認識 GIT 與常用指令

What is GIT?

Git 是一個分散式版本控制系統，廣泛應用於軟體開發中。它能夠管理程式碼的變更記錄，提供方便的協作機制。

最早的版本控制就是反覆複製資料夾，並以資料夾命名來記錄不同的程式碼狀態。後來就發展出版本控制系統，並分為集中式版本控制系統和分散式版本控制系統。

集中式版本控制系統對中央伺服器的依賴，單點故障風險高，且協作效率較低。Git 的分散式架構解決了這些問題，每個使用者都有完整的儲存庫，可以在本地進行操作，無需中央伺服器的持續連線。

GIT 相關名詞解釋

版本控制系統：版本控制系統用於追蹤文件或程式碼的變更歷史。

分散式版本控制系統：每個使用者都擁有一個完整的儲存庫，可以在本地進行操作，無需依賴中央伺服器。

工作區、暫存區和儲存庫：工作區是您當前工作目錄，暫存區是將要提交的變更暫存區域，儲存庫是包含所有歷史記錄的地方。

GIT 基本指令

```
# 初始化儲存庫
$ git init
# 從遠端儲存庫複製
$ git clone
# 儲存目前工作的變更
$ git add
# 提交變更至儲存庫
$ git commit
# 從遠端儲存庫更新本地儲存庫
$ git pull
# 上傳本地變更至遠端儲存庫
$ git push
```

GIT 的分支管理

```
# 創建分支
$ git branch

# 切換分支
$ git checkout

# 合併分支
$ git merge
```

GIT SSH 金鑰操作步驟

1. 產生 SSH 金鑰

```
ssh-keygen -t ed25519 -C "yourname@email.com"
```

1. 印出公鑰資訊

```
cd ~/.ssh
cat id_ed25519.pub
```

1. 將公鑰資訊貼到 GitHub 網站（GitHub → Settings → SSH and GPG keys）

設定 GIT 基本設定

為了讓你建立的 git commit 會是記錄正確的名稱和 Email，要進行全域的設定變更

```
git config --global user.name "Your Name"
git config --global user.email "yourname@email.com"
```

GIT 心智模型（一眼看懂）

工作區（Working Directory） → 暫存區（Staging Area） → 本地儲存庫（Local Repo） → 遠端（Remote）

- `git add` 放入暫存區
- `git commit` 寫入本地儲存庫
- `git push` 上傳到遠端
- `git pull` 更新本地（可用 `--rebase` 減少分叉）
- 隨時 `git status`、`git log` 掌握狀態

最小實務流程（從修改到上傳）

```
git status
git add path/to/file
git commit -m "feat: add intro section to README"
git push
```

分支基礎（建/切/合併）

```
git switch -c feature/intro      # 建並切到新分支
git branch --all                 # 列出所有分支
git merge feature/intro          # 在目標分支上合併
```

⚠️ 若發生衝突，打開檔案處理 `<<<<<<<` 標記，測試通過後再提交。

常見還原與後悔藥（安全優先）

```
git restore --staged file.js     # 從暫存區移除，保留檔案變更
git restore file.js              # 還原檔案到最後一次提交
git reset --soft HEAD^           # 回到前一個提交，保留變更於暫存
# 若只是補字：git commit --amend --no-edit
```

診斷好幫手（掌握脈絡）

```
git status
git log --oneline --graph --decorate -n 10
git diff                                # 看工作區差異
git diff --staged                       # 看暫存區差異
```

常見誤區（避免踩雷與補救）

- 在 `main` 直接開發：改用 `git switch -c feature/x`
- 忘了 `.gitignore`：補上後 `git rm -r --cached node_modules`
- 名字/Email 設錯：`git config --global user.name/user.email`，必要時 `--amend`
- SSH 金鑰未加入 GitHub：到 GitHub Settings → SSH and GPG keys 貼上 `~/.ssh/id_ed25519.pub`
- 使用錯誤金鑰或未載入代理：確認 `ssh -T git@github.com`；macOS 建議啟用 Keychain 與 `ssh-agent`
- 本地落後導致推不上去：`git pull --rebase` 再 `git push`
- 強推共享分支：避免 `--force`，必要時用 `--force-with-lease`

圖片：危險指令示意（來源見下）

來源：參考連結（見教學說明）

圖片中的指令會發生什麼事？（類型與後果）

- 破壞檔案系統：`rm -rf /`、`rm -rf ~`、`mkfs.* /dev/sdX`、`dd if=/dev/zero of=/dev/sdX`
  - 後果：系統檔與資料被清除或覆寫，導致系統不可開機/資料全毀
- 重寫磁碟/分割區：`mkfs`/`dd`/`> /dev/sda`
  - 後果：磁碟格式化或覆蓋原資料，無法輕易復原
- 遠端腳本直通執行：`curl ... | sh`、`wget ... | sh`
  - 後果：在未審核的情況下執行任意程式碼，可能植入惡意程式
- Fork Bomb：`:(){ :|: & };:`
  - 後果：快速建立大量行程耗盡資源，系統當機
- 權限誤用：`chmod -R 777 /` 等

Git Cheat Sheet（10 行）

```
git status                          # 目前狀態
git add <path>                      # 放入暫存區
git commit -m "msg"                 # 建立提交
git push                            # 推到遠端
git pull --rebase                   # 取回並線性化
git switch -c feature/x             # 新分支並切換
git merge feature/x                 # 合併分支
git log --oneline --graph -n 10     # 精簡歷史
git diff / git diff --staged        # 差異比較
git restore/restore --staged        # 檔案/暫存還原
```