那些跟 skills 有關的兩三事



Maple Kuo

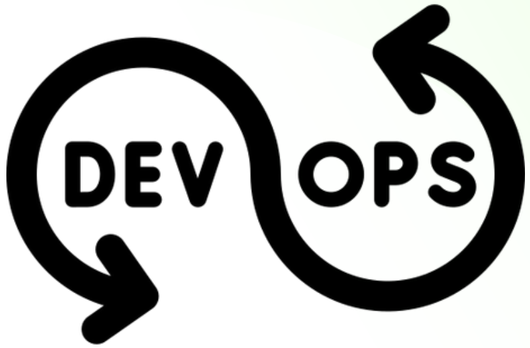
2026/04/15

skills !



從我的第一個 skills 開始聊起

最初的痛點



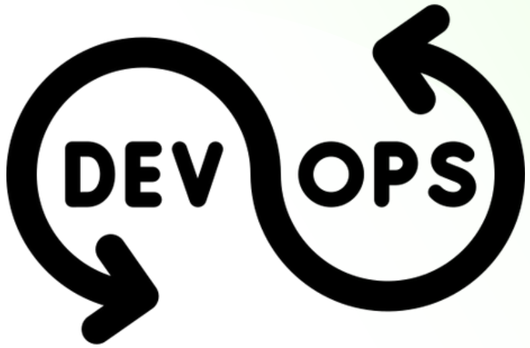
每次 push 之後要自己看 CI

最初的痛點



壞了叫 Claude 去修復

最初的痛點



Push 之後繼續看 CI

最初的痛點



於是我的第一個 skills 誕生了

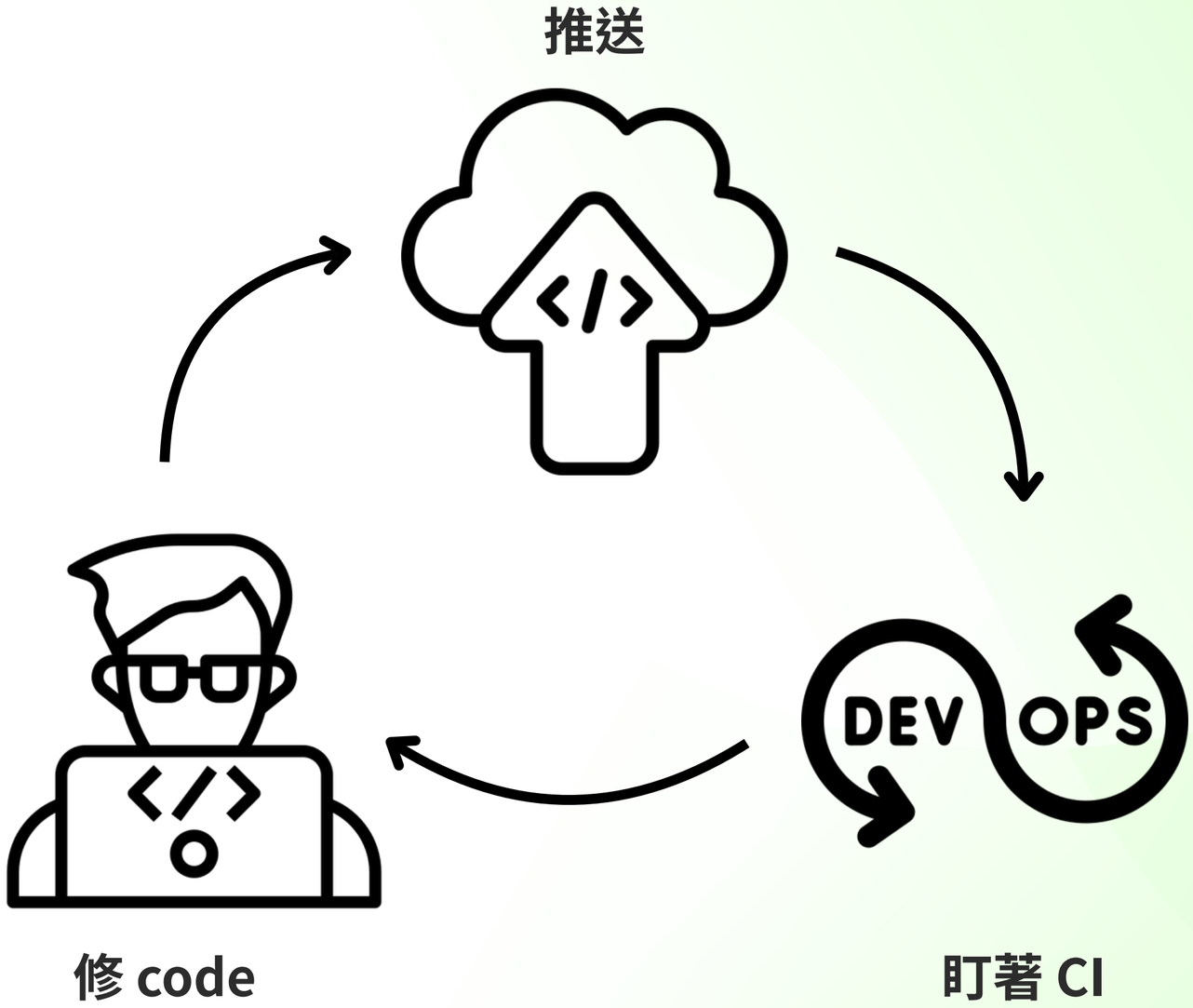
Push 之後繼續看 CI

肯定有辦法自動化！

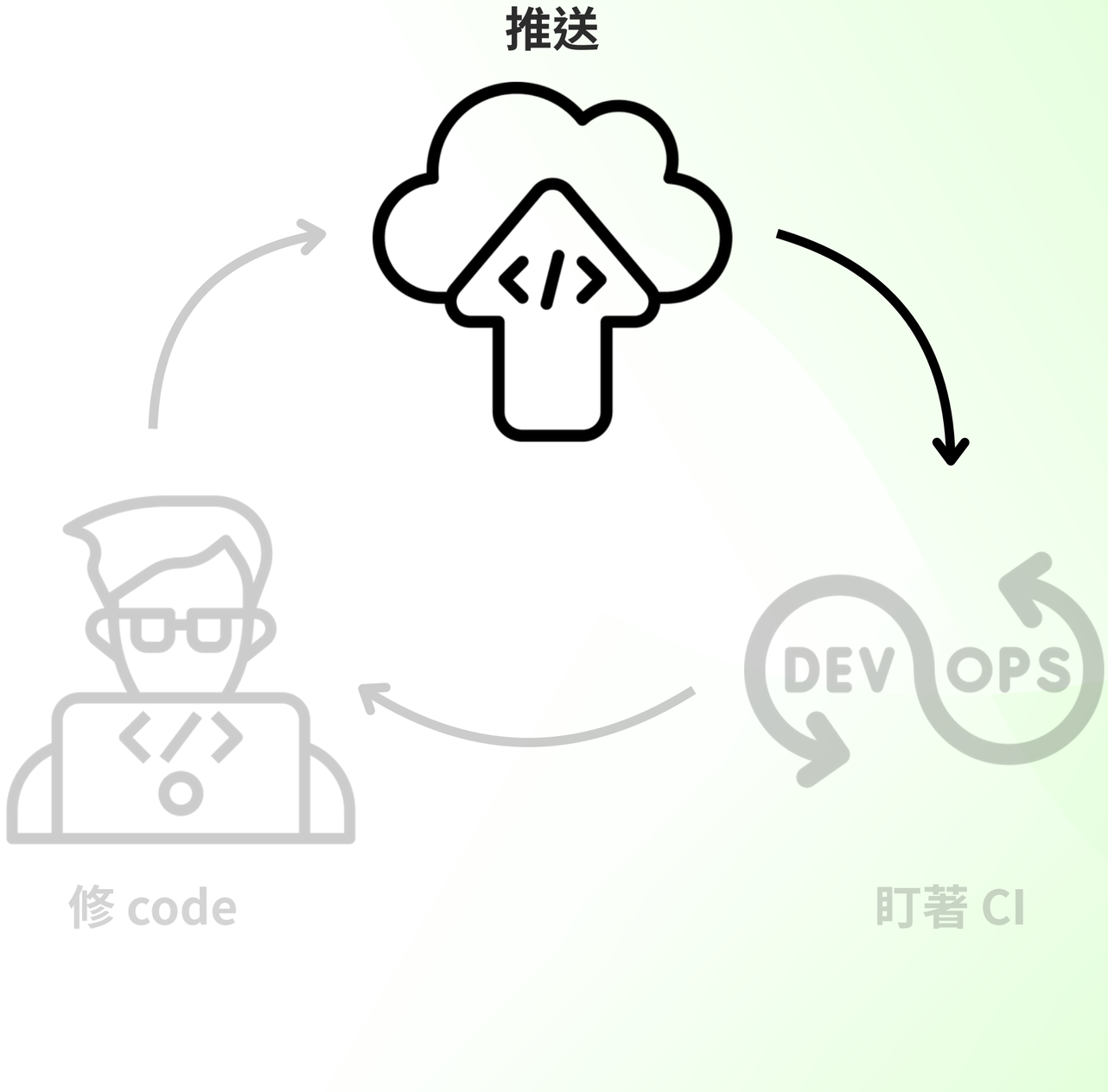


自己盯著 CI 自己修吧

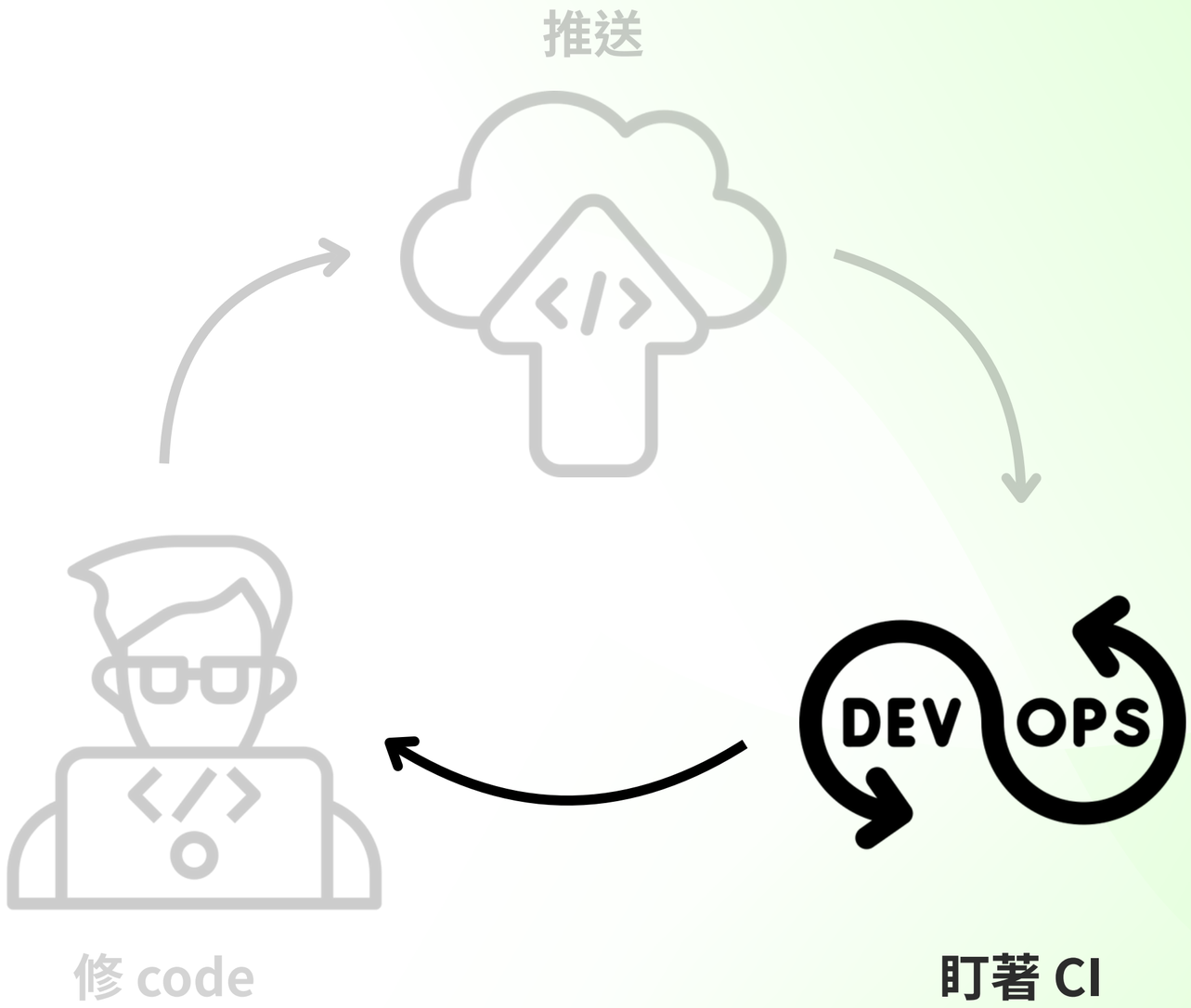
流程轉變



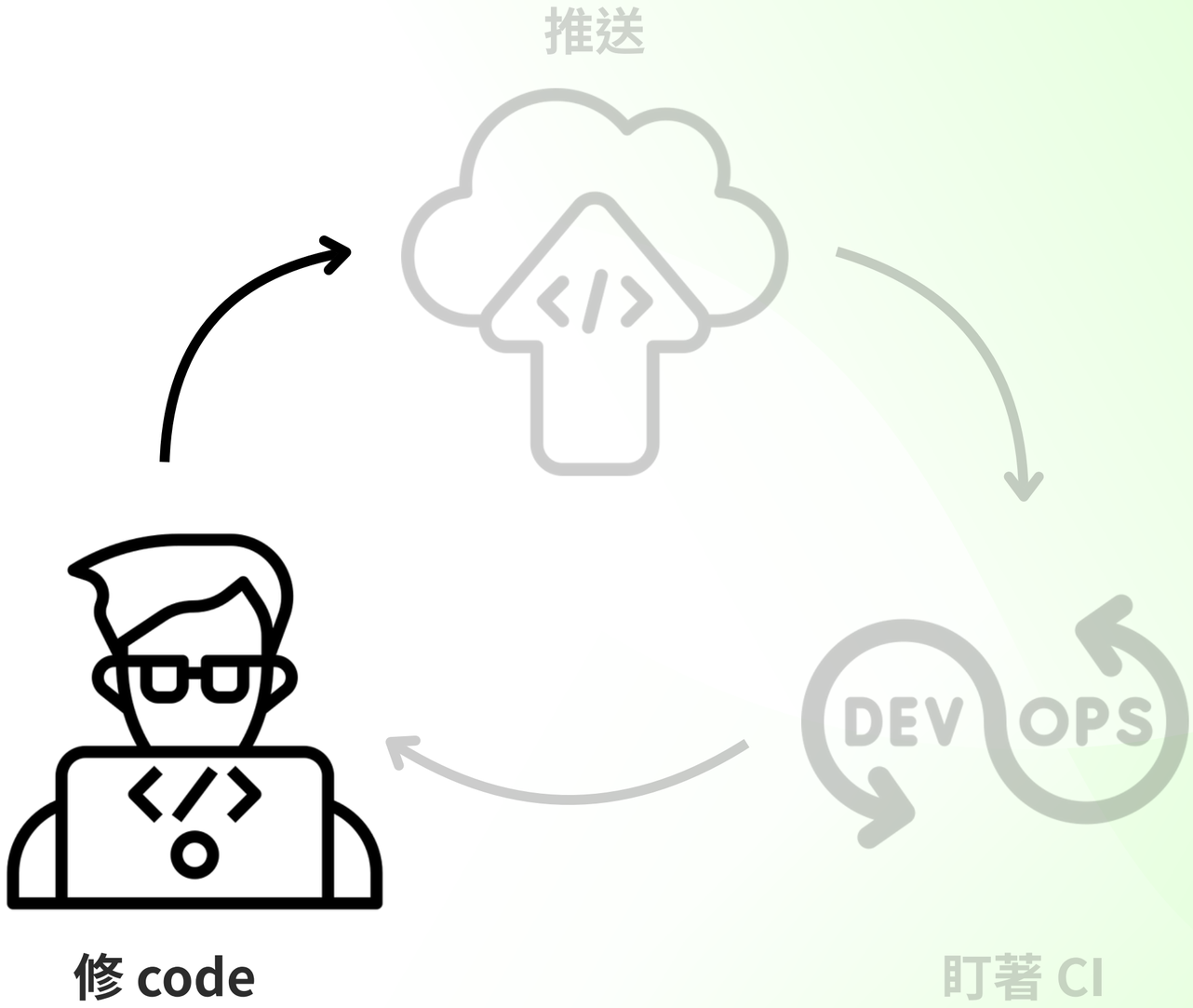
流程轉變



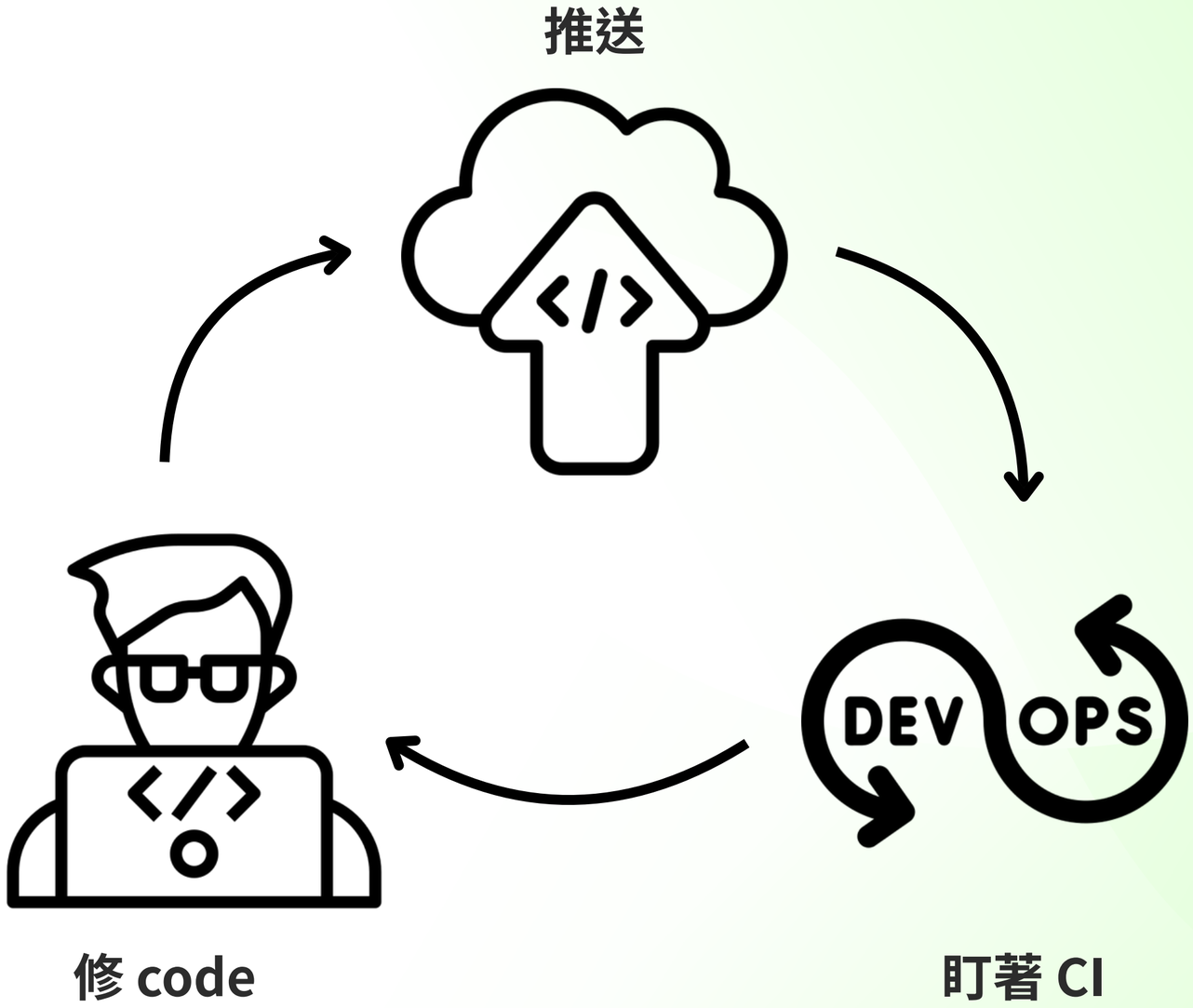
流程轉變



流程轉變



流程轉變



流程轉變



/pr-to-staging

流程轉變

```
---
name: pr-to-staging
description:  建立 PR 到 staging。分析變更、生成 PR 內容、推送並建立 PR。
---

# PR to Staging Branch Workflow

完成功能開發後，執行完整檢查流程並建立 PR 到 staging 分支。

## 執行流程

### Step 1: 確認當前分支狀態

```bash
# 檢查當前分支
git branch --show-current

# 確認不是在 main/dev/staging 上直接開發
# 如果是，提醒用戶應該先建立 feature branch
```

檢查點：
- [ ] 當前分支不是 `main`、`dev`、`staging`
- [ ] 有未推送的 commits 或本地變更

Step 2: 執行所有檢查

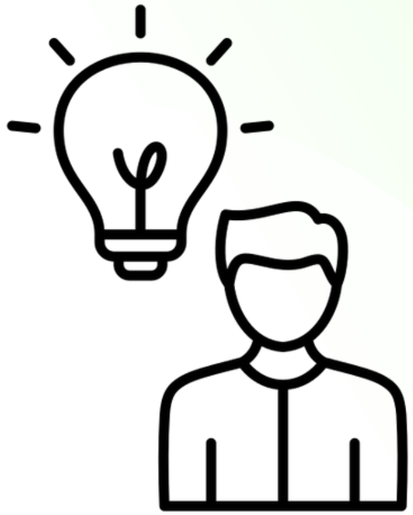
依序執行以下檢查，任一失敗則停止流程：
```

玩過 skills 了



重複性解決了但 token 燒不完

來玩 git worktree !



來試試看 worktree 吧

前情提要：什麼是 worktree

詳細比較

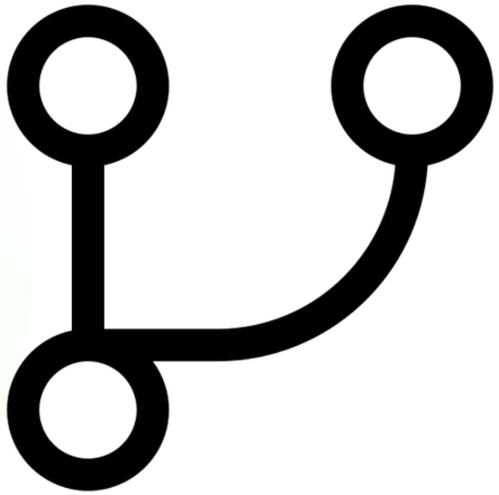
從「一個 repo 同時只能一個分支」
變成「一個 repo 可以同時跑多個分支」

舉例來說 ...



我想「同時」開發兩個功能

只有兩個選擇

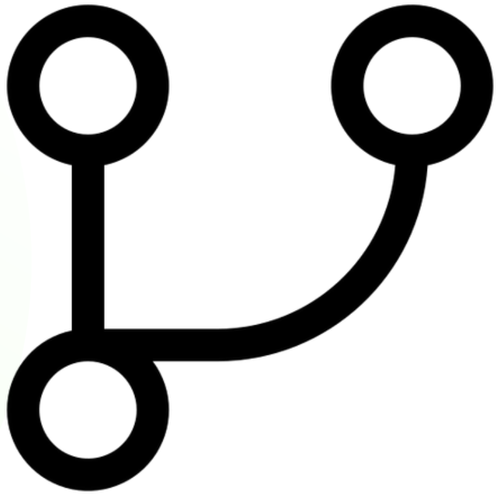


同一個 branch



Clone 兩個資料夾

只有兩個選擇



同一個 branch



Scope 變大
耦合性高

Clone 兩個資料夾

只有兩個選擇

多一個平行
多 clone 一次

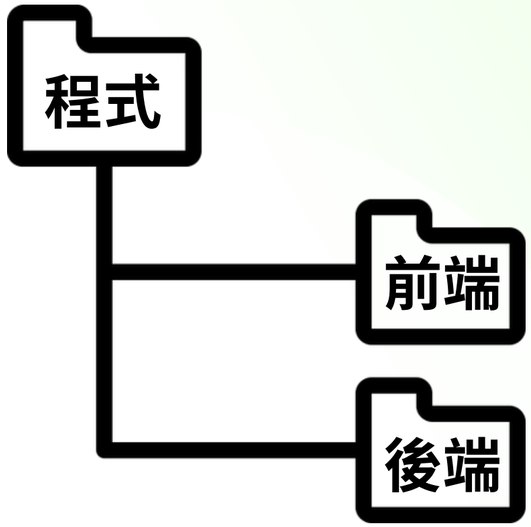
同一個 branch





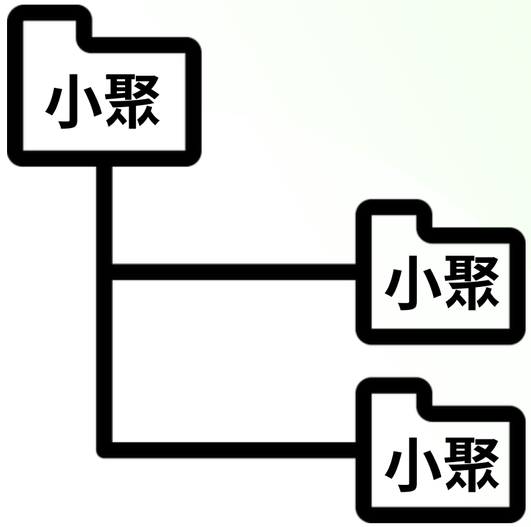
Clone 兩個資料夾

舉例來說 ...



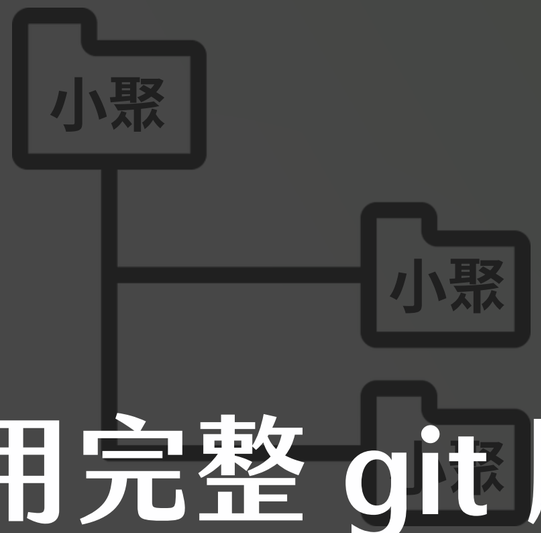
worktree就像前後端放在一個資料夾

舉例來說 ...



只是從前後端變成同一個 repo

舉例來說 ...



但共用完整 git 歷史！

只是從前後端變成同一個 repo

前情提要：什麼是 worktree

詳細比較

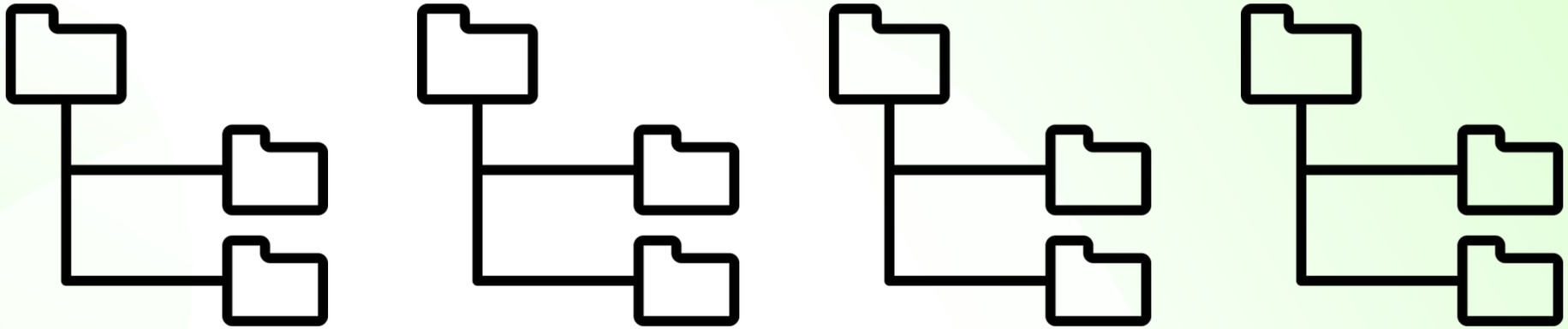
從「一個 repo 同時只能一個分支」
變成「一個 repo 可以同時跑多個分支」

來玩 git worktree !



/wt

來玩 git worktree !





得到很多 worktree 了

來玩 git worktree !

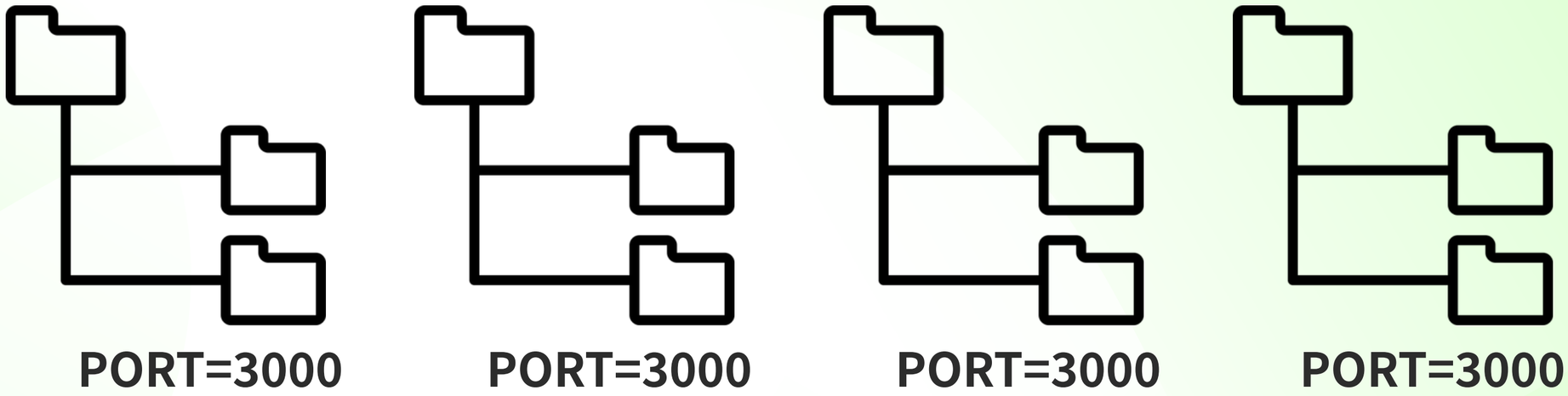


再來要人工驗收！



得到很多 worktree 了

來玩 git worktree !





來玩 git worktree !

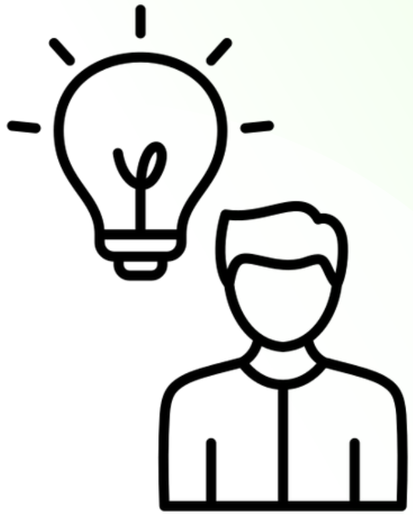
Port 衝突了，導致 local server 跑不起來

來玩 git worktree !

Port 衝突了，導致 local server 跑不起來

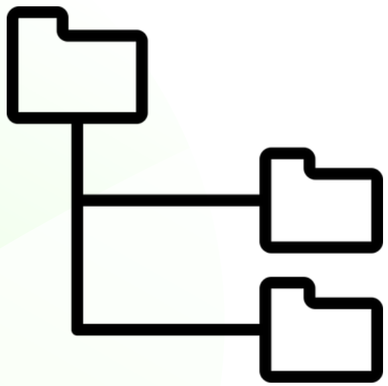
登入用的還是 Google OAuth

來玩 git worktree !

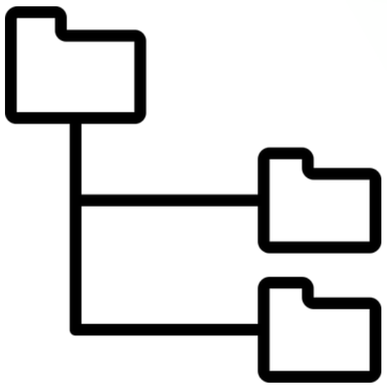


那不然做個註冊機制吧

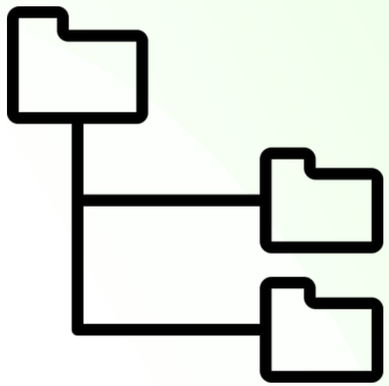
來玩 git worktree !



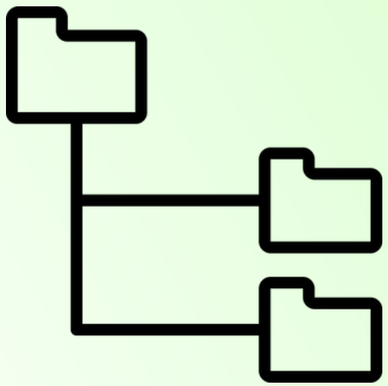
PORT=3000



PORT=3001



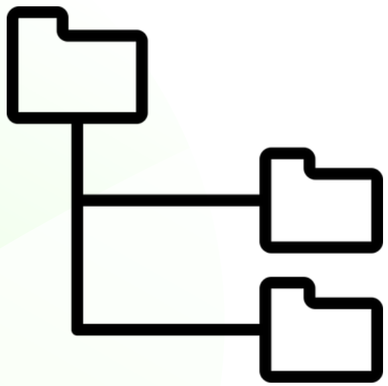
PORT=3002



PORT=3003

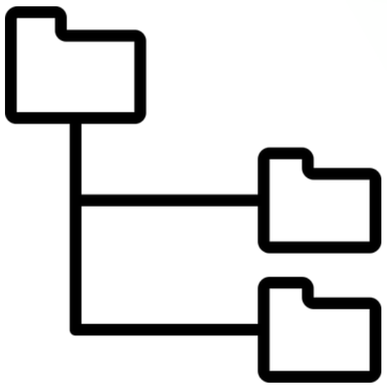


來玩 git worktree !



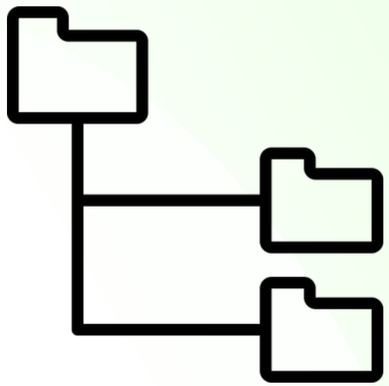
PORT=3000

改商業邏輯



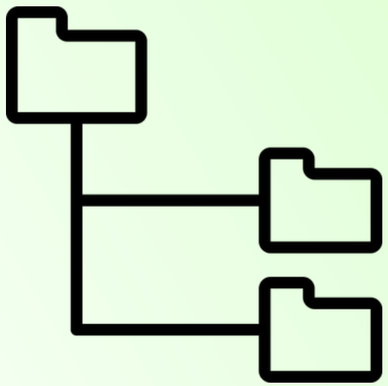
PORT=3001

改商業邏輯 + DB



PORT=3002

改商業邏輯 + DB

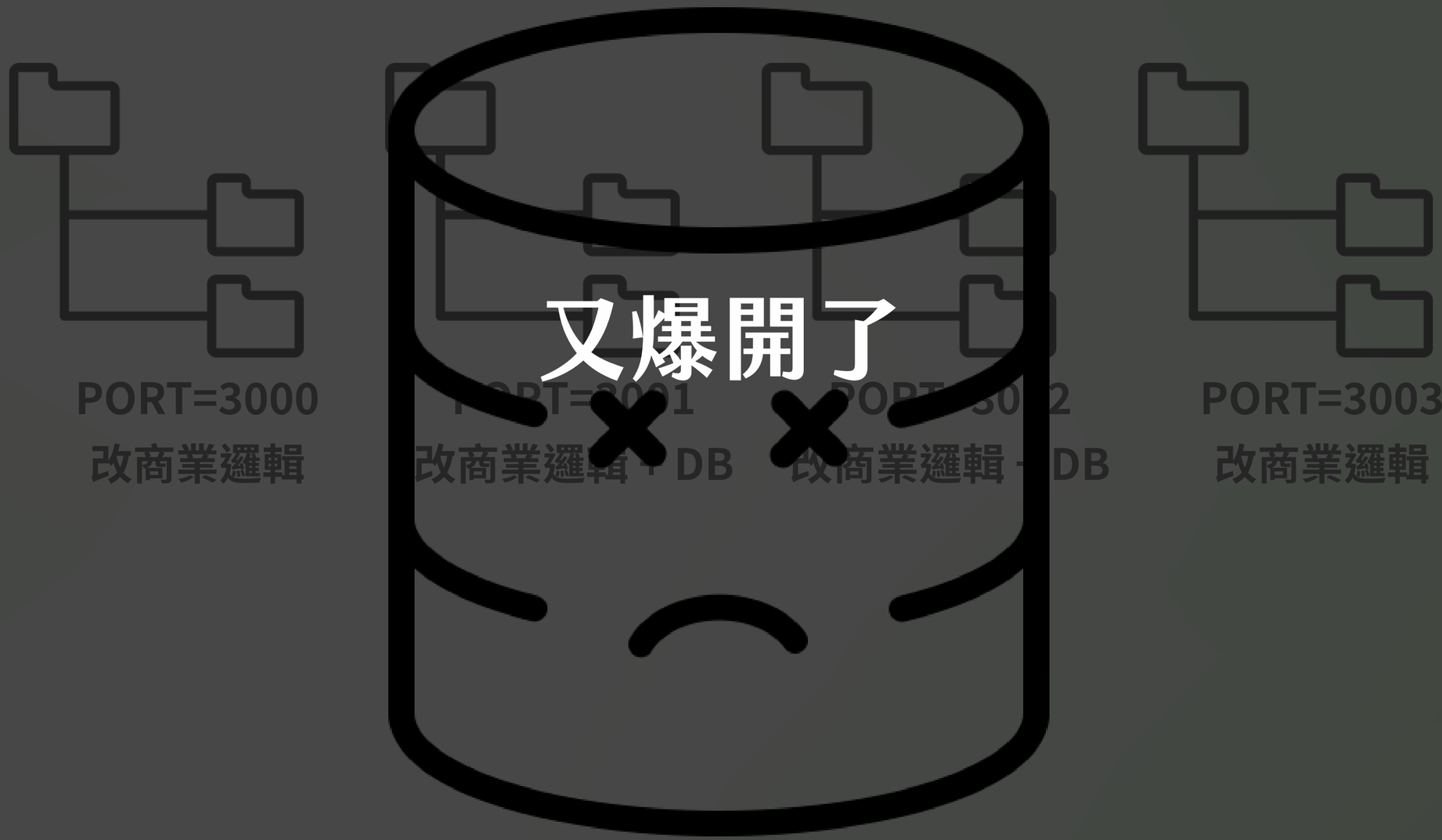


PORT=3003

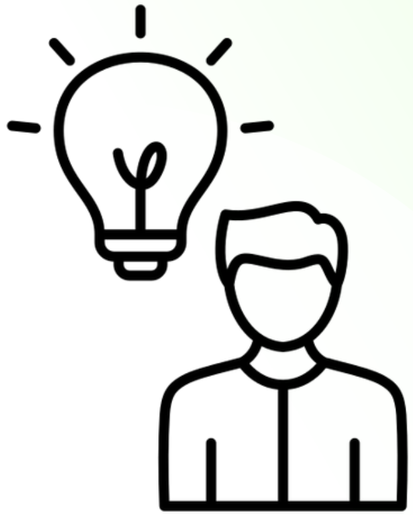
改商業邏輯



來玩 git worktree !

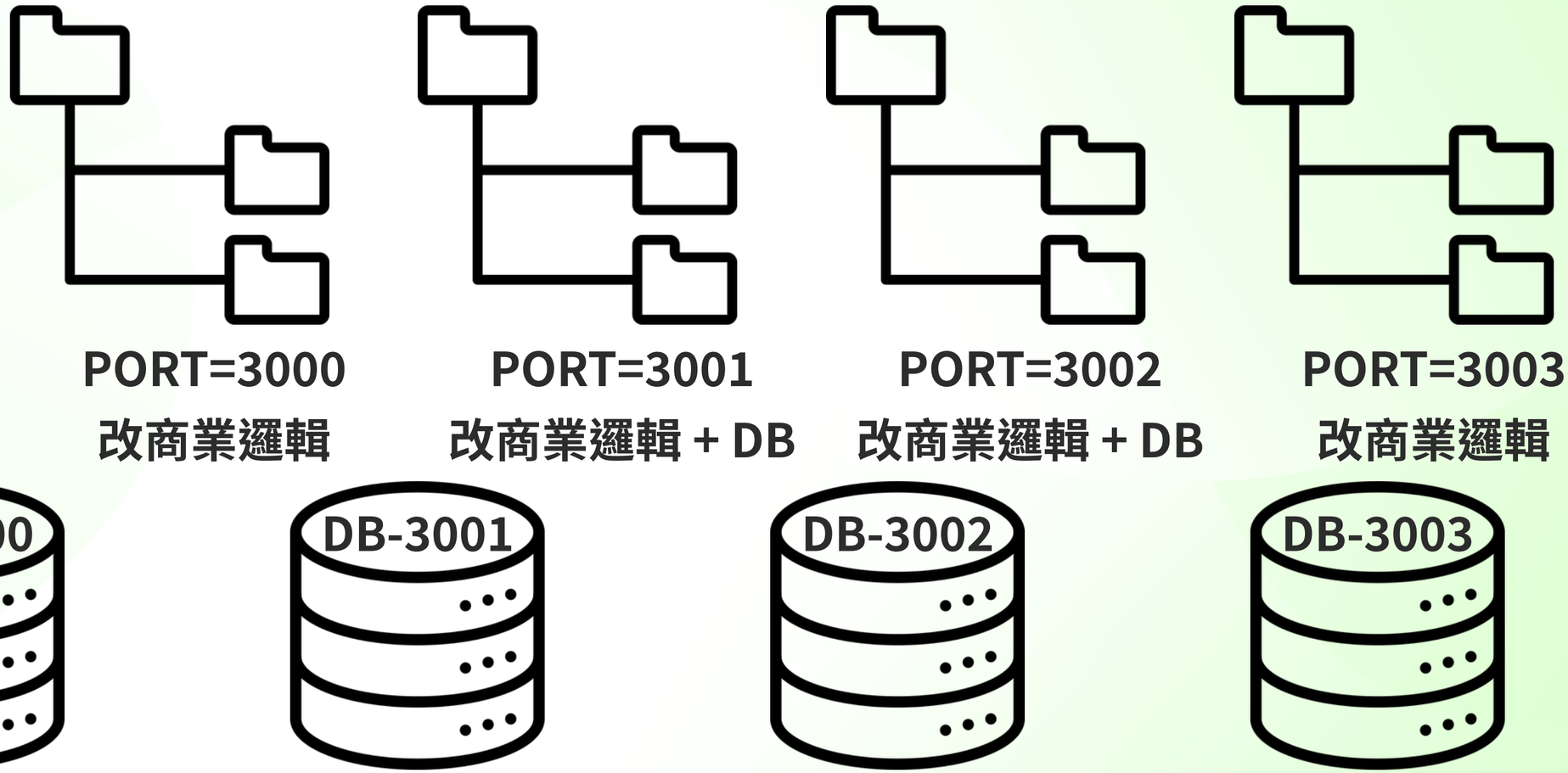


來玩 git worktree !



那再做個平行擴展吧

來玩 git worktree !



過程還有碰到一些問題 ...

- 剛開始的時候 local server 跑不起來，因為沒有 symlink .env

過程還有碰到一些問題 ...

- 剛開始的時候 local server 跑不起來，因為沒有 symlink .env
- 即便 DB 自動擴展，migration 還是沒有跑在對應的 DB

過程還有碰到一些問題 ...

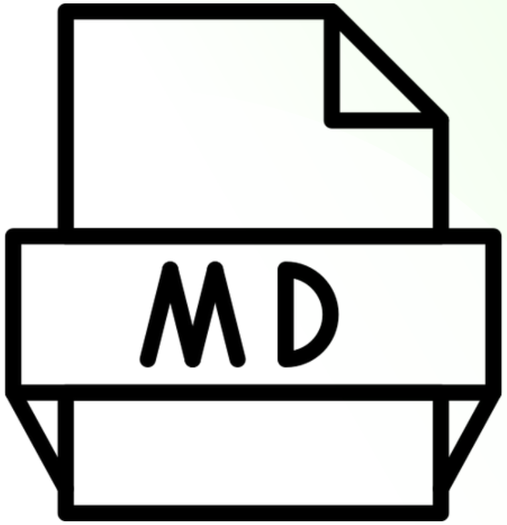
- 剛開始的時候 local server 跑不起來，因為沒有 symlink .env
- 即便 DB 自動擴展，migration 還是沒有跑在對應的 DB
- 不知道自己要在 worktree 上開發

skills?



skills 是什麼?

skills 是...



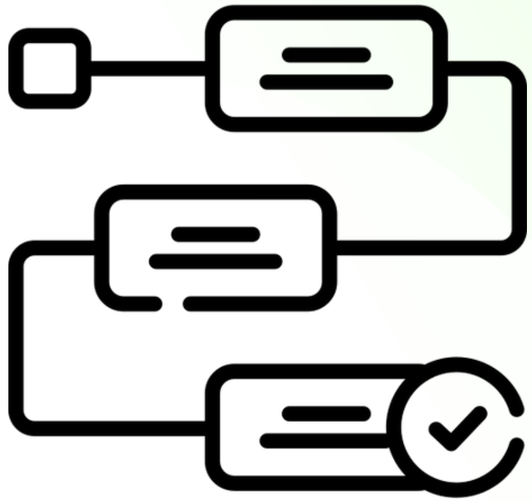
一系列 markdown 文件

skills 是...



重複指令的 shortcuts

skills 是...



別人寫好的 **workflow**

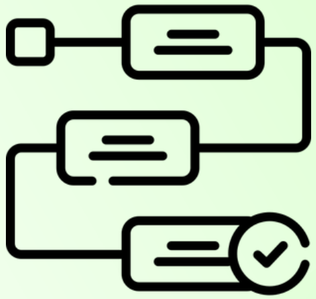
skills 是...



一系列 markdown 文件

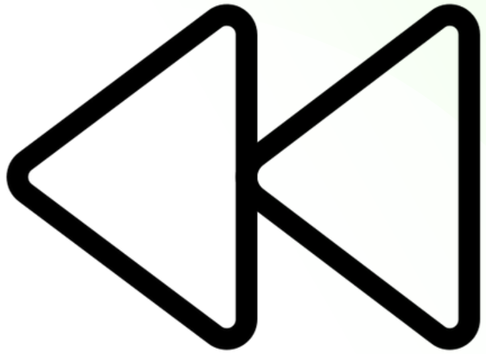


重複指令的 shortcuts



別人寫好的 workflow

時間倒轉之術

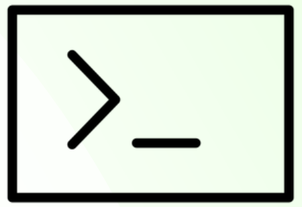


時間回到 skills 出現之前

Claude 很愛寫...



一堆 markdown 文件



一次性的 scripts

Claude 很愛寫...



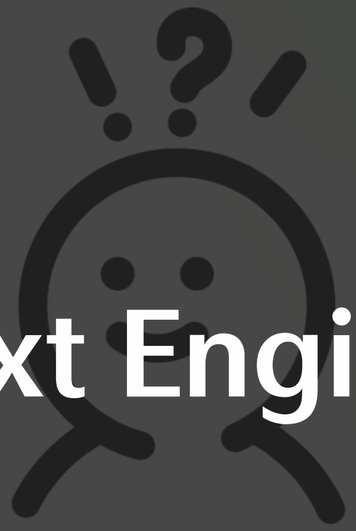
一堆 ots

skills?



skills 的本質到底是什麼?

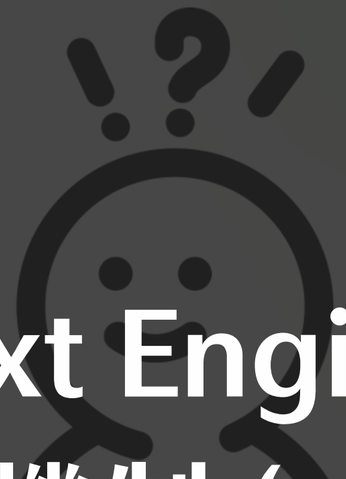
skills?



Context Engineering

skills 的本質到底是什麼?

skills?



Context Engineering
再加一點點的機制 (e.g, 漸進式揭露)
skills 的本質到底是什麼?

Context Enginnering?

- Context 終有限，LLM 不可能記住所有事情

Context Enginnering?

- Context 終有限，LLM 不可能記住所有事情
- LLM 具有隨機性，每次生成的 bash command 不盡相同

Context Enginnering?

- Context 終有限，LLM 不可能記住所有事情
- LLM 具有隨機性，每次生成的 bash command 不盡相同

因此 skills 誕生了！

skills = Context Engineering

要把 skills 玩好

=

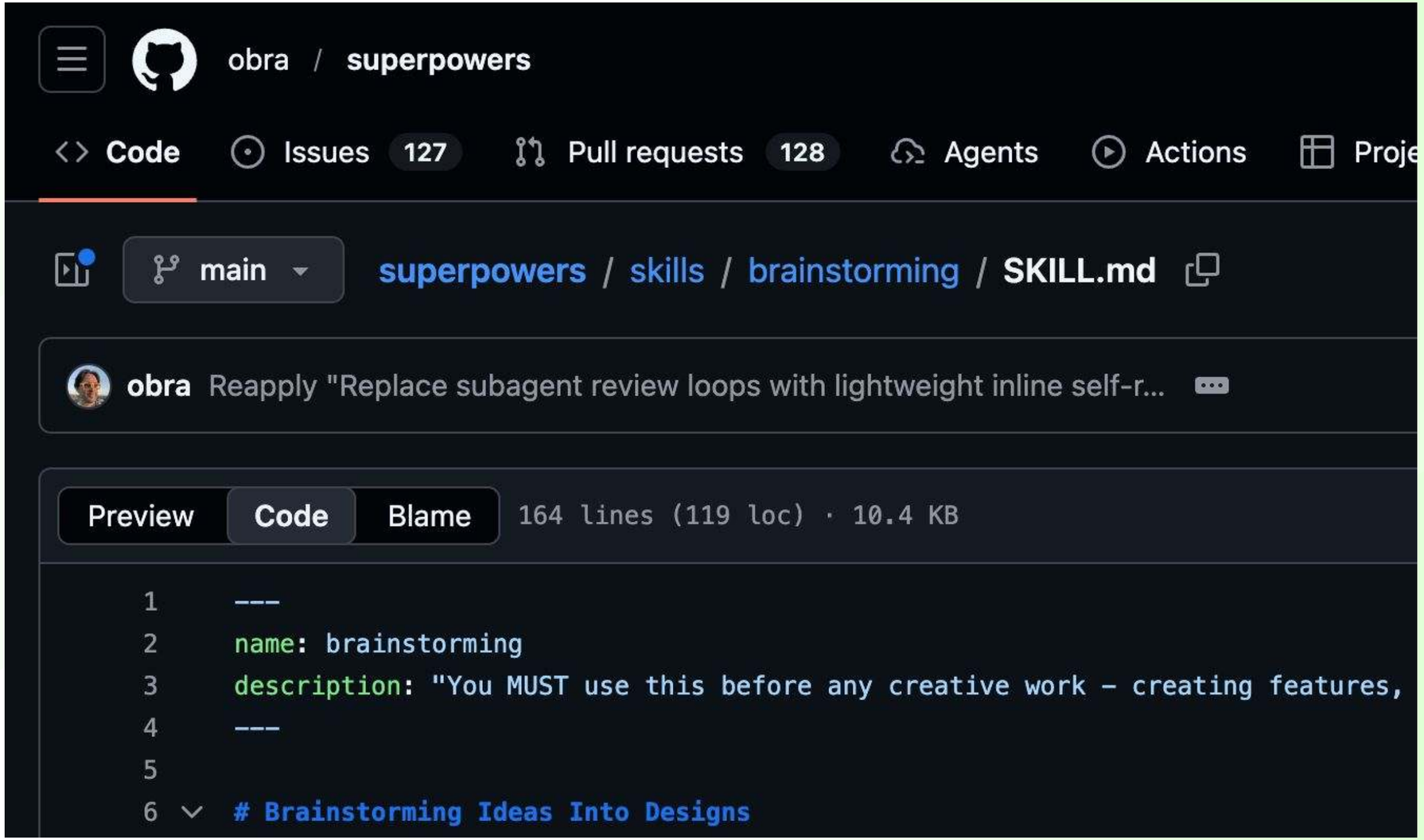
把 Context Engineering 玩好

skills = Context Engineering

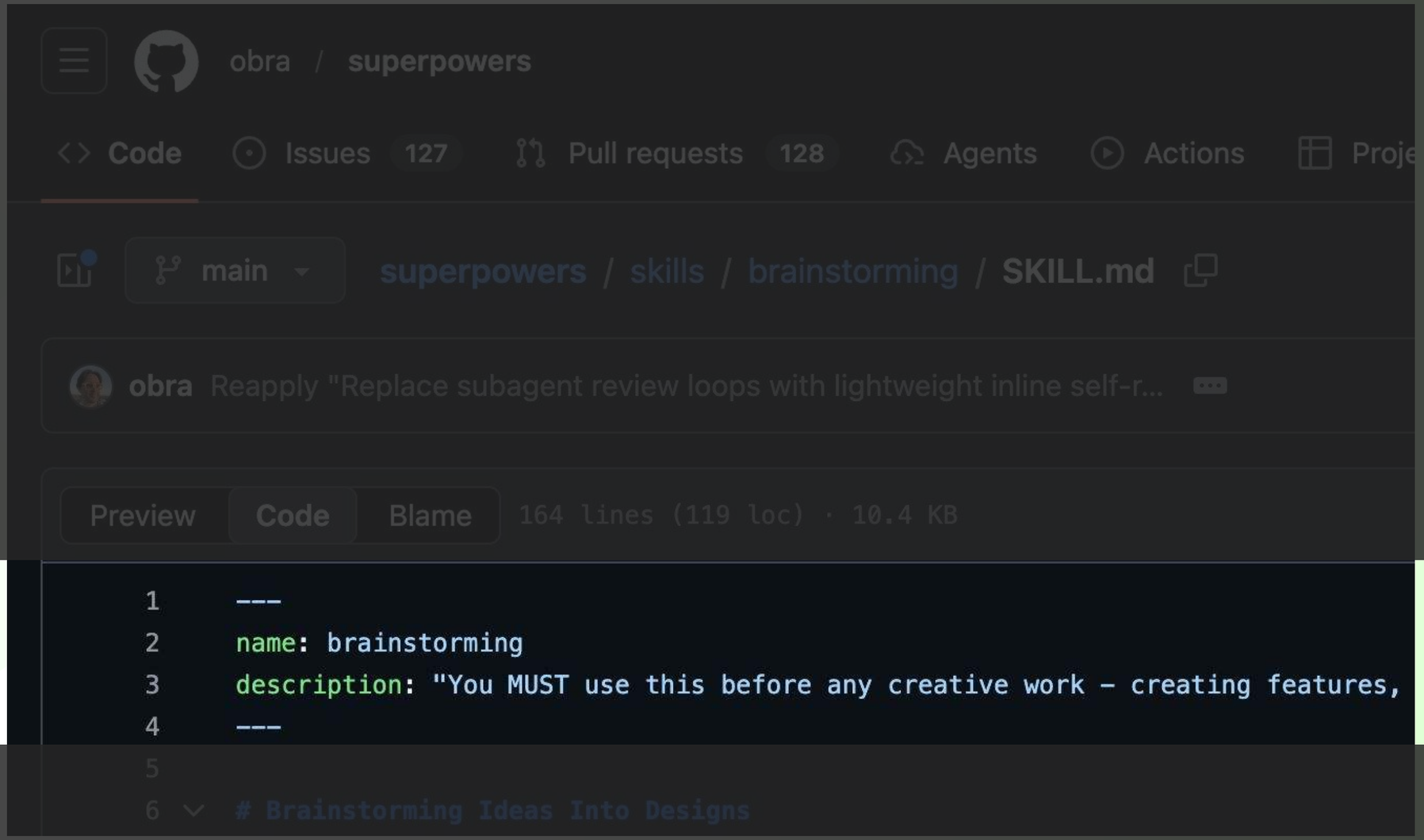


如何透過 skills 做到呢

先認識機制


obra / superpowers
Code
Issues 127
Pull requests 128
Agents
Actions
Proje
main
superpowers / skills / brainstorming / SKILL.md
obra Reapply "Replace subagent review loops with lightweight inline self-r...
Preview
Code
Blame
164 lines (119 loc) · 10.4 KB
1 ---
2 name: brainstorming
3 description: "You MUST use this before any creative work - creating features,
4 ---
5
6 # Brainstorming Ideas Into Designs

先認識機制


obra / superpowers
Code
Issues 127
Pull requests 128
Agents
Actions
main
superpowers / skills / brainstorming / SKILL.md
obra Reapply "Replace subagent review loops with lightweight inline self-r...
Preview
Code
Blame
164 lines (119 loc) · 10.4 KB
1 ---
2 name: brainstorming
3 description: "You MUST use this before any creative work - creating features,
4 ---
5
6 # Brainstorming Ideas Into Designs

但 skills 可以做更多事

```yaml
---
name: deep-code-review
description: 深度審查 PR 變更，產出結構化報告

disable-model-invocation: true
argument-hint: "[PR number]"
allowed-tools: Bash(gh *), Bash(git diff *), Read, Grep
paths: "src/**/*.ts"
user-invocable: true

context: fork
agent: Explore
model: sonnet
effort: high

hooks:
  PreToolUse:
    - matcher: Bash
      command: echo "Tool used: $TOOL_NAME"
shell: bash
---
```

但 skills 可以做更多事

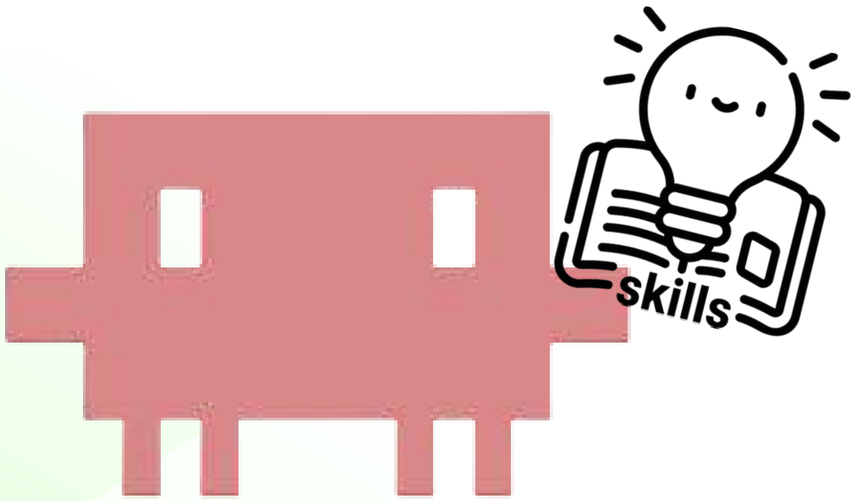
```
---
name: deep-code-review
description: 深度審查 PR 變更，產出結構化報告

disable-model-invocation: true
argument-hint: "[PR number]"
allowed-tools: Bash(gh *), Bash(git diff *), Read, Grep
paths: "src/**/*.ts"
user-invocable: true

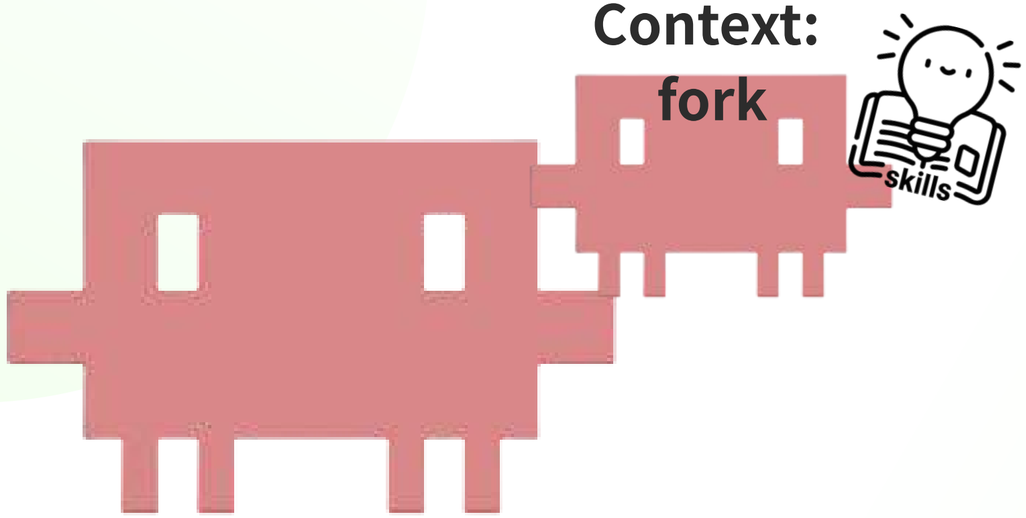
context: fork
agent: Explore
model: sonnet
effort: high

hooks:
  PreToolUse:
    - matcher: Bash
      command: echo "Tool used: $TOOL_NAME"
shell: bash
---
```

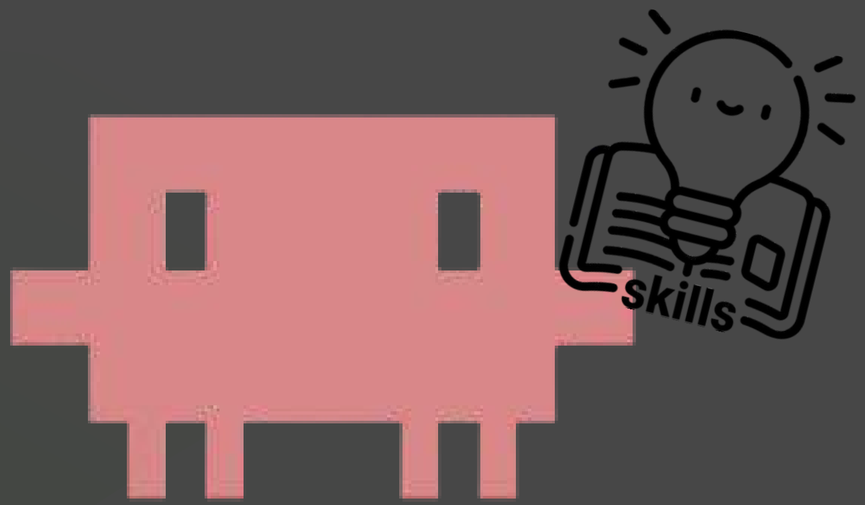
code-review: skills vs fork agent skills



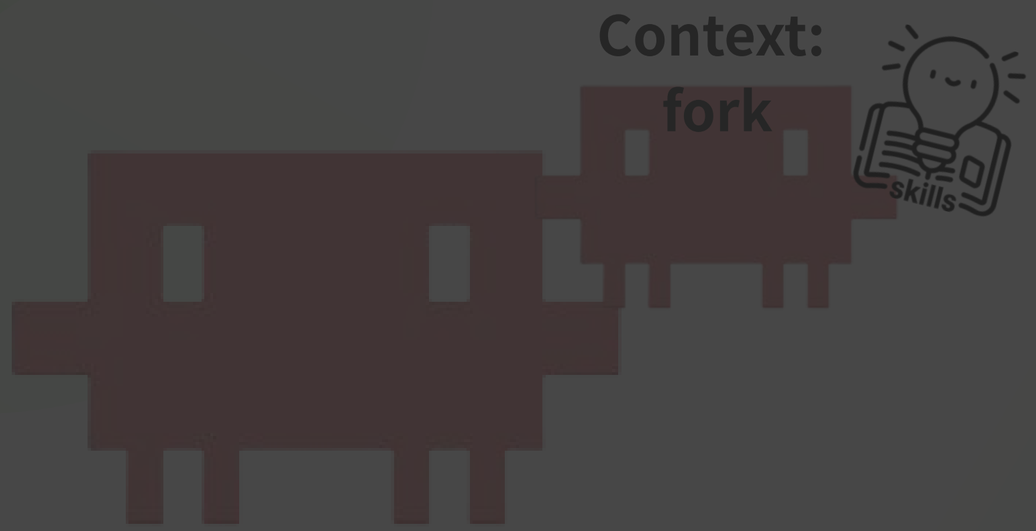
Context:
fork



code-review: skills vs fork agent skills

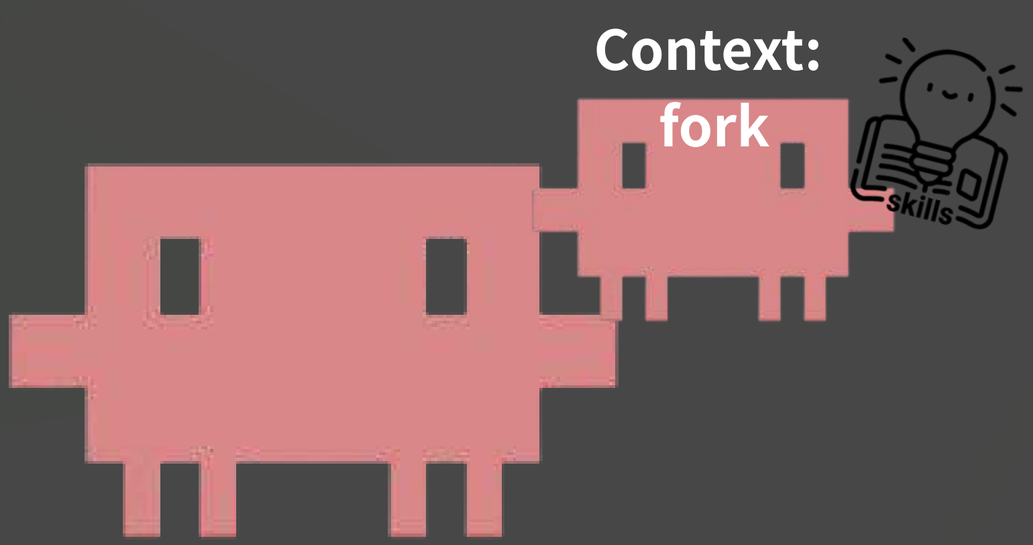


污染上下文



code-review: skills vs fork agent skills





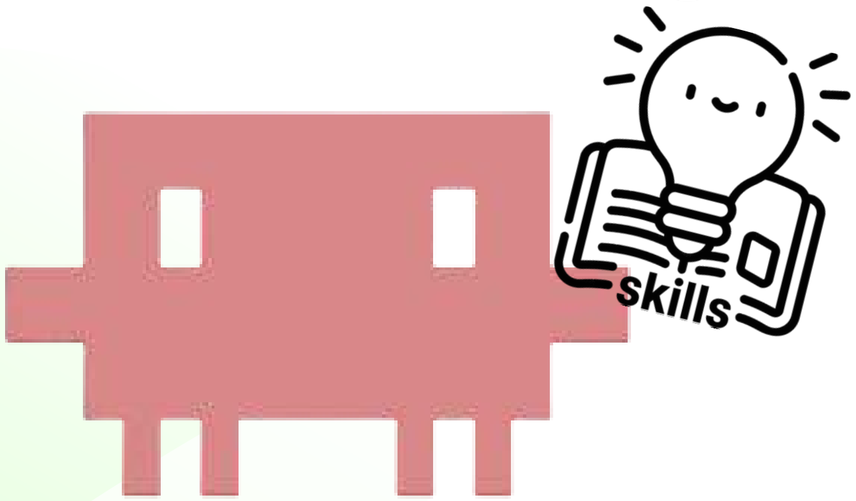
比較燒 token

skills vs fork agent skills

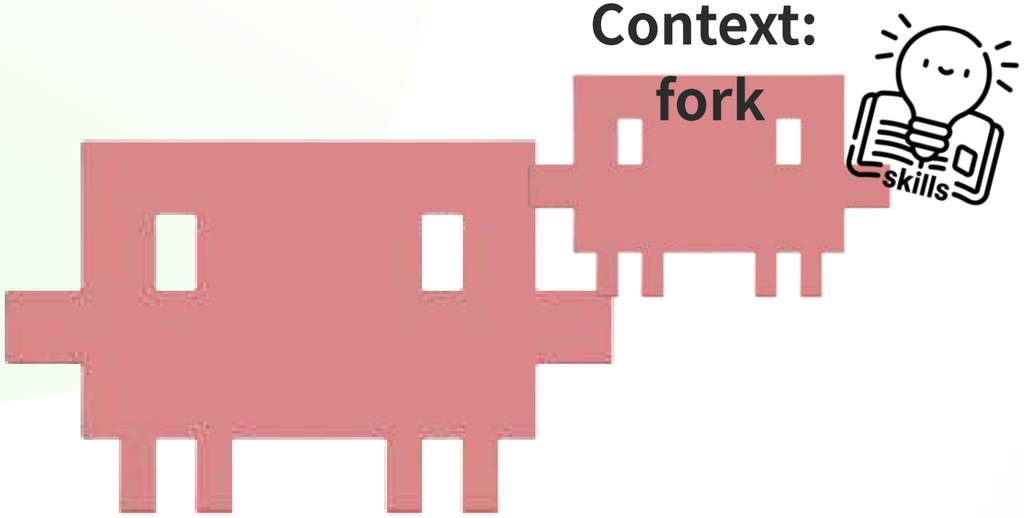


兩者使用情境的差異在哪

skills vs fork agent skills



- 常用 prompt 的快捷鍵
- 能力擴充 e.g., /wt, /reddit-search
- 按需載入的背景知識擴充



- 通用性的重複任務
- 不在意過程只在意結果

sub-agent vs fork agent skills



fork agent skills 要隔離上下文

那我不如用 sub-agent？

sub-agent vs fork agent skills



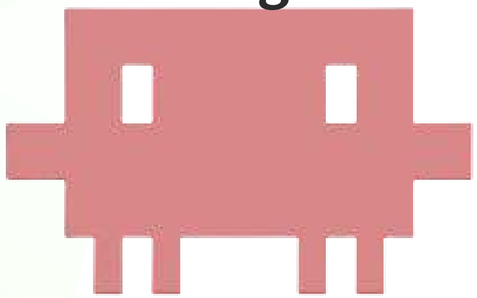
還真的差不多

fork agent skills 要隔離上下文

那我不如用 sub-agent？

sub-agent and fork agent skills

sub-agent



+

專案 skills(fork agent)



帶 Domain 提示詞　　寫專案知識及目標

想分享一個自己想法

skills 是需要管理的

鼓勵你用完別人的 skills 然後爆改成自己的

最後想分享 ...

出處（Meetup） Meetup 心得

未來或許我們不再 Import Library
而是 Import Skills

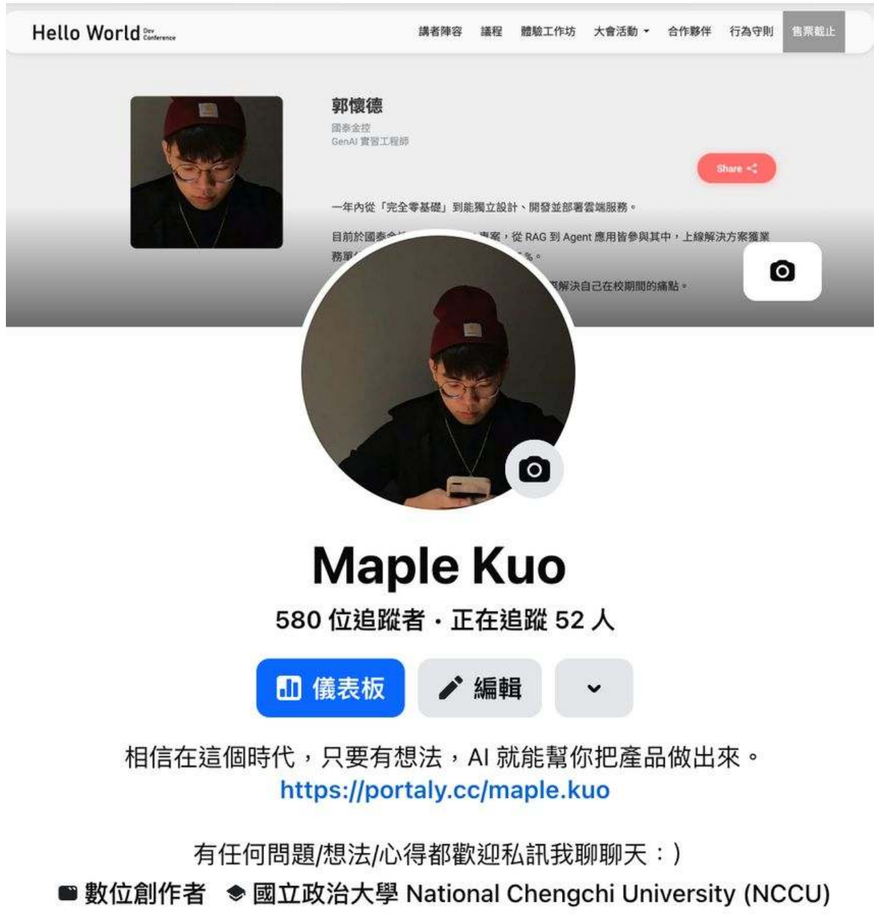
把 skills 代換成套件



總結

- skills 可以定義成簡單、通用任務
- 常常在重複敘述一件事，考慮把它變成 skills
- 有想要能重複執行的工作流，考慮把它變成 skills + scripts
- skills 的本質是在做 Context Engineering，需要搭配機制
- sub-agent 是有領域知識的專家，skills 是特定任務
- 複雜（過程很花 token 但不重要）的任務搭配 skills fork agent

經歷



- 國立政治大學 英國語文學系
- 人工智慧跨域微學程修畢
- Hello World Dev Conference 講者
- 衛生所「來學 AI 吧！數位轉型的第一步」講者
- Generative AI 小聚 講者
- Google Certified Educator
- Zeabur “Ship It” Hackathon 第三名
- 一年從 0 開始和 AI 自學轉職軟體工程師
- AI 重度付費使用者

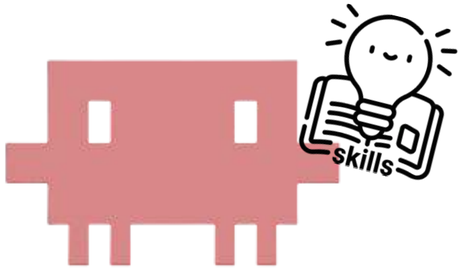


Maple Kuo

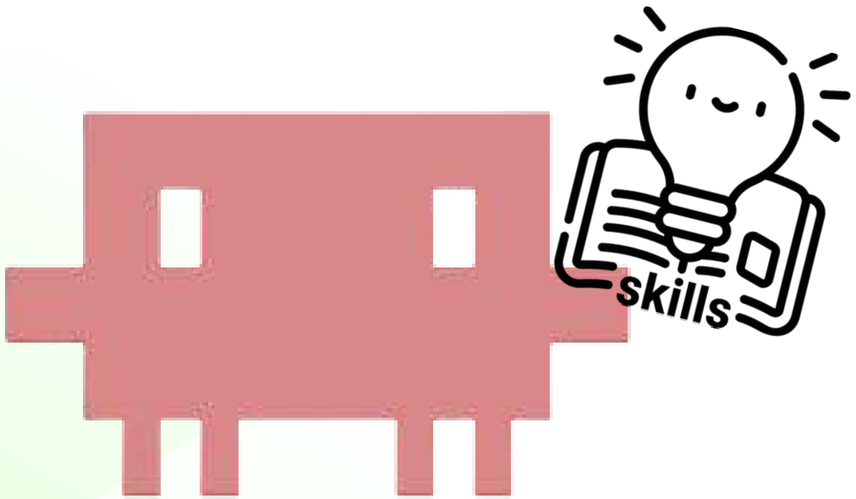
AI 時代出身的軟體工程師

感謝你！

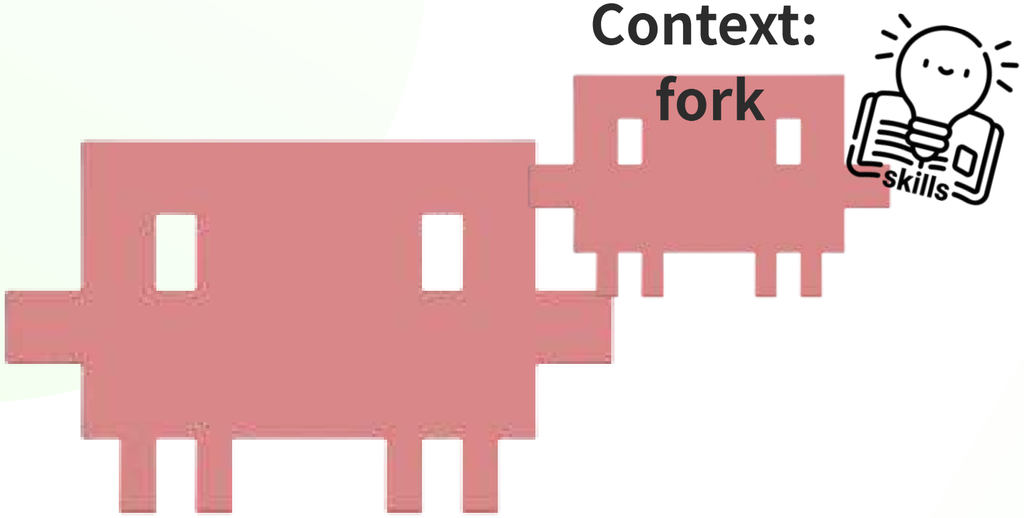
contact.maple.kuo@gmail.com



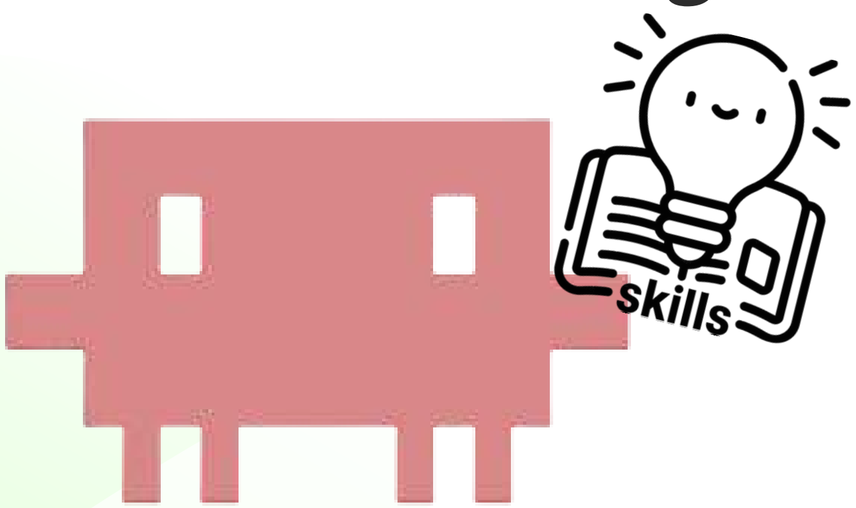
skills vs fork agent skills



Context:
fork

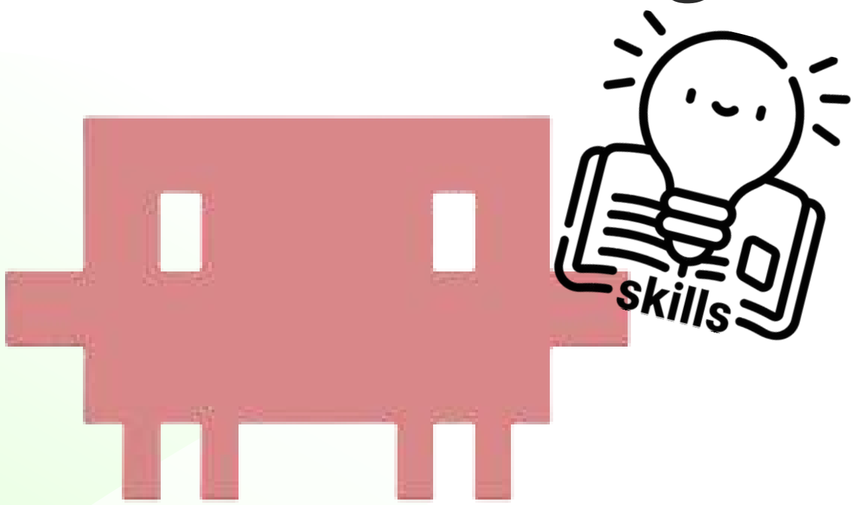


skills vs fork agent skills



- 常用 prompt 的快捷鍵

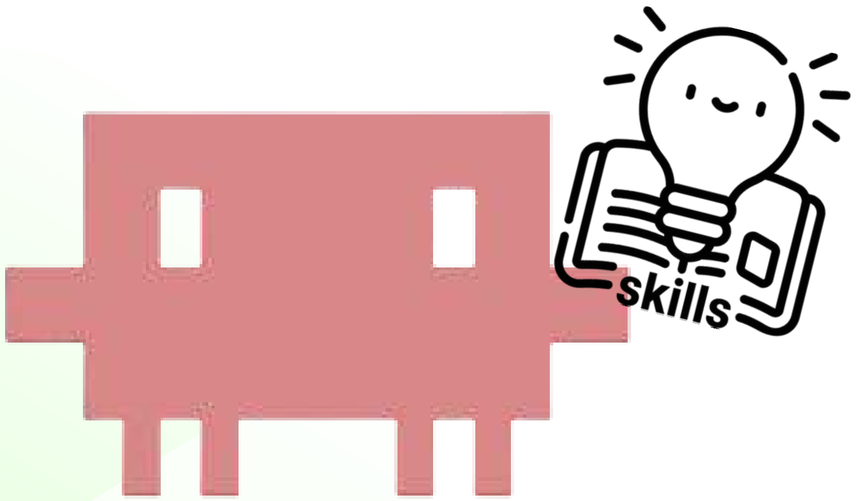
skills vs fork agent skills



- 常用 prompt 的快捷鍵
- 能力擴充 e.g., /wt, /reddit-search

Context: fork

skills vs fork agent skills

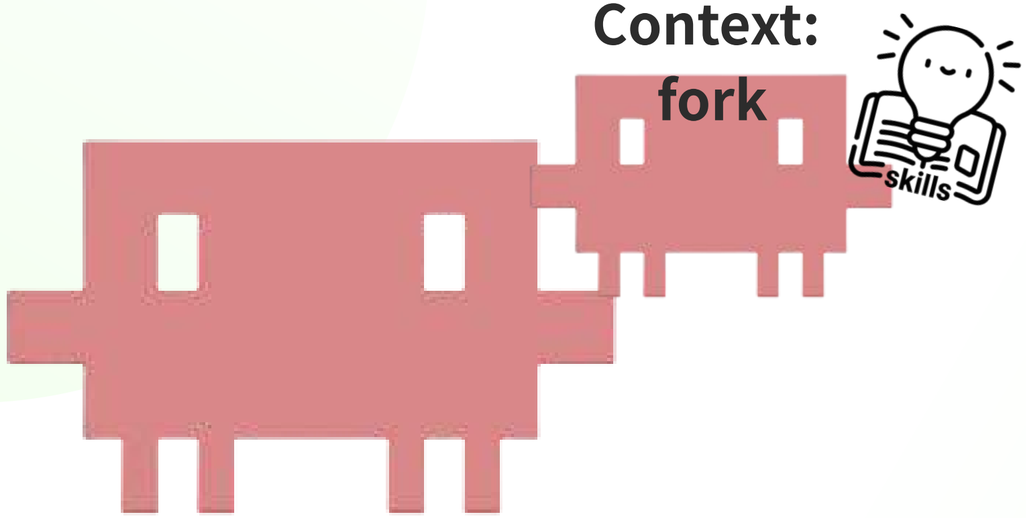


- 常用 prompt 的快捷鍵
- 能力擴充 e.g., /wt, /reddit-search
- 按需載入的背景知識擴充

Context:
fork

skills vs fork agent skills

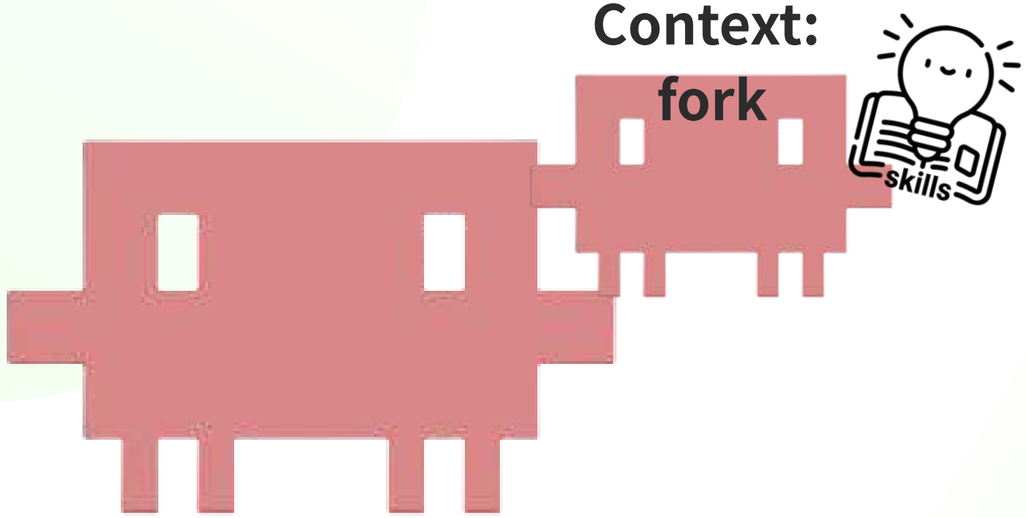
Context:
fork



- 通用性的重複任務

skills vs fork agent skills

Context:
fork



- 通用性的重複任務
- 不在意過程只在意結果

總結

- skills 可以定義成簡單、通用任務

總結

- skills 可以定義成簡單、通用任務
- 常常在重複敘述一件事，考慮把它變成 skills

總結

- skills 可以定義成簡單、通用任務
- 常常在重複敘述一件事，考慮把它變成 skills
- 有想要能重複執行的工作流，考慮把它變成 skills + scripts

總結

- skills 可以定義成簡單、通用任務
- 常常在重複敘述一件事，考慮把它變成 skills
- 有想要能重複執行的工作流，考慮把它變成 skills + scripts
- skills 的本質是在做 Context Engineering，需要搭配機制

總結

- skills 可以定義成簡單、通用任務
- 常常在重複敘述一件事，考慮把它變成 skills
- 有想要能重複執行的工作流，考慮把它變成 skills + scripts
- skills 的本質是在做 Context Engineering，需要搭配機制
- sub-agent 是有領域知識的專家，skills 是特定任務