pem
key = privkey.pem
TIMEOUTclose = 0
```

3.4 Nginx

3.5 NSSM

3.6 Let’s Encrypt