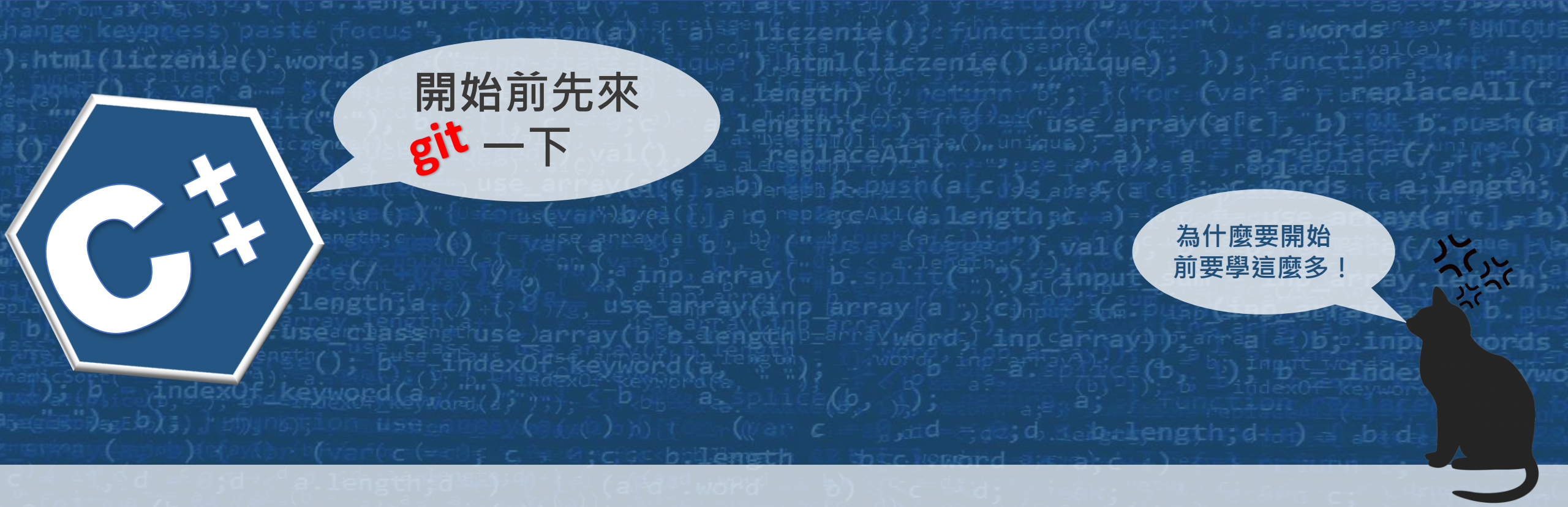


推廣教育資料結構與演算法
程式課前補充說明

Kuan-Teng Liao (廖冠登)

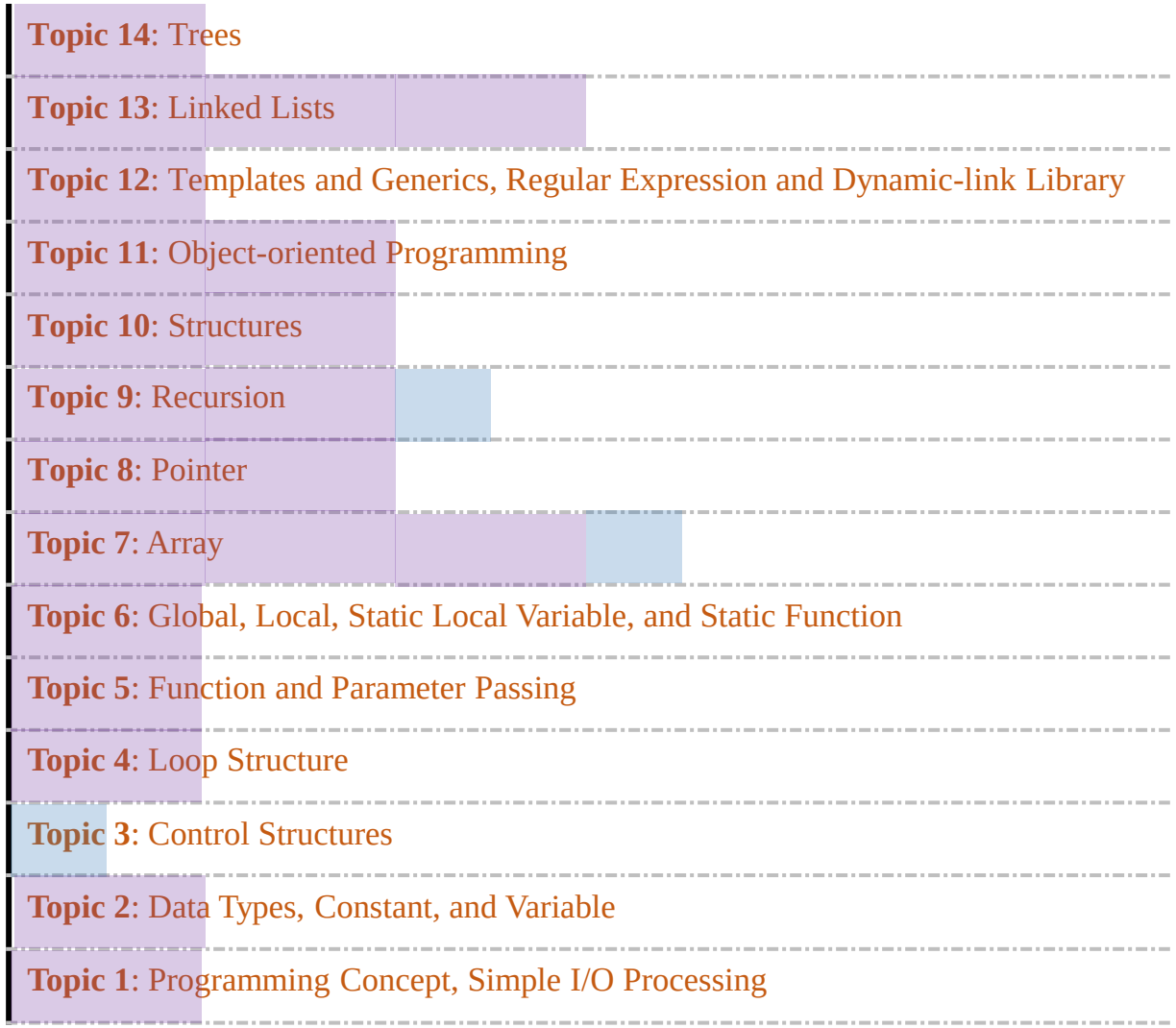
2021/02/27

大綱

- Etutor註冊與使用與Discord伺服器登入
- Git簡介、優缺點及運作原理介紹
- 流程說明
- Git指令
- 替代方案

課程大綱

C++程式培養學習堆疊與教學時間



著色部分為**教學時間**，單位(**week**)，Topic 1長度的紫色區塊為一周；藍色為半周

課程進度

- 資料結構與演算法
  - 基礎操作
    - Topic 1 前言：陣列、指標與結構回顧
    - Topic 2 前言：物件導向設計與繼承回顧
    - Topic 3 堆疊與佇列
    - Topic 4 排序(1)—插入、合併及快速排序
    - Topic 5 排序(2)—基數、計數及桶排序
    - Topic 6 串列
    - Topic 7 樹(1)—二元樹、二元搜尋樹
    - Topic 8 樹(2)—紅黑樹、R樹
    - Topic 9階段性整合複習及實作(堆疊、佇列、排序、串列與樹)
  - 進階操作
    - Topic 10 堆積(1)
    - Topic 11 堆積(2)
    - Topic 12 動態規劃(1)
    - Topic 13 動態規劃(2)
    - Topic 14 常見決策機制(1)
    - Topic 15 常見決策機制(2)
    - Topic 16 圖算法(1)
    - Topic 17 圖算法(2)
    - Topic 18 階段性整合複習及實作(堆積、動態規劃、決策機制與圖算法)

E-tutor註冊與使用(1)

- 課程網址與未來程式測驗上傳網址
  - ✓https://e-tutor.itsa.org.tw/e-Tutor/

E-tutor註冊與使用(2)

- 註冊

請不要用Facebook註冊，
如果你不想老師登記分數，
找不到你的名字的話。

不要懷疑，都必填

E-tutor註冊與使用(3)

- 註冊完後，需認證，請打開Email信箱，完成認證

E-tutor註冊與使用(4)

- 認證後，可到下面下載每個禮拜的投影片(上傳程式日後也在該塊)

E-tutor註冊與使用(5)

- 進入登入頁面，輸入登入帳密

E-tutor註冊與使用(6)

- 進入後，會進入到個人的平台選擇

成功登入後右上方會有自己名字

點選學習平台選項

E-tutor註冊與使用(7)

- 選擇頁籤上的老師專區→台北市→台北科技大學→劉傳銘老師

E-tutor註冊與使用(8)

- 選擇頁籤上的老師專區→台北市→台北科技大學→劉傳銘老師
  - ✓Password: chaintohininder

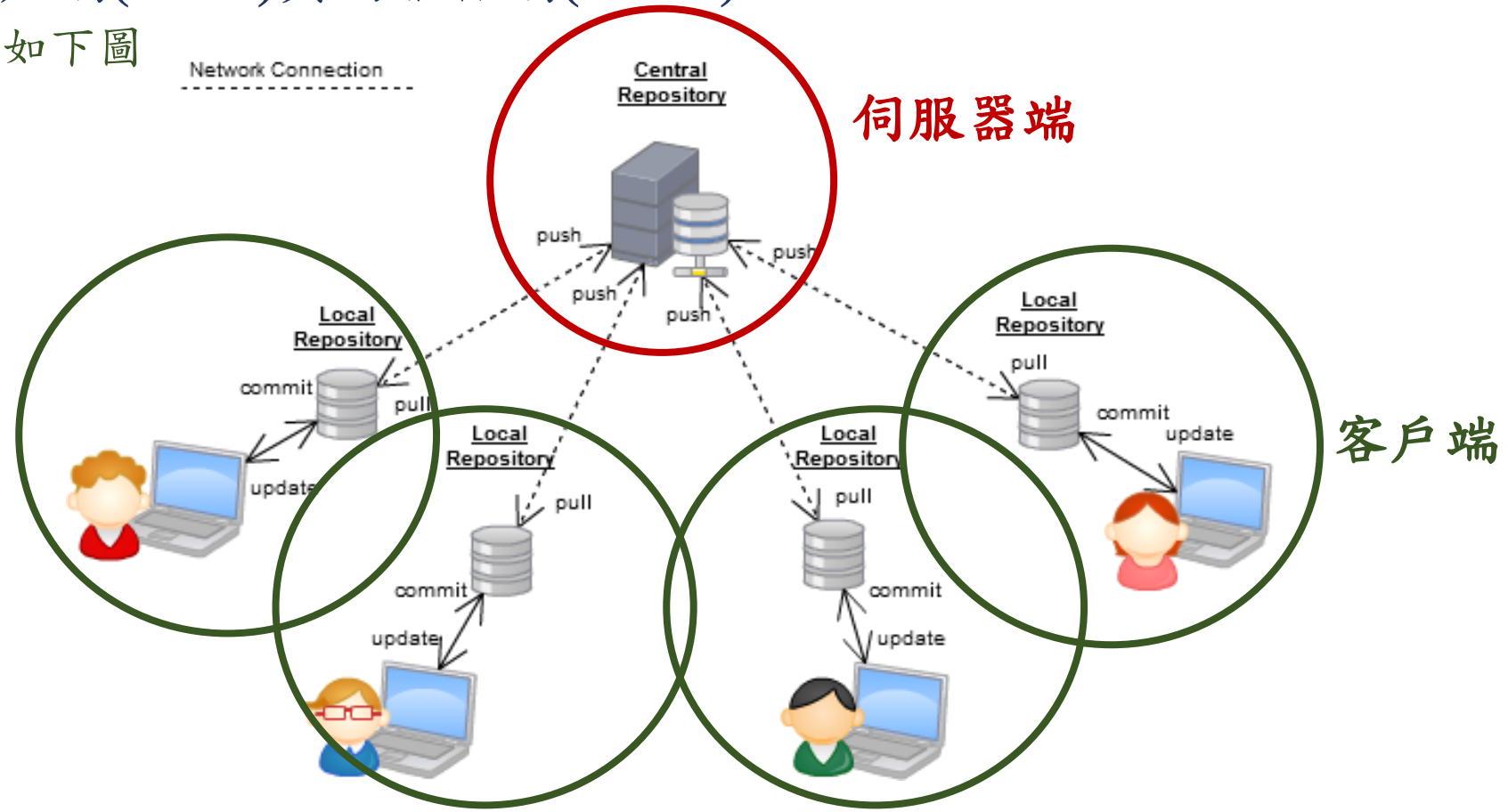
E-tutor註冊與使用(9)

- 每周投影片於頁籤活動的線上資源

Discord

Git簡介

- Git 是一種「版本控制系統」
  - ✓更精準來說是一種「分散式版本的版本控制系統」
  - ✓分為客戶端(client)與伺服器端(server)
    - e.g., 如下圖

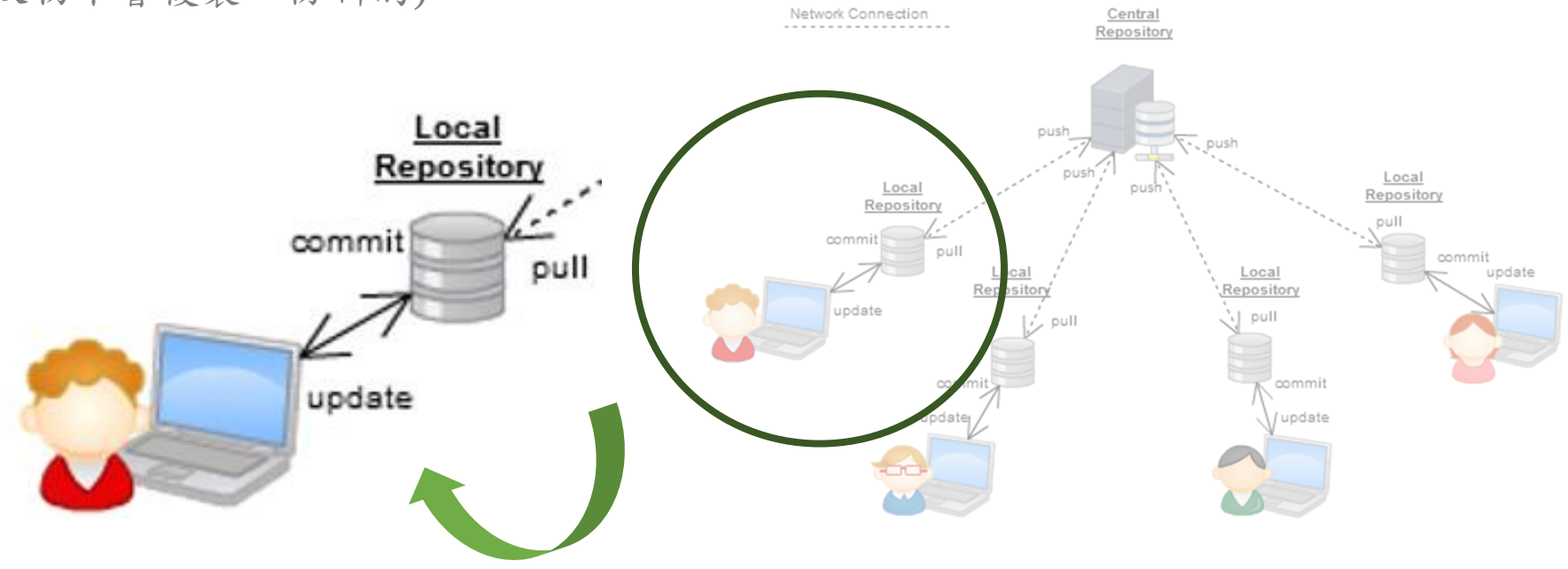


使用Git優/缺點

- 優點
  - ✓免費、開源
  - ✓速度快、檔案體積小
  - ✓**分散式系統**
    - 主要針對同一專案，多人開發下的狀況
      - 雖整體架構為主從式，但每個客戶端都有自己的倉庫(repository)，因此上端伺服器掛了，各自客戶端依舊可以對自己進行紀錄，直至伺服器好後，各自客戶端可以將其上傳，若發生衝突則需進行合併。
- 缺點
  - ✓雖好上手，卻難**精通**
    - Git 的指令有非常多，而且有的指令有點複雜

Git簡介(1)

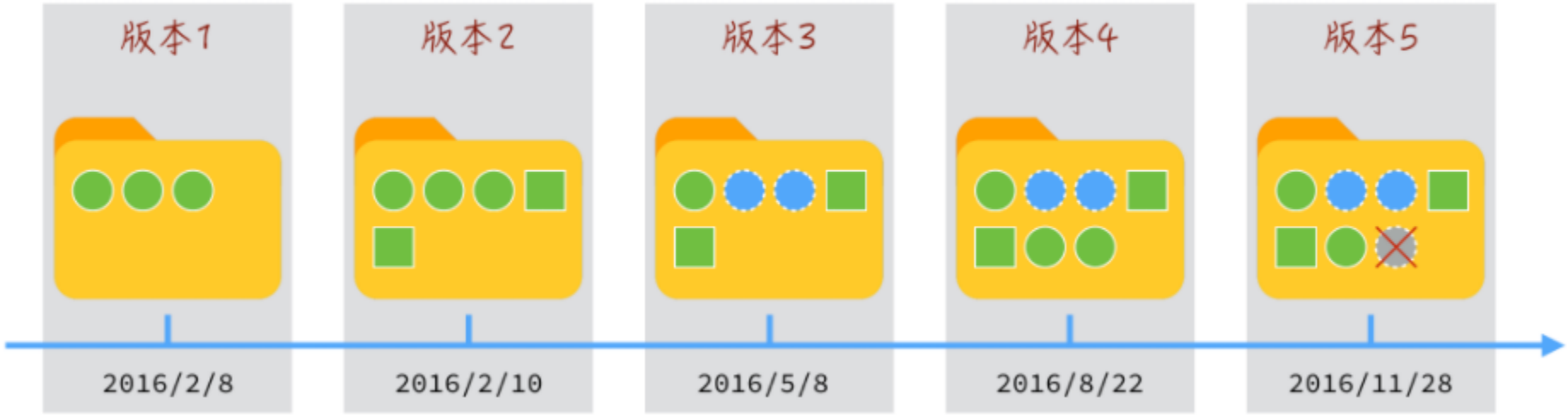
- 客戶端內
  - ✓包含**需控管的資料夾與倉庫**(repository)
    - 需控管的資料夾
      - 本機端使用者資料處理處
    - 倉庫
      - 為記錄本機端所有控管資料夾中各時期版本處(只記錄**差異點**，每個版本中同樣物或修改物不會複製一份新的)



Git簡介(2)

✓可將各類型檔案進行控管

- E.g., 如p3中，某位大大(即某綠色圈中)於每日檔案變化為下圖所示
  - 綠色為新增，藍色代表對內容進行修改，灰色叉叉代表刪除；形狀為各種不同類型的檔案
    - 注意：同一天也可能有多個版本，為了好說明這裡每天只有一個版本
- 為何會知道檔案新增/修改/刪除狀況？
  - 主要會比對在電腦中的控管處(通常為資料夾)下，與客戶端的倉庫(repository)間的差異。

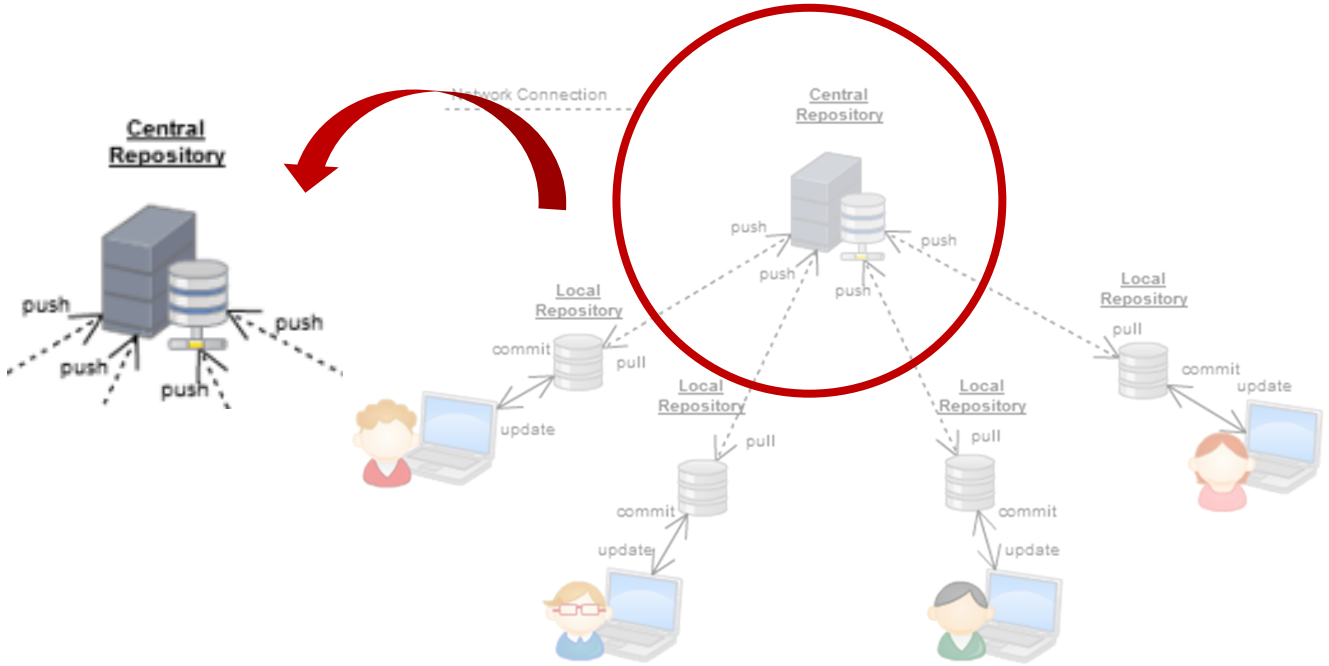


該圖來源取自：高見龍網站
https://gitbook.tw/chapters/introduction/what-is-git.html

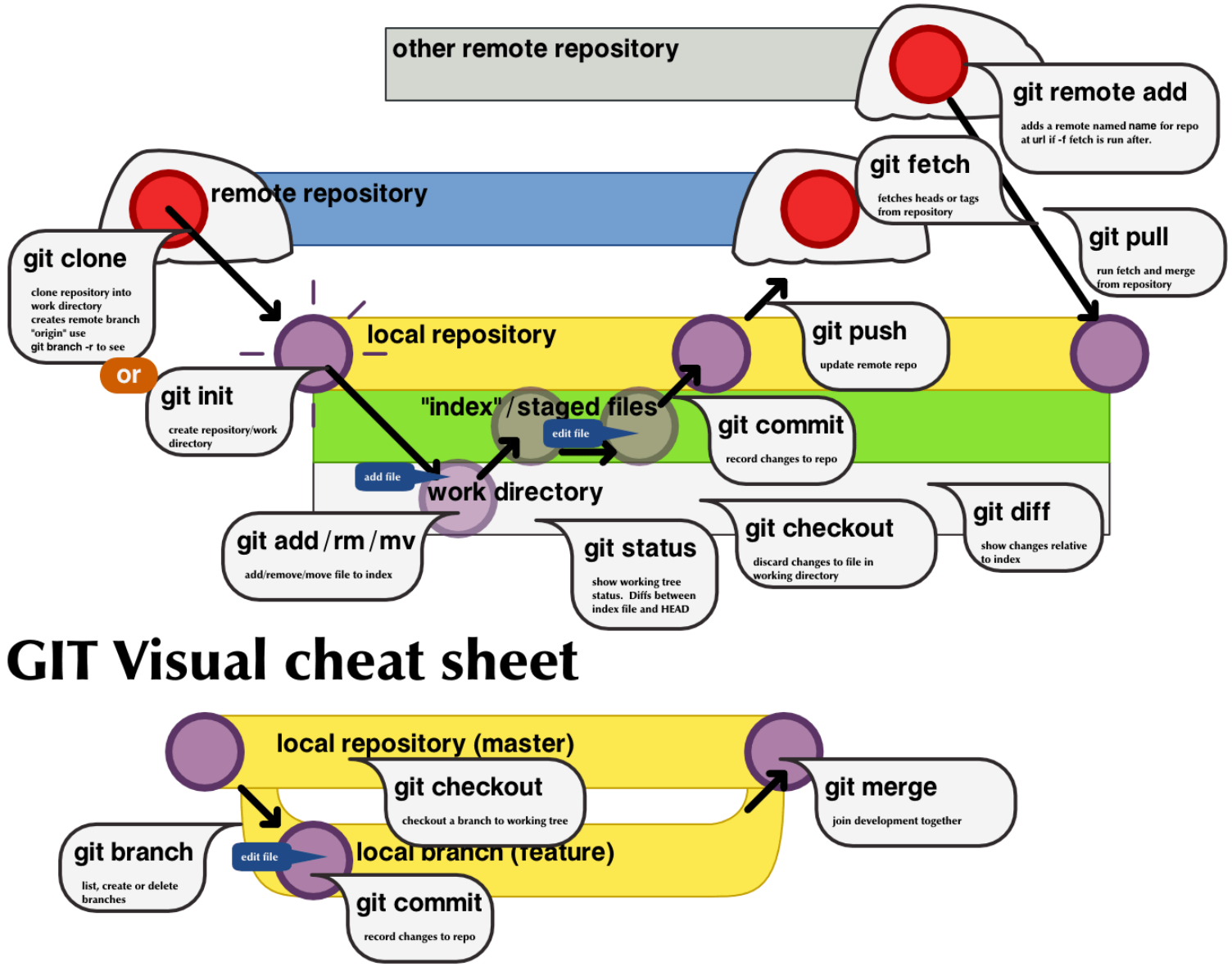
Git簡介(3)

✓伺服器端

- 為統合所有客戶端之資料，只有一個儲存空間，為集中式管理，裡面包含許多不同人的倉庫。
- 一般來說可以自行架設，但更多時候可以有更好的取代方案，如：github等
  - 在本次課程中，將採用github做為git 伺服器端以減少自行架設時間。

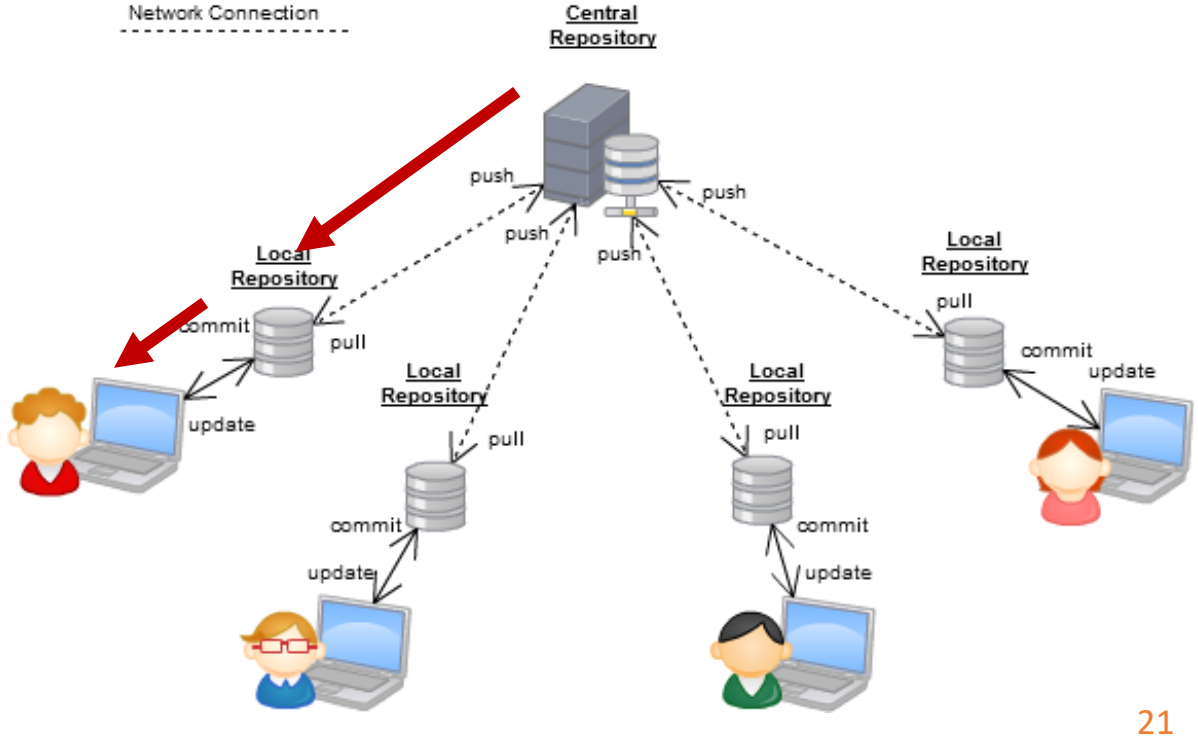


Git指令備忘圖


other remote repository
git remote add
adds a remote named name for repo at url if -f fetch is run after.
git fetch
fetches heads or tags from repository
git pull
run fetch and merge from repository
remote repository
git clone
clone repository into work directory creates remote branch "origin" use git branch -r to see
or
git init
create repository/work directory
local repository
git push
update remote repo
"index" / staged files
edit file
git commit
record changes to repo
add file
work directory
git add / rm / mv
add/remove/move file to index
git status
show working tree status. Diffs between index file and HEAD
git checkout
discard changes to file in working directory
git diff
show changes relative to index
GIT Visual cheat sheet
local repository (master)
git checkout
checkout a branch to working tree
git merge
join development together
git branch
list, create or delete branches
edit file
local branch (feature)
git commit
record changes to repo

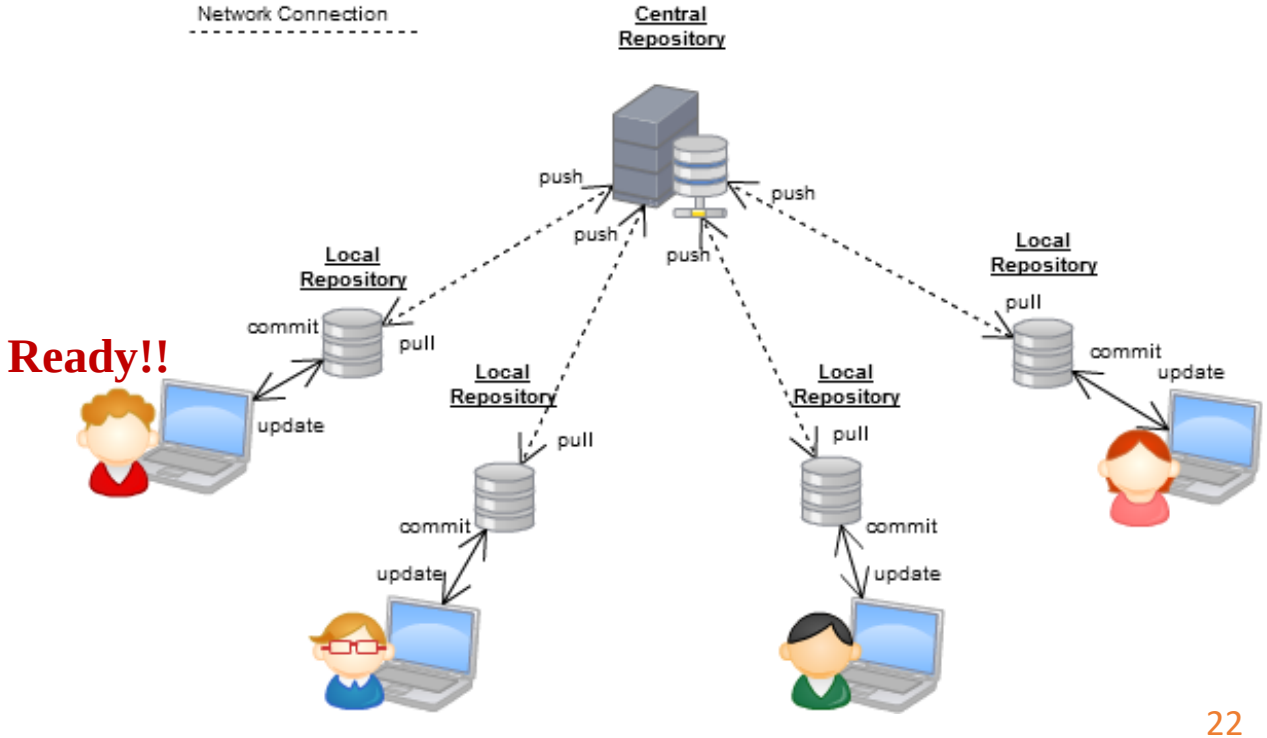
Git常用指令(1)

- clone (克隆)
  - ✓將伺服器端的倉庫(repository)**複製**一份至**客戶端的倉庫**與電腦內的專案集(project set)集中，其複製至專案集內會產生一份與伺服器端的倉庫一樣檔名的資料夾
  - ✓通常用在第一次，「本機端無伺服器倉庫的資料夾」時使用



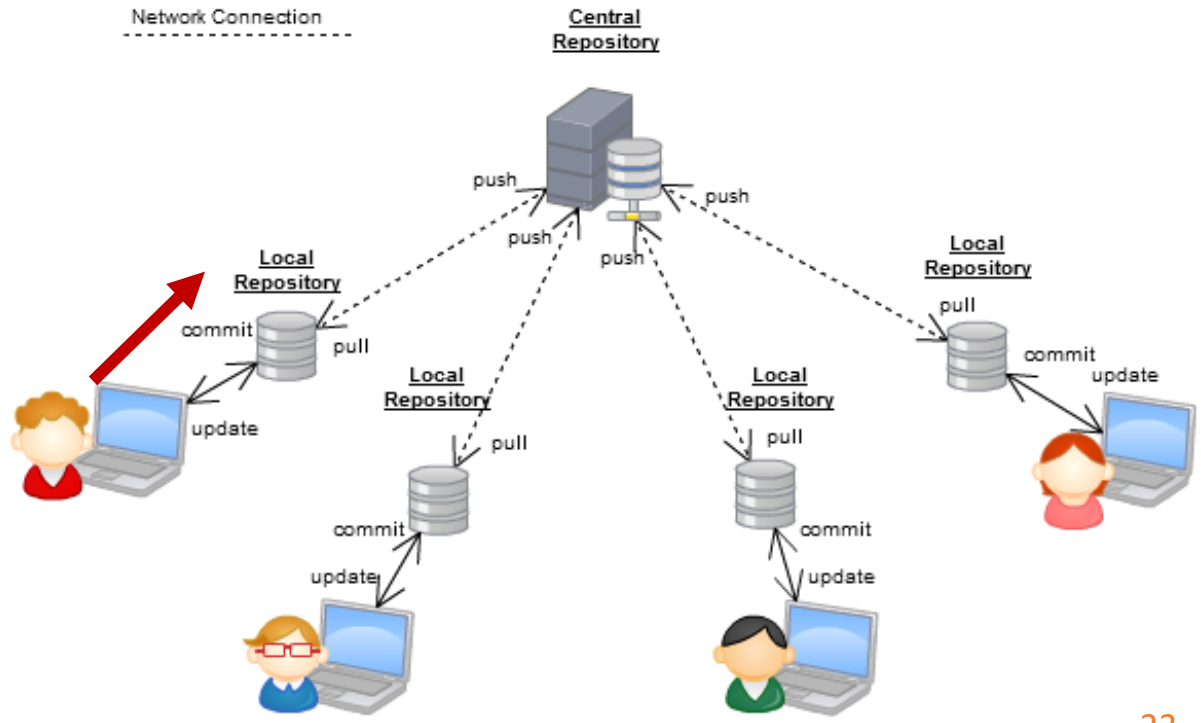
Git常用指令(2)

- add (加入)
  - ✓將變動(新增、修改、刪除)於客戶端準備好，以便之後將通過加入指令的變動進行commit指令，使其將變動提交至客戶端的倉庫中。



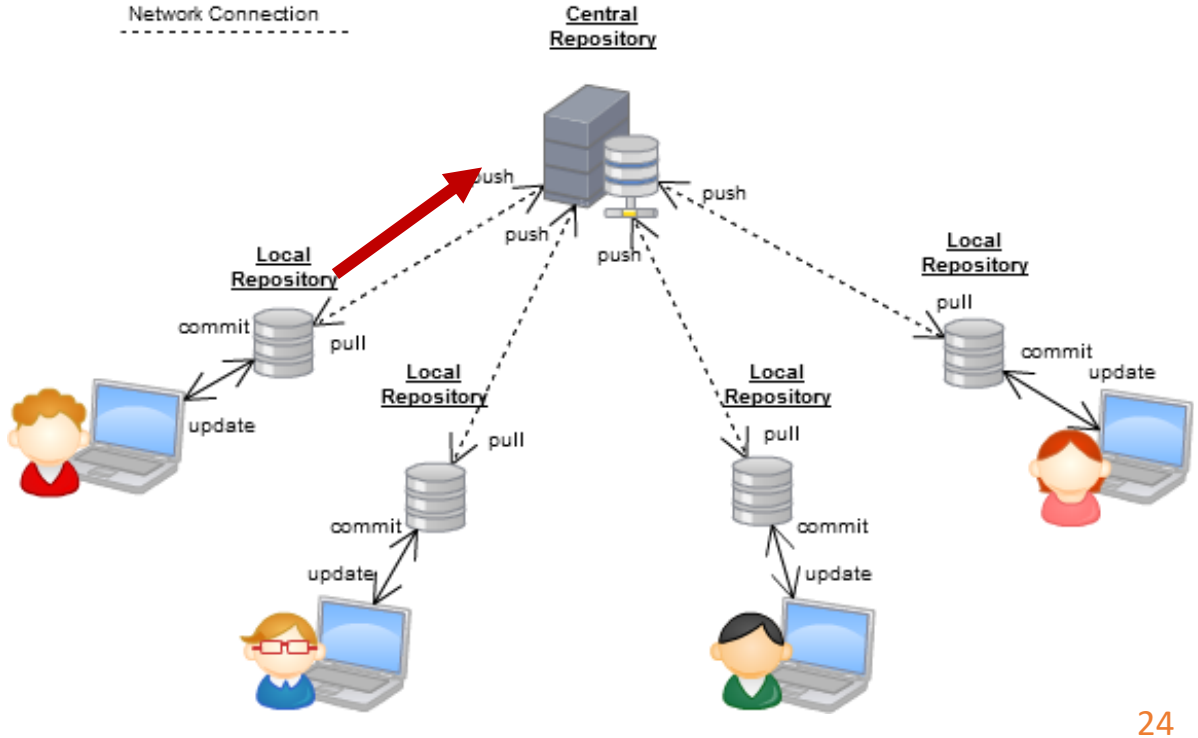
Git常用指令(3)

- commit (提交)
  - ✓將客戶端需控管的資料夾中的任何變動(新增、修改、刪除)更動至使用者端的倉庫內
  - ✓注意commit會發生在同一分支(branch)上



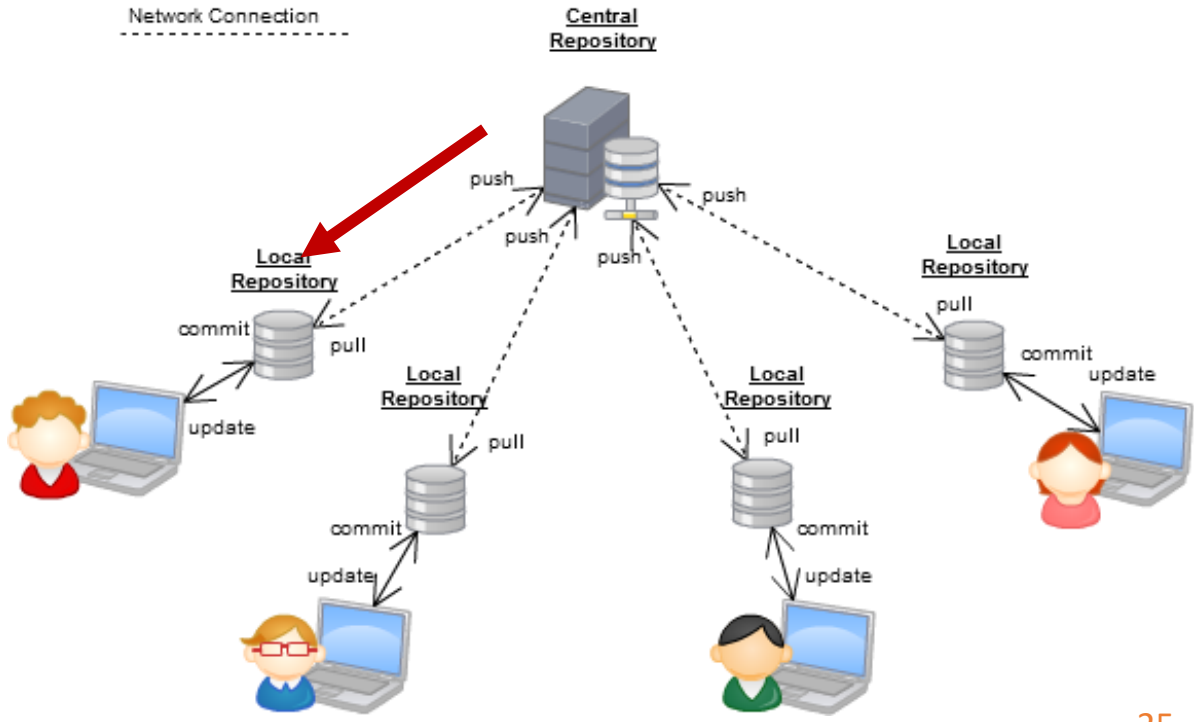
Git常用指令(4)

- push (推送)
  - ✓在所在的分支(branch)變動從客戶端的倉庫推向伺服器端的倉庫。



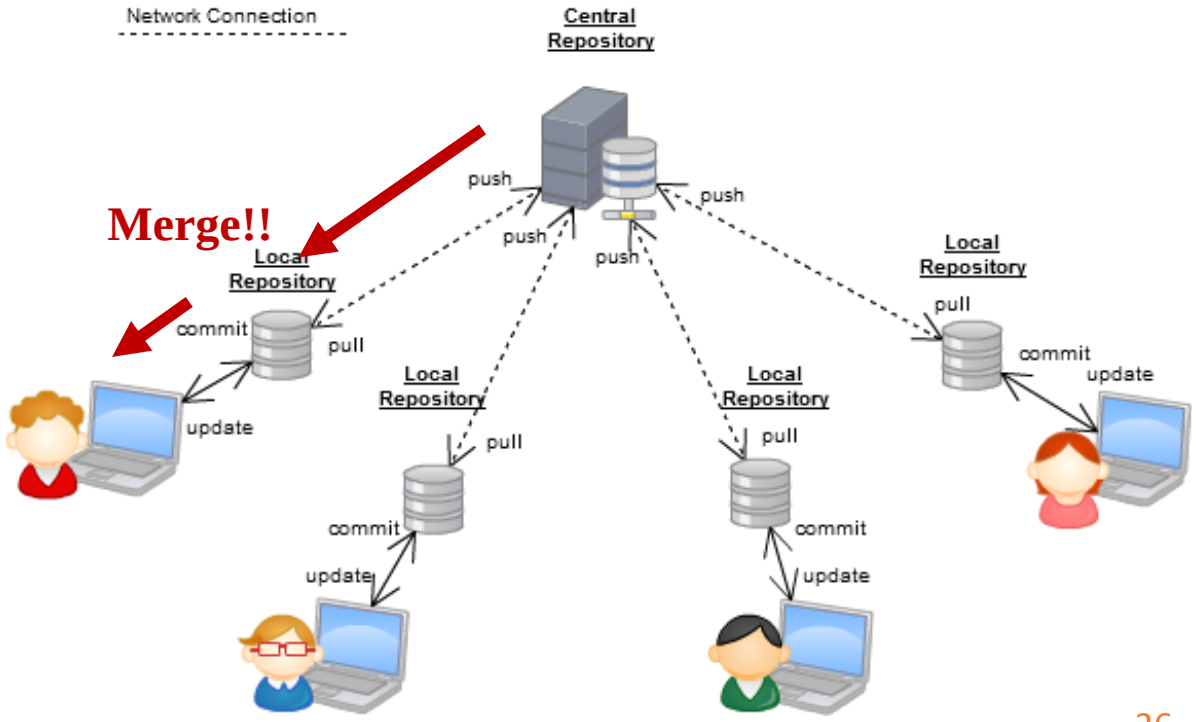
Git常用指令(5)

- fetch (獲取)
  - ✓使用時機為當客戶端已經有與伺服器端倉庫相同名稱之資料夾
  - ✓需由伺服器端獲取**全部的倉庫分支**(branch)資料至**客戶端的**倉庫時，所使用的指令。



Git常用指令(6)

- pull(拉取)
  - ✓使用時機為當客戶端已經有與伺服器端倉庫相同名稱之資料夾
  - ✓需由**伺服器端拉取最新的倉庫分支**(branch)資料與**客戶端的倉庫中最新的倉庫分支**進行對應分支**合併**(merge)，所使用的指令。
  - ✓事實上實際功能為
    - fetch+merge

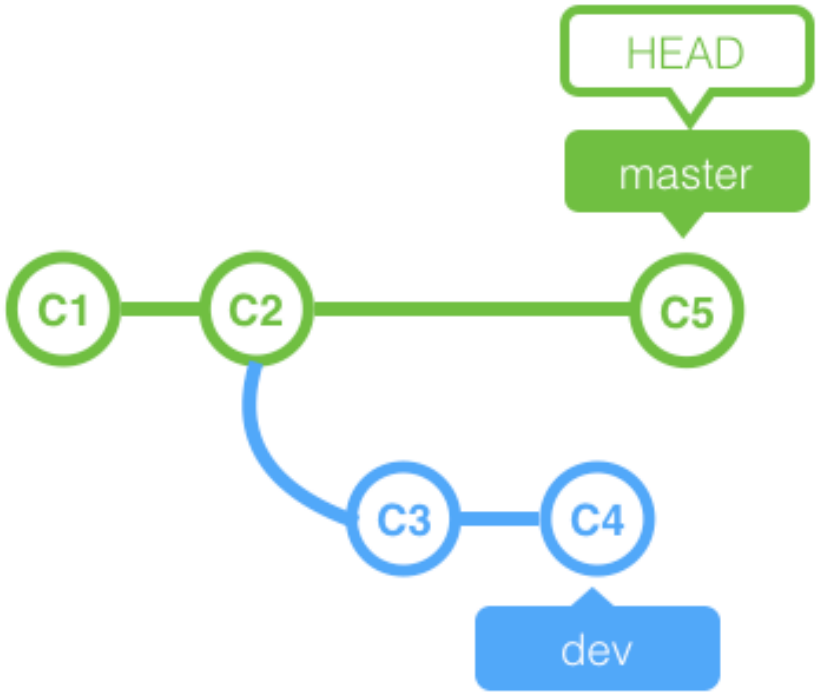


Git常用指令(7)

- 在上述的指令中，一直提到「分支」(branch)一詞，這裡需先說明
  - ✓當第一次執行完clone後，伺服器端會於倉庫內產生兩條分支
    - 一條名為**origin/master**，一條為origin/Head
  - ✓客戶端的倉庫中會產生一條分支
    - 名稱為master
  - ✓客戶端倉庫內的**master**若對其進行**push**時，會影響伺服器端內倉庫的**origin/master**
- 在客戶端，使用者可以於倉庫內對任一分支(e.g., master)上的任意時間點或是無分支的時間點進行新增分支，並給予名字，其作用在於
  - ✓於該條新名字的分支將成為倉庫內另一種發展的可能
    - E.g., p15

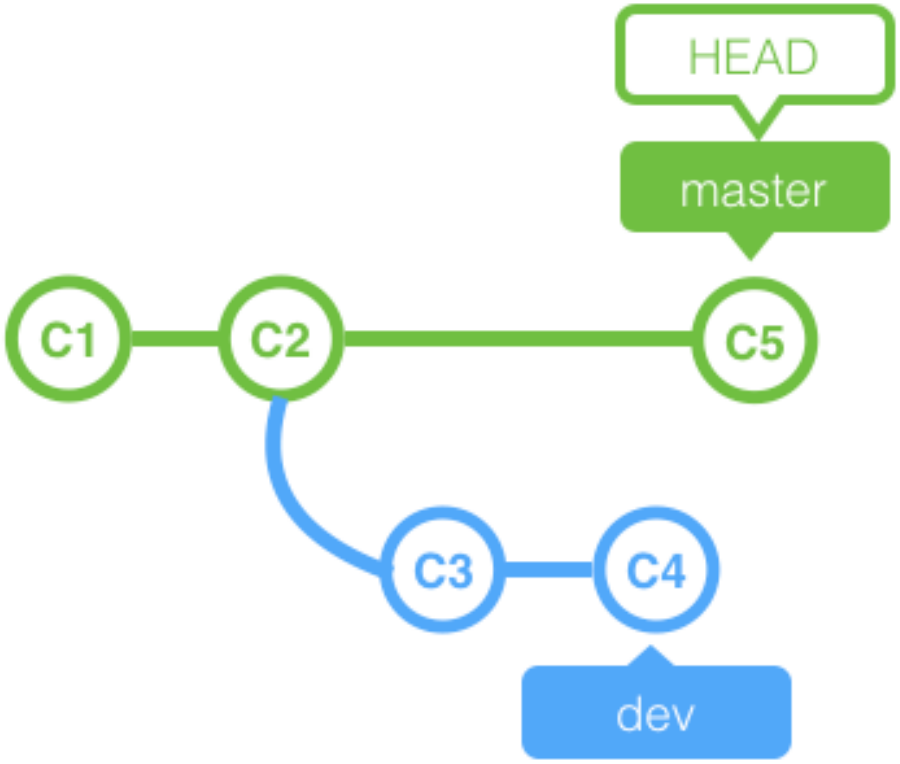
Git常用指令(8)

- 若時間軸為從左邊至右邊行進master分支於綠色線，可看出使用者於C2時開出分支名為dev，這意味在C3時間內檔案的變動可能是一種對C2的延伸，最後更新至C4
- 同時於master線上，C2又可進行撰寫延續master至下一變更上，即C5
- 因此當使用者欲作任何測試時，皆可開出分支進行測試，也不需保存測試前之結果



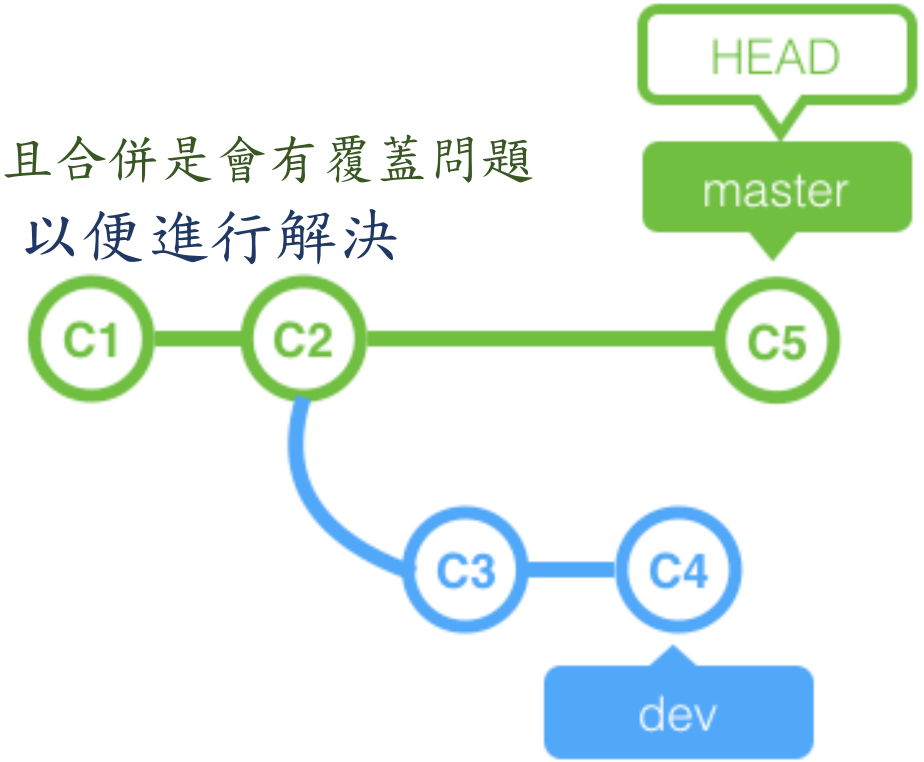
Git常用指令(9)

- checkout(切換)
  - ✓於客戶端的倉庫內分支進行切換(預設會站該條分支的最新時間點上)
    - E.g., 假設目前站在master的C5，切換至dev將會到C4



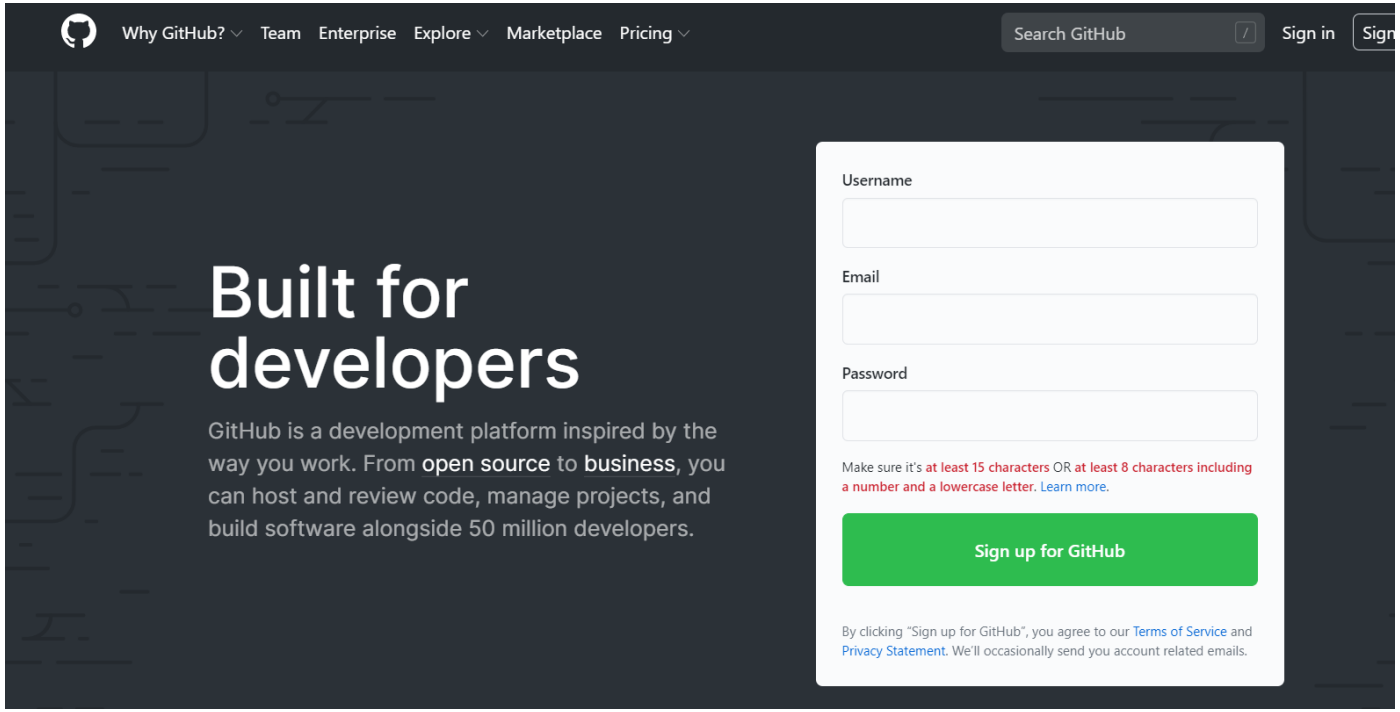
Git常用指令(10)

- merge(合併)
  - ✓ 將目前所在的分支與指定分支合併(是檔案內容合併，不是覆蓋)
  - ✓合併方式：所在分支節點合併指定分支節點，並於所在分支上產生一新的時間點
  - ✓合併會發生衝突
    - 若兩分支的對同一檔案內的內容修改且合併是會有覆蓋問題
  - ✓當發生衝突時，git將會提示衝突，以便進行解決



替代方案

- 架設Git伺服器於目前往往不符合時間成本考量，因此可選擇其他替代方案
- 於本次課程中將使用github做為Git伺服器
  - ✓https://github.com/



設定與安裝教學

- 該篇包含Git客戶端安裝、Github註冊與設定，還有最重要的，可以直接使用圖形介面進行操作—TortoiseGit
  - ✓使用TortoiseGit原因在於，如果初學者一開始直接學習下指令，~~可能會一堆人放棄……~~
  - ✓安裝說明(將以課堂上實際演示，並可於下面網址觀看安裝過程)
    - 點我觀看

Github優點

- 減少git伺服器架設時間
- 可視覺化呈現個人上傳工作時間
  - ✓E.g.,