多奇·教育訓練

2025/7/16

# AI 程式設計代理人開發全攻略

## Claude Code 代理人開發全攻略

多奇數位創意有限公司

技術總監 黃保翕（Will 保哥）

https://blog.miniasp.com


# 簡介 Claude Code

# 關於 Claude Code

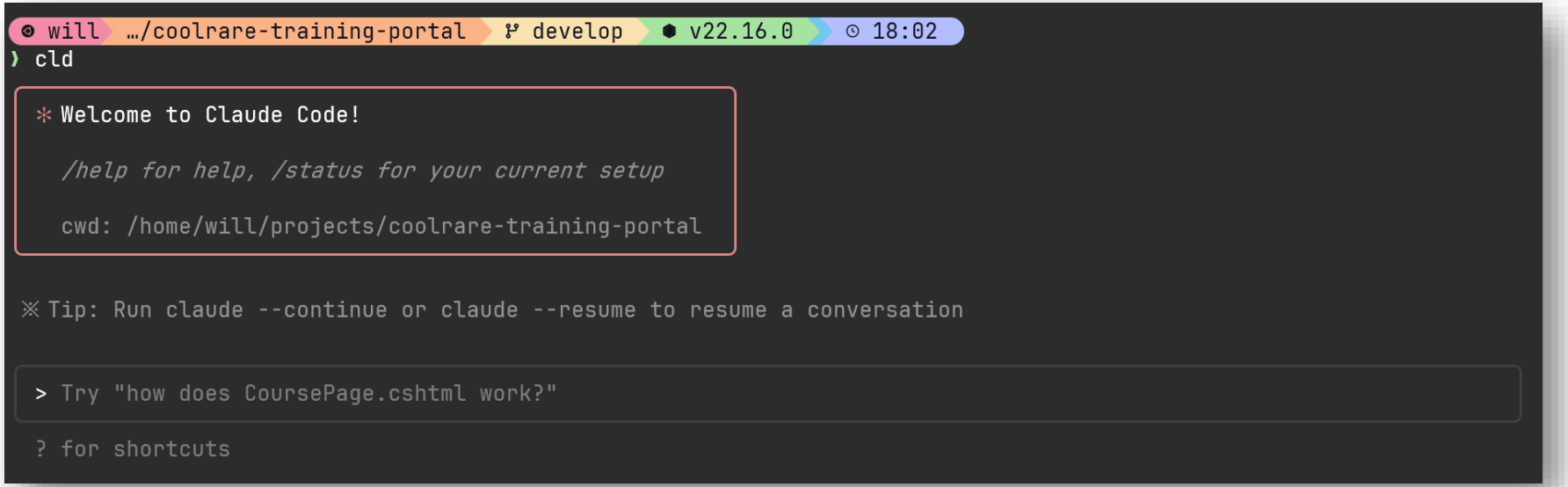
- Claude Code 是由 Anthropic 推出的命令列環境下的 AI 程式設計助手
  - Claude Code 定位為**終端機**中的 AI 夥伴
  - 開發者可以**在終端機中**與 Anthropic 的大型語言模型 Claude 互動
  - 可用於實現所謂的 Agentic Coding 或 Vibe coding (代理人式開發流程)
  - 自動規劃任務、自動執行命令、自動讀取/修改程式碼，協助完成各種開發任務
  - Claude Code 讓 Anthropic 的 AI 模型更無縫地融入開發者的工作流程，提供類似Pair Programming的體驗，同時具備執行實際命令的能力
- **設計哲學**
  - 強調低層次、高客製化的設計哲學，強調靈活的控制而不強制特定工作流程
  - 這樣的彈性也意味著使用 Claude Code 需要一些**學習曲線**才有辦法產生生產力
  - 但一旦熟悉，就能將其作為強大的**腳本化工具**整合進**日常開發流程**！

# 準備 Claude Code 執行環境

- **最小安裝**
  - 確認系統需求並確保安裝 Node.js 18 or newer、Git、GitHub CLI 就可以執行 Claude Code 工具
- **Windows 作業系統**
  - 從 v1.0.51 開始支援原生 Windows 命令列環境執行，無論命令提示字元或 PowerShell 都可以
  - 但我依然建議使用 **WSL 2 + Ubuntu 22** 當成主要環境，因為 Shell 工具鍊非常完整！
  - 可參考我替這堂課精心準備的 最佳 WSL + Ubuntu 22 環境設定 文件進行設定
  - 容器環境: 如何移除 Docker Desktop 並在 Windows 與 WSL 2 改安裝 Docker Engine
- **macOS / Linux 作業系統**
  - 也可參考 最佳 WSL + Ubuntu 22 環境設定 文件進行工具設定
  - 容器環境: Docker Desktop on Mac, apple/container

# 啟動 Claude Code 的方法

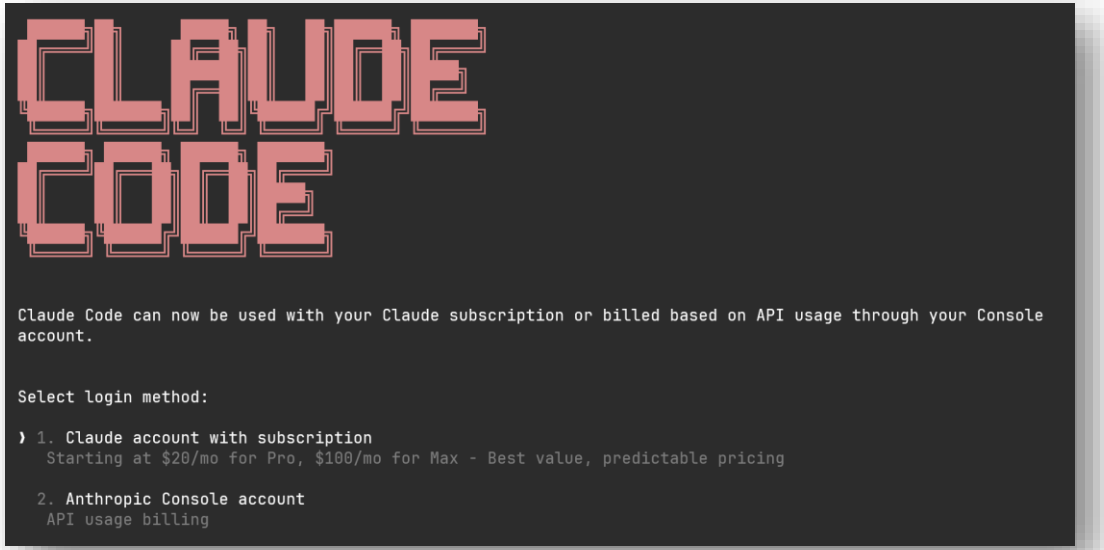
- **全域安裝**
  - `npm install -g @anthropic-ai/claude-code`
  - `alias cld='claude'`
  - `cld`
- **直接執行**
  - `npx @anthropic-ai/claude-code`




Claude Code on Windows 新手上路

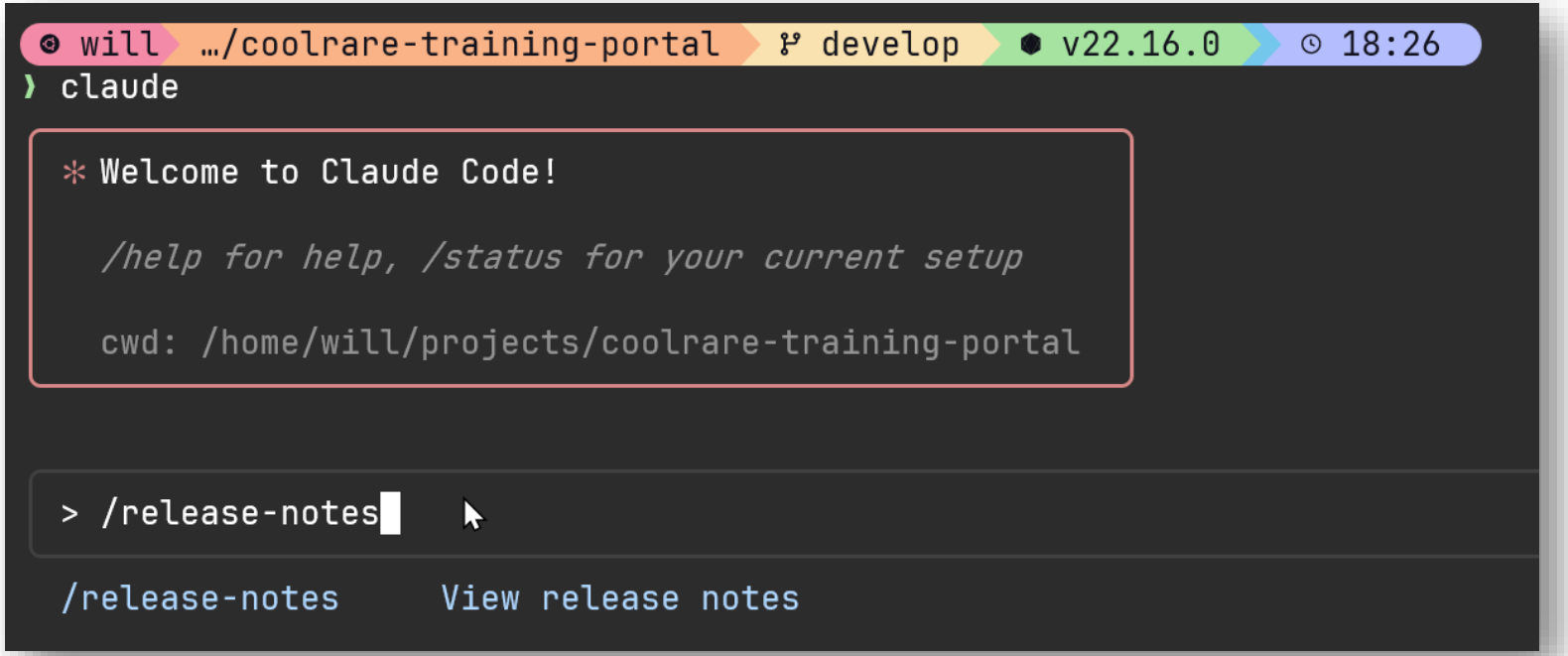
# 搞定 Claude Code 身分驗證

- 第一次登入時就會自動引導你通過認證，比 Gemini CLI 簡單 1 萬倍吧！ XD
- **Claude 帳號 (需 Pro 或 Max 方案)**
  - 以限制**使用量**的方式提供
  - 短時間不要超量就不會達標
- **Anthropic Console 帳號**
  - 以 **Tokens** 用量計費
  - 付費 **$5** 美元就可以開始用
- **企業平臺**
  - 使用 Amazon Bedrock 或 Google Vertex AI 進行企業部署，只要依照文件設定環境變數即可。你甚至可以使用其他 non-Claude 的模型使用！



# Claude Code 並非開放原始碼軟體

- 回報問題: https://github.com/anthropics/claude-code/issues
- 發行記錄: **/release-notes**



# 簡介 Anthropic 的 Claude 模型家族

# Claude 模型的特性

- Anthropic 的核心模型，經過**大量微調**！
  - 擁有 200,000 Tokens 的上下文窗口 (Context Window)
  - 具備強大的**自然語言理解**、**圖像處理**、**多語言任務**、**分析與推理**與**程式設計**等能力
- Claude 模型系列與定位
  - Claude **Opus**　　專門處理**複雜推理**任務，Anthropic 最強大的模型
  - Claude **Sonnet**　　平衡的中型模型，是兼具**能力**和**效能**的模型，表現依然傑出！
  - Claude **Haiku**　　針對**速度優化**的**輕量級**模型，適合即時應用
- 不同系列
  - 2024 年發表 Claude 3.x 家族代號 Haiku、Sonnet 等版本
  - 2025 年 5 月發布 **Claude 4** 家族代號 Haiku、Sonnet、Opus！

# Anthropic 是一間做 LLM 的公司

- 這意味著
  - **越多人用**越好，他們進步越快
  - **越多管道**使用他們的模型越好，他們收入會更高
  - 做 LLM 的門檻極高，唯有成為世界第一，他們才能生存下去

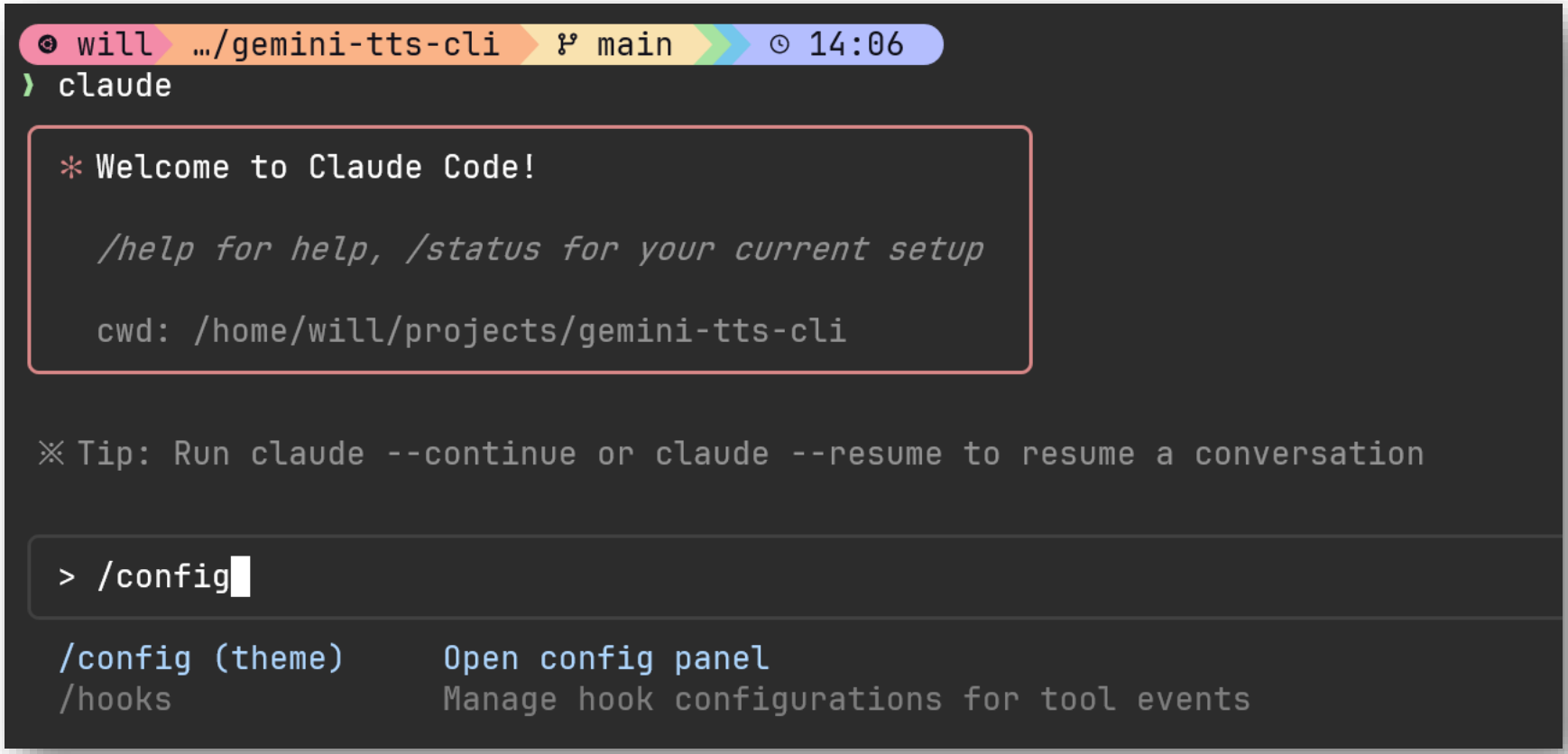
- 他們重視開發者體驗
  - 打從一開始 Claude 對於「開發人員」就特別友善
  - 我們從 Claude Code 的設計細節經常會看出很多巧思
  - 他們有 Claude Code SDK 可以用！🔥

# 認識 Claude Code 指令與互動方式

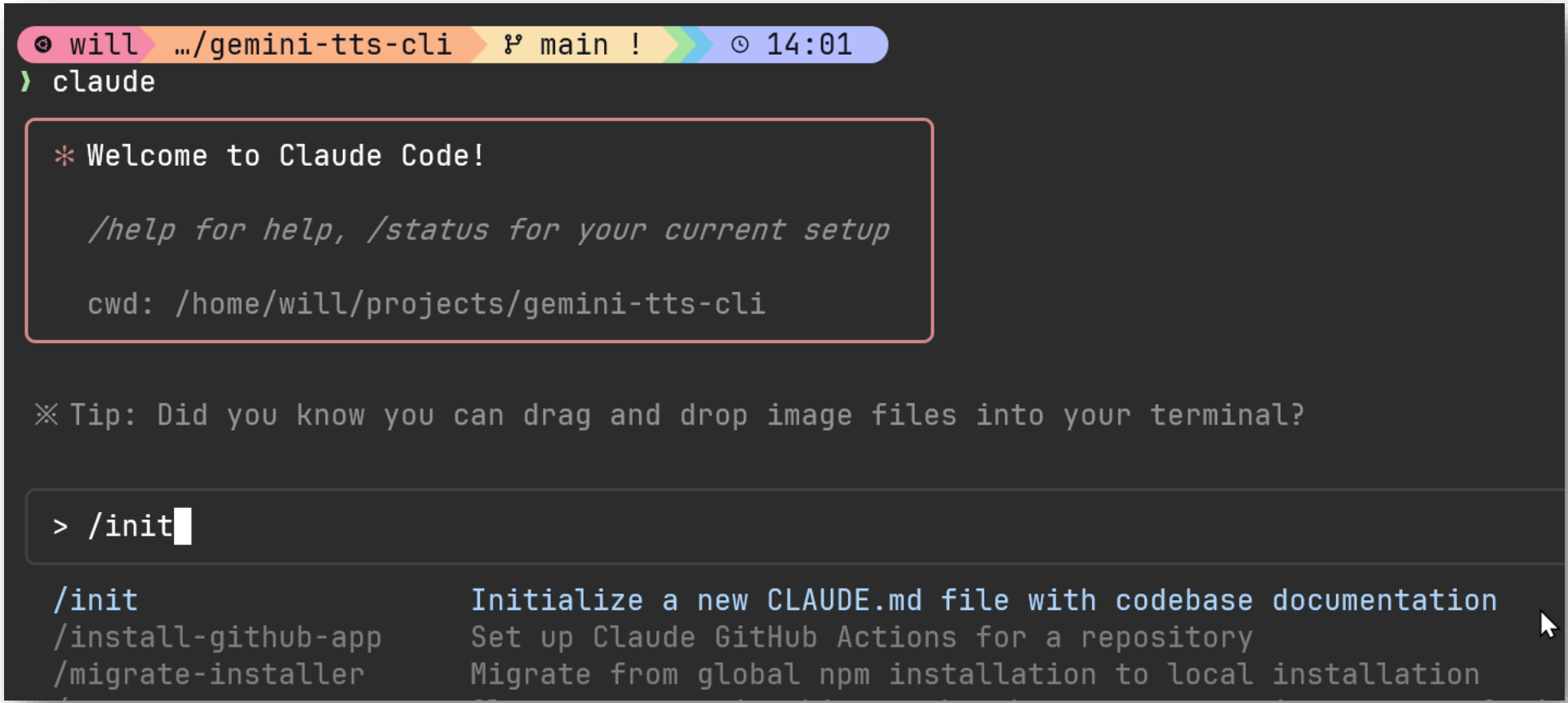
# 主要的使用情境

- **透過對話式指令來互動**
  - Claude 會根據上下文理解您的意圖，可能採取一系列步驟來完成任務
  - 可使用內建的斜線命令 (slash commands) 快速完成特定工作
  - 可自定義特殊的斜線命令 (Custom slash commands)
  - 使用 ! 執行 bash 指令將結果納入提示
  - 使用 @ 參考專案內的檔案
  - 搭配 Hooks 等機制，讓使用者能細緻控制 AI 行為
- **透過無互動的方式自動化開發**
  - 通常會透過 Claude Code SDK 來實現
  - 支援 CLI, TypeScript, Python 三種 SDK

# 主題和外觀設定 /config



# 初始化專案 /init



# 提交產生的 CLAUDE.md 檔案

```
● The CLAUDE.md file has been updated with minor improvements:

  1. Added missing macOS build commands - Added the macOS x64 and ARM64 publish commands that were referenced
  development commands section.
  2. Enhanced release trigger documentation - Added a concrete example of how to trigger releases with git ta

  The existing CLAUDE.md was already comprehensive and well-structured, covering all the essential informatio
  - Complete development commands for building, testing, and publishing
  - Clear architecture overview with single-file design explanation
  - Key component breakdown and API integration details
  - Voice management system and audio processing standards
  - Testing structure and environment requirements
  - Development guidelines from copilot instructions
  - CI/CD integration details

  The file effectively captures the "big picture" architecture and development workflow without being overly


> git add, commit, push
```

# 退出 Claude Code


- 快速連續按兩次 **Ctrl+D** 或 **Ctrl+C** 即可

# 短指令

- 輸入 ? 可以查看輸入提示
- 可按 ↑ 查看命令歷史
- 輸入 @ 可以選取檔案或資料夾
  - 檔案也可以包含「圖片」內容，按下 Tab 鍵可以快速自動完成/自動補全
- 輸入 ！ 可以輸入 bash 命令
- 輸入 / 可以輸入 Claude Code 專屬的斜線命令 (Slash commands)
- 輸入 # 可以加入一段記憶到指定的上下文檔案中（Bug #2062）

# 使用 # 不會加入到記憶中的應變措施

- 自己手動打以下這段提示

  - add this to CLAUDE.md

```
# Always use dotnet build with exising @gemini-tts-cli.csproj file
```

```
Where should this memory be saved?

 ❯ 1. Project memory           Checked in at ./CLAUDE.md
   2. Project memory (local)   Gitignored in ./CLAUDE.local.md
   3. User memory              Saved in ~/.claude/CLAUDE.md

Example project memory: “Run lint with the following command after major edits: npm run lint”
```

# 斜線命令 (Slash commands) 1/2

| 命令 | 用途 |
| --- | --- |
| /init | 使用 CLAUDE.md 指南初始化專案 |
| /login | 切換 Anthropic 帳戶 |
| /logout | 登出您的 Anthropic 帳戶 |
| /mcp | 管理 MCP 伺服器連線和 OAuth 驗證 |
| /memory | 編輯 CLAUDE.md 記憶檔案 |
| /model | 選擇或變更 AI 模型 |
| /permissions | 檢視或更新權限 |
| /pr_comments | 檢視拉取請求評論 |
| /review | 請求程式碼審查 |
| /status | 檢視帳戶和系統狀態 |
| /terminal-setup | 安裝 Shift+Enter 按鍵綁定以換行（僅限 iTerm2 和 VSCode） |
| /vim | 進入 vim 模式以交替插入和命令模式 |

# 斜線命令 (Slash commands) 2/2

| 命令 | 用途 |
| --- | --- |
| /add-dir | 新增額外的工作目錄 |
| /bug | 回報錯誤（將對話傳送給 Anthropic） |
| /clear | 清除對話歷史記錄 |
| /compact [instructions] | 壓縮對話，可選擇性地提供焦點指示 |
| /config | 檢視/修改設定 |
| /cost | 顯示 token 使用統計 (使用 Claude 訂閱認證會看不到) |
| /doctor | 檢查您的 Claude Code 安裝健康狀況 |
| /help | 取得使用說明 |
| /add-dir | 新增額外的工作目錄 |
| /bug | 回報錯誤（將對話傳送給 Anthropic） |
| /clear | 清除對話歷史記錄 |
| /compact [instructions] | 壓縮對話，可選擇性地提供焦點指示 |

# 鍵盤輸入的技巧

- 可以按 **Ctrl+V** 貼上大量「文字」內容 (不能貼圖)
- 可以按 **Ctrl+U** 復原上次輸入，雙擊 **Escape** 鍵可以清空輸入
- 可以按 **Ctrl+R** 顯示上一個步驟的完整回應內容，再按一次 **Ctrl+R** 可以復原
- 可以按 **Ctrl+Z** 暫時退出 Claude Code 環境，然後按 **fg** 跳回來
- 可以按 **Shift+Tab** 切換不同模式：**自動接受編輯模式**、**計畫模式** (唯讀)、**一般模式**
- 在目前輸入的最後面輸入一個反斜線 **** 並按下 **Enter** 即可換行
  - macOS 可以按 **Option+Enter** 或 **Meta+Enter**
  - Windows 可以按 **Ctrl+J** 或 **Ctrl+Enter**

# 快速入門

- 建議認真閱讀官方的快速入門文件
  - 步驟 1：開始您的第一個會話
  - 步驟 2：提出您的第一個問題
  - 步驟 3：進行您的第一次代碼更改
  - 步驟 4：將 Git 與 Claude Code 一起使用
  - 步驟 5：修復錯誤或添加功能
  - 步驟 6：測試其他常見工作流程
  - 基本命令
  - 初學者專業提示
    - 使用 `claude commit` 就可以幫你 commit 一份完美的 git commit

# Vibe coder 的提示參考

- generate .gitignore
- setup quality control
- fix it
- fix the code
- bump major version
- bump minor version
- bump patch version
- add, commit, push
- 啟動不同的思考等級
  - think
  - think hard
  - think harder
  - ultrathink
- refactor this function to be more readable
- create a new branch and fix this error, then push to the remote
- amend the current commit and add summary of this session in it

# 使用 Claude Code 的基本心法

- **斜線指令（/）或 At 指令（@）或 Shell 模式與傳遞指令（!）**

  無論是哪一種，其目的只有一個：**準備好上下文！**

- **提示建議**

  - 不要將模糊不清的需求轉化成「提示」，讓 Claude 先幫助你理解問題！
    - 在進行任務之前，讓 Claude 了解您的程式、錯誤訊息、常用命令、截圖、商業邏輯
    - 不要說：「修復錯誤」，嘗試：「修復登錄錯誤，用戶輸入錯誤憑據後看到空白屏幕」
  - 將複雜任務分解為步驟，分批送出，逐一確認
    - 釐清「脈絡」比什麼都還重要！脈絡就是 "Context"
  - 重構程式碼不要一次到位
    - 先識別需要重構的舊程式碼，獲得重構建議，安全地應用變更，驗證重構結果

在非互動模式下執行 (CI)

# Claude Code SDK

# 認識 Claude Code SDK

- Claude Code SDK 主要用於**非互動模式**下進行延伸！👍
- **CLI SDK**
  - Claude Code SDK 最基本的形式就是以 CLI 介面執行，加上 **-p** 參數即可
    - `npm install -g @anthropic-ai/claude-code`
- **TypeScript SDK**
  - TypeScript SDK 包含在 NPM 上的主要 **@anthropic-ai/claude-code** 套件中
    - `npm install @anthropic-ai/claude-code`
- **Python SDK**
  - Python SDK 在 PyPI 上以 **claude-code-sdk** 的形式提供
    - `pip install claude-code-sdk`

# Claude Code SDK for CLI

- 三種輸出格式

```
# 純文字
cat main.py | claude -p 'Review this code for bugs'
# JSON
claude -p 'Generate a hello world python app' --output-format json
# 串流 JSON
claude -p 'Generate a hello world python app' --output-format stream-json --verbose
```

- 進階用法
  - 支援「多輪對話」，可以**恢復對話**或從最近的會話**繼續**
  - 支援「自訂系統提示」或「附加系統提示」
  - 完整範例

# 多輪對話

```
# 繼續最近的對話
$ claude --continue

# 繼續並提供新的提示
$ claude --continue "Now refactor this for better performance"

# 開始會話並捕獲會話 ID
$ claude -p "Initialize a new project" --output-format json | jq -r '.session_id' > session.txt

# 使用相同會話繼續
$ claude -p --resume "$(cat session.txt)" "Add unit tests"

# 透過會話 ID 恢復特定對話
$ claude --resume 550e8400-e29b-41d4-a716-446655440000

# 在列印模式（非互動）中恢復
$ claude -p --resume 550e8400-e29b-41d4-a716-446655440000 "Update the tests"

# 在列印模式（非互動）中繼續
$ claude -p --continue "Add error handling"
```

~/.claude/projects/

# 自訂系統提示

```
# 具有特定要求的系統提示
$ claude -p "Create a database schema" --system-prompt "You are a database architect.
Use PostgreSQL best practices and include proper indexing."

# 覆蓋系統提示 (僅適用於 --print 的使用情境)
$ claude -p "Build a REST API" --system-prompt "You are a senior backend engineer.
Focus on security, performance, and maintainability."

# 附加系統提示 (僅適用於 --print 的使用情境)
$ claude -p "Build a REST API" --append-system-prompt "After writing code, be sure to
code review yourself."
```

# 串流 JSON 輸入

- 透過 **stdin** 提供的訊息串流，其中**每個訊息代表一個使用者輪次**。
- 這允許多輪對話而**無需重新啟動 `claude` 執行檔**，並允許在模型處理請求時向模型提供指導。
- 每個訊息都是一個 JSON 資料，內容必須是完整的 **user message** 物件，遵循與輸出訊息架構相同的格式。
- 訊息使用 **`jsonl`** 格式格式化，其中每行輸入都是一個完整的 JSON 物件。串流 JSON 輸入需要 -p 和 --output-format stream-json。
- 範例

```
echo '{"type":"user","message":{"role":"user","content":[{"type":"text","text":"Explain this code"}]}}' | claude -p \
  --output-format=stream-json \
  --input-format=stream-json \
  --verbose
```

# Claude Code SDK for TypeScript

```typescript
import { query, type SDKMessage } from "@anthropic-ai/claude-code";

const messages: SDKMessage[] = [];

for await (const message of query({
  prompt: "Write a haiku about foo.py",
  abortController: new AbortController(),
  options: {
    maxTurns: 3,
  },
})) {
  messages.push(message);
}

console.log(messages);
```

https://docs.anthropic.com/en/docs/claude-code/sdk#typescript

# Claude Code SDK for Python

```python
from claude_code_sdk import query, ClaudeCodeOptions
from pathlib import Path

options = ClaudeCodeOptions(
    max_turns=3,
    system_prompt="You are a helpful assistant",
    cwd=Path("/path/to/project"),  # Can be string or Path
    allowed_tools=["Read", "Write", "Bash"],
    permission_mode="acceptEdits"
)

async for message in query(prompt="Hello", options=options):
    print(message)
```

https://docs.anthropic.com/en/docs/claude-code/sdk#python

# GitHub Actions

- **REPL**
  - 可以使用 **`/install-github-app`** 斜線命令快速設定 GitHub Actions workflows
  - 安裝好就可以在 Issues 或 Issue comment 透過 **`@claude`** 呼叫 Claude Code 出來做事
  - 文件: Claude Code GitHub Actions

Settings

# 深入講解 Claude Code 設定

# 階層化的設定檔

- **使用者設定** (套用於所有使用 Claude Code 的情境)
  - 位於 `~/.claude/settings.json`，適用於該用戶在所有專案中的預設設定
  - 例如您個人偏好的編碼風格、全域允許的工具清單等，可在此配置
- **專案設定** (套用於整個專案並應加入版控)
  - 位於當前專案目錄下的 `.claude/settings.json`
  - 這通常由團隊共同維護，團隊內所有人在此專案使用 Claude Code 時都共享一致的行為。
  - **專案設定**可覆蓋**使用者設定**對應項目，以符合專案需求。
- **專案本地設定** (個人在此專案暫時想調整某些 Claude 行為)
  - 位於 `.claude/settings.local.json`，用於開發者在某專案中的**個人設定**
  - 一般該檔案不會納入版控，而且會加入 .gitignore 檔案中（`*.local.json`）

💡 Anthropic 文件提到以前也有 `CLAUDE.local.md` 作為個人記憶檔，但現已不推薦，用 **imports** 方法取代

# 終端機設定的技巧

- 在 REPL 互動介面下，可以輸入 **/config** 斜線命令來快速設定

| 設定項目 | 目前值 | 說明 |
| --- | --- | --- |
| 自動壓縮 | true | 啟用自動壓縮功能 |
| 使用待辦清單 | true | 啟用待辦事項清單功能 |
| 詳細輸出 | false | 停用詳細輸出模式 |
| 自動更新 | true | 啟用自動更新功能 |
| 主題 | Dark mode | 深色模式 |
| 通知 | Terminal Bell (\a) | 終端機提示音 |
| 編輯器模式 | normal | 標準編輯器模式 |
| 模型 | Default (recommended) | 只有訂閱 Max plan 與 API key 可以選用模型 |
| 使用自訂 API 金鑰 | false | 當有 ANTHROPIC_API_KEY 環境變數才會出現 |

# 認識 Claude Code for VSCode

- **自動安裝**
  - 當您從 VSCode 的終端機啟動 Claude Code 時，它會自動偵測並安裝擴充功能 (Bug #3431)
- **選取內容**
  - 編輯器中選取的文字會自動加入 Claude 的內容中（透過 Claude Code: Insert At-Mentioned 命令）
- **差異檢視**
  - 程式碼變更可以直接顯示在 VSCode 的差異檢視器中，而非終端機！
- **鍵盤快速鍵**
  - 支援 Alt+Cmd+K / Ctrl+Alt+K 等快速鍵，將選取的程式碼推送到 Claude 的 CLI 提示中
- **分頁感知**：Claude 可以看到您在編輯器中開啟了哪些檔案
- **自動設定**：可以在 `/config` 斜線命令中將差異工具設定為自動，以啟用 IDE 整合功能

# 整合 IDE 的示範 ( VS Code )

- Claude Code 雖然運行在命令列，但它與現代 IDE 可以深度整合，提供更順暢的體驗
  - Claude Code [Beta] Plugin for JetBrains IDEs | JetBrains Marketplace
  - Claude Code for VSCode - Visual Studio Marketplace

- **VS Code 內建的命令**
  - Run Claude Code　　快速鍵: `Ctrl+Escape`（建議改為 `Ctrl+Shift+`` ）
  - Claude Code: Insert At-Mentioned　　快速鍵: `Ctrl+Alt+K`
  - Fix with Claude Code　　快速鍵: 無預設綁定

- **中肯建議**
  - 只要你的桌子夠大，建議螢幕能買多寬就買多寬！😄

Custom slash commands

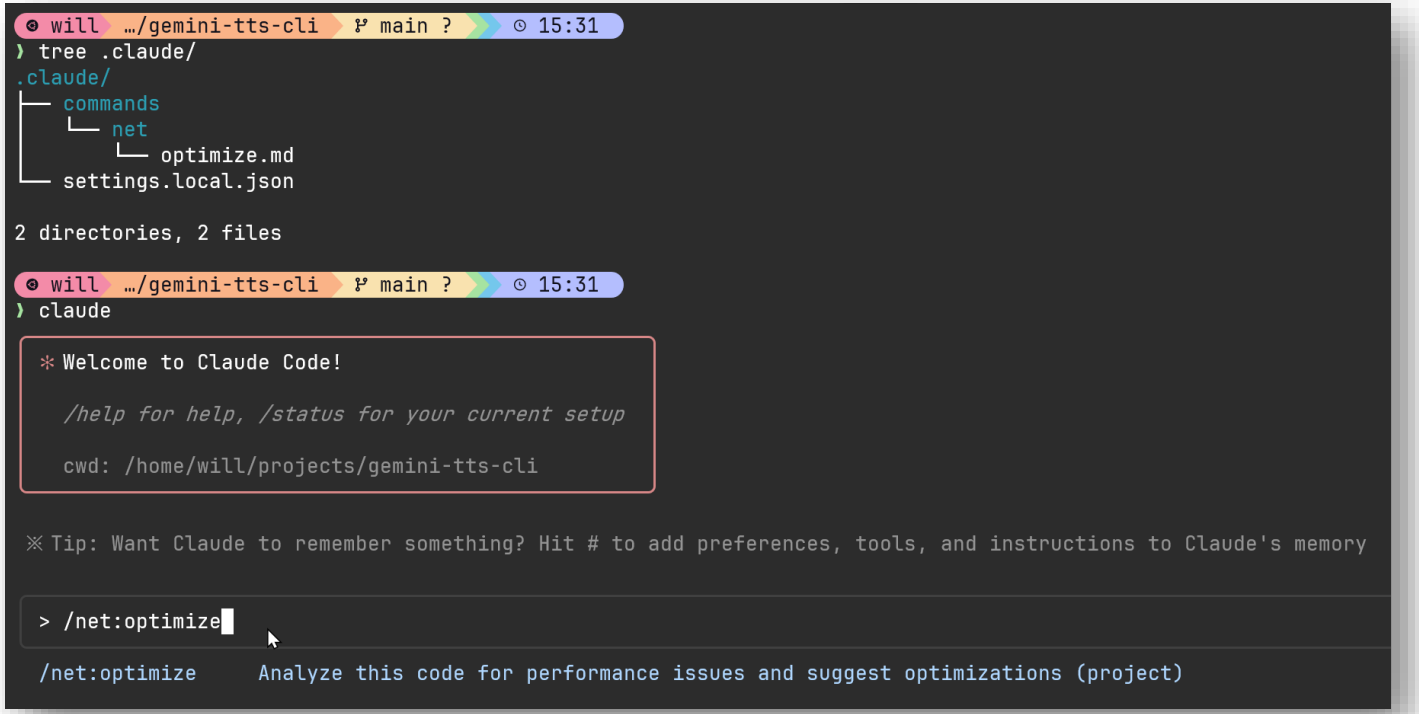
# 自訂斜線命令

# 自訂斜線命令

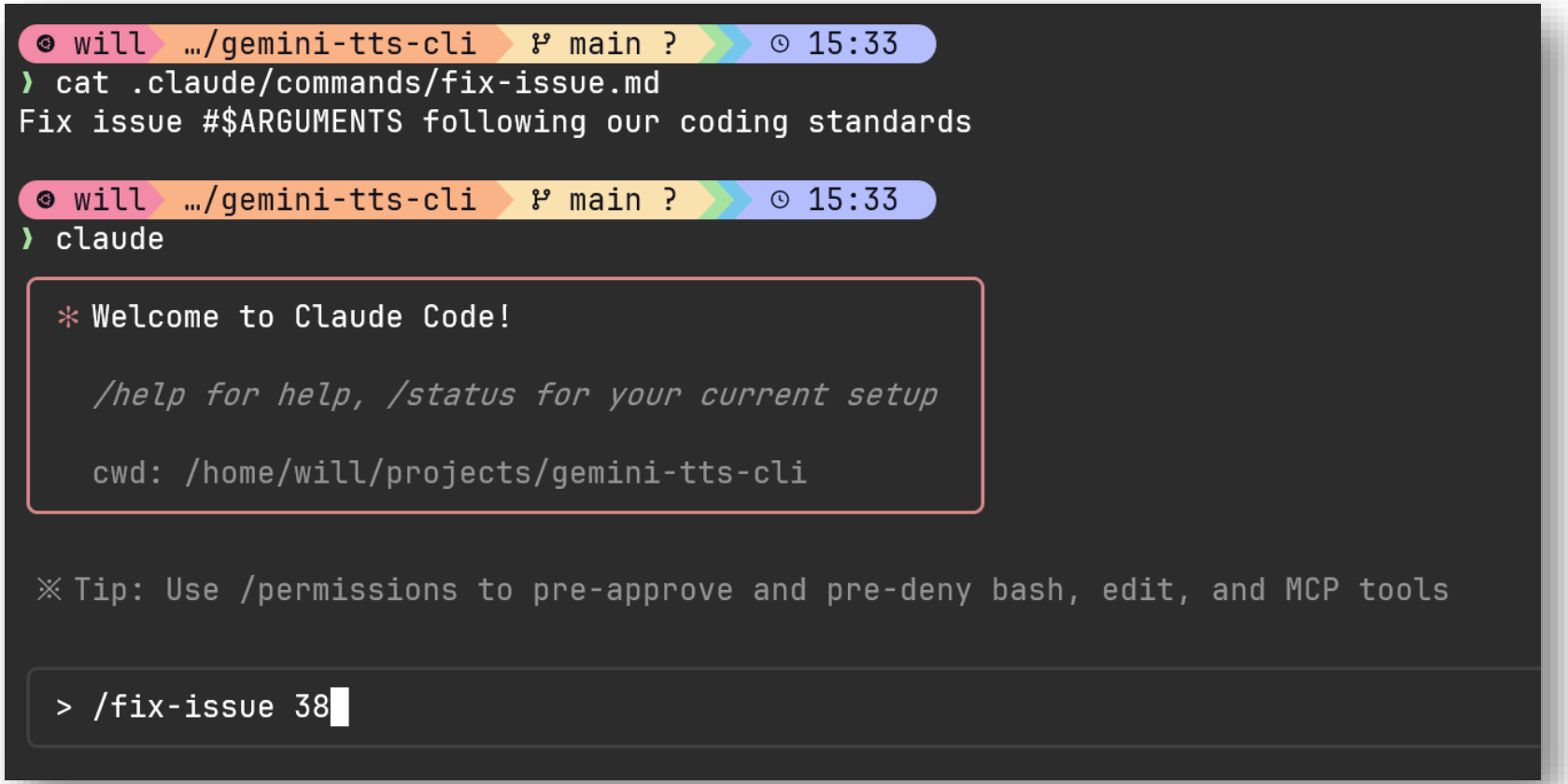
- 自訂斜線命令允許您將常用提示定義為 Markdown 檔案
  - Claude Code 可以執行這些檔案，使用方式與 GitHub Copilot 完全相同
  - 自訂命令可以依照範圍組織 (專案、個人)，並透過目錄結構支援命名空間
- 基本語法
  - `/<command-name> [arguments]`
    - `<command-name>` 從 Markdown 檔案名稱衍生的名稱（不含 .md 副檔名）
    - `[arguments]` 傳遞給命令的可選參數
- 專案命令
  - **`.claude/commands/*.md`**
- 個人命令
  - **`~/.claude/commands/*.md`**
- **Claude Code 官網文件中文版更新非常慢，建議看英文版！🔥**

# 自訂斜線命令 - 命名空間

- 目錄結構　`.claude/commands/NAMESPACE/命令名稱.md`
- 呼叫方式　`/net:optimize`



# 自訂斜線命令 - 使用 $ARGUMENTS 參數



# 自訂斜線命令 - 執行 Bash 指令

```
❯ bat .claude/commands/commit.md
       File: .claude/commands/commit.md
   1   ---
   2   allowed-tools: Bash(git add:*), Bash(git status:*), Bash(git commit:*)
   3   description: Create a git commit
   4   ---
   5
   6   ## Context
   7
   8   - Current git status: !`git status`
   9   - Current git diff (staged and unstaged changes): !`git diff HEAD`
  10   - Current branch: !`git branch --show-current`
  11   - Recent commits: !`git log --oneline -10`
  12
  13   ## Your task
  14
  15   Based on the above changes, create a single git commit.

 will …/gemini-tts-cli  main ?  15:39
❯ claude
 ✻ Welcome to Claude Code!

   /help for help, /status for your current setup

   cwd: /home/will/projects/gemini-tts-cli

 > /commit
```

- 宣告工具使用
  - 使用 YAML frontmatter 語法
  - **allowed-tools**
    - 指令可使用的工具清單
  - **description**
    - 指令的簡短描述

- 執行命令的宣告語法

  **!`git status`**

# 自訂斜線命令 - 使用 @ 參考指定檔案

**檔案參考**

使用 `@` 前綴將檔案內容包含在指令中，以 **參考檔案** 。

例如：

```
# Reference a specific file
Review the implementation in @src/utils/helpers.js

# Reference multiple files
Compare @src/old-version.js with @src/new-version.js
```

# 自訂命令會送給 Claude 判斷該呼叫什麼工具

```
> /fix-issue is running… 38

● I'll help you fix issue #38. Let me first fetch the details of this issue to understand what needs to be fixed.

  Bash(gh issue view 38)
  └ Running…

 Bash command

   gh issue view 38
   Get details of issue #38

 Do you want to proceed?
 ❯ 1. Yes
   2. Yes, and don't ask again for gh issue view commands in /home/will/projects/gemini-tts-cli
   3. No, and tell Claude what to do differently (esc)
```

Hooks

# 使用 Hooks 擴充 Claude Code 行為

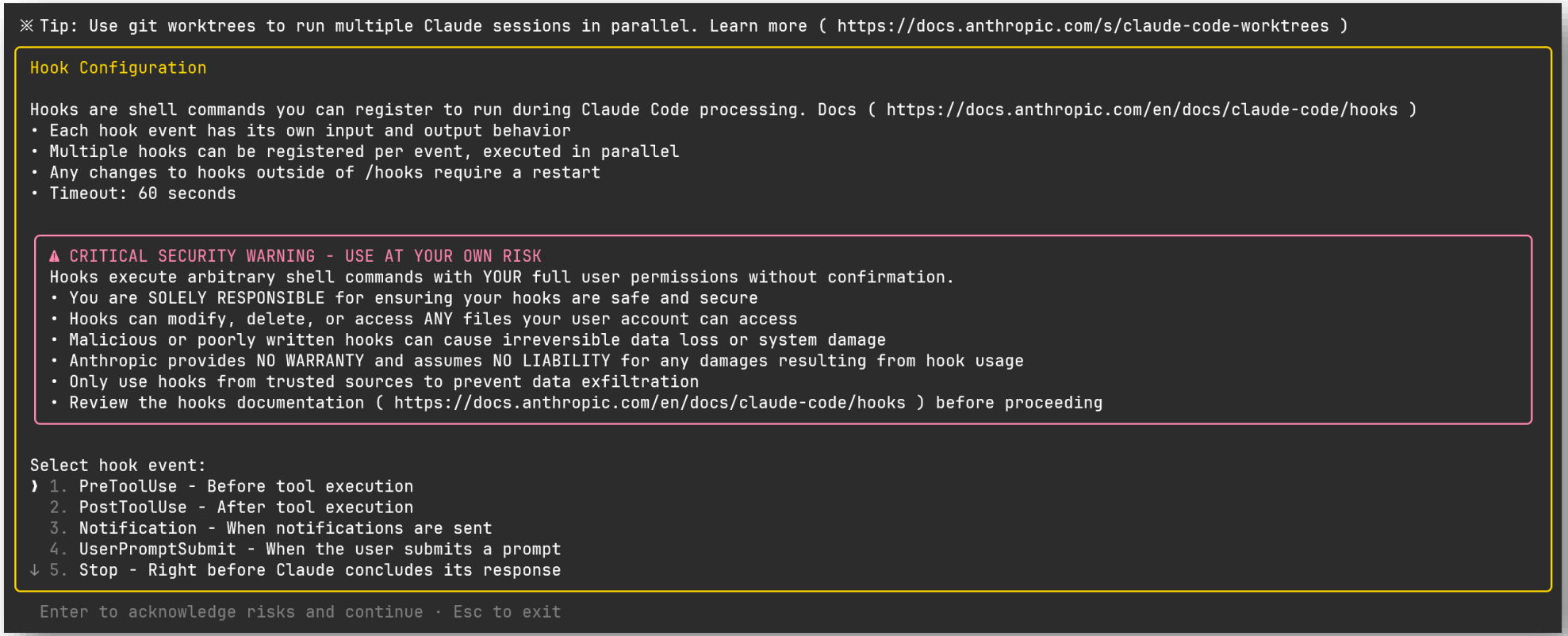
# 使用 Hooks 擴充 Claude Code 行為

- Hooks 會在 Claude Code 生命週期的不同階段執行
- Hooks 提供對 Claude Code 行為的確定性控制,
  - Claude Code 會確保特定動作**一定會執行**，而**不是依賴 LLM 來選擇執行它們**！（重要）
- Hooks 的範例用途
  - **執行通知**　：客製化當 Claude Code 等待您的輸入或執行權限時的通知方式。
  - **自動格式化**：每次檔案編輯後，在 .ts 檔案上執行 prettier，在 .go 檔案上執行 gofmt 等。
  - **執行記錄**　：追蹤和計算所有已執行的指令，以用於合規性或偵錯。
  - **意見回饋**　：當 Claude Code 產生的程式碼不符合您的程式碼庫慣例時，提供自動化意見回饋。
  - **自訂權限**　：封鎖對生產環境檔案或敏感目錄的修改。
  - **如果把這些用途寫在「提示」裡，會有一定的機率「不會執行」！**

# Hook 事件

- Claude Code 提供數個 hook 事件，會在工作流程的不同階段執行
  - `PreToolUse`
    - 在工具呼叫前執行（可封鎖工具呼叫）
  - `PostToolUse`
    - 在工具呼叫完成之後執行
  - `Notification`
    - 在 Claude Code 發送通知時執行（會延遲幾秒才通知，如果你在終端機就不會發通知）
  - `Stop`
    - 當 Claude Code 完成回應時執行
  - `SubagentStop`
    - 在子代理任務完成時執行
- 每個事件都會接收不同的資料，並能以不同的方式控制 Claude 的行為。

# 使用 /hooks 快速開啟設定畫面




- 超大的風險自負提示，大家注意看！ 😅

# /hooks 設定範例：Notification

- `Notification`
  - `Matcher`
    - `""`
  - `Command`
    - `paplay /home/will/.claude/Notification.wav`

- **相關說明**
  - **paplay** 命令用來在 WSL 2 播放聲音，工具來自於 PulseAudio 套件！
  - WSL 2 的 Ubuntu 有 PulseAudio 這個套件，可以直接安裝
    - apt install -y pulseaudio
  - 語音自動生成可以用我的 GeminiTtsCli 工具
    - dotnet tool install -g GeminiTtsCli

# /hooks 設定範例： PreToolUse/PostToolUse

- PreToolUse
  - Matcher
    - Bash
  - Command
    - `jq '.' >> ~/.claude/PreToolUse-logs.jsonl`
    - `jq -r '"\(.tool_input.command) - \(.tool_input.description // "No description")"' >> ~/.claude/bash-command-log.txt`
- PostToolUse
  - Matcher
    - `Edit|MultiEdit|Write`
  - Command
    - `jq -r '.tool_input.file_path' | while read -r f; do case "$f" in *.js) npx prettier --write "$f";; esac; done`

Bug #3179: 目前官方文件幾乎全是錯誤範例！😅

# 直接編輯設定檔 JSON 設定 Hooks

- Hooks 設定檔具層次結構，通常位於
  - `.claude/settings.json` 專案設定
  - `~/.claude/settings.json` 使用者全域設定

- 手動編輯的彈性比較大，但是容易改錯！

# 自訂斜線指令和 Hooks 的心得分享

- Claude Code 的互動方式透過自訂斜線命令與 Hooks 非常強大
  - 它讓開發流程變得既高階又低階
  - 高階在於開發者可以**抽象出複雜流程**為**簡單命令**
  - 低階在於可以掌控**底層行為**，在每個關鍵點掛載自己的邏輯
  - 這讓 Claude Code 成為一個**可塑性極強**的開發助手，能夠迎合各種開發風格與 "vibe" (無論是**愛聊天**的**程式小白**，還是偏好**嚴格規範自動化流程**的專業開發者)

Memory

# 掌控 Claude Code 的記憶能力

# 認識 CLAUDE.md 上下文檔案

- Claude Code 提供三種上下文檔案位置

| 記憶體類型 | 位置 | 目的 | 使用案例範例 |
| --- | --- | --- | --- |
| **專案記憶** | `./CLAUDE.md` | 專案的團隊共用指示 | 專案架構、編碼標準、常用工作流程 |
| **使用者記憶** | `~/.claude/CLAUDE.md` | 所有專案的個人偏好設定 | 程式碼風格偏好設定、個人工具捷徑 |
| **專案記憶 (本機)** | `./CLAUDE.local.md` | 個人專案特定偏好設定 | *(已棄用)* |

# 整合不同來源的上下文檔案

- 理解 CLAUDE.md imports 能力

```
 will …/pytest  main ?  v22.16.0  v3.10.12  18:12
❯ bat CLAUDE.md
       File: CLAUDE.md
   1   See @README for project overview and @package.json for available npm commands for this project.
   2
   3   # 載入額外專案指示
   4   - git workflow @docs/git-instructions.md
   5
   6   # 載入個人化的指令檔
   7   - @~/.claude/my-project-instructions.md
   8
   9   # 載入為了避免潛在的碰撞，匯入不會在 markdown 程式碼範圍和程式碼區塊中進行評估
  10   This code span will not be treated as an import: `@anthropic-ai/claude-code`
  11
  12   # GitHub Copilot Instructions and GEMINI.md
  13   See @copilot-instructions.md and @GEMINI.md for more context for this project.
```

Tool use

# 深入理解 Claude Code 工具調用

# 窺探 Claude Code 的內部工具

- **Claude Code 並沒有文件說明有哪些工具可用，只有 Claude API 文件有寫**
  - 我有點不太明白為什麼會這樣？不開源就算了，有哪些工具也不寫！😈
  - 💡 我的猜測：**他們就是要你想清楚，工具不是隨便讓你亂執行的，有危險！**
- **常見的內部工具**
  - Task, TodoWrite, Read, Edit, MultiEdit, Write
  - Bash, Glob, Grep, LS, exit_plan_mode
  - NotebookRead, NotebookEdit
  - WebFetch, WebSearch
  - MCP
- **神奇的命令**

```
claude -p 'Generate a hello world python app' \
  --output-format stream-json --verbose | jq
```

```
will …/pytest main ? v22.16.0 v3.10.12 18:19
❯ claude  -p 'Generate a hello world python app' --output-format stream-json --verbose | jq
{
  "type": "system",
  "subtype": "init",
  "cwd": "/home/will/projects/pytest",
  "session_id": "3de260ba-f2eb-44e6-8cbd-3e2cdf4a8a30",
  "tools": [
    "Task",
    "Bash",
    "Glob",
    "Grep",
    "LS",
    "exit_plan_mode",
    "Read",
    "Edit",
    "MultiEdit",
    "Write",
    "NotebookRead",
    "NotebookEdit",
    "WebFetch",
    "TodoWrite",
    "WebSearch"
  ],
  "mcp_servers": [],
  "model": "claude-sonnet-4-20250514",
  "permissionMode": "default",
  "apiKeySource": "none"
}
```

# Claude 支援兩種工具

- **用戶端工具 (Client tools)**
  - 在你的電腦執行的工具，包括：
    - 您自行建立和實作的使用者定義自訂工具
    - 需要用戶端實作的 Anthropic 定義工具，例如 computer use 與 text editor

- **伺服器工具 (Server tools)**
  - 在 Anthropic 伺服器上執行的工具，例如 `WebFetch` 或 `WebSearch` 工具
  - 這些工具必須在 API 要求中指定，但不需要您自行實作

# 你必須明確定義允許的工具清單

- 分享我的 .claude/settings.json 版本
  - 建議不要一口氣全部加入

- 以下是 Claude Code 的 YOLO mode 參數

```
claude --dangerously-skip-permissions
```

File: .claude/settings.json

```
{
  "permissions": {
    "allow": [
      "Edit",
      "WebFetch",
      "Bash(dotnet:*)",
      "Bash(ls:*)",
      "Bash(chmod:*)",
      "Bash(git status)",
      "Bash(git diff:*)",
      "Bash(git log:*)",
      "Bash(git add:*)",
      "Bash(git rm:*)",
      "Bash(git commit:*)",
      "Bash(git push:*)",
      "Bash(git push)",
      "Bash(git tag:*)",
      "Bash(find:*)",
      "Bash(sed:*)",
      "Bash(rg:*)",
      "Bash(npx husky:*)",
      "Bash(npm init:*)",
      "Bash(npm test:*)",
      "Bash(npm install:*)",
      "Bash(npm run:*)",
      "Bash(npm run quality:fix:*)",
      "Bash(gh repo create:*)",
      "Bash(gh repo edit:*)",
      "Bash(gh pr:*)"
    ],
    "deny": []
  },
  "hooks": {
```

💡 YOLO = You Only Live Once = 你只活一次

# 設定 MCP 伺服器

- **新增 MCP stdio 伺服器**

```
# Basic syntax
claude mcp add <name> <command> [args...]
# Example: Adding a local server
claude mcp add my-server -e API_KEY=123 -- /path/to/server arg1 arg2
```

- **新增 MCP SSE 伺服器**

```
# Basic syntax
claude mcp add --transport sse <name> <url>
# Example: Adding an SSE server
claude mcp add --transport sse sse-server https://example.com/sse-endpoint
# Example: Adding an SSE server with custom headers
claude mcp add --transport sse api-server https://api.example.com/mcp --header "X-API-Key: your-key"
```

# 設定 MCP 伺服器

- **新增 MCP HTTP 伺服器**

```
# Basic syntax
claude mcp add --transport http <name> <url>
# Example: Adding a streamable HTTP server
claude mcp add --transport http http-server https://example.com/mcp
# Example: Adding an HTTP server with authentication header
claude mcp add --transport http secure-server https://api.example.com/mcp --header
"Authorization: Bearer your-token"
```

- **管理 MCP 伺服器**

```
# List all configured servers
claude mcp list
# Get details for a specific server
claude mcp get my-server
# Remove a server
claude mcp remove my-server
```

# 加入 GitHub MCP Server

- **取得 GitHub PAT 金鑰**
  - https://github.com/settings/tokens
- **設定環境變數**
  - `export GITHUB_PERSONAL_ACCESS_TOKEN=your-github-pat`
- **加入 MCP 伺服器**
  - 
```
claude mcp add --transport http  github https://api.githubcopilot.com/mcp/ \
  --header "Authorization: Bearer $GITHUB_PERSONAL_ACCESS_TOKEN" > /dev/null
```
- **設定檔位址 (內含敏感金鑰資訊)**
  - `~/.claude.json`
- **移除 MCP 伺服器**
  - `claude mcp remove github`

Security

# 理解 Claude Code 的安全機制

# 權限式架構

- Claude Code 使用分層權限系統來平衡功能與安全性

| 工具類型 | 範例 | 需要核准 | 「是，不再詢問」行為 |
| --- | --- | --- | --- |
| Read-only | File reads, LS, Grep | No | N/A |
| Bash Commands | **Shell execution** | **Yes** | Permanently per project directory and command |
| File Modification | **Edit/write files** | **Yes** | Until session end |

# 內建的保護機制

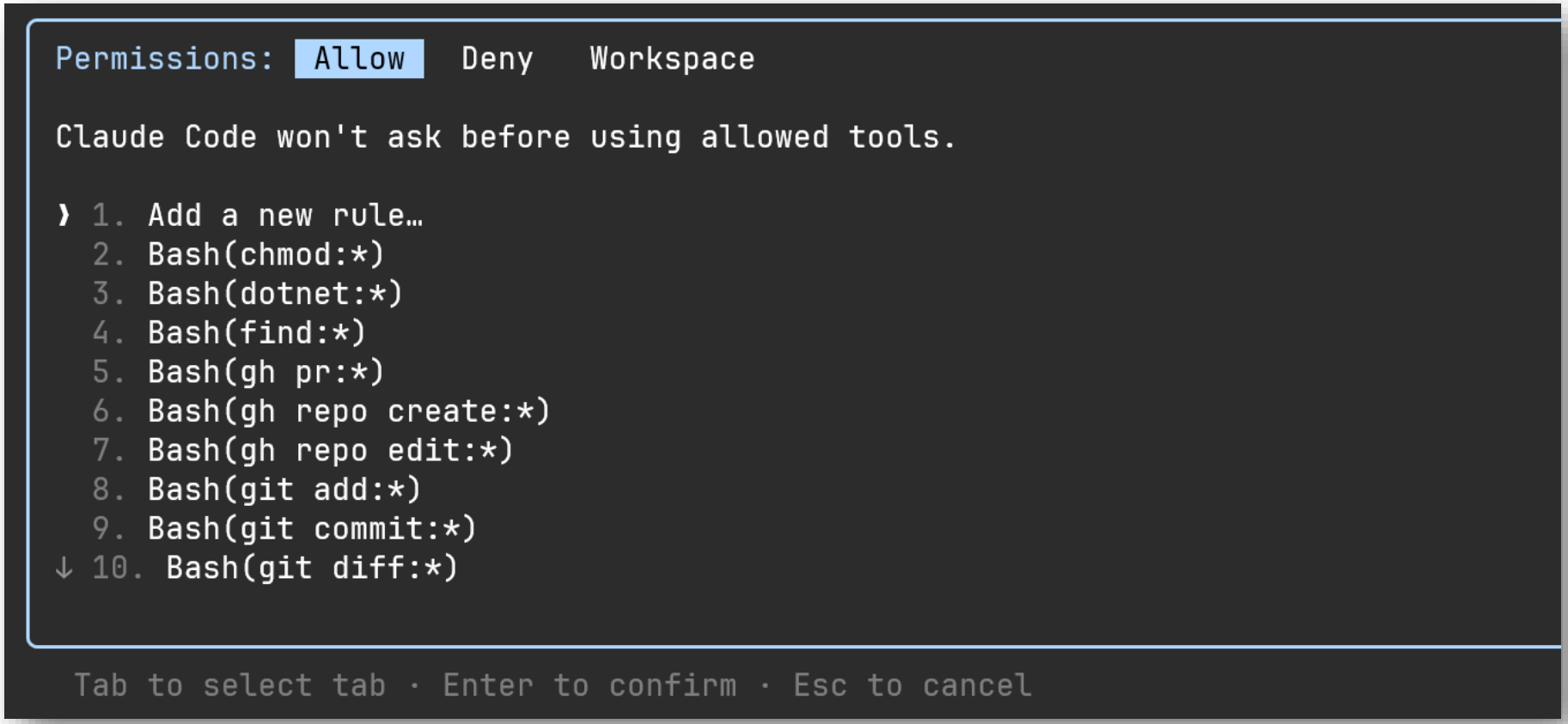
- 你可以在設定中指定**權限模式** (Permission modes)

| 模式 | 說明 |
| --- | --- |
| default | 標準行為 - 首次使用每項工具時提示權限 |
| acceptEdits | 自動接受該工作階段的檔案編輯權限 |
| plan | 規劃模式 - Claude 可以分析檔案或執行指令，但不能修改 (唯讀) |
| bypassPermissions | 略過所有權限提示（需要安全環境 - 請參閱下方警告） |

# 以不同的權限模式啟動 Claude Code

- `claude --permission-mode default`
- `claude --permission-mode acceptEdits`
- `claude --permission-mode plan`
- `claude --permission-mode bypassPermissions`

# 使用 /permissions 斜線命令進行設定



# 工具特定權限規則

- **Bash**
  - `Bash(npm run build)` **精準比對** Bash 指令 `npm run build` ， 多一點點參數都不行！
  - `Bash(npm run test:*)` 比對以 `npm run test` 開頭的 Bash 指令（可以識別用 **&&** 串接的命令）
- **Read & Edit**（規則均遵循 gitignore 規範）
  - `Edit(docs/**)` 比對你專案中 `docs` 目錄內的編輯檔案
  - `Read(~/.zshrc)` 比對你家目錄的 `~/.zshrc` 檔案的讀取
  - `Edit(//tmp/scratch.txt)` 比對對 `/tmp/scratch.txt` 的編輯
- **WebFetch**
  - `WebFetch(domain:example.com)` 比對 `example.com` 域名的擷取請求
- **MCP**
  - `mcp__puppeteer` 比對由 `puppeteer` 伺服器提供的任何工具（名稱在 Claude Code 中設定）
  - `mcp__puppeteer__puppeteer_navigate` 符合 'puppeteer' 伺服器提供的 `'puppeteer_navigate'` 工具

# 認識潛在的攻擊管道

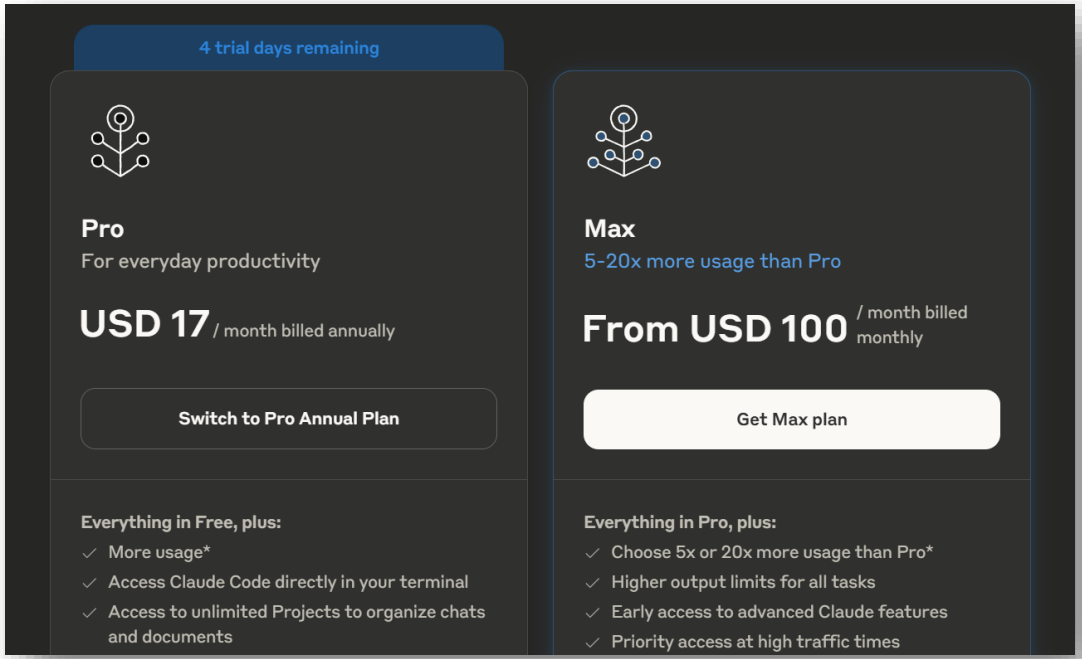
- **Bash**
  - 任何惡意的提示都有可能觸發 Bash 工具執行！🔥
- **Hooks**
  - 執行 Hooks 不會做任何權限檢查，這點要非常小心！🔥
- **你可以透過 hook 進行額外的權限控制**
  - Claude Code Hooks 提供了一種註冊自訂 Shell 命令的方式，以便在執行階段進行權限評估。
  - 當 Claude Code 發出工具呼叫時，PreToolUse hook 會在權限系統執行前運行，而 Hook 的輸出可以決定是否**允許**或**拒絕**該工具呼叫，**以取代權限系統**。

Cost

# 詳細估算 Claude Code 的導入成本

# 如何計算成本

- **購買 Claude 的 Pro 或 Max 訂閱**
  - 固定成本，較無心理壓力
  - Pro $17/mo
  - Max $100/mo
  - Max $200/mo
- **Anthropic Console**
  - 無免費額度，適合理性人士使用
  - 直接以 Tokens 用量計費
  - 至少 $5 起跳！
  - https://console.anthropic.com/settings/billing



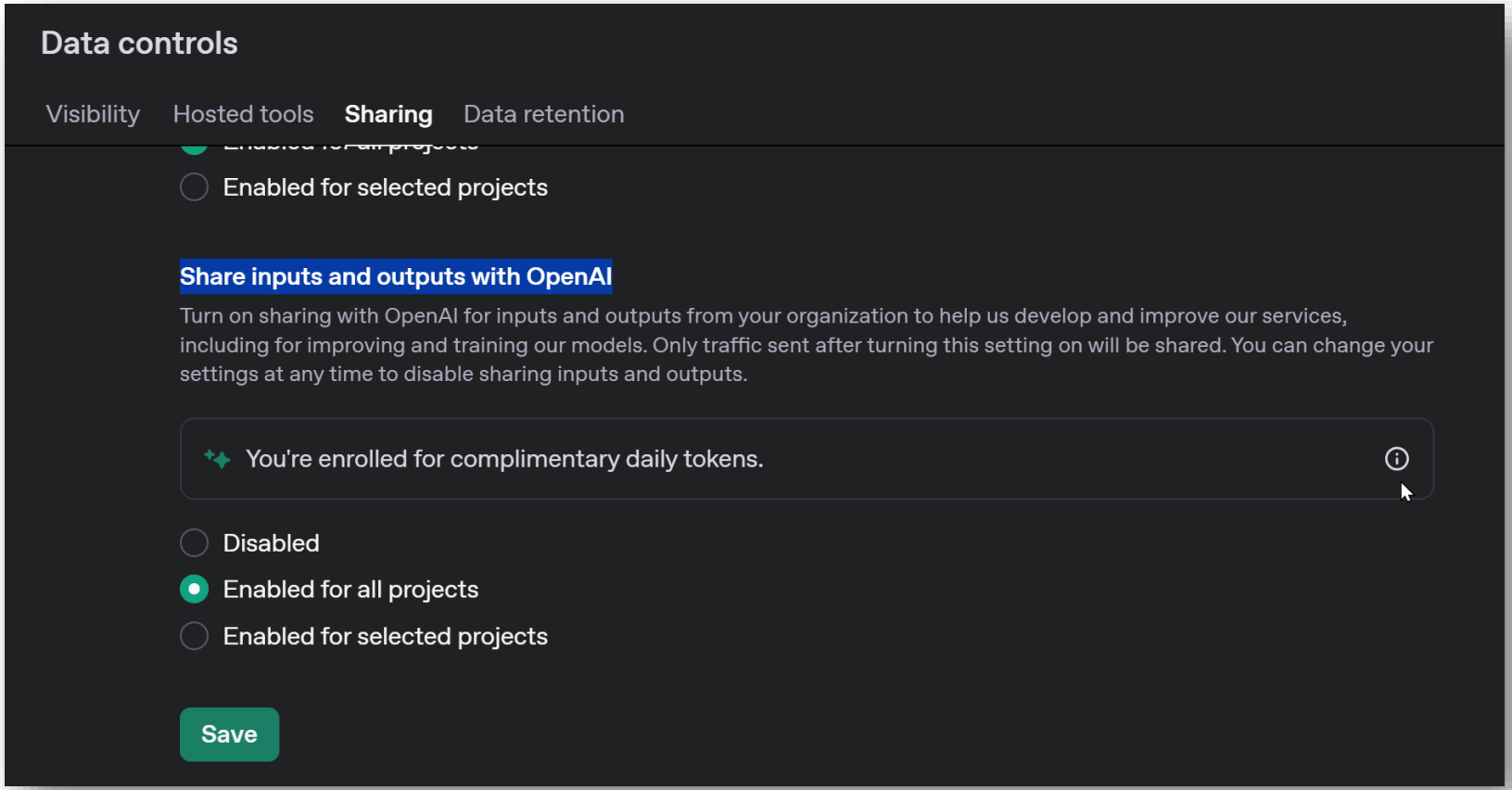
# Claude API 的 Token 價格 (USD)

| 模型 | 輸入 token 價格 | 輸出 token 價格 |
|---|---|---|
| Claude Opus 4 | $15/百萬 token | $75/百萬 token |
| Claude Sonnet 4 | $3/百萬 token | $15/百萬 token |
| Claude 3.7 Sonnet/3.5 Sonnet | $3/百萬 token | $15/百萬 token |
| Claude 3.5 Haiku | $0.80/百萬 token | $4/百萬 token |
| Claude 3 Opus | $15/百萬 token | $75/百萬 token |
| Claude 3 Haiku | $0.25/百萬 token | $1.25/百萬 token |

# 省錢的技巧分享

- 直接買最貴的 **Claude Max** 方案，就不會感覺貴了（咦？）😄
- 現在購買 **Claude Pro** 可以享受 7 天免費試用，還有前三個月 50% OFF 折扣！😍
- 不要什麼都叫 Claude Code 做事，用 GitHub Copilot 或 AIChat 會更省錢！
- 多多利用 Gemini CLI 免費額度
  - 只要是免費的額度，Google 都會拿你的 Code 去訓練
  - 雖然免費的很香，但是用到 Gemini 2.5 Flash 真的會想哭
- 使用 Codex CLI 可以享受 OpenAI 提供的「資料共享」方案
  - 每日最多 25 萬個 token，適用於 gpt-4.5-preview、gpt-4.1、gpt-4o、o1 和 o3
  - 每日最多 250 萬個 token，適用於 gpt-4.1-mini、gpt-4.1-nano、gpt-4o-mini、o1-mini、o3-mini、o4-mini 和 codex-mini-latest
- 新創公司可以加入 AWS Activate Startup 計畫，可享用 **$1000 美元**的免費額度！
- 記得清除 `ANTHROPIC_API_KEY` 環境變數：**`unset ANTHROPIC_API_KEY`**

# Share inputs and outputs with OpenAI




https://platform.openai.com/settings/organization/data-controls/sharing

# Share inputs and outputs with OpenAI

Links

# 相關連結

# 好用連結

- Claude Code 官方文件 (千萬不要看中文版)
- Claude Code: Best practices for agentic coding
- 與 Claude Code 相關的環境變數
- Claude Code for VSCode - Visual Studio Marketplace
- Anthropic Courses
  - Claude Code in Action
- YouTube
  - How I use Claude Code (+ my best tips)
- SuperClaude v3 🚀

# 聯絡資訊

The Will Will Web

網路世界的學習心得與技術分享

http://blog.miniasp.com/

Facebook

Will 保哥的技術交流中心

http://www.facebook.com/will.fans

Twitter

https://twitter.com/Will_Huang

多奇·教育訓練

# THANK YOU!

Q&A