介紹 OpenTelemetry Demo 電商系統

Astronomy Shop - 天文攝影器材商店

專門銷售相機和天文攝影器材的線上商店

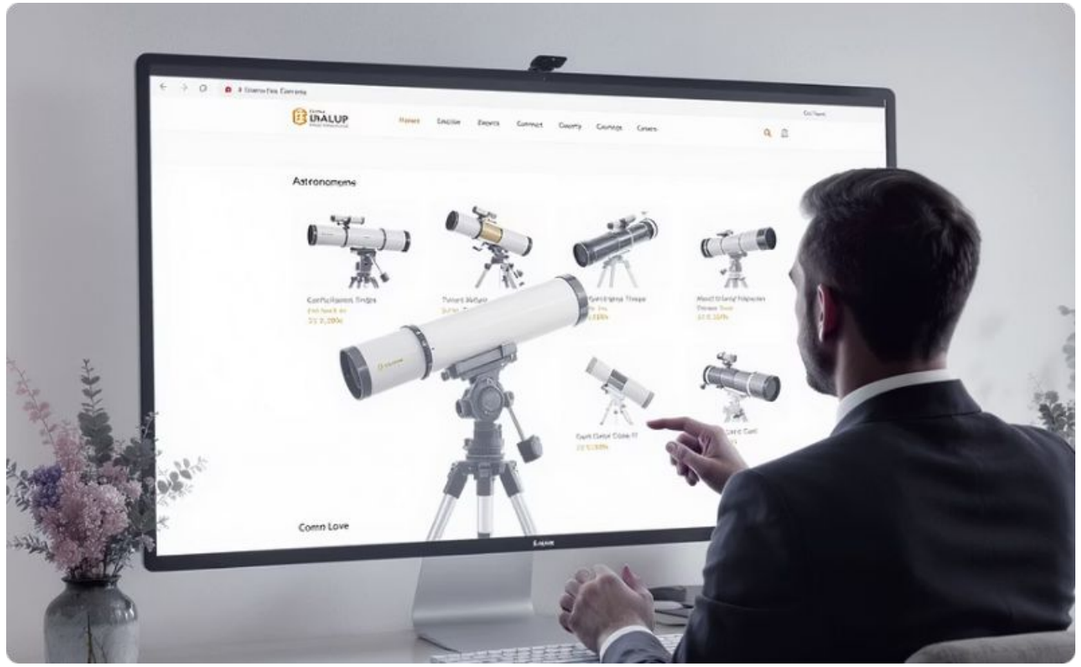
提供商品瀏覽、購物車和結帳功能

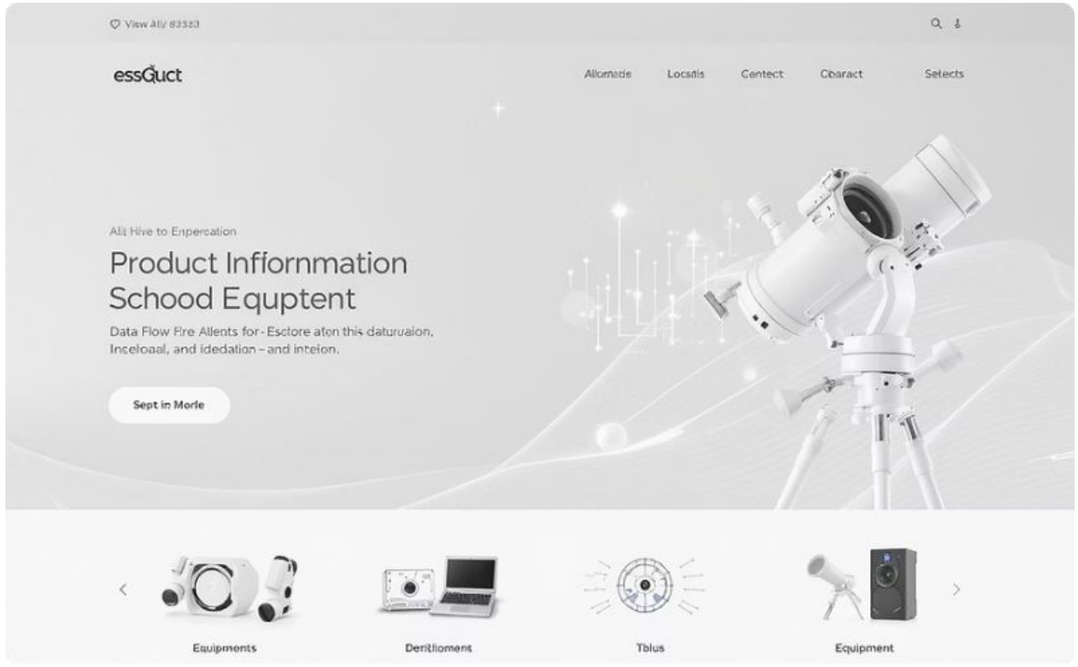
分享人：雷 N

電商網站:

http://odd.ganhua.wang:8080

使用場景:瀏覽產品目錄





使用者

透過前端網站瀏覽產品

查看商品詳細資訊

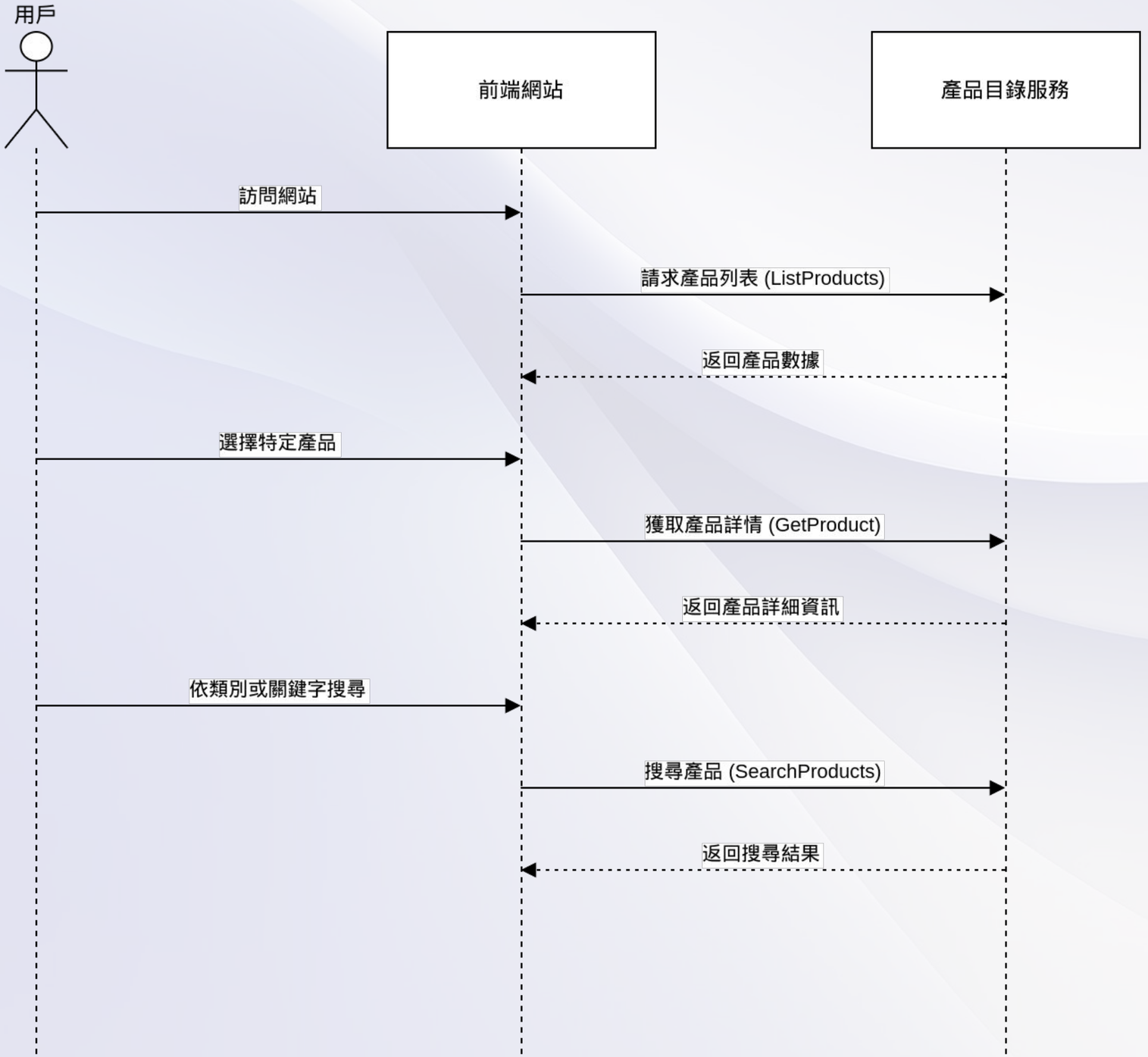
依類別或關鍵字搜尋

產品目錄服務

提供產品資訊

處理商品搜尋請求

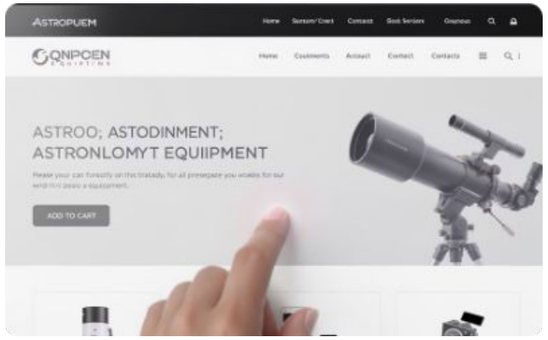
回傳商品詳細規格


用戶
前端網站
產品目錄服務
訪問網站
請求產品列表 (ListProducts)
返回產品數據
選擇特定產品
獲取產品詳情 (GetProduct)
返回產品詳細資訊
依類別或關鍵字搜尋
搜尋產品 (SearchProducts)
返回搜尋結果

使用場景:加入商品至購物車

選擇商品

使用者瀏覽並選擇心儀的天文攝影器材

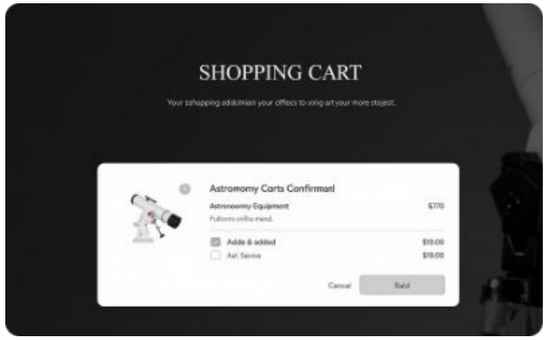


加入購物車

點擊加入購物車按鈕將商品添加到購物清單

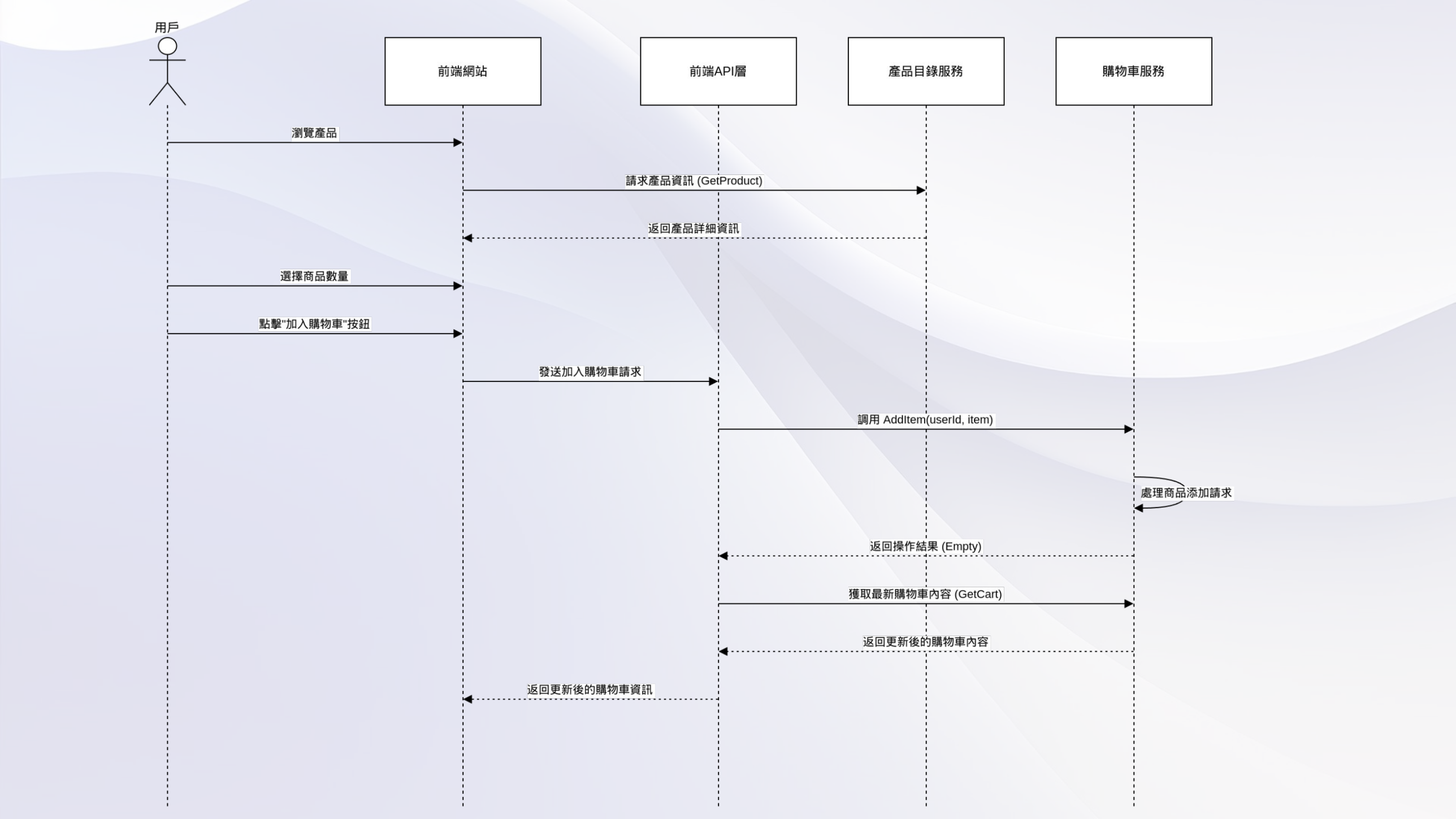
購物車服務

系統後端儲存使用者的購物車項目資訊

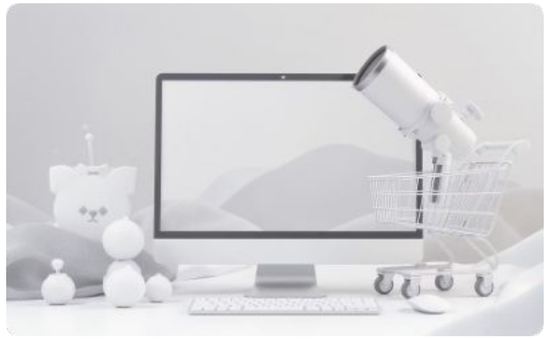


確認更新

系統顯示更新後的購物車內容給使用者確認

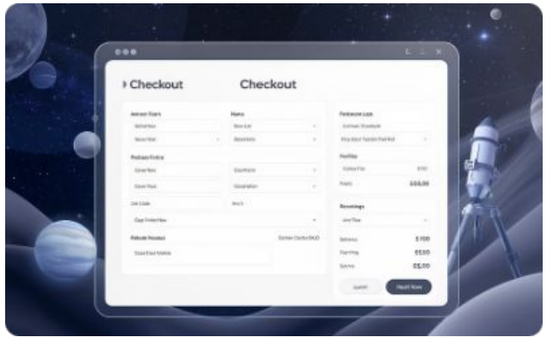

用戶
前端網站
前端API層
產品目錄服務
購物車服務
瀏覽產品
請求產品資訊 (GetProduct)
返回產品詳細資訊
選擇商品數量
點擊"加入購物車"按鈕
發送加入購物車請求
調用 AddItem(userId, item)
處理商品添加請求
返回操作結果 (Empty)
獲取最新購物車內容 (GetCart)
返回更新後的購物車內容
返回更新後的購物車資訊

使用場景:購物車結帳流程



購物車確認

使用者在前端網站上檢視購物車內容



結帳服務

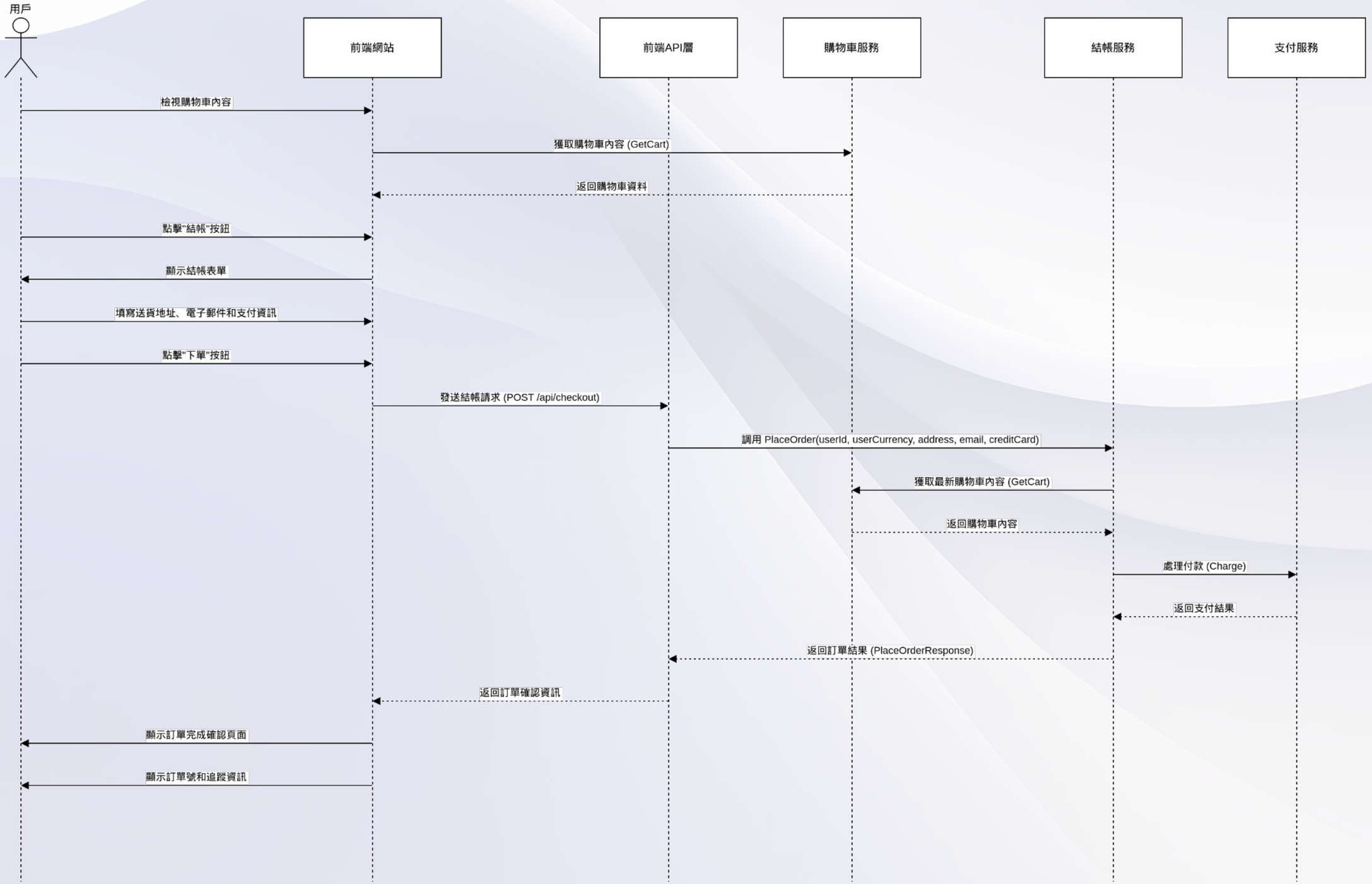
使用者啟動結帳流程,結帳服務處理訂單

支付處理

支付服務驗證並處理信用卡支付

訂單完成

結帳服務生成訂單確認資訊


用戶
前端網站
前端API層
購物車服務
結帳服務
支付服務
檢視購物車內容
獲取購物車內容 (GetCart)
返回購物車資料
點擊"結帳"按鈕
顯示結帳表單
填寫送貨地址、電子郵件和支付資訊
點擊"下單"按鈕
發送結帳請求 (POST /api/checkout)
調用 PlaceOrder(userId, userCurrency, address, email, creditCard)
獲取最新購物車內容 (GetCart)
返回購物車內容
處理付款 (Charge)
返回支付結果
返回訂單結果 (PlaceOrderResponse)
返回訂單確認資訊
顯示訂單完成確認頁面
顯示訂單號和追蹤資訊

Dashboard 監控

基礎系統指標

監控各服務系統層級效能

R.E.D. 指標

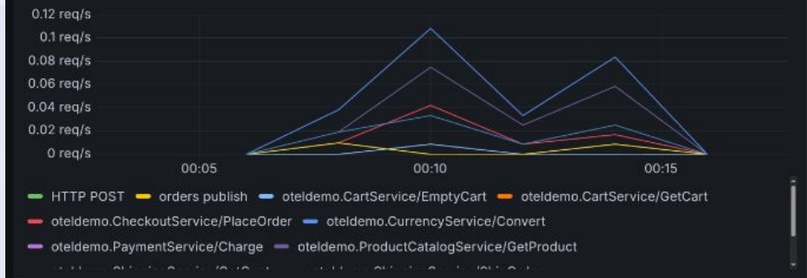
各服務的 Rate、Error、Duration 關鍵指標

Service checkout

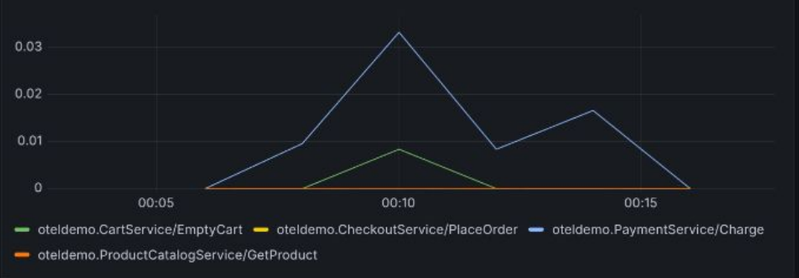
Last 15 minutes | Refresh

Spanmetrics (RED metrics)

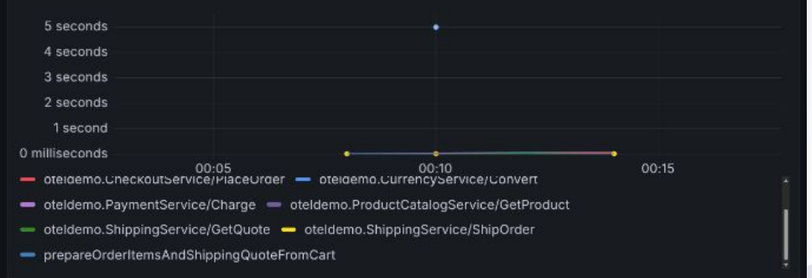
Requests Rate by Span Name



Error Rate by Span Name



Average Duration by Span Name



Application Log Records

Log Records by Severity

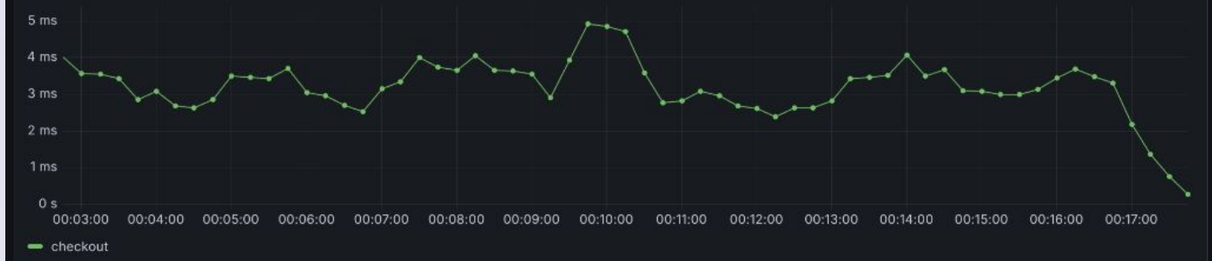
severity_text	Value #A
INFO	8

Log Records (100 recent entries)

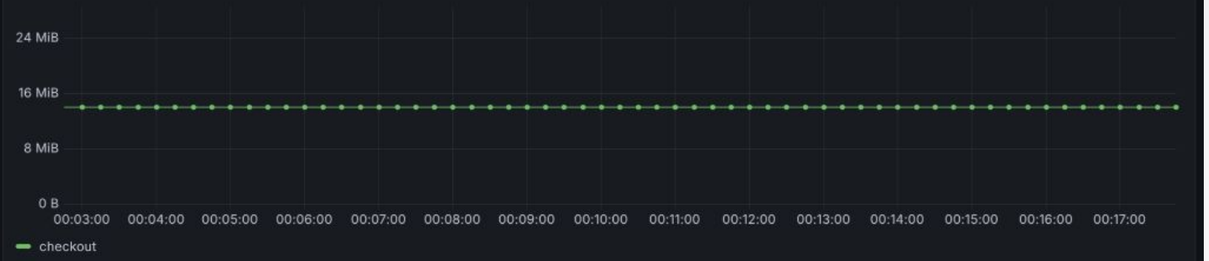
```
> starting to listen on tcp: "[::]:5050"
> Successfully initialized new client
> client/brokers registered new broker #1 at kafka:9092
> Connected to broker at kafka:9092 (unregistered)
> client/metadata fetching metadata for all topics from broker kafka:9092
> ClientID is the default of 'sarama', you should consider setting it to something application-specific.
> ClientID is the default of 'sarama', you should consider setting it to something application-specific.
> Initializing new client
```

Application Metrics

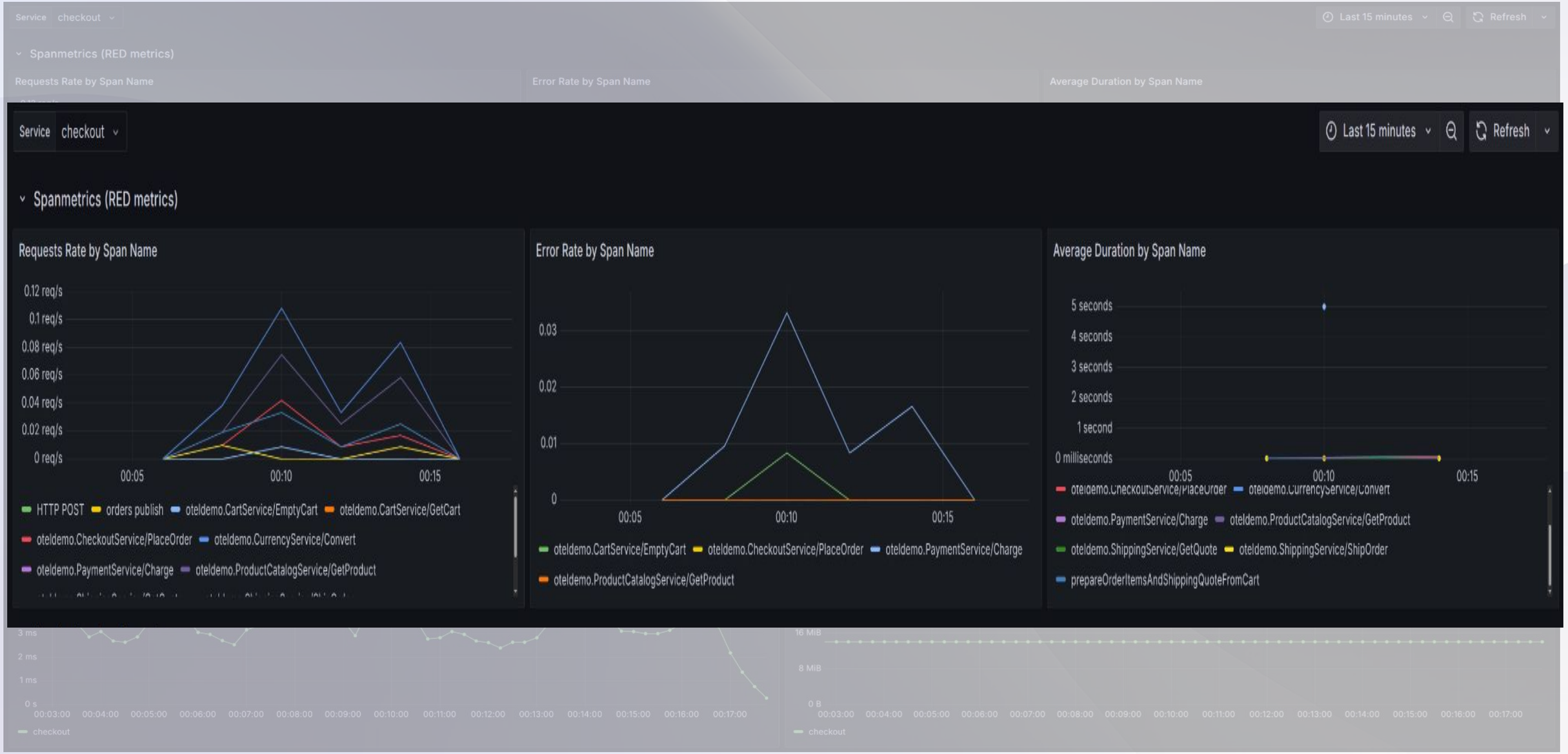
container_cpu



container_memory_max_usage_bytes




Service
checkout
Last 15 minutes
Refresh
Spanmetrics (RED metrics)
Requests Rate by Span Name
0.12 req/s
0.1 req/s
0.08 req/s
0.06 req/s
Error Rate by Span Name
0.03
0.02
Average Duration by Span Name
5 seconds
4 seconds
3 seconds
Application Metrics
container_cpu
5 ms
4 ms
3 ms
2 ms
1 ms
0 s
00:03:00 00:04:00 00:05:00 00:06:00 00:07:00 00:08:00 00:09:00 00:10:00 00:11:00 00:12:00 00:13:00 00:14:00 00:15:00 00:16:00 00:17:00
checkout
container_memory_max_usage_bytes
24 MiB
16 MiB
8 MiB
0 B
00:03:00 00:04:00 00:05:00 00:06:00 00:07:00 00:08:00 00:09:00 00:10:00 00:11:00 00:12:00 00:13:00 00:14:00 00:15:00 00:16:00 00:17:00
checkout


Service checkout
Last 15 minutes
Refresh
Spanmetrics (RED metrics)
Requests Rate by Span Name
Error Rate by Span Name
Average Duration by Span Name
0.12 req/s
0.1 req/s
0.08 req/s
0.06 req/s
0.04 req/s
0.02 req/s
0 req/s
00:05
00:10
00:15
HTTP POST
orders publish
oteldemo.CartService/EmptyCart
oteldemo.CartService/GetCart
oteldemo.CheckoutService/PlaceOrder
oteldemo.CurrencyService/Convert
oteldemo.PaymentService/Charge
oteldemo.ProductCatalogService/GetProduct
0.03
0.02
0.01
0
oteldemo.CartService/EmptyCart
oteldemo.CheckoutService/PlaceOrder
oteldemo.PaymentService/Charge
oteldemo.ProductCatalogService/GetProduct
5 seconds
4 seconds
3 seconds
2 seconds
1 second
0 milliseconds
oteldemo.CheckoutService/PlaceOrder
oteldemo.CurrencyService/Convert
oteldemo.PaymentService/Charge
oteldemo.ProductCatalogService/GetProduct
oteldemo.ShippingService/GetQuote
oteldemo.ShippingService/ShipOrder
prepareOrderItemsAndShippingQuoteFromCart
3 ms
2 ms
1 ms
0 s
16 MiB
8 MiB
0 B
00:03:00
00:04:00
00:05:00
00:06:00
00:07:00
00:08:00
00:09:00
00:10:00
00:11:00
00:12:00
00:13:00
00:14:00
00:15:00
00:16:00
00:17:00
checkout


Service checkout
Last 15 minutes
Refresh
Spanmetrics (RED metrics)
Requests Rate by Span Name
0.12 req/s
0.1 req/s
0.08 req/s
0.06 req/s
0.04 req/s
0.02 req/s
0 req/s
00:05
00:10
00:15
HTTP POST
orders publish
oteldemo.CartService/EmptyCart
oteldemo.CartService/GetCart
oteldemo.CheckoutService/PlaceOrder
oteldemo.CurrencyService/Convert
oteldemo.PaymentService/Charge
oteldemo.ProductCatalogService/GetProduct
Error Rate by Span Name
0.03
0.02
0.01
0
00:05
00:10
00:15
oteldemo.CartService/EmptyCart
oteldemo.CheckoutService/PlaceOrder
oteldemo.PaymentService/Charge
oteldemo.ProductCatalogService/GetProduct
Average Duration by Span Name
5 seconds
4 seconds
3 seconds
2 seconds
1 second
0 milliseconds
00:05
00:10
00:15
oteldemo.CheckoutService/PlaceOrder
oteldemo.CurrencyService/Convert
oteldemo.PaymentService/Charge
oteldemo.ProductCatalogService/GetProduct
oteldemo.ShippingService/GetQuote
oteldemo.ShippingService/ShipOrder
prepareOrderItemsAndShippingQuoteFromCart
Application Log Records
Log Records by Severity
severity_text
Value #A
INFO
8
Log Records (100 recent entries)
Connected to broker at kafka:9092 (unregistered)
client/metadata fetching metadata for all topics from broker kafka:9092
ClientID is the default of 'sarama', you should consider setting it to something application-specific.
ClientID is the default of 'sarama', you should consider setting it to something application-specific.
Initializing new client
Application Metrics
container_cpu
5 ms
4 ms
3 ms
2 ms
1 ms
0 s
00:03:00
00:04:00
00:05:00
00:06:00
00:07:00
00:08:00
00:09:00
00:10:00
00:11:00
00:12:00
00:13:00
00:14:00
00:15:00
00:16:00
00:17:00
checkout
container_memory_max_usage_bytes
24 MiB
16 MiB
8 MiB
0 B
00:03:00
00:04:00
00:05:00
00:06:00
00:07:00
00:08:00
00:09:00
00:10:00
00:11:00
00:12:00
00:13:00
00:14:00
00:15:00
00:16:00
00:17:00
checkout

這樣就夠了嘛?

ODD 可觀測性驅動開發

簡介 ODD

簡介 OpenTelemetry

SLO Dashboard

分享人：雷 N 

ODD（可觀測性驅動開發）

Day 0 開發設計

以系統可觀測性為核心考量

提供快速找出問題的能力

定位系統或程式碼中的問題位置

持續改進

根據監控結果優化系統表現

有效率的監控

針對 SLO 目標添加檢測能力

左移可觀測性

左移的另一個好處：成本（儲存與時間 ）

如果在 Day 2 才開始回溯系統，需要花大量時間了解整個系統的設計與架構，並對系統和程式進行大幅修改，很容易會收集到大量無關的數據和雜訊。

如果在 Day 0 就開始設計系統，可以針對這次的需求進行設計，只收集關鍵區域的相關數據，避免浪費資源去蒐集目前無用的資訊。

減少業務損失

縮短故障修復時間

OpenTelemetry

統一資料標準

OpenTelemetry 整合**上下文**於 Traces、Logs 和 Metrics，提供一致且全面的可觀測數據。

開發階段整合

在開發階段即整合 OpenTelemetry，實現可觀測性向左移動，及早發現並解決潛在問題。

快速定位問題

避免在多個工具間切換，OpenTelemetry 協助快速精準地定位問題根源，實現「**One dashboard to rule them all**」的目標。

關鍵理念：收集和分析數據，提升迭代效率。

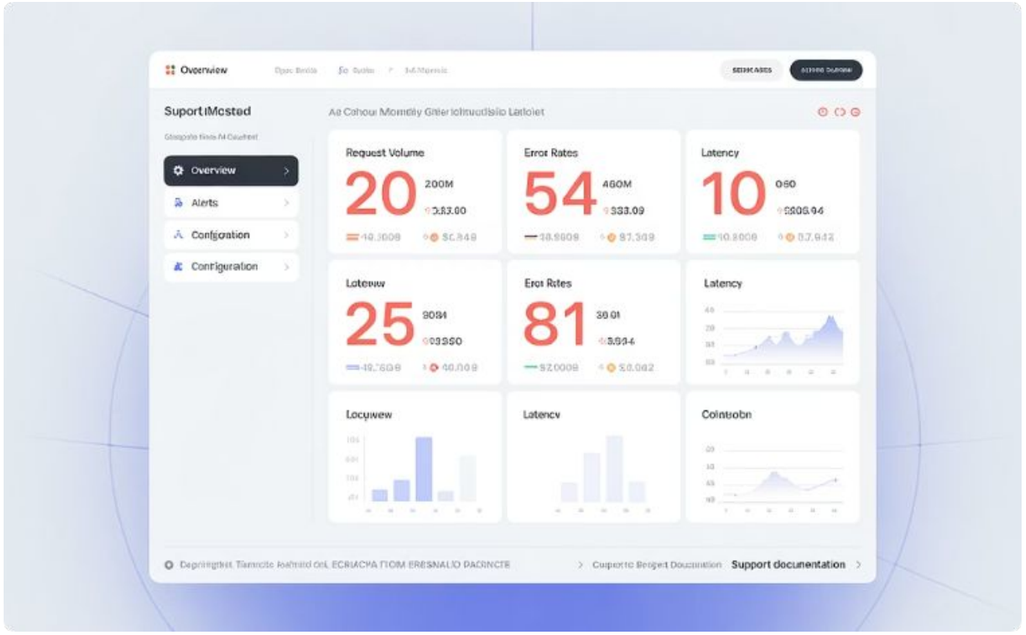
OpenTelemetry

Auto vs Manual Instrumentation

1

Auto Instrumentation

快速為每個服務建立監控指標，例如 RED 指標。這是一個很好的起點，能快速掌握系統整體運行狀況。



OpenTelemetry

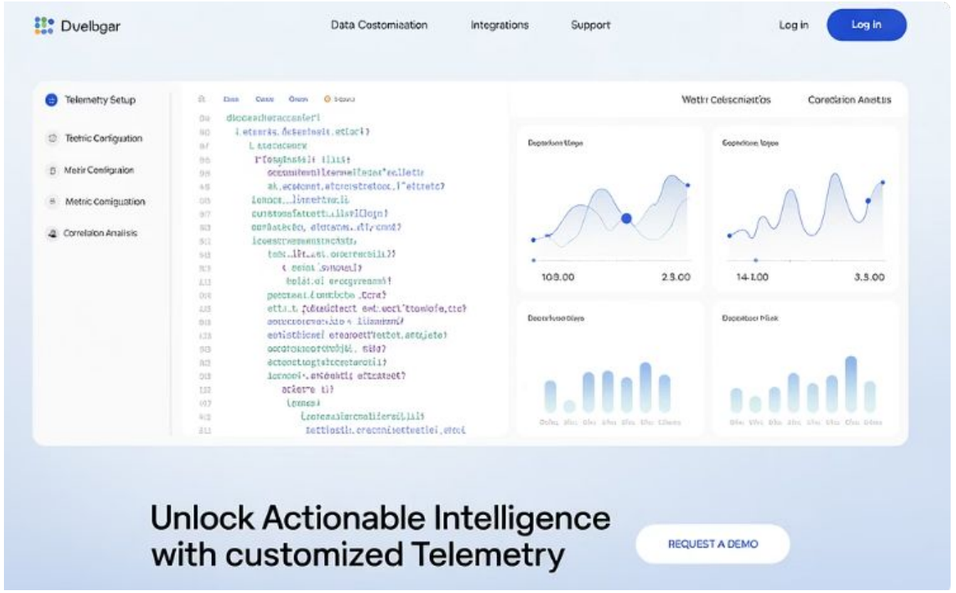
Auto vs Manual Instrumentation

2

Manual Instrumentation

根據具體業務需求，生成更多自定義的遙測數據，並建立數據之間的關聯性。有助於深入了解系統行為，支持更精細的性能優化與問題診斷。

Auto 和 Manual 檢測相輔相成，前者提供快速部署的基礎指標，後者則進一步擴展數據的深度與業務相關性。



Manual Instrumentation 起手示

SDK 設定全局 Provider

設定 Telemetry Provider	建立全局提供者
設定資料輸出	選擇儲存與分析方式
套用資源屬性	標記服務名稱與版本
設定採樣策略	控制遙測數據蒐集頻率

```go
func initLogProvider() {
    // 設定 Telemetry Data 的 Exporter
    logExporter, err := otlploggrpc.New(ctx)

    // 設定 Telemetry Data Provider
    provider := sdklog.NewLoggerProvider(
        sdklog.WithProcessor(sdklog.NewBatchProcessor(logExporter)),
        sdklog.WithResource(initResource()),
    )

    // 將 Provider 設定成全局物件
    global.SetLoggerProvider(lp)

}
```

Manual Instrumentation 第二招

Trace 與 Span 的生活化例子

Trace（追蹤）

麥當勞點餐的整個流程

從進店到拿到餐點的完整過程

Span（片段）

點餐 Span（2分鐘）

付款 Span（1分鐘）

製作漢堡 Span（3分鐘）

打包 Span（1分鐘）

點餐過程中的 Trace 與 Span

1. **整體點餐流程**

   完整 Trace 包含多個 Span

2. **各個處理階段**

   每個 Span 代表一個獨立步驟

3. **服務間的關聯**

   追蹤請求如何跨多個服務流動

4. **效能分析**

   識別系統瓶頸與延遲來源

Trace

點餐過程

Span

點餐

Span

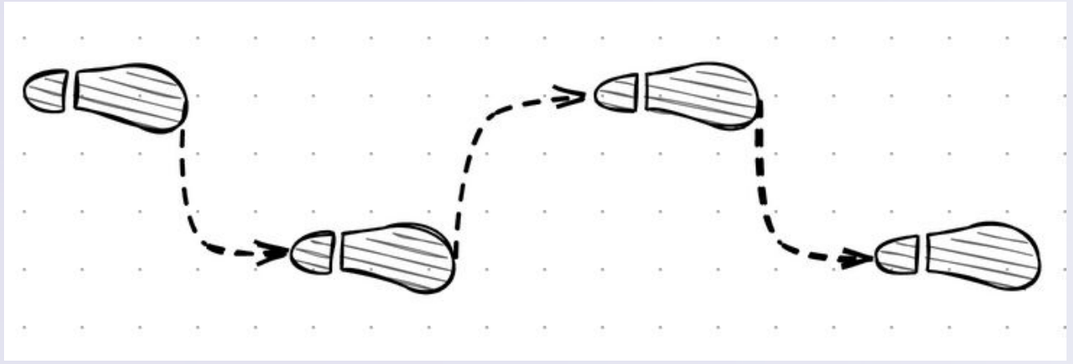
付款

Span

製作

Span

打包



PlaceOrder 關鍵流程追蹤

1. 開始訂單處理

初始化訂單追蹤

2. 取得購物車資訊

載入使用者購物項目

3. 處理支付流程

信用卡支付處理與驗證

4. 完成訂單建立

儲存訂單並通知使用者

```go
func (cs *checkout) PlaceOrder(ctx context.Context, req *pb.PlaceOrderRequest) (*pb.PlaceOrderResponse, error)
{   // 從當前的 Context 中獲取 span
    // span 可以想像成是一個追蹤片段，記錄著這個請求的軌跡
    span := trace.SpanFromContext(ctx)

    // 為這個 span 添加一些重要資訊（就像是給行李箱貼上標籤）
    // 這些資訊可以幫助我們之後在觀察系統時，更容易找到特定用戶的請求
    span.SetAttributes(
        // 記錄使用者 ID，方便之後查詢特定用戶的請求
        attribute.String("app.user.id", req.UserId),
        // 記錄用戶使用的貨幣，有助於分析不同地區的訂單
        attribute.String("app.user.currency", req.UserCurrency),
    )

    // 一般的 log 記錄，配合 trace 可以更完整地了解系統運作
    log.WithContext(ctx).Infof("[PlaceOrder] user_id=%q user_currency=%q order_id=%q",
        req.UserId, req.UserCurrency, orderID.String())
}
```

Trace aabbccdd

下單過程

aabbccdd-xy1 Span 收到請求

aabbccdd-aac Span 確認訂單

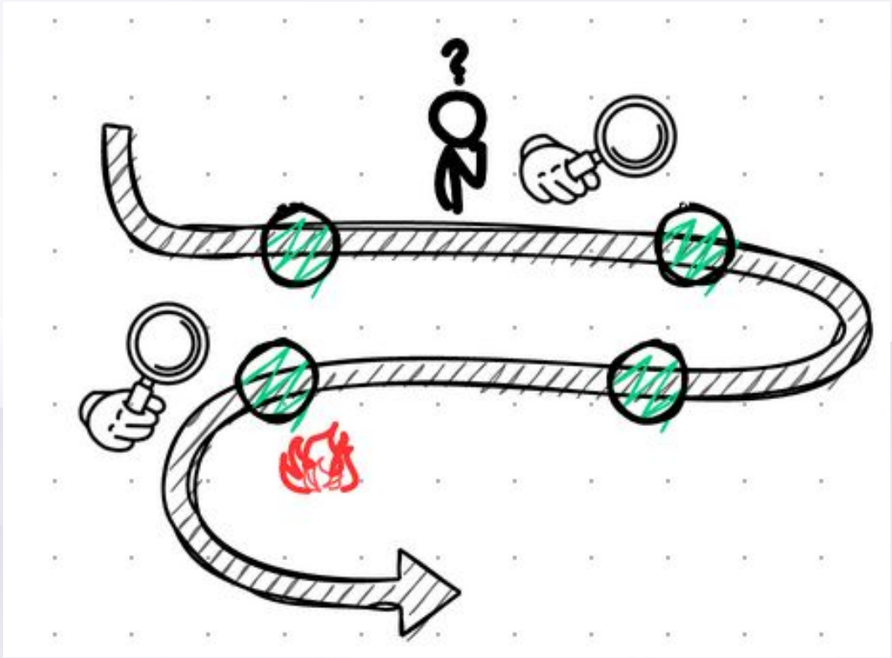
aabbccdd-bbc Span 計算總金額

aabbccdd-daa Span 信用卡付款

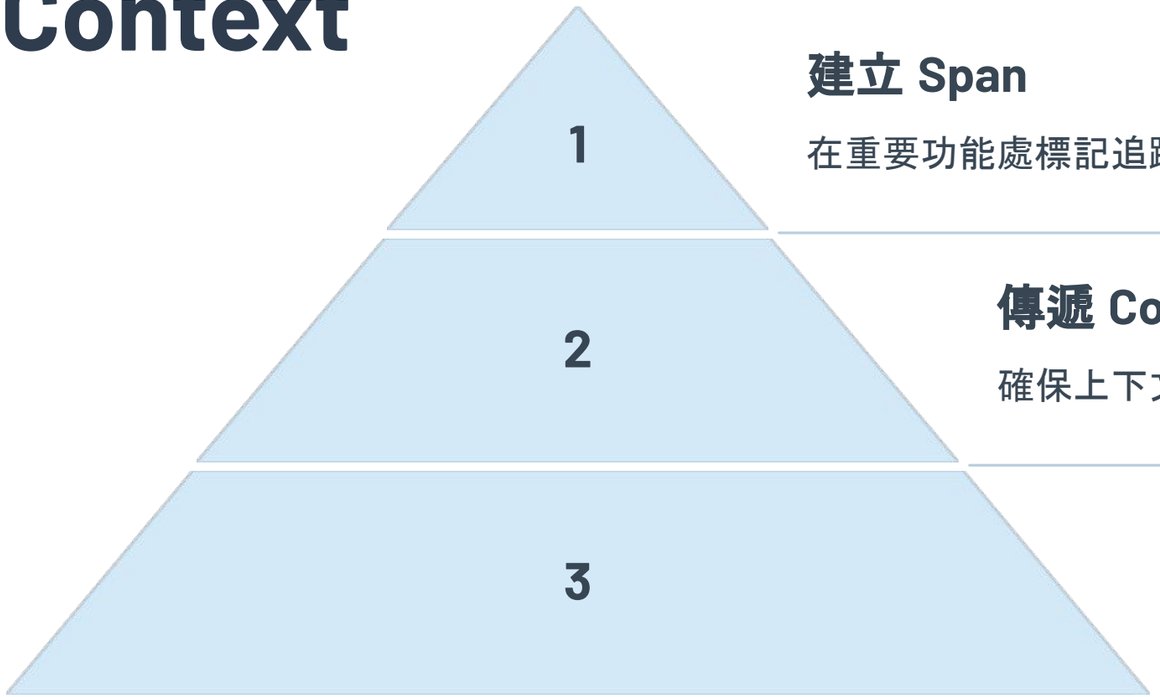
aabbccdd-aas Span 產生外送資料

aabbccdd-da2 Span 清空購物車

aabbccdd-y23 Span 完成流程



SDK 設定 建立 Span 與 Log 共用 Context



建立 Span

在重要功能處標記追蹤片段，記錄操作時間與屬性

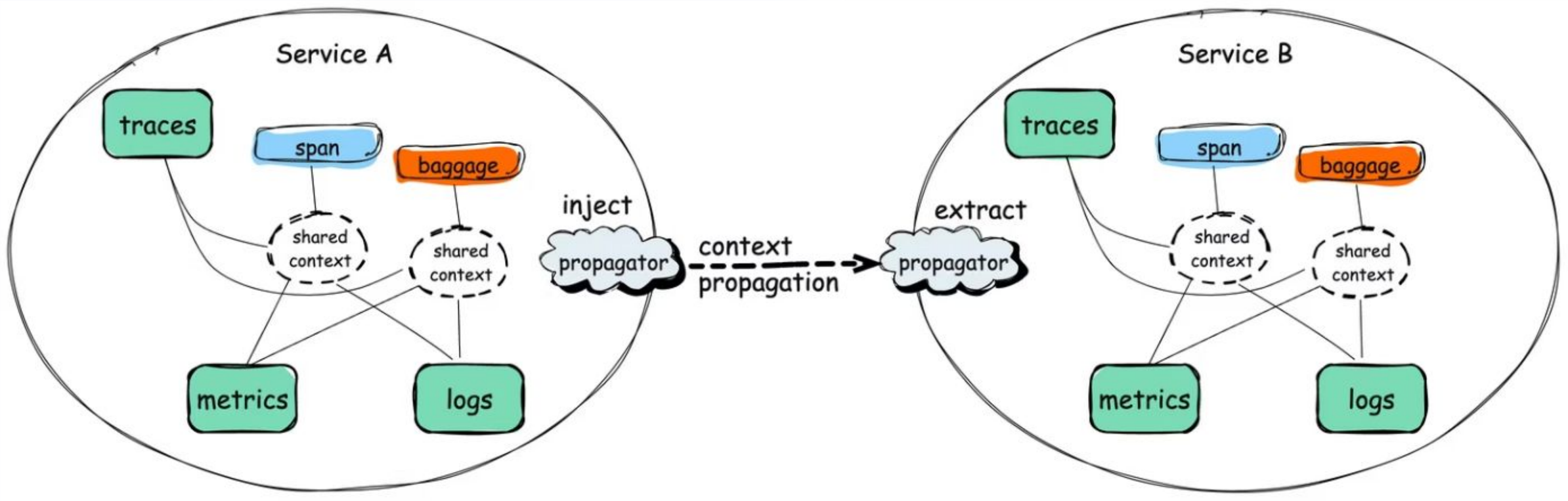
傳遞 Context

確保上下文在服務間保持，維持分散式追蹤連貫性

Log 關聯

日誌與追蹤資料連結，簡化問題定位與診斷

Trace Context 傳播與關聯



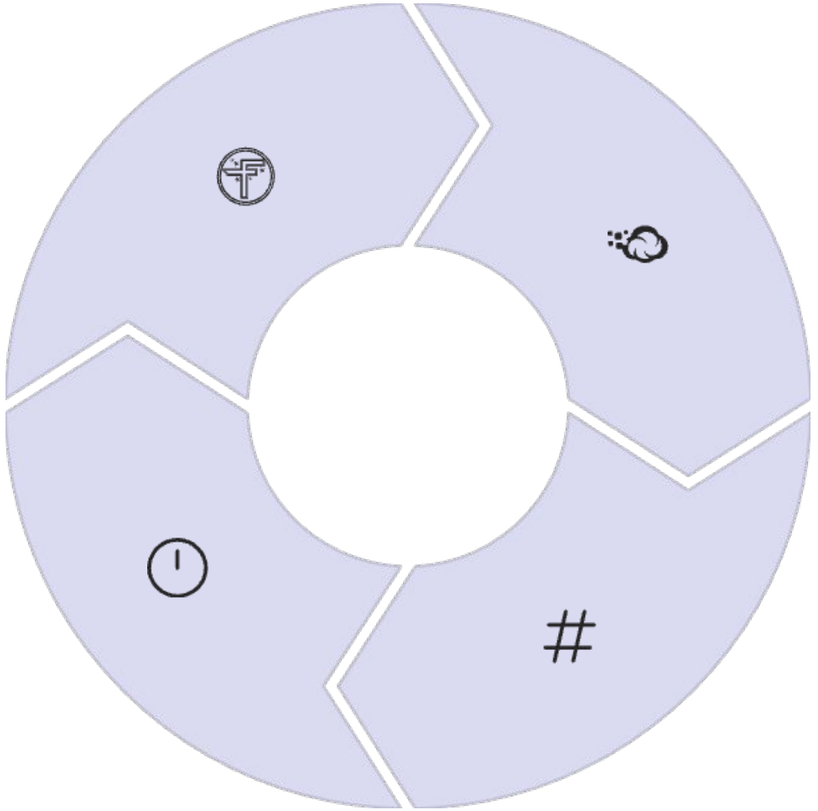
遙測資料的 4 大組成

Span Context

包含 Trace ID、Span ID 與父 Span ID，用於建立分散式追蹤的完整層級關係，確保跨服務請求能被正確關聯

時間資訊

記錄操作的開始與結束時間戳，計算持續時間與延遲情況，支援性能分析與瓶頸識別



Resource Context

記錄服務名稱、版本、部署環境、容器 ID 與主機資訊等，幫助識別產生遙測資料的來源與運行環境

屬性資訊

包含自定義標籤、HTTP 狀態碼、請求參數、用戶 ID 等關鍵業務資訊，便於篩選、分組與根因分析

Trace Span 可否取代 Log？

Trace Span

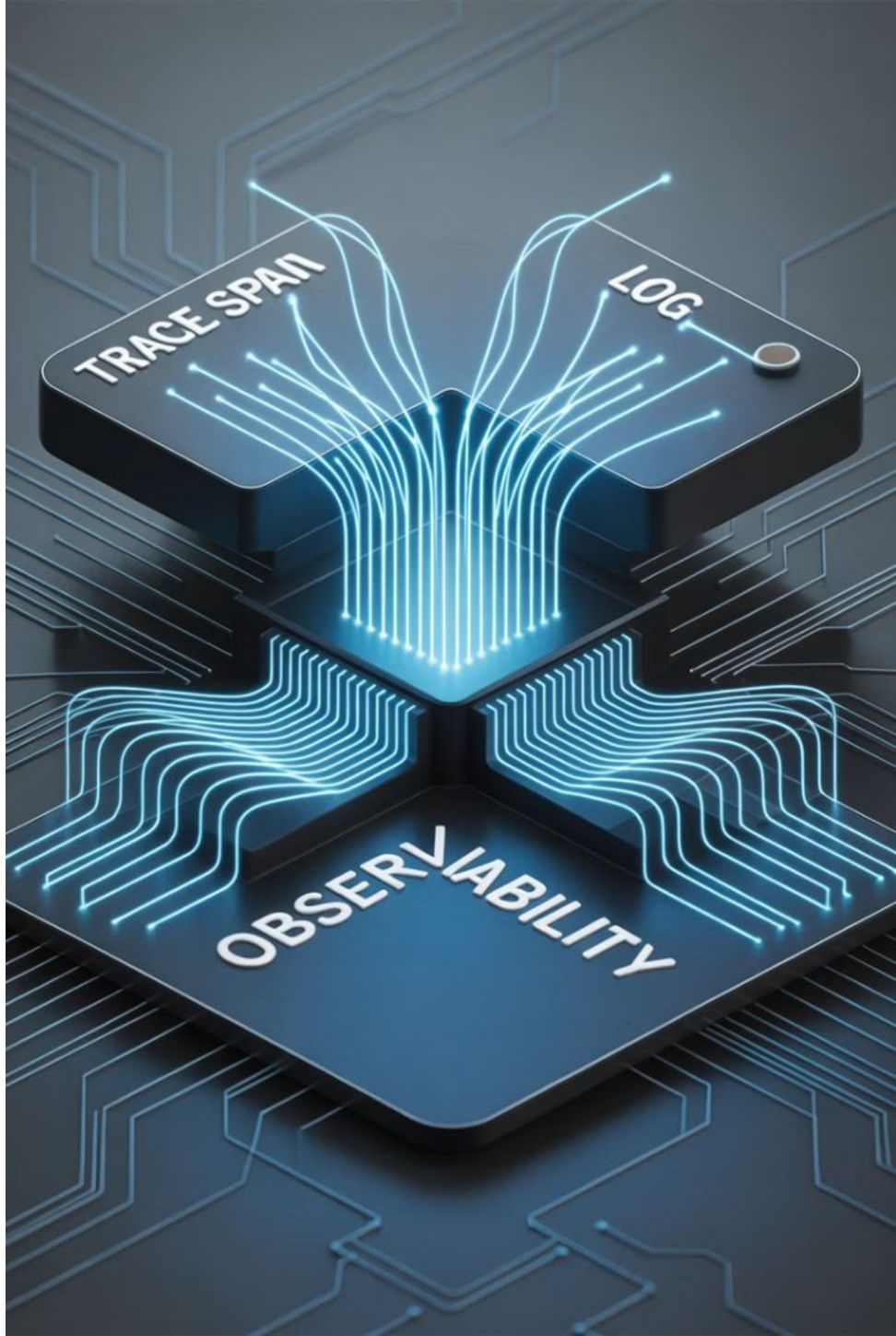
Span Context 主要用於建立分散式追蹤的完整層級關係，確保跨服務請求能被正確關聯，但因成本與效能考量，並非所有 Trace 資料都會被長期儲存。

Log

一旦產生就會被持久化儲存，提供系統行為的歷史記錄與詳細上下文，適合深入的問題診斷與稽核追蹤。

Observability

理想的可觀測性能力應將這兩種遙測資料相互關聯：在 Log 中包含 Trace ID 與 Span ID，同時在 Span 中記錄關鍵事件。這樣既能享有 Tracing 的全域視角，又能保留 Logging 的細節資訊，互相補強而非替代。



SLO Dashboard 實踐應用

關鍵業務指標

電商平台主要交易流程

- 購物車回應時間
- 結帳成功率
- 支付處理可用性

錯誤預算監控

衡量系統可接受的失敗量

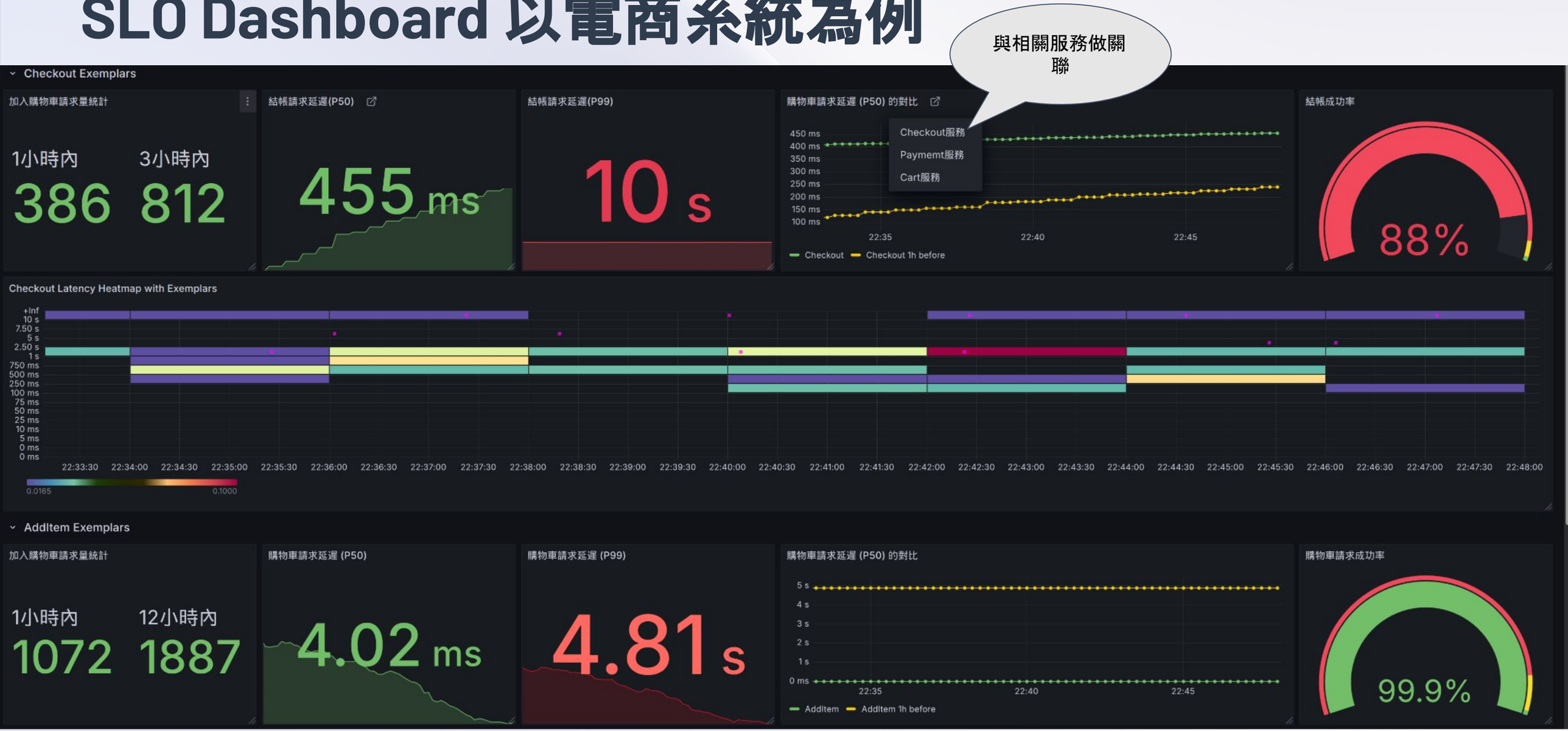
- 月度錯誤預算
- 消耗速率警告
- 歷史趨勢比較

關聯性分析

業務指標與系統性能關聯

- 依賴服務影響
- 基礎設施關聯
- 容量規劃依據

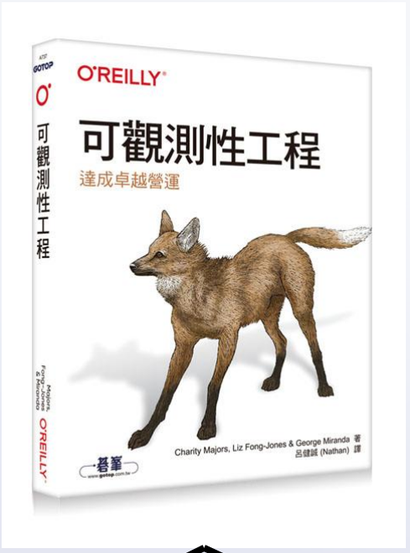
SLO Dashboard 以電商系統為例

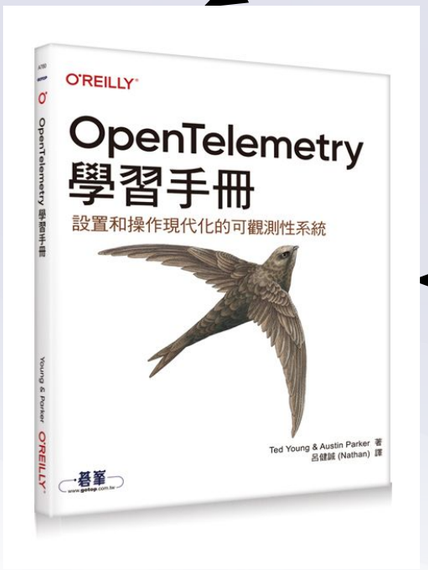


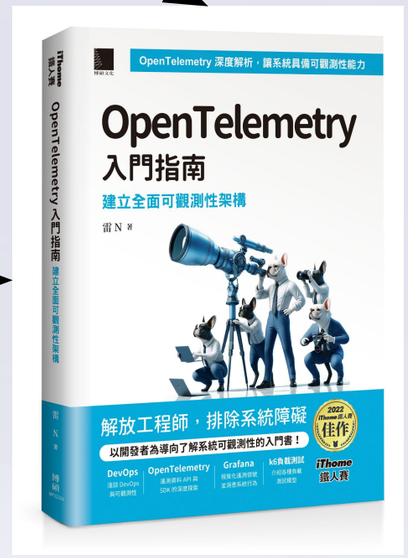
電商網站:
http://odd.ganhua.wang:8080

監控系統:
http://odd.ganhua.wang:8080/grafana

著作


O'REILLY
可觀測性工程
達成卓越營運
Charity Majors, Liz Fong-Jones & George Miranda 著
呂健誠 (Nathan) 譯
碁峰


O'REILLY
OpenTelemetry
學習手冊
設置和操作現代化的可觀測性系統
Ted Young & Austin Parker 著
呂健誠 (Nathan) 譯
碁峰


OpenTelemetry 深度解析，讓系統具備可觀測性能力
OpenTelemetry
入門指南
建立全面可觀測性架構
雷 N 著
解放工程師，排除系統障礙
以開發者為導向了解系統可觀測性的入門書！
2022 iThome 鐵人賽 佳作
iThome 鐵人賽

問卷

https://r.itho.me/dods25