如何打造企業微服務平台

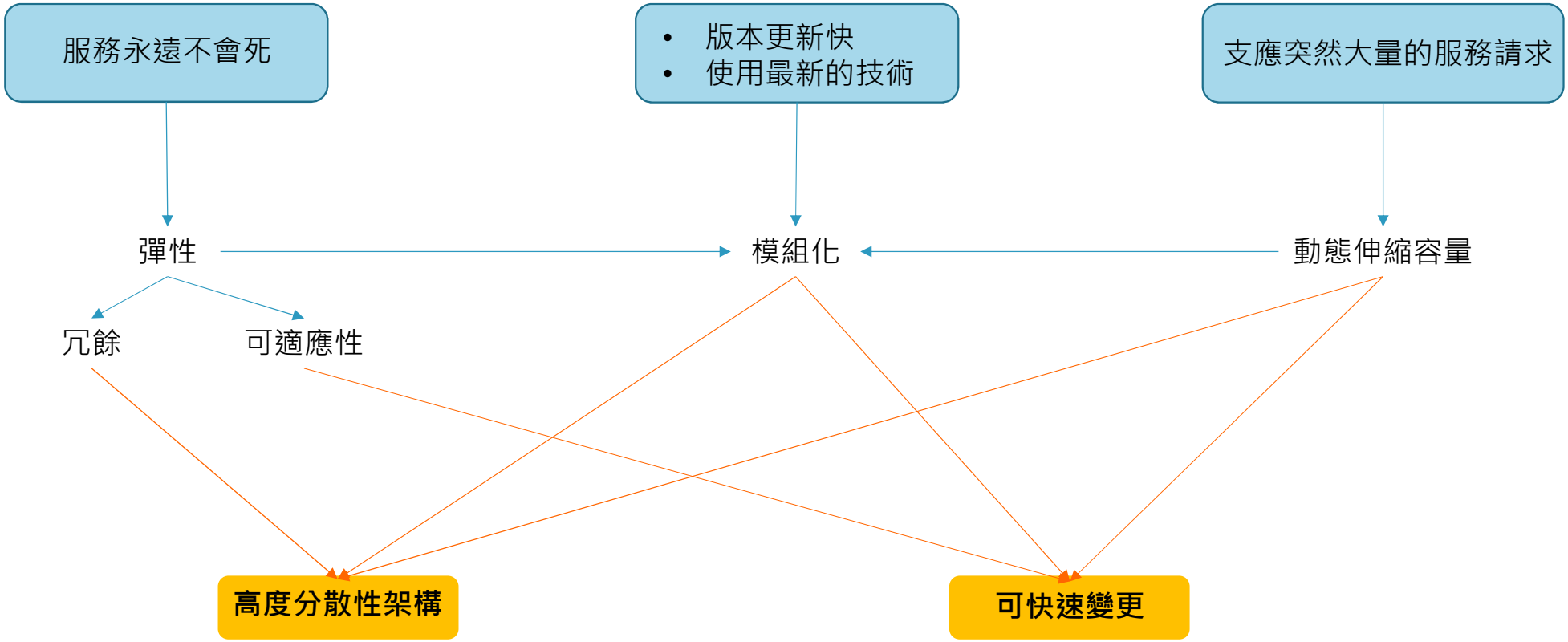
謝逸凡 IVAN HSIEH

Agenda

- 為何需要微服務架構
  - IT 應用系統架構設計重點
  - 服務架構的演進
- 如何建立微服務架構
  - 服務與資料切割
  - 建立平台安全
  - 服務DevOps
  - 服務監控
- Summary

為何需要微服務架構

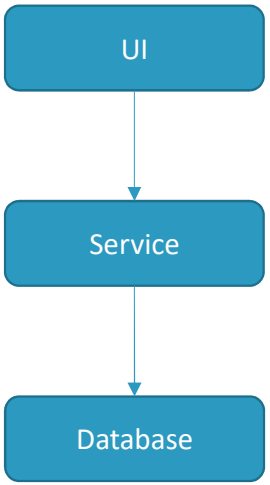
IT 應用系統架構設計重點



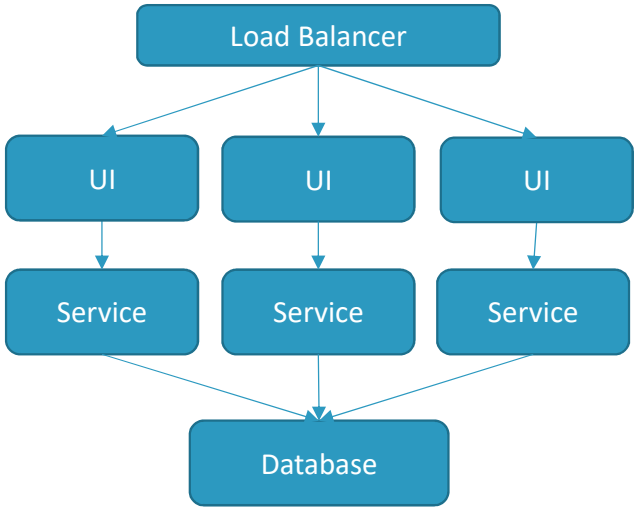
高分散式並可以不斷變化的架構

服務架構的演進

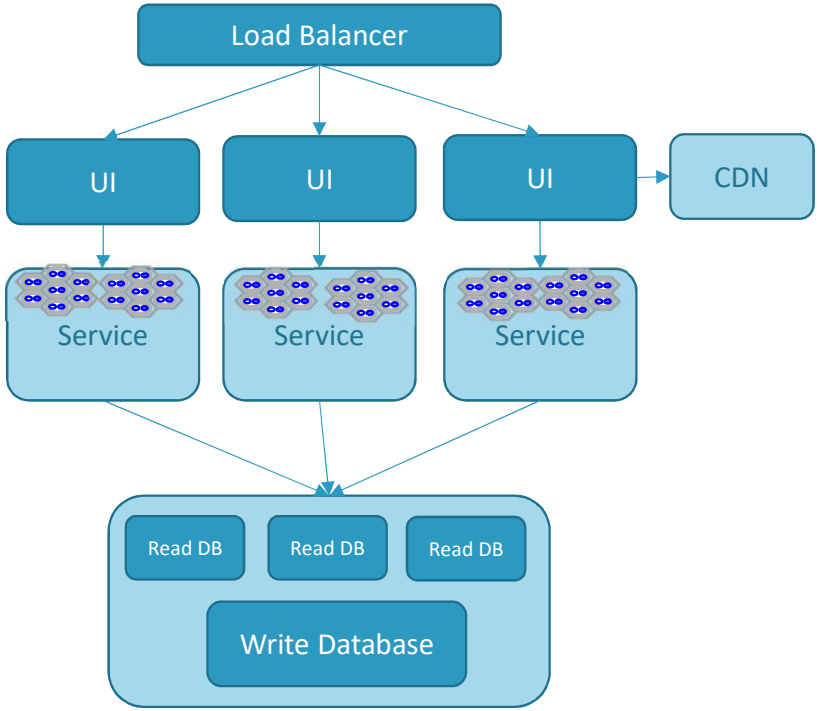
三層式架構



三層式集群架構



強化版三層式集群架構



傳統單體架構提升後還有什麼問題

應用系統需要快速彈性支援
面對海量需求資料庫還是瓶頸

新技術 新平台應用
單體程式使用單一語言架構
無法滿足目前新技術的需求

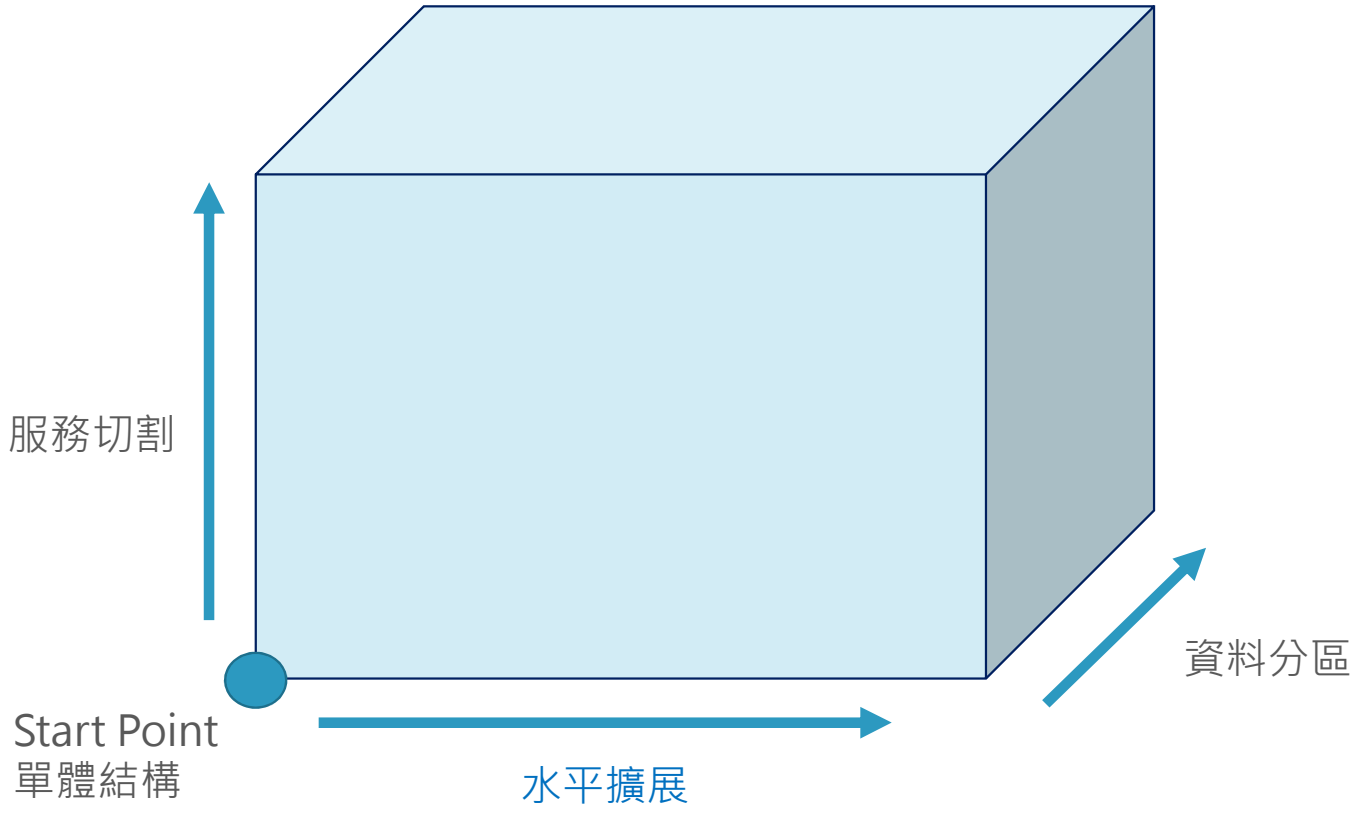
系統快速開發
程式邏輯複雜，修改成本高

系統的高可用性
單體使用資源大，對基礎建設要求多
橫向擴展成本高

快速部屬與安全
邏輯複雜需要更多更完整的測試

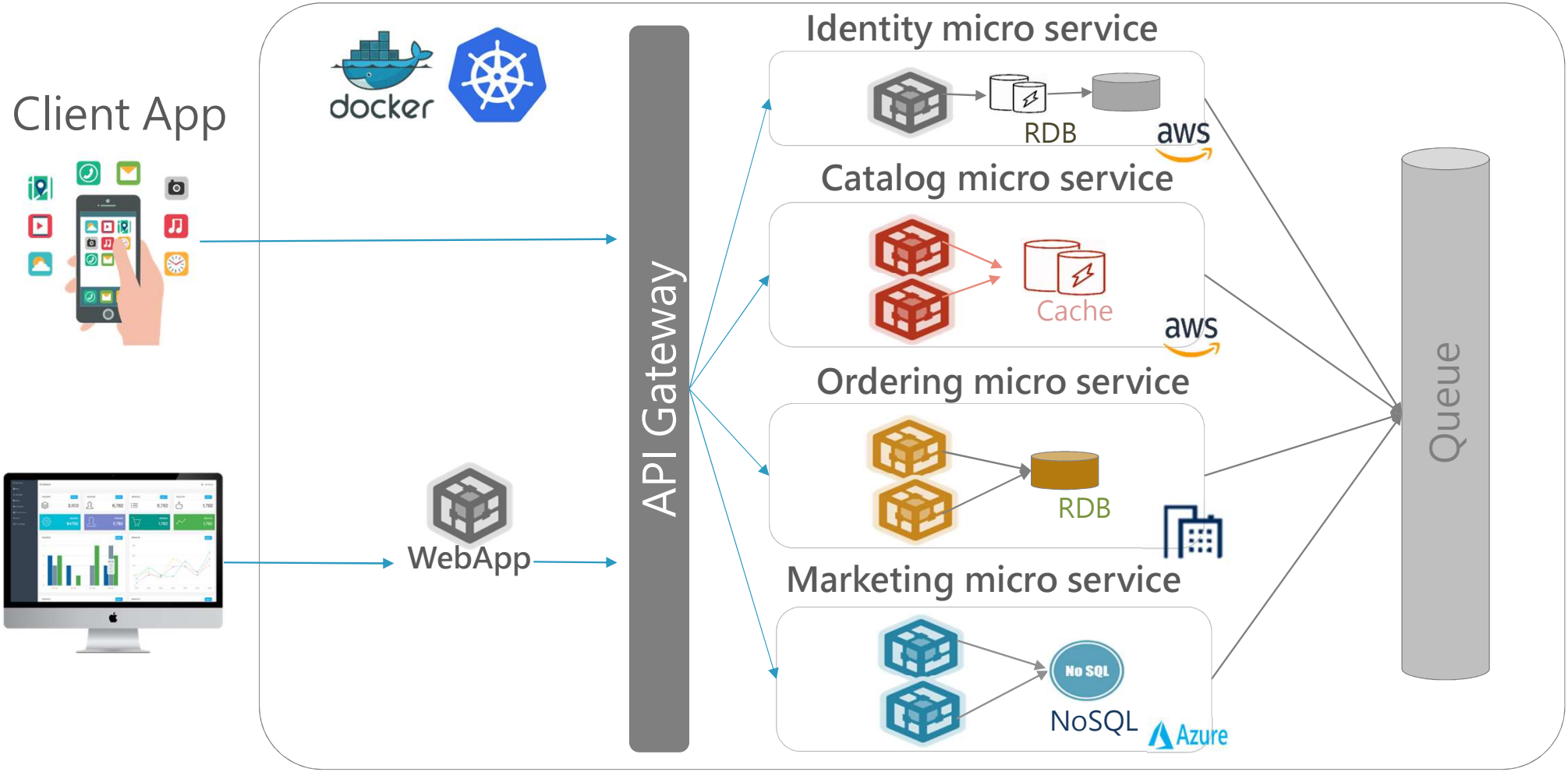
系統複雜效能出問題難以除錯
功能複雜不易監控 系統問題難查找
錯誤問題無法隔離

服務如何擴展

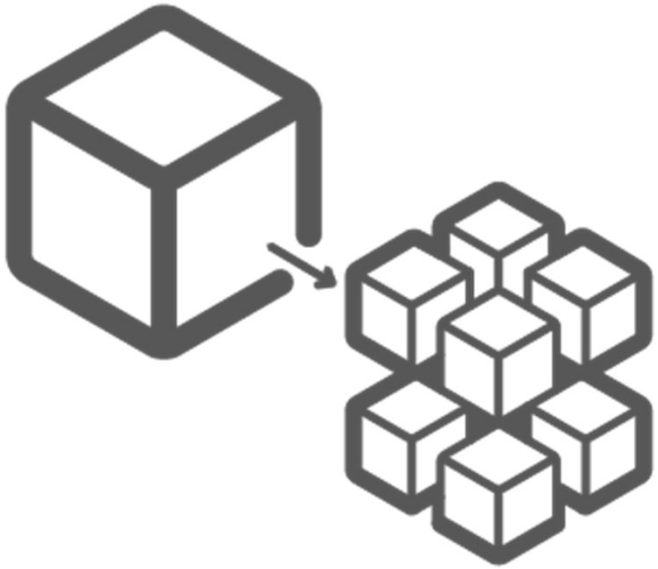


Art of Scalability, The: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise

微服務架構特性1


Client App
docker
API Gateway
Identity micro service
RDB
aws
Catalog micro service
Cache
aws
Ordering micro service
RDB
Marketing micro service
No SQL
NoSQL
Azure
WebApp
Queue

微服務架構特性2



技術的異質性

■專注服務 ■組織調整，去中心化
■持續創新

韌性 高可用度

■失敗隔離 隔艙 ■數據隔離，資料一致性
■隱藏內部細節 ■一個或多個無狀態服務一起運行

擴展

■持久化資料都在外部，快速容量擴展 • 透過http restful api 相互通信
■說明依賴關係並隔離管理依賴 ■非同步架構

容易部屬 ，自動化

■一份程式代碼 多環境部屬 ■DevOps CICD
■環境配置檔案與程式代碼分開隔離 ■開發環境與正式環境相同

服務建構開發＆運行

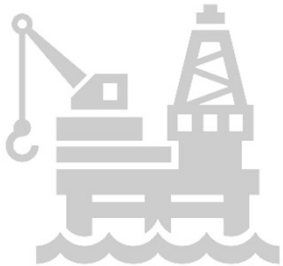
■服務重組 ■後端服務當作附加資源 ■透過網路API調用後端服務
■漸進式設計 ■Log / 管理監控機制 ■快速重啟容器特性

如何建立微服務

Microservice Architecture Focus

Service

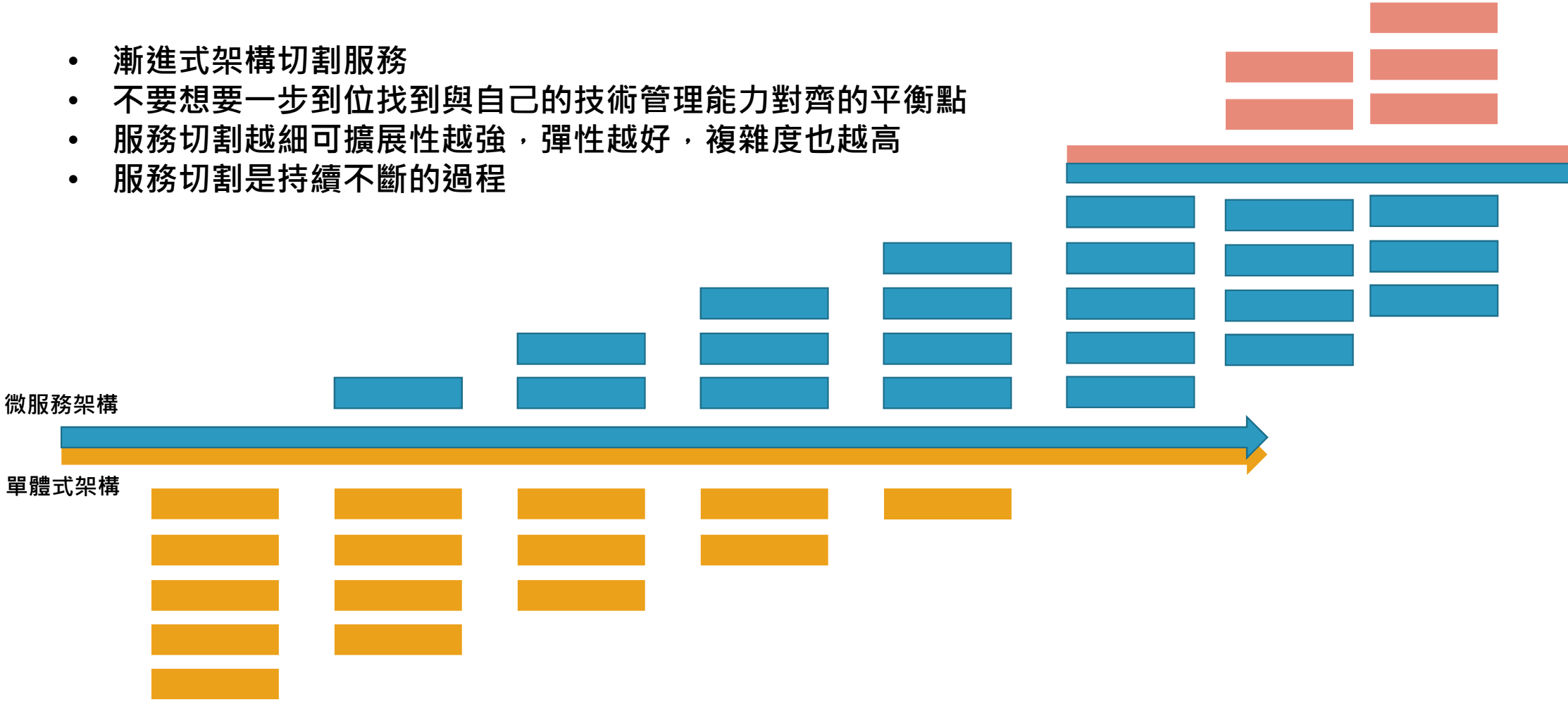
Data



Platform

建立微服務如何開始

- 漸進式架構切割服務
- 不要想要一步到位找到與自己的技術管理能力對齊的平衡點
- 服務切割越細可擴展性越強，彈性越好，複雜度也越高
- 服務切割是持續不斷的過程



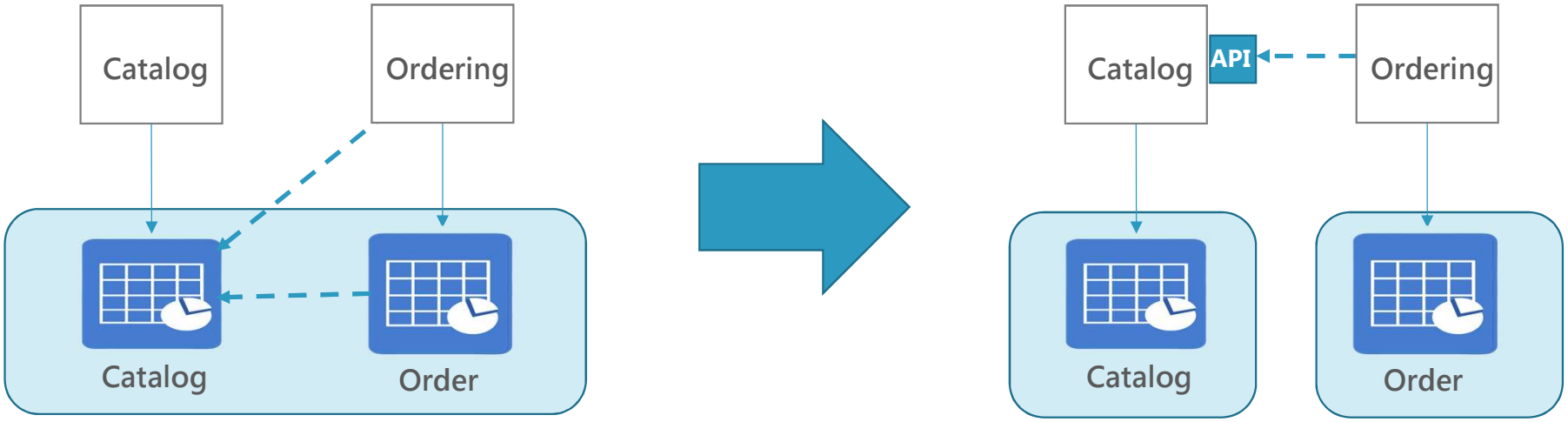
服務與資料切割

- 服務切割原則
  - 單一職責
  - 功能修改都在同一服務內
- 根據業務/服務/功能/Domain切割
  - 依賴纏繞的根源資料庫
- 服務切割後分散式架構的挑戰
  - 資料重組查詢效能問題
  - 服務修改過渡期
  - 服務間調用可用性
- 最終一致性的架構 (CAP)

依賴纏繞的根源 資料庫

檢視服務程式存取資料庫的狀況，找出服務主要使用Table

Ordering 需要 Catalog 資料

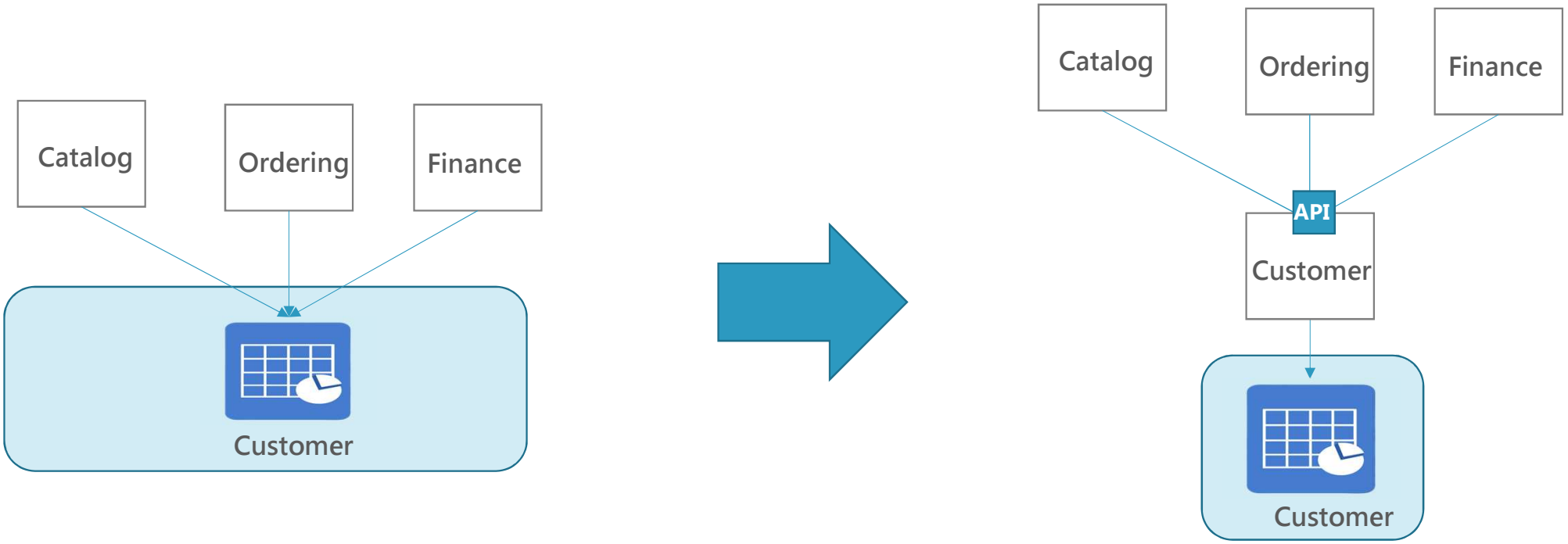


直接Access Table 或 Table Join 使用

透過API Access Catalog 資料

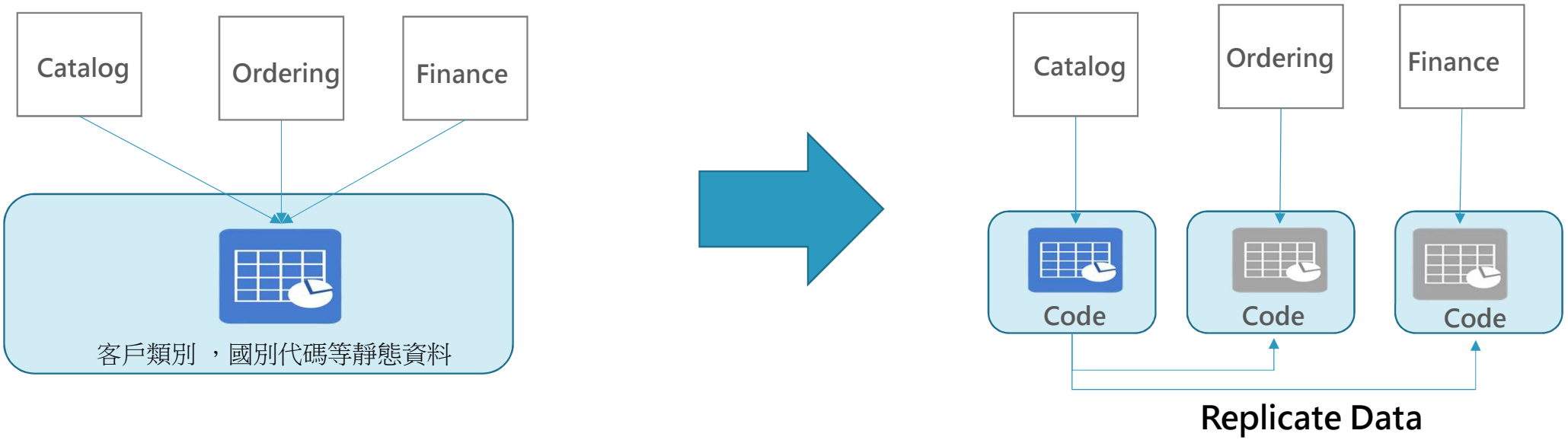
依賴纏繞的根源 資料庫

Ordering、Finance、Catalog 都需要客戶資料



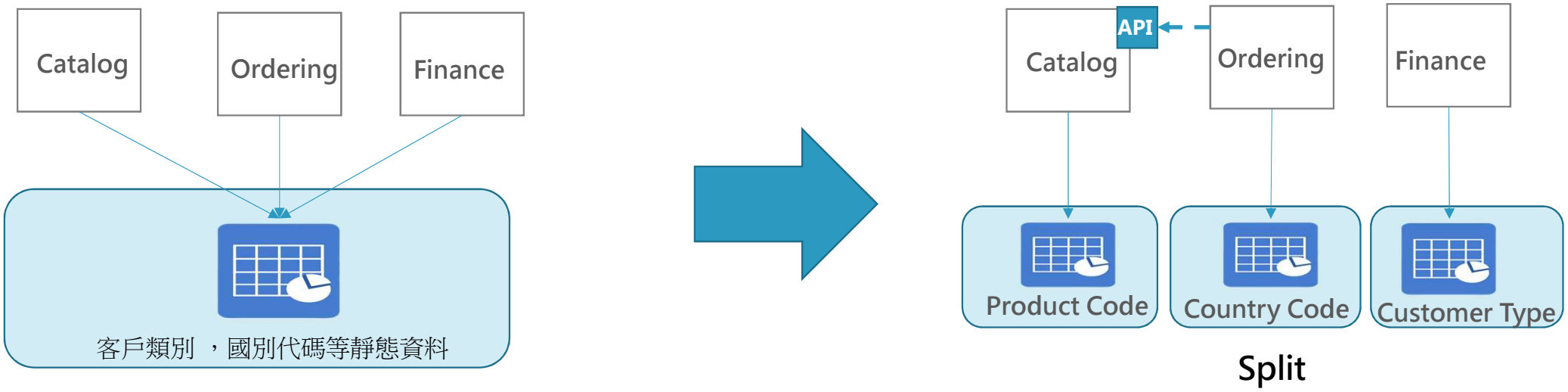
依賴纏繞的根源 資料庫

Ordering Finance Catalog 都需要客戶類別 ，國別代碼，產品代碼等靜態資料



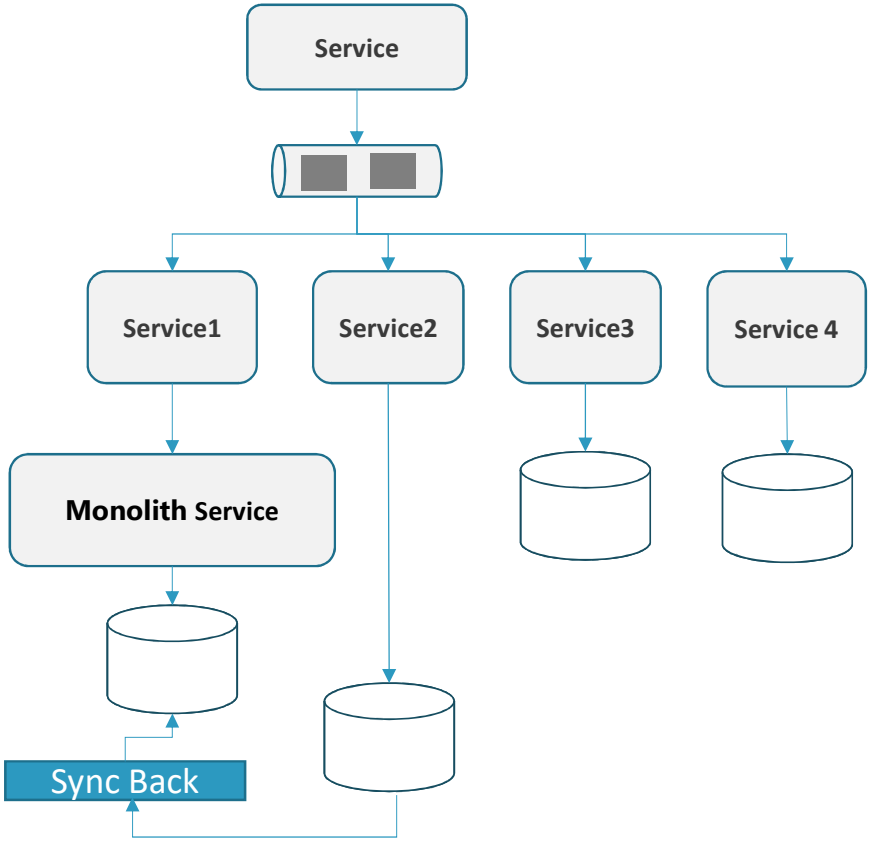
依賴纏繞的根源 資料庫

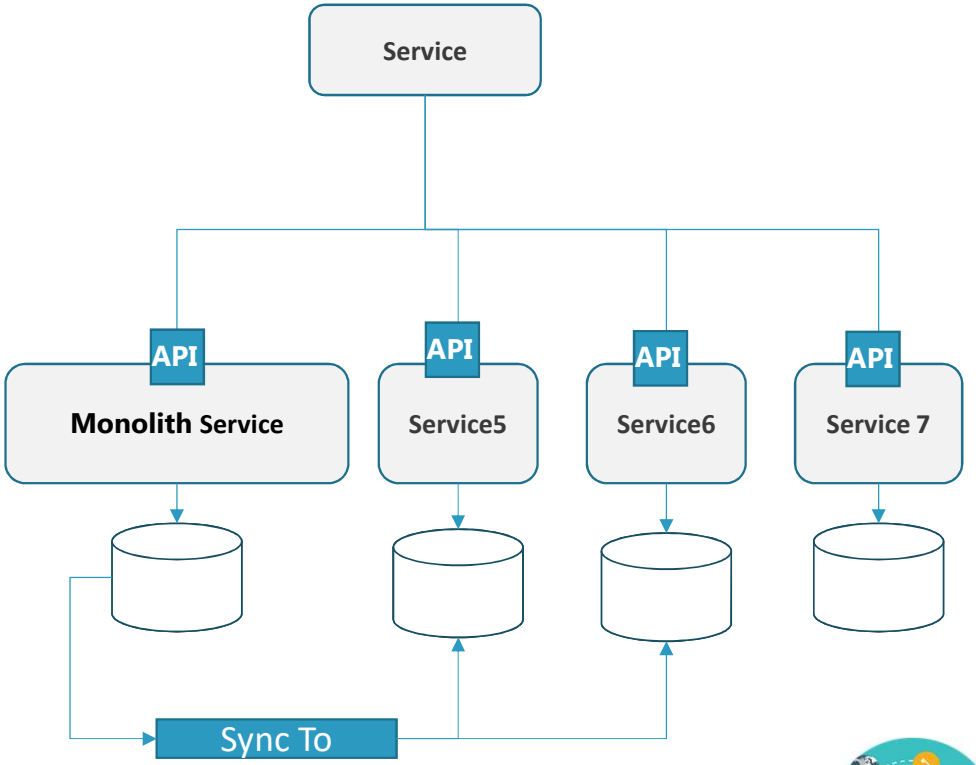
Ordering Finance Catalog 都需要客戶類別 ，國別代碼，產品代碼等靜態資料



服務修改過渡期

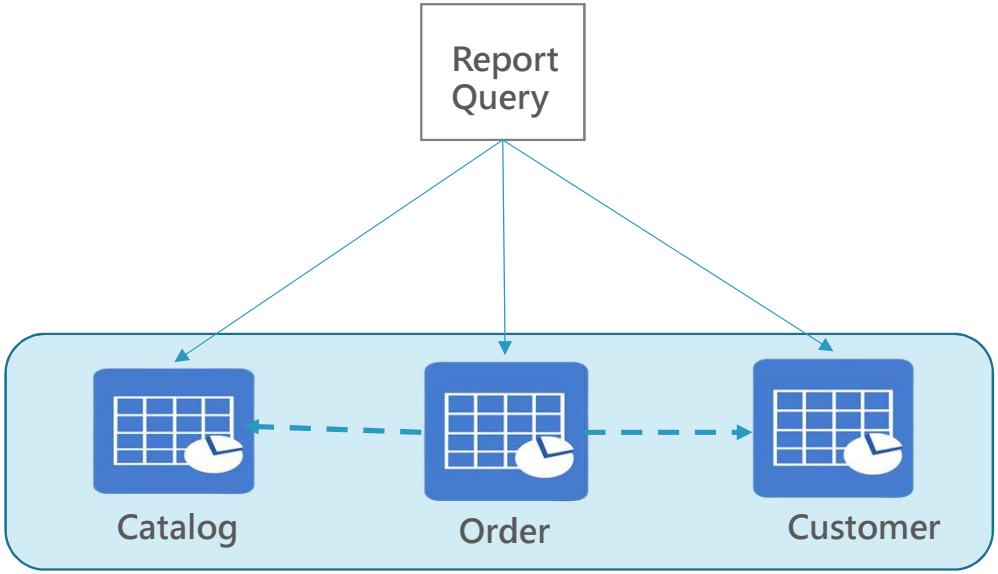
微服務化無法同時完成，過程中需要保持服務可以正常執行

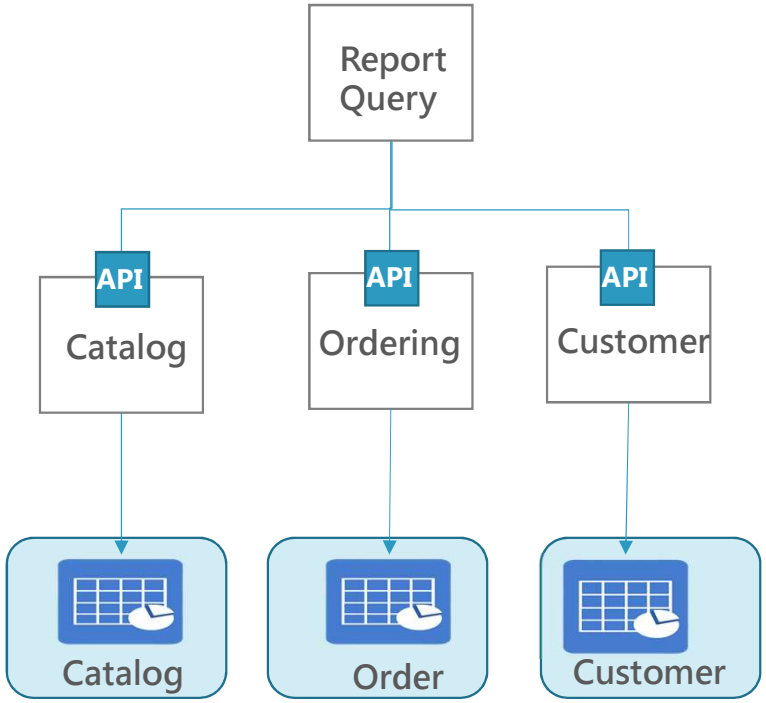




查詢功能效能問題

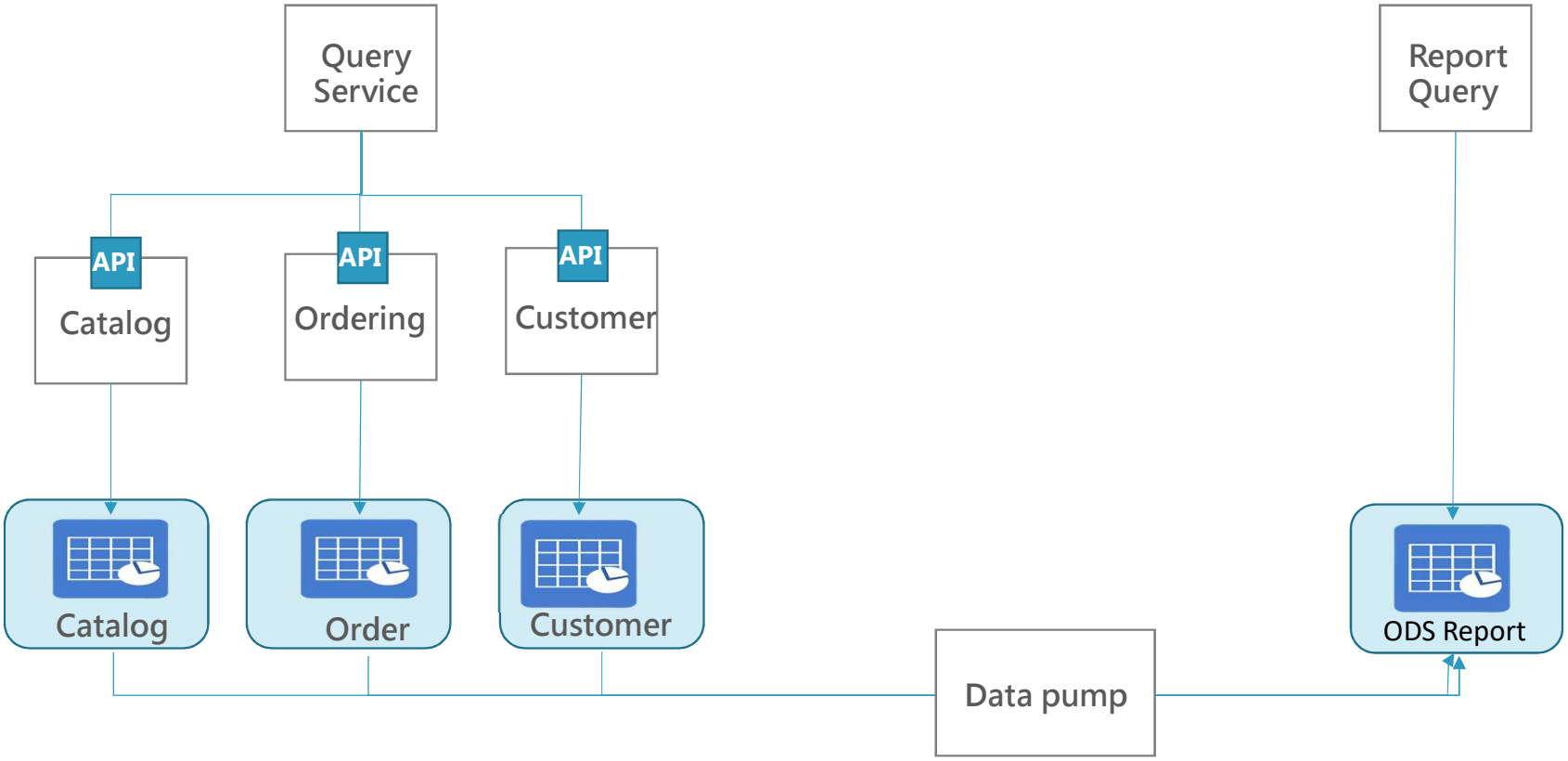
- 同一資料庫可以利用 Table Join 資料庫 Index 等功能提供整合查詢
- 資料庫切割後查詢功能需透過 API 整合資料，
  如回傳資料量過大透過程式比對程式效能問題與網路頻寬負擔大





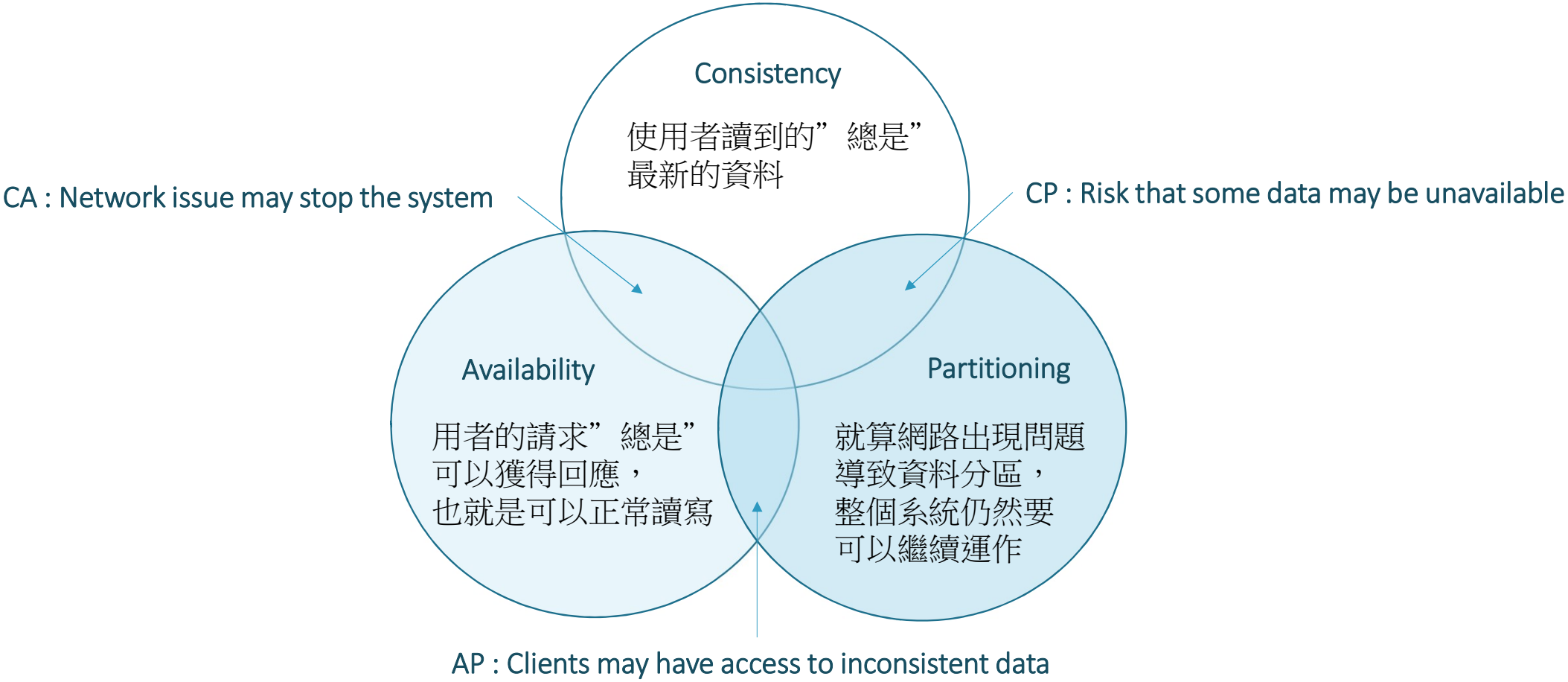
資料重組查詢效能問題

將相關資料整理至Report Server，提供整合查詢使用



微服務是最終一致性的架構

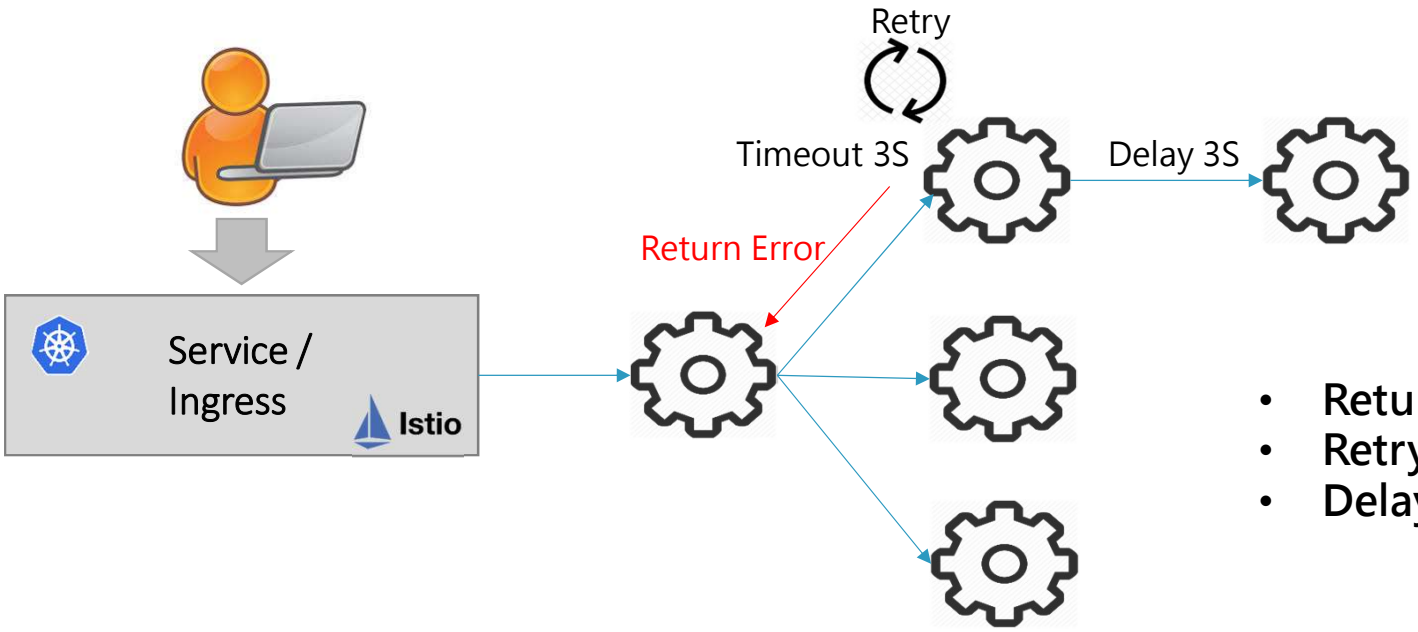
Data Eventual Consistency (CAP)



服務間調用可用性

部份服務不可用時確保服務可用度

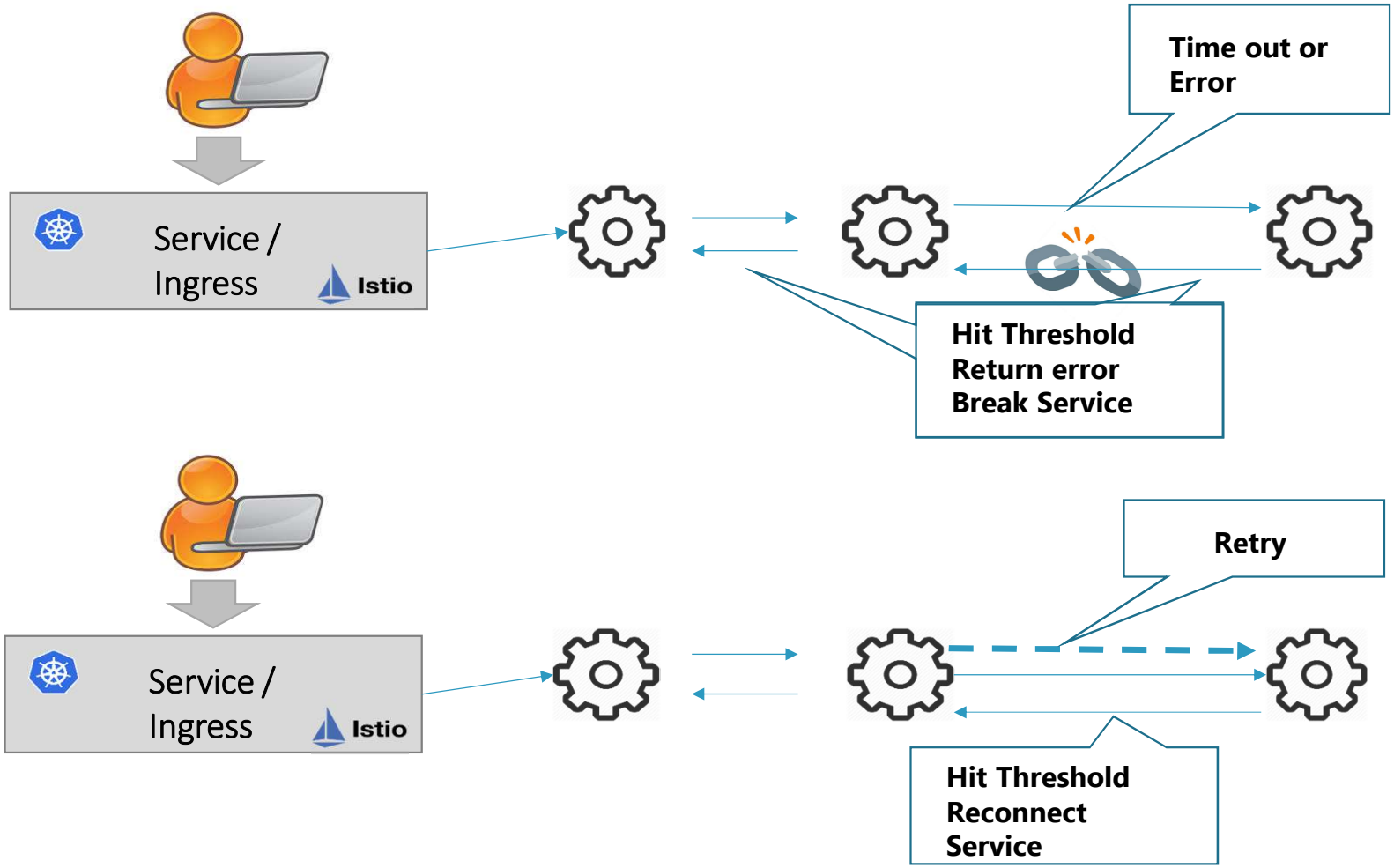
- 網路延遲
- 流量增加系統效能緩慢
- 軟硬體問題



- Return Error
- Retry times
- Delay

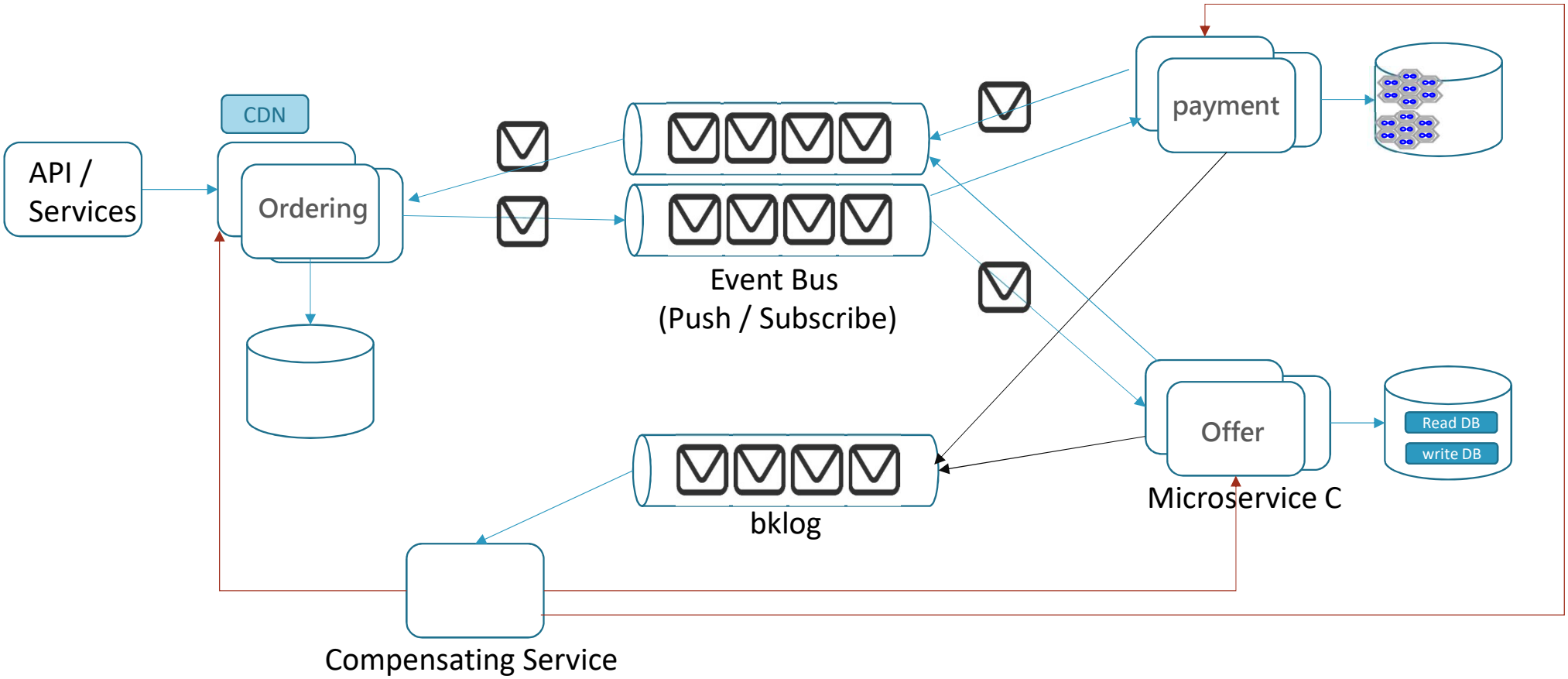
服務間調用可用性

後端服務有問題自動斷開與服務恢復自動復原服務連線



分佈式架構資料一致性問題

Data Eventual Consistency (CAP)



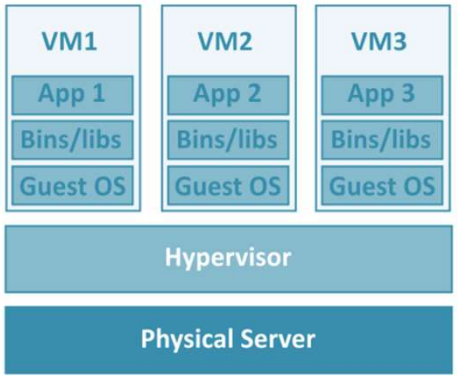
Microservice Architecture Focus

Service

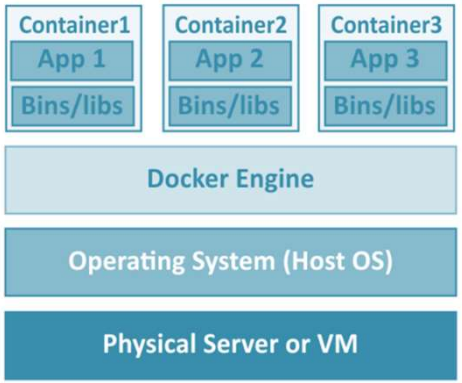
Data

Platform

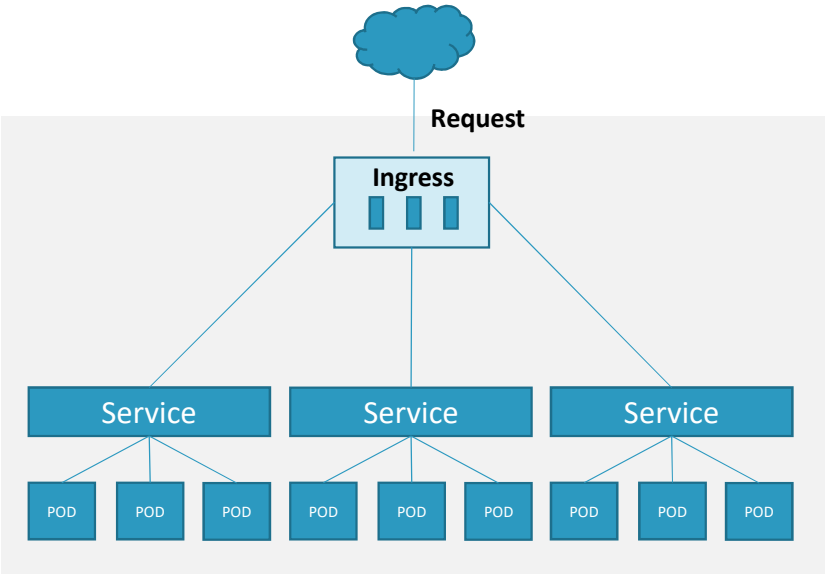
Microservice Platform



Virtualization

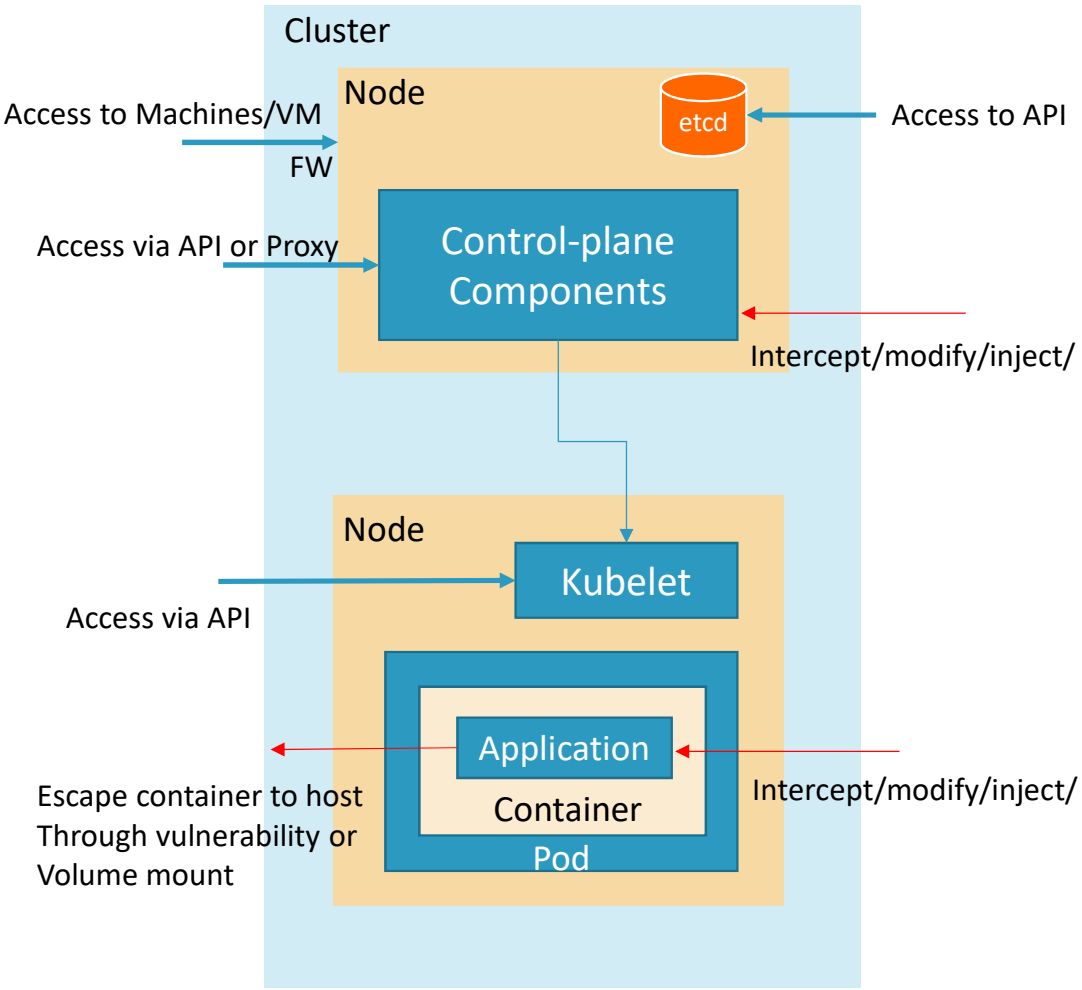


Container



Kubernetes

Kubernetes 平台安全準備事項



Infrastructure

- 通過網路訪問API服務 ，不允許公開在 Internet 上
- 通過網路訪問 Node，不允許公開在 Internet 上
- 需透過網路訪問控制列表控制
- 認證使用 Key control ，Key須妥善保管

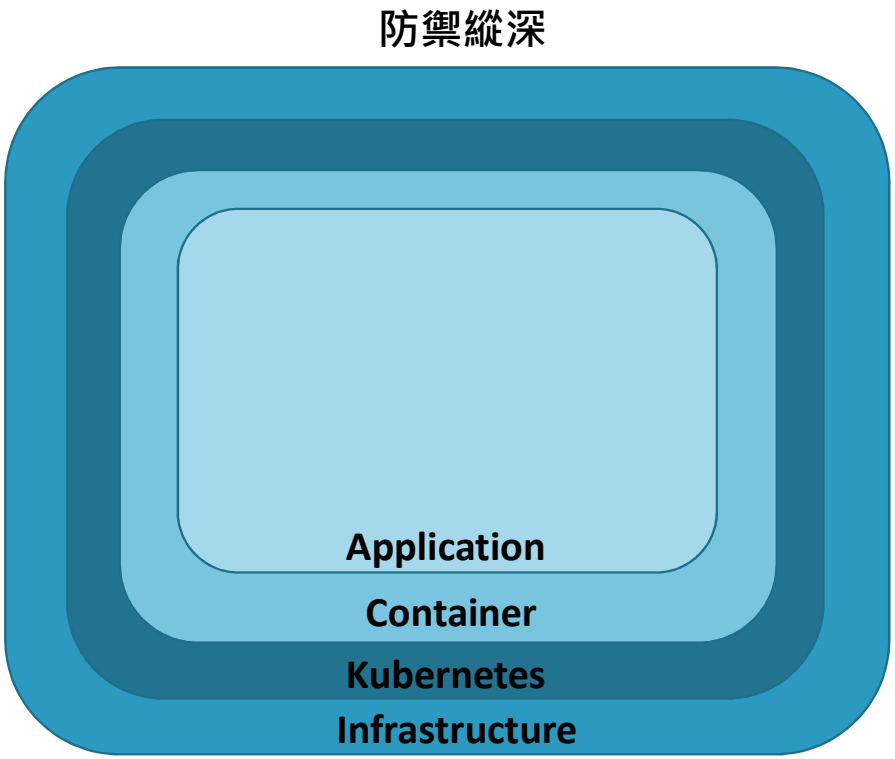
Kubernetes

- Kubernetes 基于角色的權限控管（RBAC)
- 權限最小化原則
- Master ，etcd 加密 ，訪問 etcd通過 TLS 來使用 etcd
- 用新版本 & Turn on log

Pod

- 使用Namespace 隔離敏感工作負載
- 預設 Pod 都是可以互聯的，使用Network policy 控制 Pod 和集群之间的流量
- Pod 安全策略，設定檔檢查，修改預設的組態設定 1. CPU Memory 2. Healthy Check 3 Root 權限

Kubernetes 平台安全準備事項



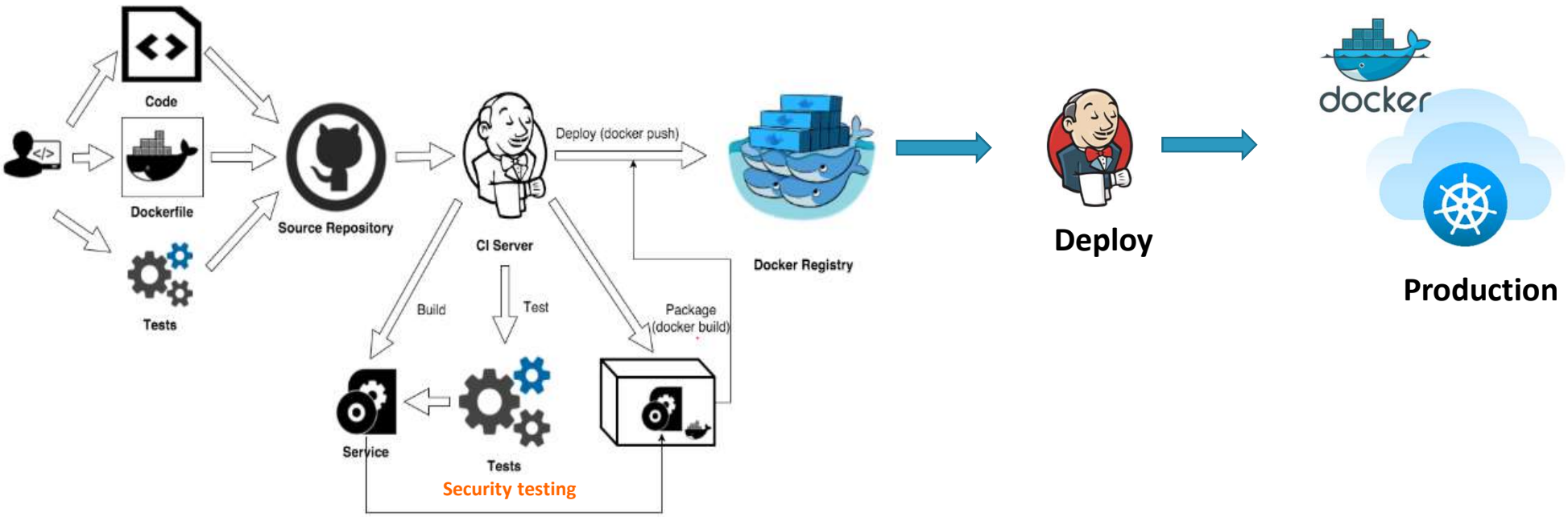
Container

- 必須制定鏡像管理政策
- 最小基礎鏡像，關掉不使用的組件
- 使用最新版本的鏡像
- 定期掃描鏡像是識別修復已知漏洞
- 評估預設設定是否適合
  - 不使用Root 執行程式
  - 不允許特權升級
  - Patch level

AP

- 在運行時監控和限制程式執行資源與及時發現和反應。
- Application Log Distributed Tracing
- DevOps 檢核點 include Code scan / Security testing
- AP 弱點掃描入侵偵測

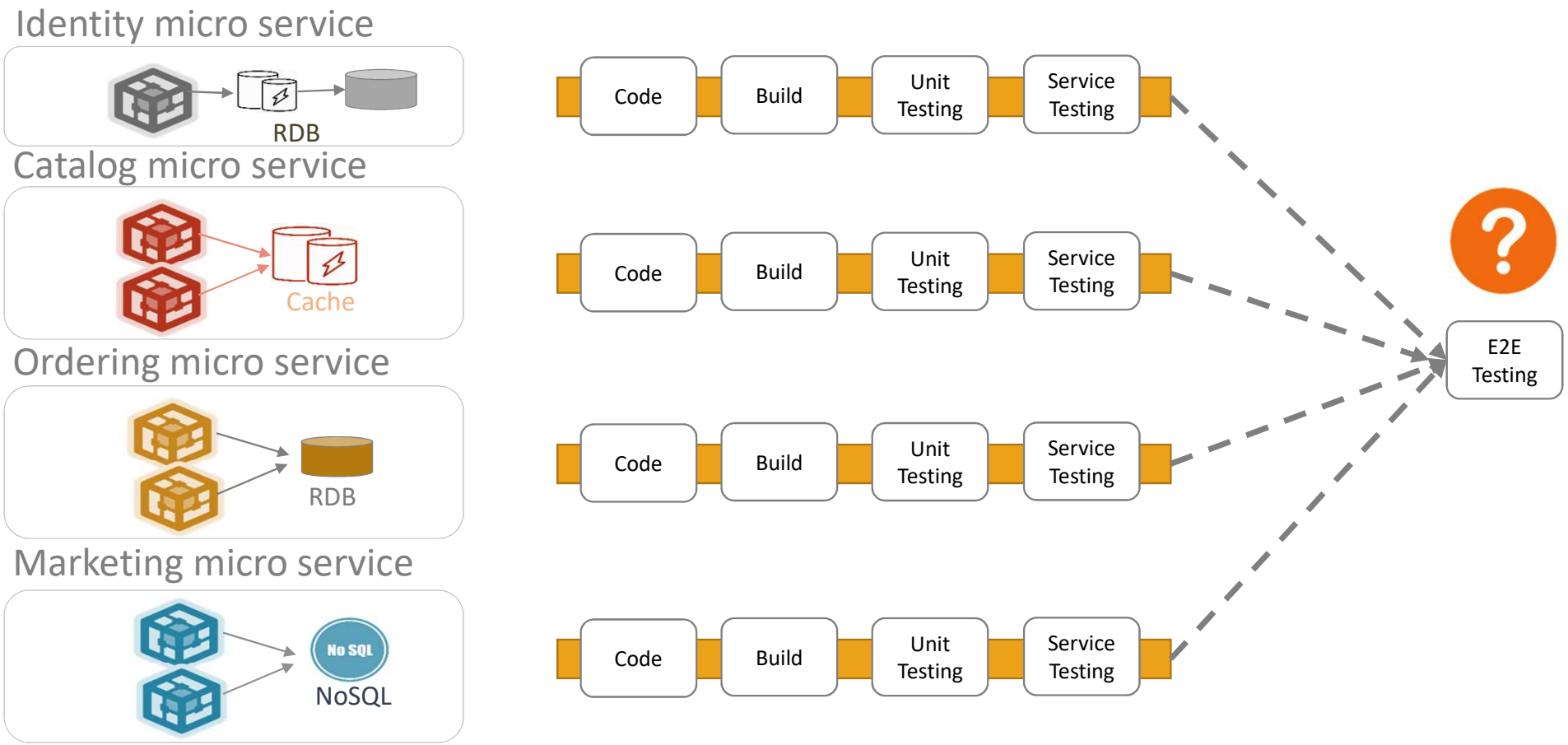
服務上版 DevOps



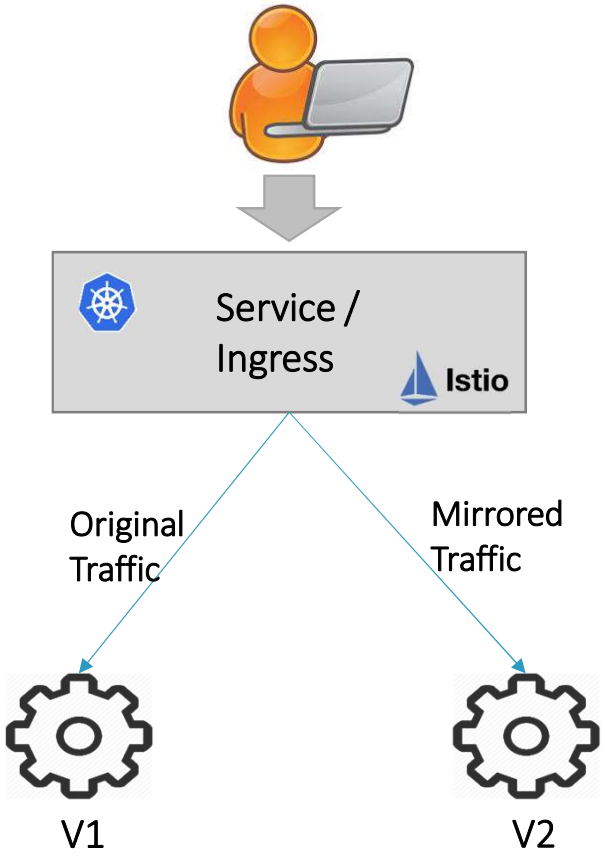
CI

CD

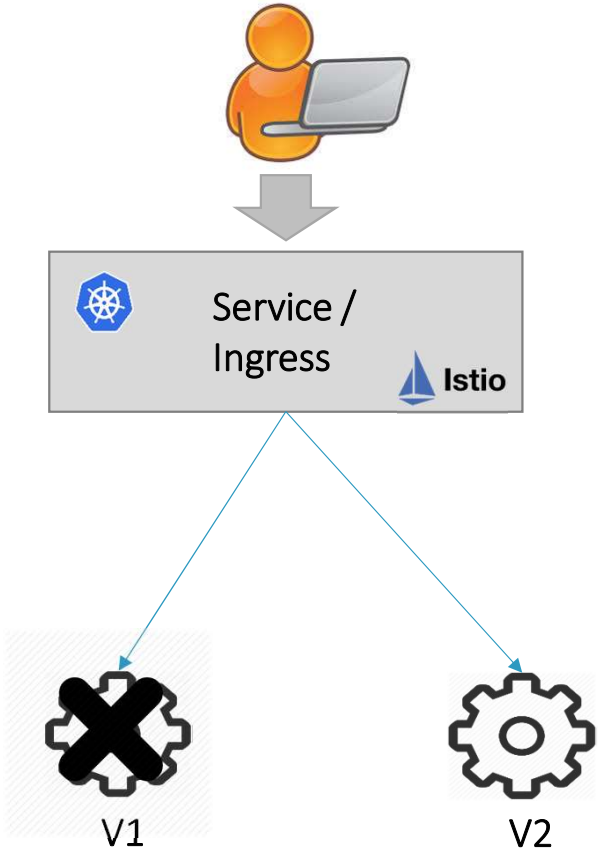
DevOps 測試需求


Identity micro service
RDB
Catalog micro service
Cache
Ordering micro service
RDB
Marketing micro service
No SQL
NoSQL
Code
Build
Unit Testing
Service Testing
Code
Build
Unit Testing
Service Testing
Code
Build
Unit Testing
Service Testing
Code
Build
Unit Testing
Service Testing
E2E Testing

真實環境測試

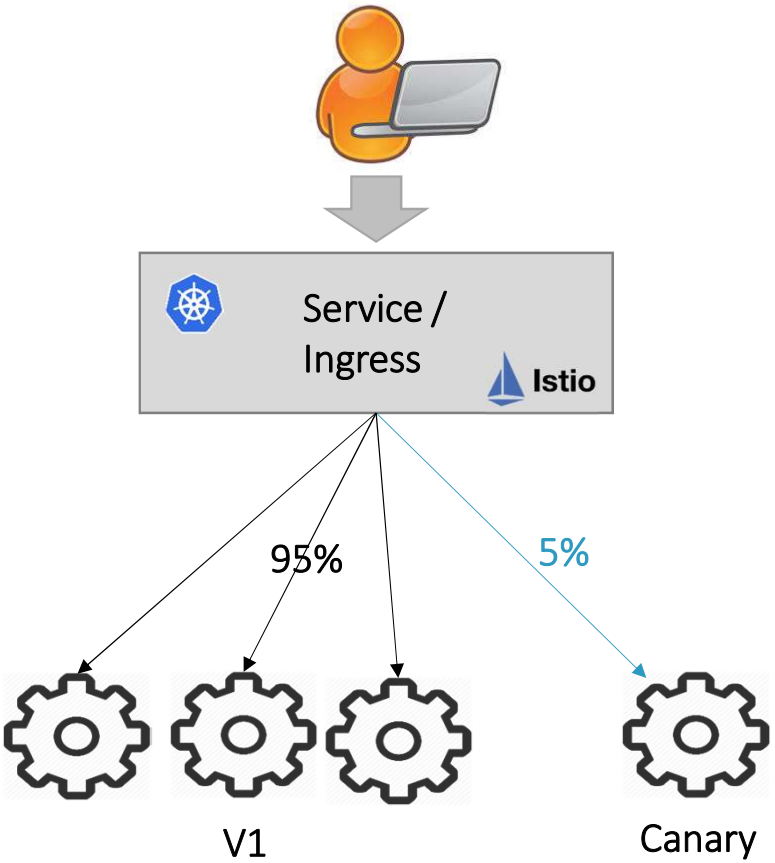


Shadow

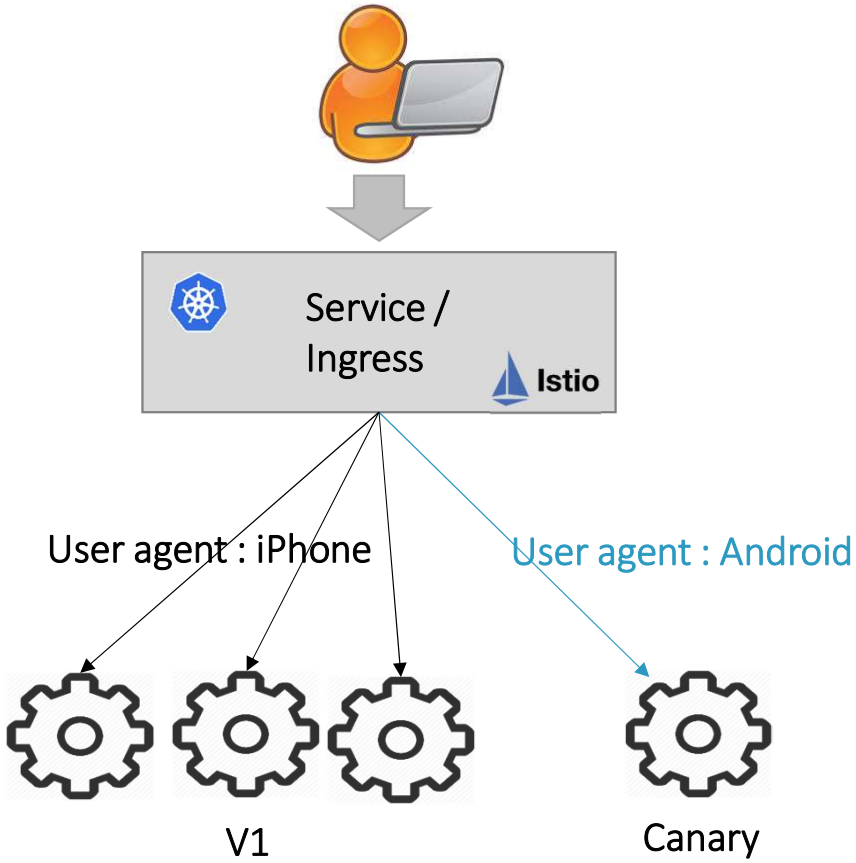


Bule /Green

金絲雀佈署架構



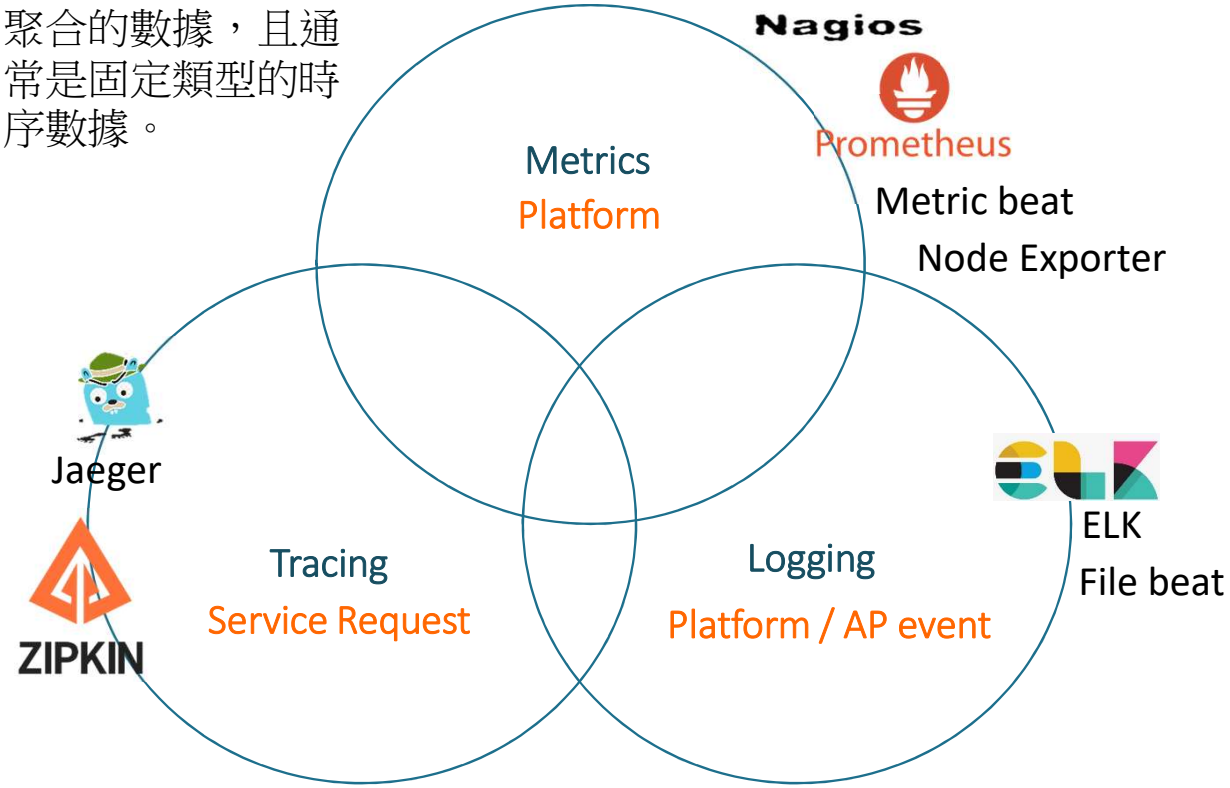
Traffic split



Contain Base

分散式微服務架構監控

聚合的數據，且通常是固定類型的時序數據。



記錄單個服務請求的處理流程，其中包括服務調用順序和處理時間。

用於記錄離散的事件，包含程序執行到某一點或某一階段的詳細信息。例如，應用程序的調試信息或錯誤信息

Summary

- microservice applications focus on **Service, Data, Platform**

- Focus on **service requirement**
- **Not all in** Microservice
- **Non service requirement** is also important

- **Tracing** continue to enhance
- Use **service mesh** to support operation
- K8S **security** enhance