版本控制系統概論

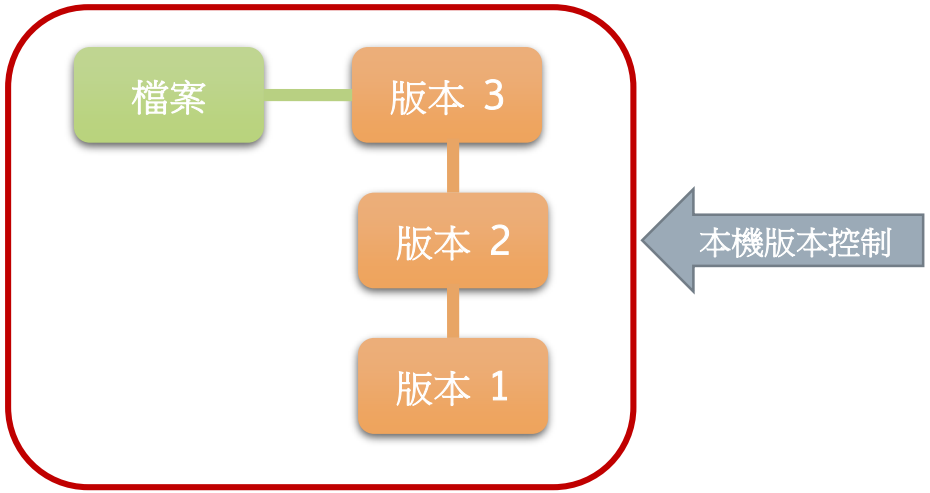
- 版本控制系統
  - Version Control System；VCS
  - 為程式碼管理軟體的通稱
  - 是一套系統
  - 該系統按時間順序記錄某一個或一系列檔案的變更
  - 可以檢視其以前特定的版本

- 版本控制系統
  - 功用：
    - 保存程式檔的修改紀錄
    - 保存程式檔的歷史版本
    - 記錄檔案的內容變化
    - 查詢每個版本所更動的內容

版本控制系統概論

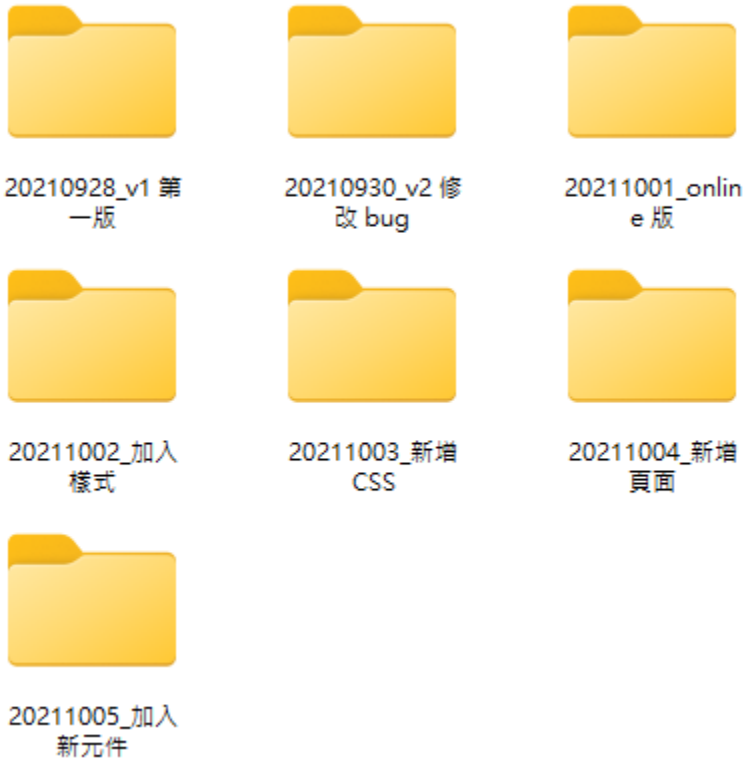
- 版本控制系統(VCS)
  - 本機版本控制系統：
    - 傳統版本控制：
      - 檔案複製到資料夾
      - 資料夾命名(特定名稱、時間)
      - 容易錯誤
        - 路徑
        - 覆蓋
        - 名稱類似

- 版本控制系統
  - 本機版本控制系統：
    - 本機版本控制：
      - 使用簡單的資料庫儲存檔案的所有變更



版本控制系統概論

- 版本控制系統
  - 本機版本控制系統

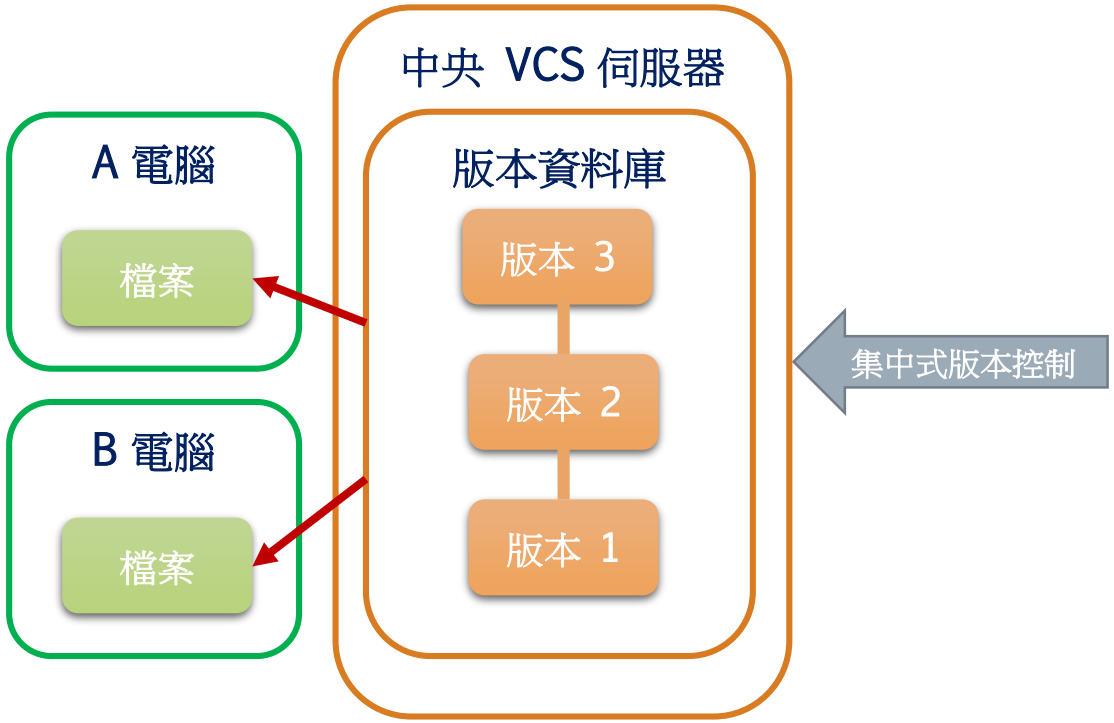


- 版本控制系統
  - 本機版本控制系統：
    - 缺點：
      - 效率低
      - 以資料夾做版本控制，不知道裡面有哪些檔案更動過。
      - 集中式系統
        - 必須有一台伺服器
        - 需要有網路

版本控制系統概論

- 版本控制系統
  - 集中式版本控制系統：
    - Centralized Version Control System；CVCS
    - 集中控管
    - 檔案置於單一伺服器，多用戶可取出檔案，修改某一程式檔案時，必須先將其鎖定，再取出修改，在完成修改與回傳之前，任何人都不能更動此程式檔案。

- 版本控制系統
  - 集中式版本控制系統：



版本控制系統概論

- 版本控制系統
  - 集中式版本控制系統：
    - 優點：
      - 有效避免衝突發生(不同人同時修改同一程式碼，造成混淆情況)
      - 所有人可以在某種程度上掌握他人在專案中的進度
      - 管理員可以精準地控制每個人的許可權
      - 維護集中式版本控制系統比本機版本控制系統要簡單

- 版本控制系統
  - 集中式版本控制系統：
    - 缺點：
      - 伺服器當機
        - 任何人都無法協作、修改
      - 中央資料庫受損(如硬碟損壞)
        - 檔案將流失
      - 以上缺點也可能對應到本機版本控制系統

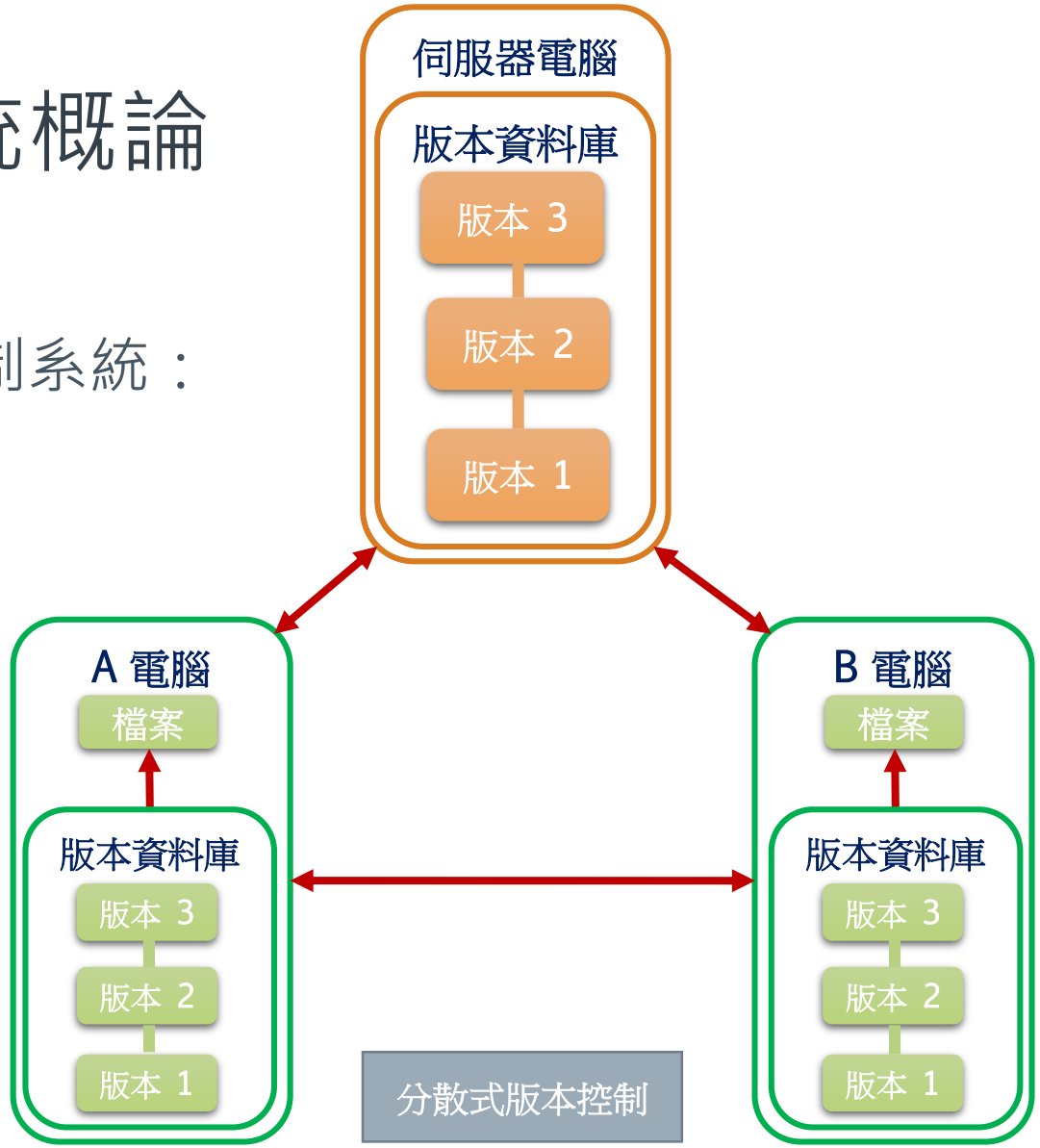
版本控制系統概論

- 版本控制系統
  - 分散式版本控制系統：
    - Distributed Version Control System；DVCS
    - 為因應集中式版本控制系統的缺點而生
    - 任何人都可以隨時取的任一檔案來編輯
    - 軟體多半可在本機單獨操作

- 版本控制系統
  - 分散式版本控制系統：
    - 通常有共同伺服器(需要網路)
    - 若沒有網路環境也可以自行控制，待有網路時再行同步即可。
    - 對整個程式庫(repository)進行完整的映像檔
    - 即使伺服器故障，個別本機也可使用自己的映像檔操作

版本控制系統概論

- 版本控制系統
  - 分散式版本控制系統：



版本控制系統概論

- 版本控制系統
  - VCS 軟體：Git
    - 在 Git 中每一個傳統管理的資料夾都視為一個版本，並記錄了每個版本修改了那些檔案。
    - 可看到設計者修改那些程式碼
    - 顯示該修改版本是由哪位設計者更新
    - 以輸入指令的方式操作 VCS
    - 可在網路、單機操作
    - 免費、開源、易學
    - 分散式系統
      - 沒有網路連線依舊可以操作

- 版本控制系統

操作 Git

- Git
  - 功能：
    - 安裝在電腦中的版本管理工具
    - 將檔案存入檔案庫(儲存庫)
    - 儲存在檔案庫中的檔案即為檔案的『歷史備份』
    - 檔案需要使用時再從檔案庫中取出使用
    - 每次版本有變化，會更新並記錄整個目錄與檔案的樹狀結構。

- Git
  - 使用：
    - 命令列(Command Line)：
      - 執行 Git 指令
    - 圖形介面工具
      - Microsoft：
        - Visual Studio
        - Visual Studio Code
          - VS Code
      - Git GUI 操作介面：
        - GitHub Desktop
        - Sourcetree

建立本機檔案庫

- 檔案庫(Repository)
  - 功用：
    - 存放檔案的地方(資料夾)
    - 檔案存放於檔案庫，Git 即可追蹤、控管。
    - Git 可以管理電腦中任何一個資料夾中的檔案、子資料夾。

- 檔案庫(Repository)
  - 設定：
    - 資料夾中執行管理指令：
      - git init
    - 執行後會建立一個檔案庫
    - 被建立的『檔案庫』是名稱為 .git 的資料夾
      - 負責程式的版本控制
    - 執行後即可讓 Git 完成管理前的準備工作，由 Git 接手管理資料夾。
    - 檔案庫中儲存被管理的檔案與資料夾、曾經被加入的歷史版本

建立本機檔案庫

- Git 追蹤檔案的方式
  - Git 將檔案與資料夾分為：
    - 被追蹤的(tracked)
    - 忽略的(ignored)
    - 未追蹤的(untracked)
  - 指令：git status
    - 列出 untracked 檔案清單，可提醒那些檔案還沒有被歸類。
    - 初始時，所有檔案都是 untracked

- Git 追蹤檔案的方式
  - untracked：
    - Git 偵測到有該檔案，但還不是 Git 追蹤的對象。
    - 必須先將該檔案加入到 staging area(索引)，即可將該檔案加入到追蹤對象。

建立本機檔案庫

- Git 追蹤檔案的方式
  - 正常情況下，資料夾中所有的檔案都應該被分類為『tracked』或是『ignored』。
    - tracked：
      - 檔案已經被加入 Git 檔案庫
    - ignored：
      - 要求 Git 不要檢查檔案
- 檔案加入 Git 追蹤
  - 加入索引指令：
    - git add 檔案名稱1 檔案名稱2...
    - git add .
      - 檔案若有更改，一定要 add
    - 加入索引的檔案即將會被提交成一個新版本(commit)
    - 將檔案加入暫存區
    - add 後可使用『git status』指令查看狀態
      - Git 即偵測到該資料夾中有新增的檔案

建立本機檔案庫

- 提交檔案新版本
  - 提交新版本指令：
    - git commit -m “新版本資訊”
      - 執行後才算完成整個檔案版本操作流程
  - 修改最後一次 commit 的訊息：
    - git commit --amend –m “修改的資訊”
  - 修改 commit 的歷史紀錄：
    - 重置 commit (reset)
    - 更改 commit 歷史紀錄(rebase)
- 查詢版本
  - 查詢指令：
    - git log
- Git 操作基本流程
  - 工作目錄初始化
    - init
  - 加入索引(暫存區)
    - add
  - 提交(檔案庫)
    - commit