# 以產品思維驅動的 DevOps 策略
# 金融業 AI 團隊的實踐之旅

吳明倫 @ 玉山銀行智能金融處

# About Me

## 吳明倫 (Ming-Lun)

- **DevOps Engineer @ E.SUN Bank**
- 喜歡旁軸相機
- 正在準備人生第一場半馬

# TABLE OF CONTENTS

01

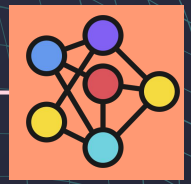
# 多團隊環境下的 CI/CD 策略選擇

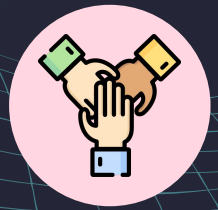
# 關於玉山銀行智能金融處

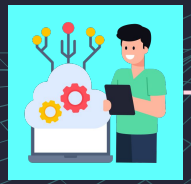
## 智能金融處

**60+** Members

**8** Teams







**30+** Members

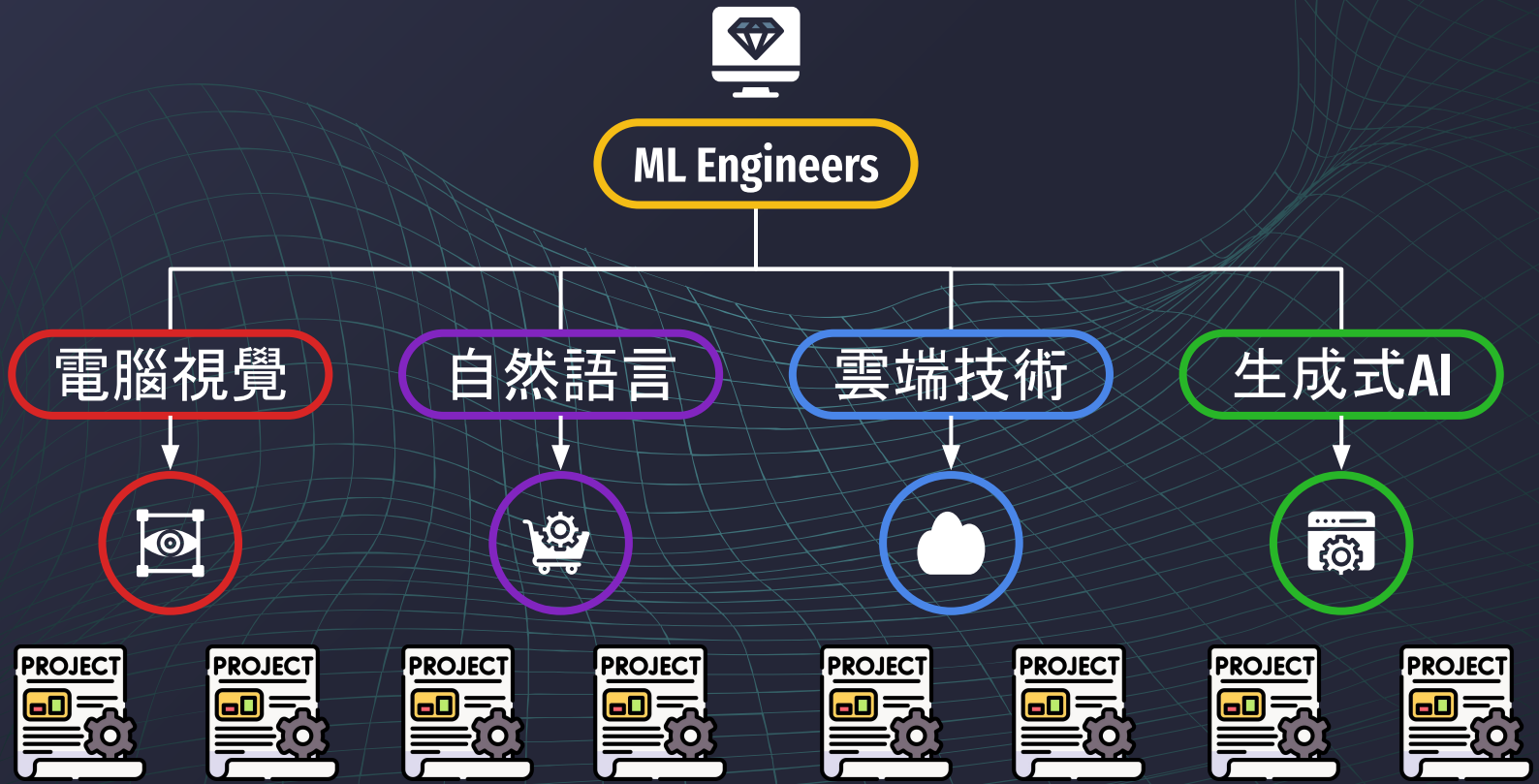
## ML Engineers

AI 新興技術研發
資料分析與模型訓練
Computer Vision, HLP, RPA

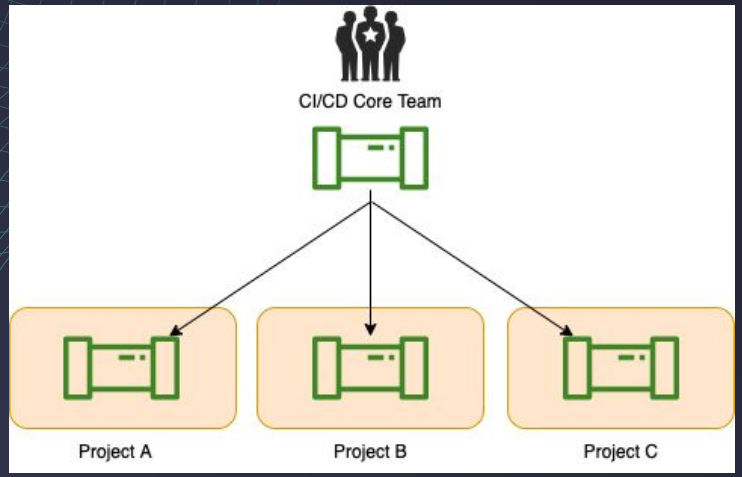
## Data Engineers

建立 AI 研發環境
維護 ML 服務框架
DevOps, Kubernetes, Database

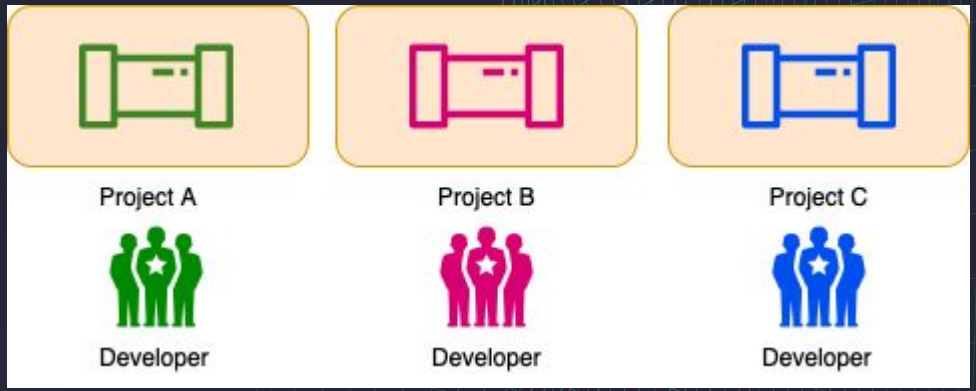
# 關於玉山銀行智能金融處 (Cont.)



# CI / CD Strategies for Multiple Teams




Centralized CI / CD




Decentralized CI /CD

# CI / CD Strategies for Multiple Teams

## Centralized Strategy

由一個 **Core Team** 負責組織內的 **CI/CD** 流程

**一致性高**
成員調動不需重新適應 CI/CD 流程

**啟動快速**
Project 初始化時，不需特別規劃 CI/CD

適合大型團隊、傳統企業

## Decentralized Strategy

由開發團隊自行設計 **CI/CD** 流程

**靈活度高**
可自行規劃最符合團隊的 CI/CD 流程

**擁抱 DevOps**
令每個團隊成員皆有機會接觸 DevOps

適合高度 DevOps、Agile 的團隊

# CI / CD Strategies for Multiple Teams

## Centralized Strategy

由一個 **Core Team** 負責組織內的 **CI/CD** 流程

**一致性高**
成員調動不需重新適應 CI/CD 流程

**快速啟動**
專案初始化時，不需從頭規劃 CI/CD

適合大型團隊、傳統企業

## Decentralized Strategy

由開發團隊自行設計 **CI/CD** 流程

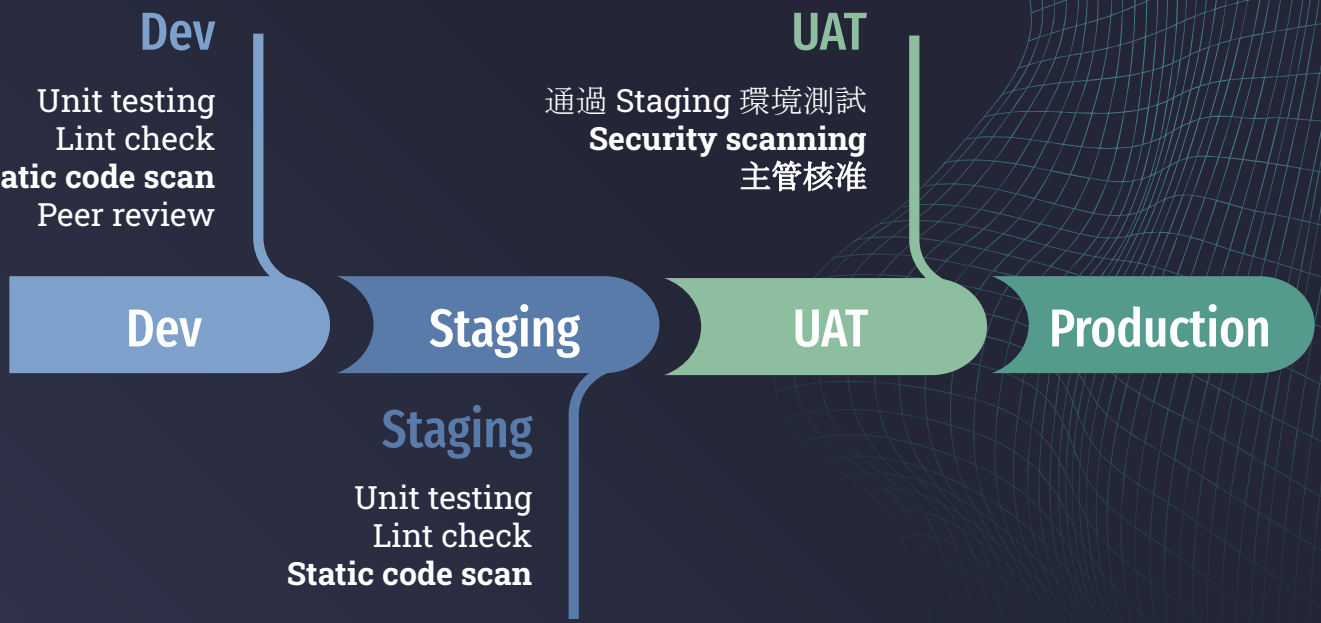
**細緻度高**
可自行規劃最符合團隊的 CI/CD 流程

**擁抱 DevOps**
令每個團隊成員皆有機會接觸 DevOps

適合高度 DevOps、Agile 的團隊

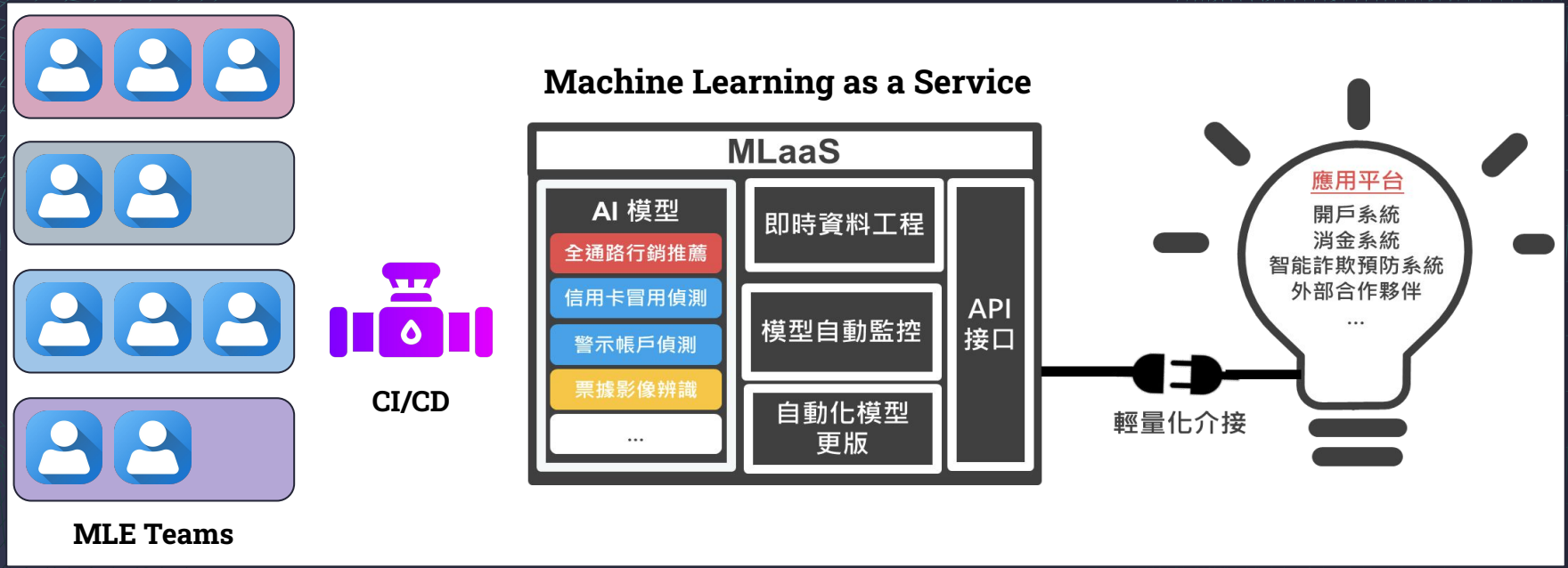
# Why Centralized Strategy ?

- 金融業為高度監管行業, 必須確保所有 Project 的 CI/CD 流程皆符合規範



# Why Centralized Strategy ?

- 希望 MLE 團隊專注於建模, 透過共通的 CI/CD Pipeline 部署機器學習服務



02

# 產品思維的建立

## Core Team 建立產品思維

- 所有 ML Engineer 皆依賴 CI/CD 流程部署服務
- CI/CD 流程的順暢度，與 ML Engineer 的開發體驗有極大關聯

基於以上特性, CI/CD Core Team 開始出現一種思維：

- CI/CD Pipeline 是玉山智金處的「**內部產品**」
- **ML Engineers** 是該產品的「**內部顧客**」

**「內部顧客」透過「產品」達成「部署機器學習服務」的目標**

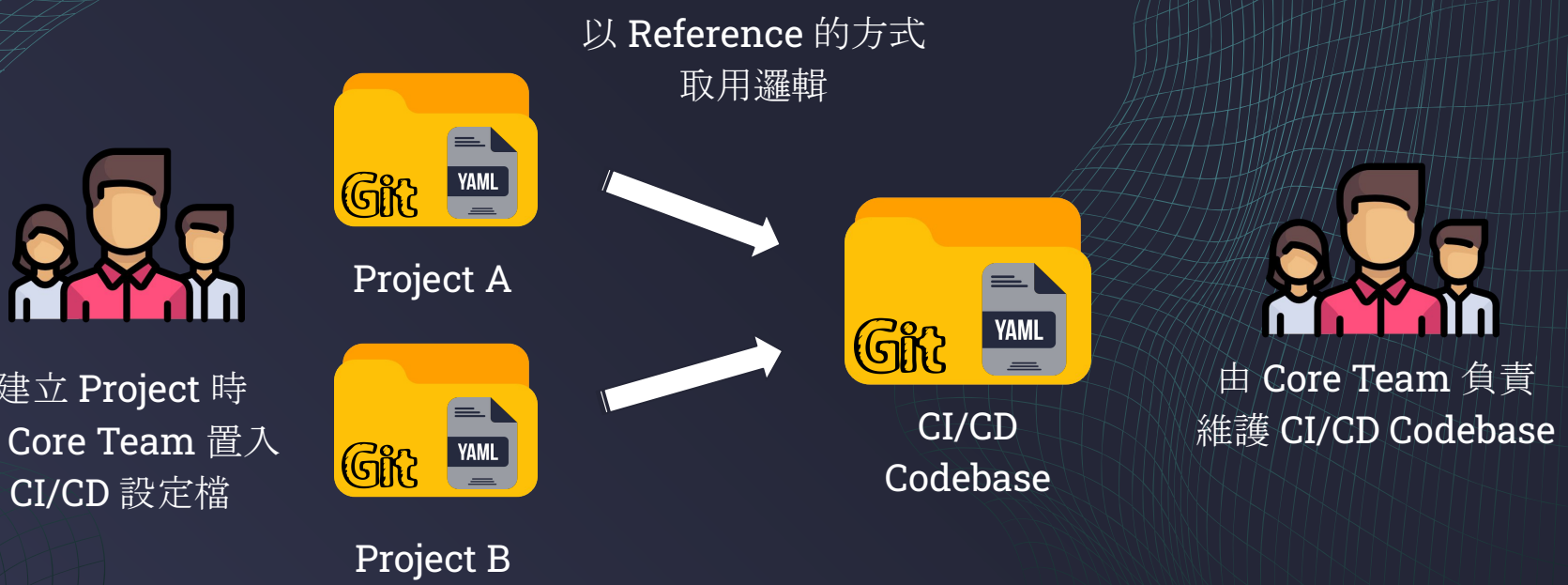
“Satisfy the needs of users internal to your organization or enables your organization to satisfy the needs of its customers.”

03

# 我們遭遇了什麼樣的挑戰？

# 挑戰 1：無力優化產品 - 維運成本過高

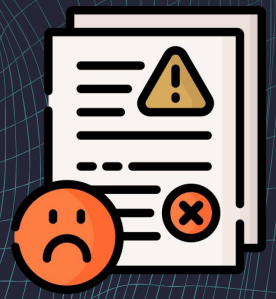
**Centralized Strategy** 的高維運成本



## 挑戰 1 : 無力優化產品 - 維運成本過高

在初始化階段, 除了 **Yaml** 檔案外, 仍有其他設定:

- **Project** 成員 & 群組權限
- 互動元件的 **Service Connection**
- **Branch Policy**



建立 1 個 **Project** 需要經歷 **14** 個步驟, 約 **30** 分鐘的時間

在人力不足的情況下, 產品的 **Feature Request** 常被積壓在 **Backlog** 中

| 60+ | 30+ | 180 + | 4 |
|---|---|---|---|
| ML Engineers | Data Engineers | Projects | CI/CD Core Members |

## Solution - 建立自動化流程

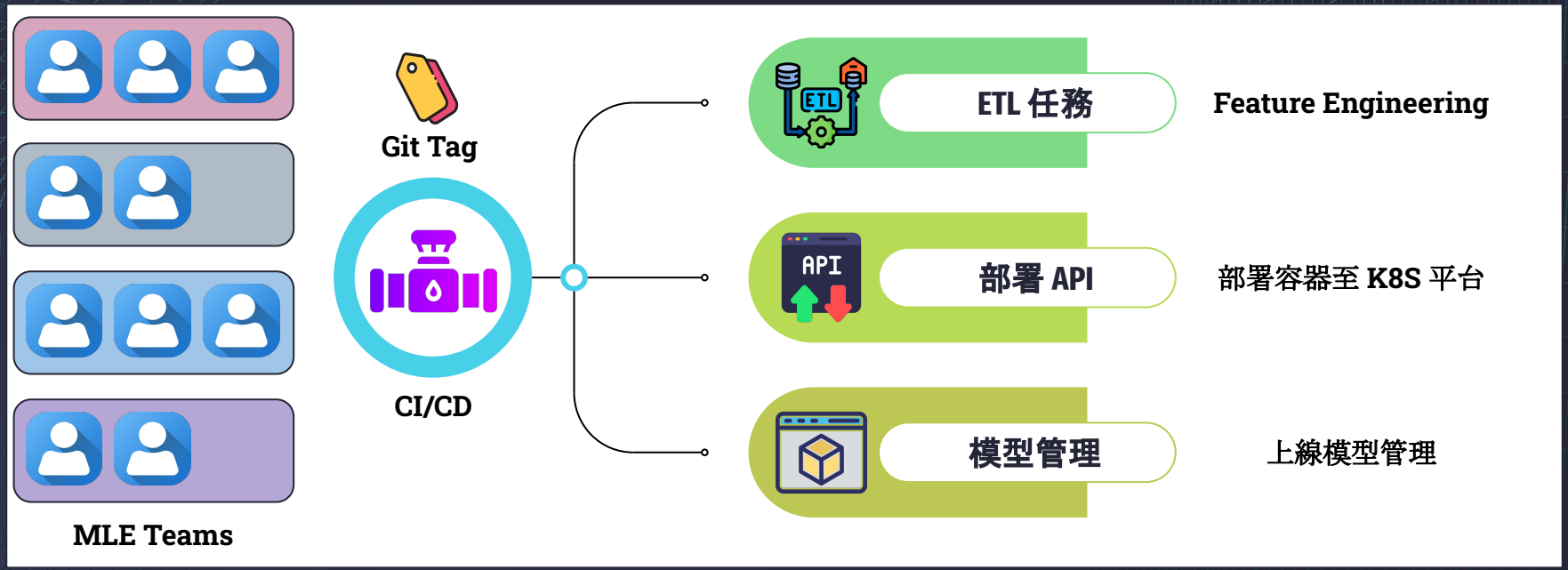
1. 梳理既有流程 : 建立「手動」的標準作業程序
2. 封裝 Azure DevOps 的 API, 將原先的 UI 操作改為「半自動」的 Runbook


效益 :

- **降低例行性的維運工時**: 由 30 分鐘 降為 30 秒 (縮短 **98 %**)
- **提高作業正確率**: 以 Runbook 自動設定後, 不再出現「人為設定錯誤」的事件

# 挑戰 2 : 產品使用者的客服需求多 - 溝通成本高

ML Engineers 可透過 Git Tag 選擇執行不同類型的任務



# 挑戰 2 : 產品使用者的客服需求多 - 溝通成本高

CI/CD Core Team

# Solution - 來寫文件吧！


使用者透過文件
排除問題

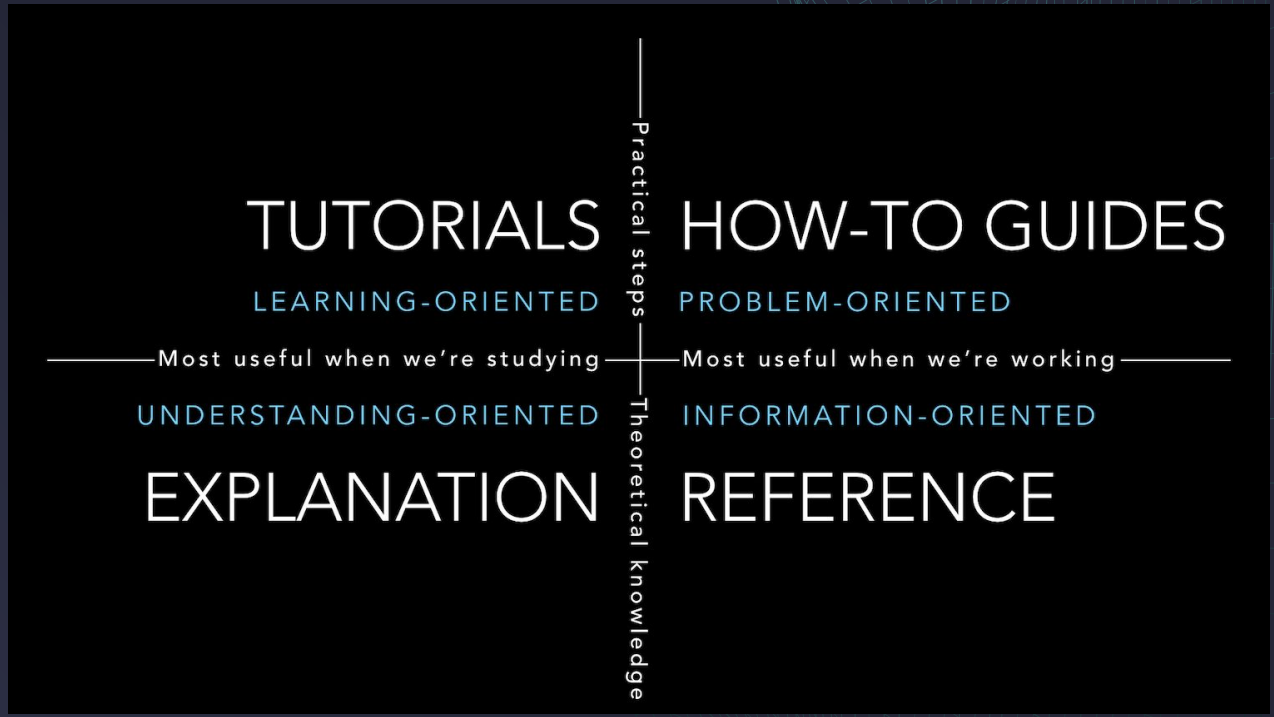
使用者不看文件

CI/CD Core Team


使用者看完文件
仍然尋求客服

# Solution - Documentation System

導入文件框架
**Documentation System**




Source : https://documentation.divio.com

# Solution - Documentation System

導入文件框架
**Documentation System**

- 由 DIVIO 提出的文件分類系統, 將技術文件依照「目的」、「目標客群」劃分為四大類
- 針對產品功能提出「有效」文件, 降低工程師的認知負荷

- 效益 :
  - 作者可藉此了解不同類型文件的所需內容
  - 讀者可根據自身需求, 快速找到符合自己的文件
  - 確保團隊在討論的過程中, 有相同的Context

# 挑戰 3：內部產品缺乏量化指標

「內部產品」通常著重在「用戶體驗」，所收集的指標偏向「質化」而非「量化」

常見的做法包含：

- 用戶訪談
- 內部回饋系統

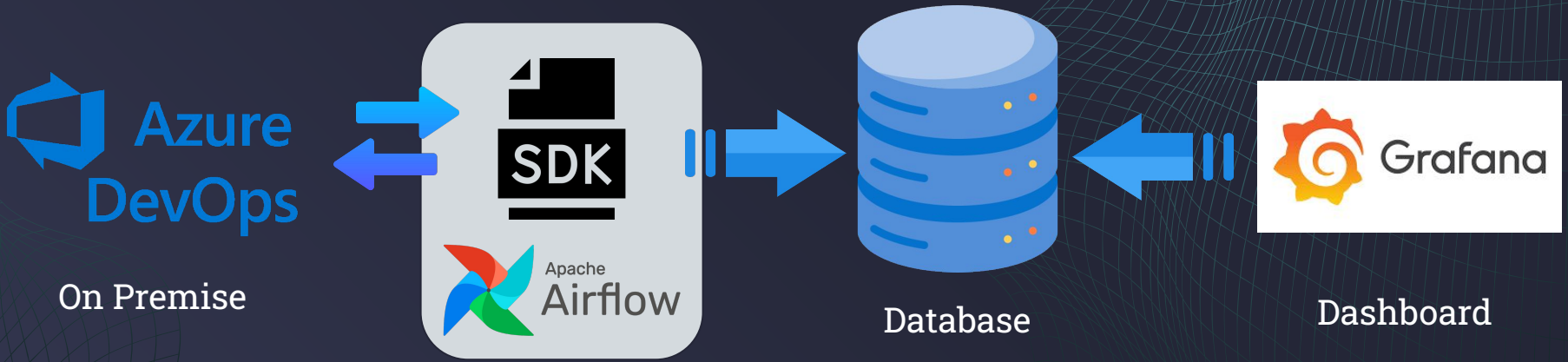
然而，對於 CI/CD 認知等級不同的團隊，可能給出完全不同的評價

較難取得可參考的數據

## Solution - 建立 CI/CD 可觀測性

透過排程工具, 收集 Azure DevOps 的使用數據:

- 使用 On Premise 版本的 Azure DevOps
- 需要根據用戶的需求, 自行探索「有效」的指標



## Solution - 建立 CI/CD 可觀測性

Why not DORA ?

- 衡量的是 Project 的 DevOps 效率
- 站在 Core Team 的角度, 需要「產品該如何修正」的指標

尋找「內部產品」的北極星指標 - **“從「提供客戶價值」作為起點”**

- 追求內部用戶的**使用者體驗**
- **MLE 追求部署的穩定性**

  “在 ML Code 未更動的情況下, 避免 CI/CD 出現意料之外的錯誤”

# Solution - 建立 CI/CD 可觀測性

01

## 掌握產品現況

依靠量化數據衡量 CI/CD Pipeline 的穩定度

02

## **識別錯誤情境**

在什麼情境下, 使用者發生錯誤?

03

## 定位用戶痛點

發生錯誤時, 具體是哪個階段造成失敗?

04

## 尋找錯誤根因

錯誤根因分析, 提高修正效率

# Step 1 - 掌握產品現況

- 取得產品當前的使用狀況 - 各環境 CI/CD 失敗率

Production 環境

Staging 環境

UAT 環境

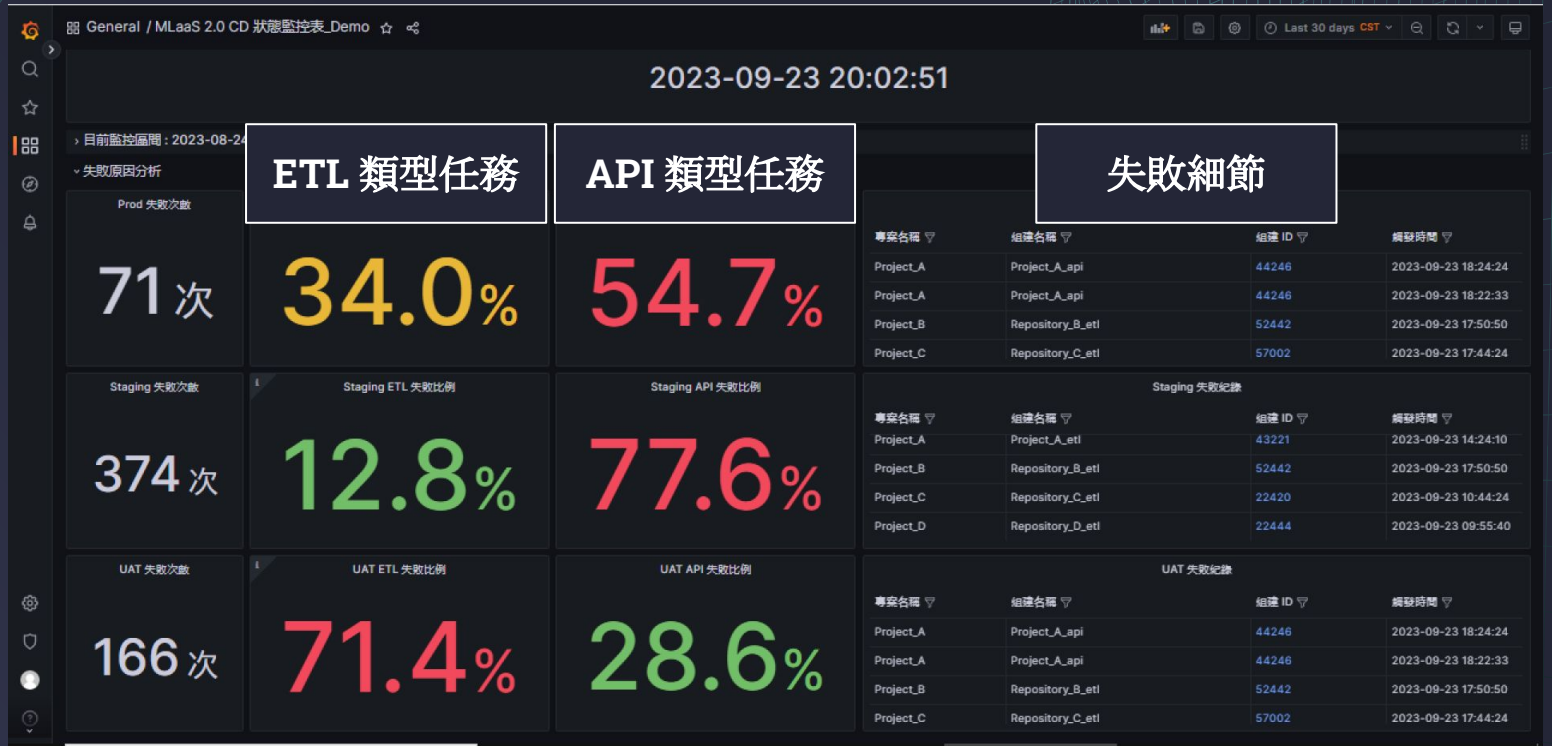
# Step 2 - 識別錯誤情境 (How ?)

- 識別用戶在什麼情境下出現錯誤

Production 環境

Staging 環境

UAT 環境



# Step 3 - 定位用戶痛點 (Which ?)

- 了解現況後，深入尋找用戶的「功能痛點」

Production 環境

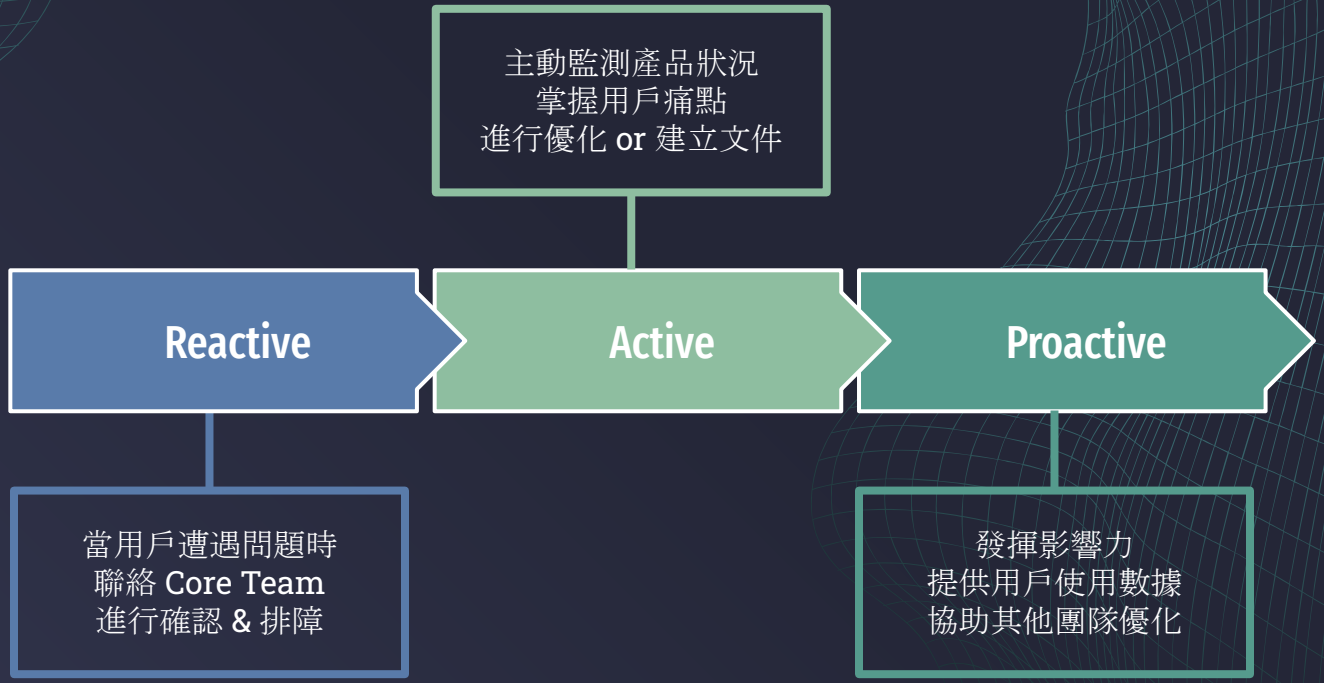
Staging 環境

UAT 環境

# Step 4- 尋找錯誤根因 (Why ?)

- 針對單一「功能痛點」，取得相關數據，作為「優化」的起點

# Solution - 建立 CI/CD 可觀測性



04

# 結論及反思

# 結論

- 多團隊情境下的CI/CD 部署策略
  - Centralized Strategy
  - Decentralized Strategy
- CI/CD Core Team 在優化產品過程中遭遇的挑戰
  - 維運成本高 : 例行性、重複性高的維運工作
  - 溝通成本高 : 重複性的使用者客服
  - 缺乏量化數據
- 解決方式

# Takeaway

## 思考流程痛點

根據團隊現況
思考產品所需的策略
尋找產品的「痛點」

## 優化產品

優化使用者體驗
修正產品
提供用戶數據給其他團隊


## 標準化

盤點、檢視作業流程
建立有效的文件

## 自動化

建立品質流程
提高效率及作業正確率
降低維運工時

## 建立產品指標

以「提供用戶價值」為目標
掌握產品現況
收集產品數據

# THANKS!