

>_ WHOAMI?

0sGa or os24

- CTF Player | Developer
- A student from NYUST
- btw I use Vim

more on <https://osga.dev>



>_ 今天要做啥？

- 著重帶大家熟悉兩個工具
 - `tmux`
 - `nvim`

>_ Slido

有任何問題都歡迎提問！

CODE: 9025087







>_ tmux

🧠 什麼是 tmux？

- **tmux** 是一個 終端機多工器。
 - 它可以讓你在 一個終端視窗中同時跑多個視窗或分割畫面。
 - 常用在 Linux 或 macOS 的命令列環境。
-

🔍 為什麼要用 tmux？

-  **同時開多個程式**（像開多個視窗），但都在同一個終端。
 -  **不中斷的背景執行**: 就算你關掉筆電，程式還是繼續跑！
 -  **隨時恢復工作狀態**: 重開電腦也可以回到之前的工作畫面。
 -  **分割畫面**: 上下左右分割，同時看 log、編碼、測試。
-

❓ tmux 跟什麼像？

- 有點像終端機版的「分頁式 Chrome」或「多視窗 VSCode」。
- 把「每一個工作」放在自己的小空間中，不打架。

>_ tmux

📦 安裝 tmux

Linux:

```
sudo apt install tmux
```

MacOS:

```
sudo brew install tmux
```

👉 進入 tmux

安裝完成後直接在終端輸入

```
tmux
```





看到終端下面出現一條 bar 代表進入 tmux 了

>_ tmux

🧠 什麼是 tmux？

- **tmux** 是一個 終端機多工器。
 - 它可以讓你在 一個終端視窗中同時跑多個視窗或分割畫面。
 - 常用在 Linux 或 macOS 的命令列環境。
-

🔍 為什麼要用 tmux？

-  **同時開多個程式**（像開多個視窗），但都在同一個終端。
 -  **不中斷的背景執行**: 就算你關掉筆電，程式還是繼續跑！
 -  **隨時恢復工作狀態**: 重開電腦也可以回到之前的工作畫面。
 -  **分割畫面**: 上下左右分割，同時看 log、編碼、測試。
-

❓ tmux 跟什麼像？

- 有點像終端機版的「分頁式 Chrome」或「多視窗 VSCode」。
- 把「每一個工作」放在自己的小空間中，不打架。

>_ tmux

📦 安裝 tmux

Linux:

```
sudo apt install tmux
```

macOS:

```
brew install tmux
```

👉 進入 tmux

安裝完成後直接在終端輸入


```
tmux
```

看到終端下面出現一條 bar 代表進入 tmux 了

`Ctrl + D` 或打 `exit` 可以退出


>_ tmux

 操作練習: 啟動與離開 tmux

 建立一個新的 tmux 工作區

```
tmux new -s demo
```


這樣你就進入一個叫做 **demo** 的 tmux session
終端機下方會出現一條綠色或藍色的狀態列

 離開 tmux (但不關掉)

按下:

```
Ctrl + b 然後按 d
```

這會將你「detach」，tmux 還在背景運行！

 回到剛剛的 session

```
tmux attach -t demo
```

你會重新連回剛剛的工作區，畫面恢復原樣！

>_ tmux

■ □ Pane 分割練習

▮ ✂ 上下分割畫面

```
Ctrl + b 然後按 "
```

▮ ➦ 左右分割畫面

```
Ctrl + b 然後按 %
```

你現在畫面應該出現兩三個視窗，互相獨立！

▮ ⇐⇒ Pane 之間切換

```
Ctrl + b 然後按 方向鍵 (←↑↓→)
```

移動游標來切換不同區塊

▮ ✖ 關閉 Pane


在你想關掉的畫面輸入:

```
exit
```


或使用 `Ctrl + d`，畫面就會自動消失～

>_ tmux

 管理多個視窗 (Window)

 + 新增視窗

```
Ctrl + b 然後按 c
```

 切換視窗

```
Ctrl + b 然後按 n (下一個)  
Ctrl + b 然後按 p (上一個)
```

 查看視窗清單

```
Ctrl + b 然後按 w
```



 重新命名視窗

```
Ctrl + b 然後按 ,
```

為你的工作取一個清楚的名字吧！

>_ tmux

  清理與結束

  查看所有 tmux 工作區

```
tmux ls
```

  結束一個 session

```
tmux kill-session -t demo
```

這樣就會徹底關閉剛剛的工作環境

  小挑戰

請嘗試完成以下操作:

1. 建立一個 `practice` 工作區
2. 分割上下與左右各一次
3. 在一個 pane 執行 `top`
4. 使用快捷鍵切換 Pane
5. 使用 `Ctrl+b d` 離開，再 attach 回來
6. 清理所有 Pane 並 kill 該 session

>_ LazyVim

🌟 什麼是 LazyVim?

- LazyVim 是一個為現代開發者打造的 Neovim 配置框架。
- 基於 Neovim + Lua + lazy.nvim (插件管理器) 。
- 開箱即用，預設就包含 LSP、Telescope、Treesitter 等強大工具。

📦 LazyVim 的特色

- ⚡ **快速啟動**: 使用 lazy.nvim 延遲加載，提高啟動速度
- 🧠 **智能補全**: 內建 nvim-cmp + LSP，支援自動補全、診斷
- ✏️ **即時測試**: 預設支援多種 test runner 與編譯器
- 🍷 **美觀介面**: 美化狀態列、Telescope 選單、UI 樣式
- ⚙️ **模組化設計**: 方便客製與擴充插件

>_ LazyVim

📦 安裝 LazyVim

📋 Step 1: 先安裝 Neovim (v0.9+)

🍏 macOS:

```
brew install neovim
```

🐧 Ubuntu / WSL:

```
sudo apt install neovim
```

📋 Step 2: 安裝 LazyVim 基礎配置

📌 建請備份你原本的 Neovim 設定！

```
mv ~/.config/nvim ~/.config/nvim.backup  
mv ~/.local/share/nvim ~/.local/share/nvim.backup
```

然後 clone 官方 LazyVim:



```
git clone https://github.com/LazyVim/starter ~/.config/nvim  
nvim
```




LazyVim 會在首次啟動時自動安裝所需插件。



>_ LazyVim

  第一次啟動後你會看到:

- 左下角是 **狀態列** (lualine)
- **<space>** 是主要功能鍵 (Leader Key)
- 啟動畫面可以用 **⌂** 快速載入最近開啟的檔案

  試試這些常用操作

- **<space> f f:**  打開檔案搜尋 (Telescope)
- **<space> f w:**  搜尋目前資料夾內文字
- **<space> b b:**  開啟 buffer 清單
- **<space> g g:**  開啟 lazygit (需安裝)

  編輯功能與補全

- 使用 **Tab** 自動補全
- 使用 **K** 檢視語言說明文件 (LSP hover)
- 使用 **<space> c a** 觸發 code actions (LSP)

>_ LazyVim

🔧 客製化 LazyVim

📁 LazyVim 的設定結構

```
~/.config/nvim/  
├── init.lua  
├── lua/  
│   ├── plugins/  
│   └── config/  
│       └── ...
```

- **plugins/**: 自定義插件清單與設定
- **config/**: 設定 LSP、UI、快捷鍵等
- **init.lua**: 入口，載入所有模組

📁 🍴 新增一個自訂插件

在 **lua/plugins/myplugin.lua** 中新增:

```
return {  
  "tpope/vim-surround",  
  event = "VeryLazy",  
}
```

儲存後重啟 Neovim 或執行:

```
:Lazy sync
```


>_ LazyVim

🧠 小挑戰: LazyVim 初體驗

請同學們嘗試完成以下練習:

1. 安裝 LazyVim 並成功啟動
2. 使用 `<space> f f` 搜尋一個檔案
3. 使用 `<space> c a` 觸發一次 LSP code action
4. 自訂一個插件
5. 嘗試改變顏色主題

