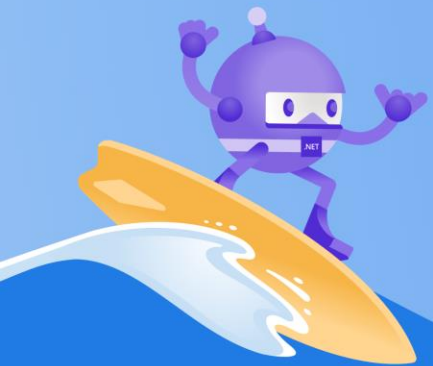


.NET Conf

探索 .NET 新世界



微服務下的 NET Core MongoDB Redis

Ben Lu 陸浩翔

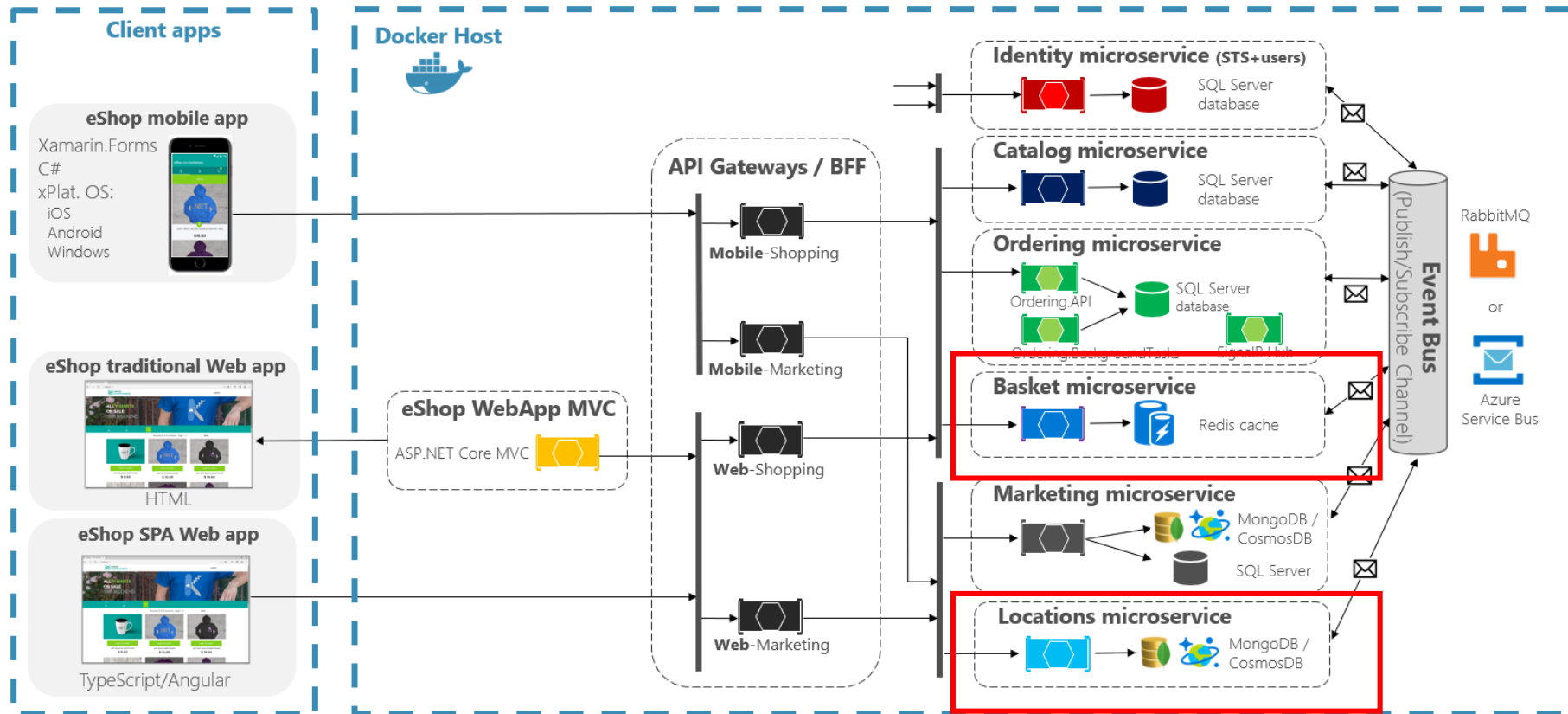


**RMDBS 與 noSQL 是合作關係、非取代對方
各自有其優點與適用的情境**



eShopOnContainers reference application

(Development environment architecture)



<https://docs.microsoft.com/zh-tw/dotnet/architecture/microservices/multi-container-microservice-net-applications/implement-api-gateways-with-ocelot>

假設要做一個的 Uber

想像一下需要完成的功能項目

- **App**每**20**秒 發出一台車子的**GPS**位置
- **User**可以 使用**APP** 來送出叫車的需求
- 司機 可以 使用**APP** 來接受**User**叫車
- 當司機接到任務後，他就不能再接其他工作了
- 當一個區域，呼叫**Uber**的人很多，但是車子很少，你需要把這個區域的車資提高，吸引**Uber**司機前往這個區域載客

適合Redis 資料庫的場域

變動非常的頻繁，但是又不具需要永久儲存特性的資訊。

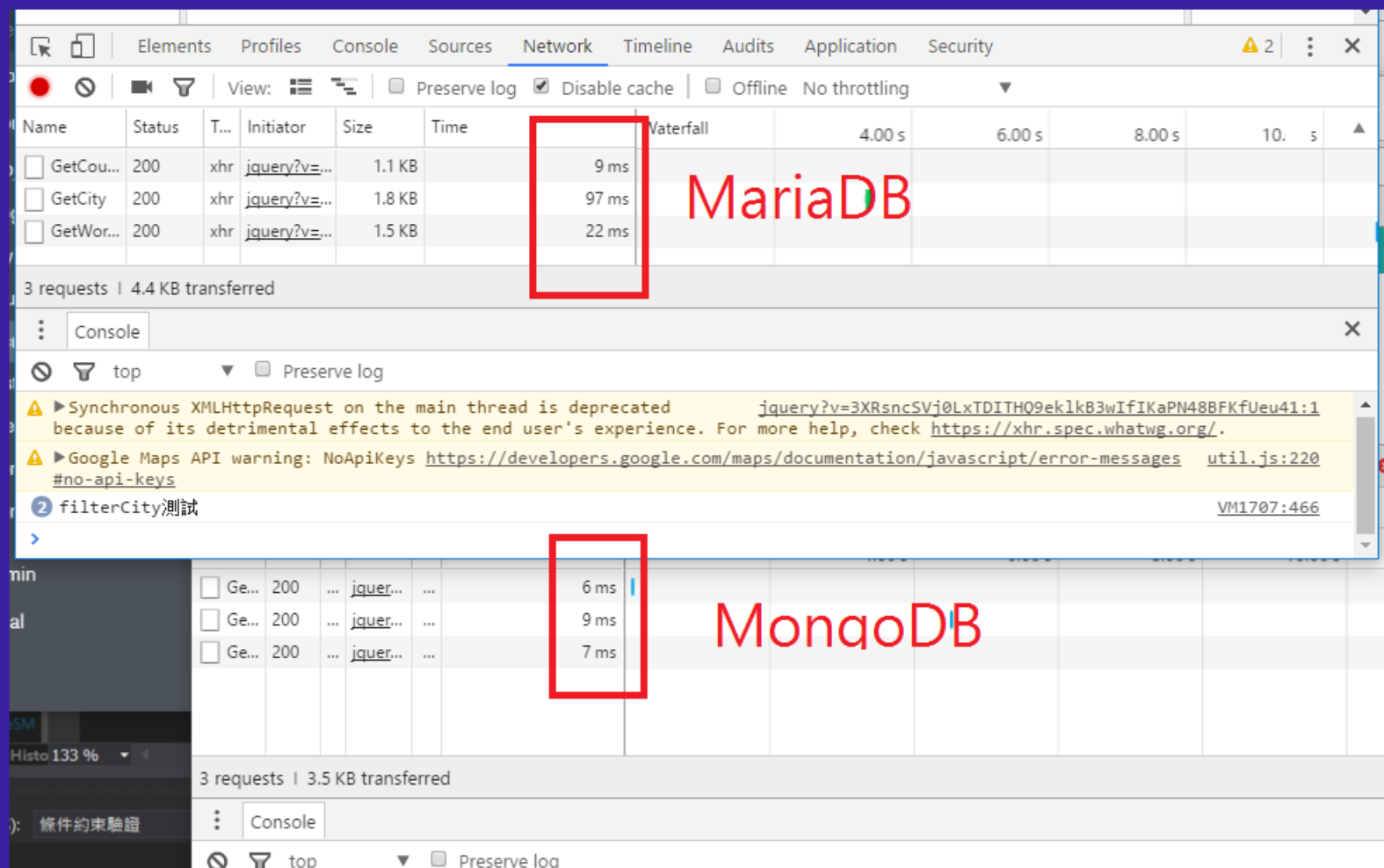
- 尋車過程以鄰近的行政區為主。(e.g. 將台灣做行政區域切割，車子動態註冊到區域)
- User App 輸入呼叫車子的需求，此時User 可能會尋不到車，或是任何理由而放棄他要原本叫車的需求。(e.g. 呼叫了Uber，剛好有小黃來了,取消原本的呼叫。)
- Backend 使用 cookie 模式驗證，Cookie 加密金鑰的儲存。
- 類 FaceBook Messenger 的聊天功能。(e.g. SignalR會給每一個連線一個 Connection id做識別。而User經常在電腦上一次就開一堆頁籤。
- 具時效性的訂票、訂位系統。User Booking了位置，還沒結帳前，有五分鐘可以猶豫。在此之前，位子並不會被釋放。

適合 MongoDB 資料庫的場域

主體的關聯資料龐大。此類資料會在 Join 指令時消耗大量的效能

- 一輛車子一天產生 約2000 筆GPS資訊。5000台車跑Uber。一天就會有超過1000萬筆的資料。
- FB 的文章按讚，一篇文章可能有時會到數百人、數千人按讚。
- 聊天室的訊息內容。你的Line上面會有一堆聊天室，群組聊天。每個聊天室裡面，訊息內容一定相對龐大。

結構不同所造成的效能差異



我們在企業內開發常看或是你正面臨的問題



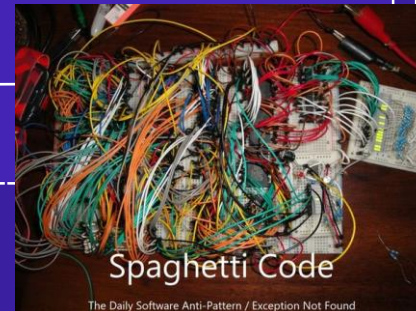
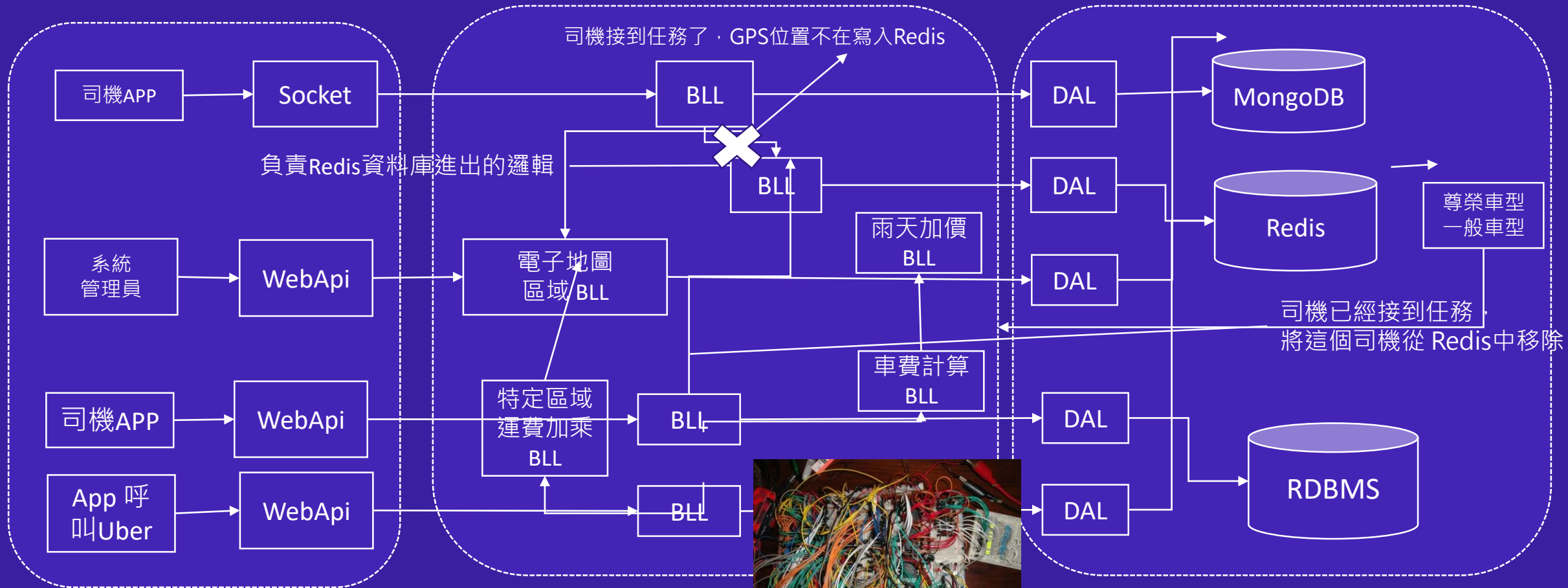


需求變動是常態，是正常的

- 一個案子在開發時，需求一定是在過程中改來改去不斷變動
- 許多延伸想法往往會在專案開發到一半後如雨後春筍的冒出
- 隨著加入測試的人越來越多，大家都提出越來越多的想法，有些需求還不一定能擋住.....。
- 商務邏輯的增加，新需求加入最早專案，會造成專案越漸臃腫，難以維護。
- 當系統上線一段時間，隨著商業發展，開始擴展原本的架構。
例如: 新的金流支付出現，你的付款模式變得複雜....。

Spaghetti code 出現在商業邏輯間的依賴

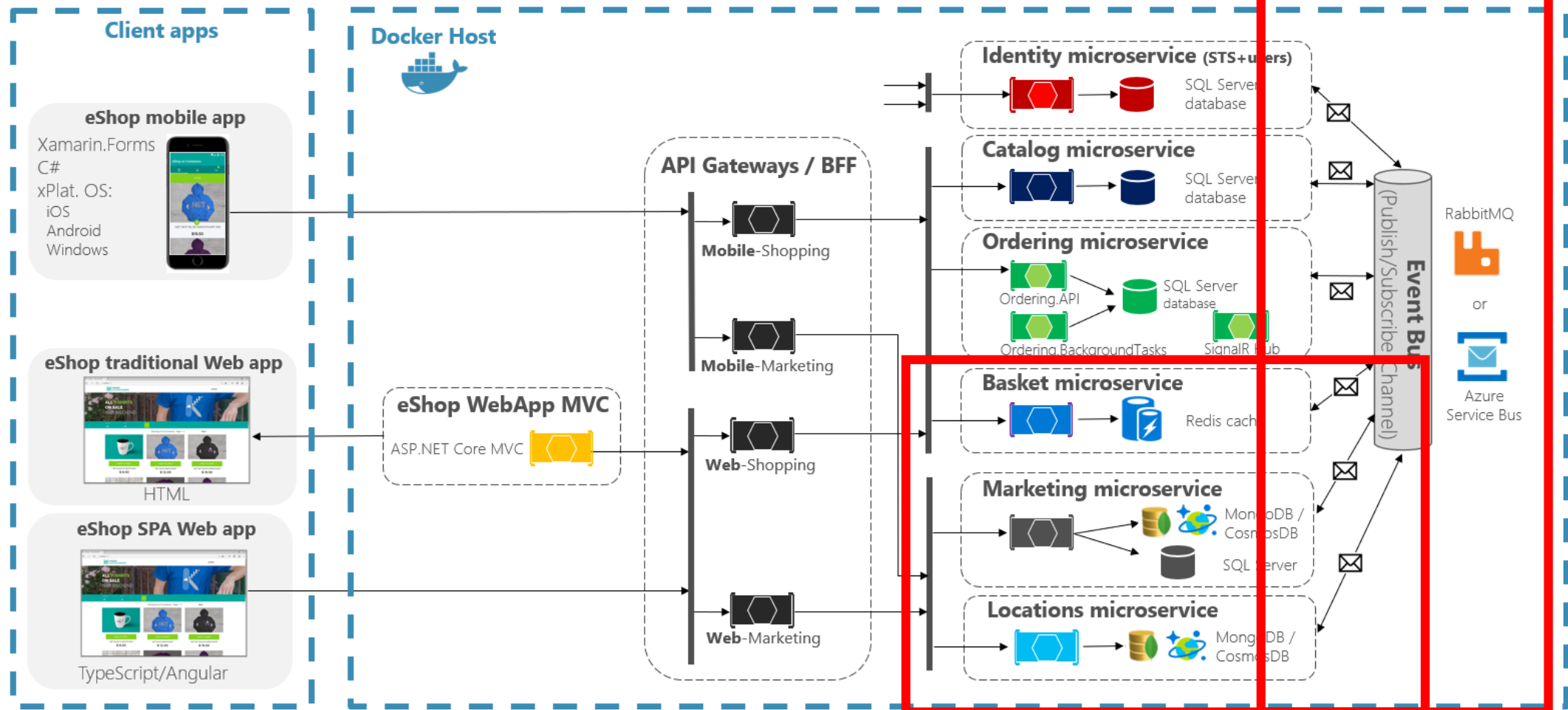
商務邏輯





eShopOnContainers reference application

(Development environment architecture)





Services



Basket



Catalog



Identity



Location



Marketing



Ordering

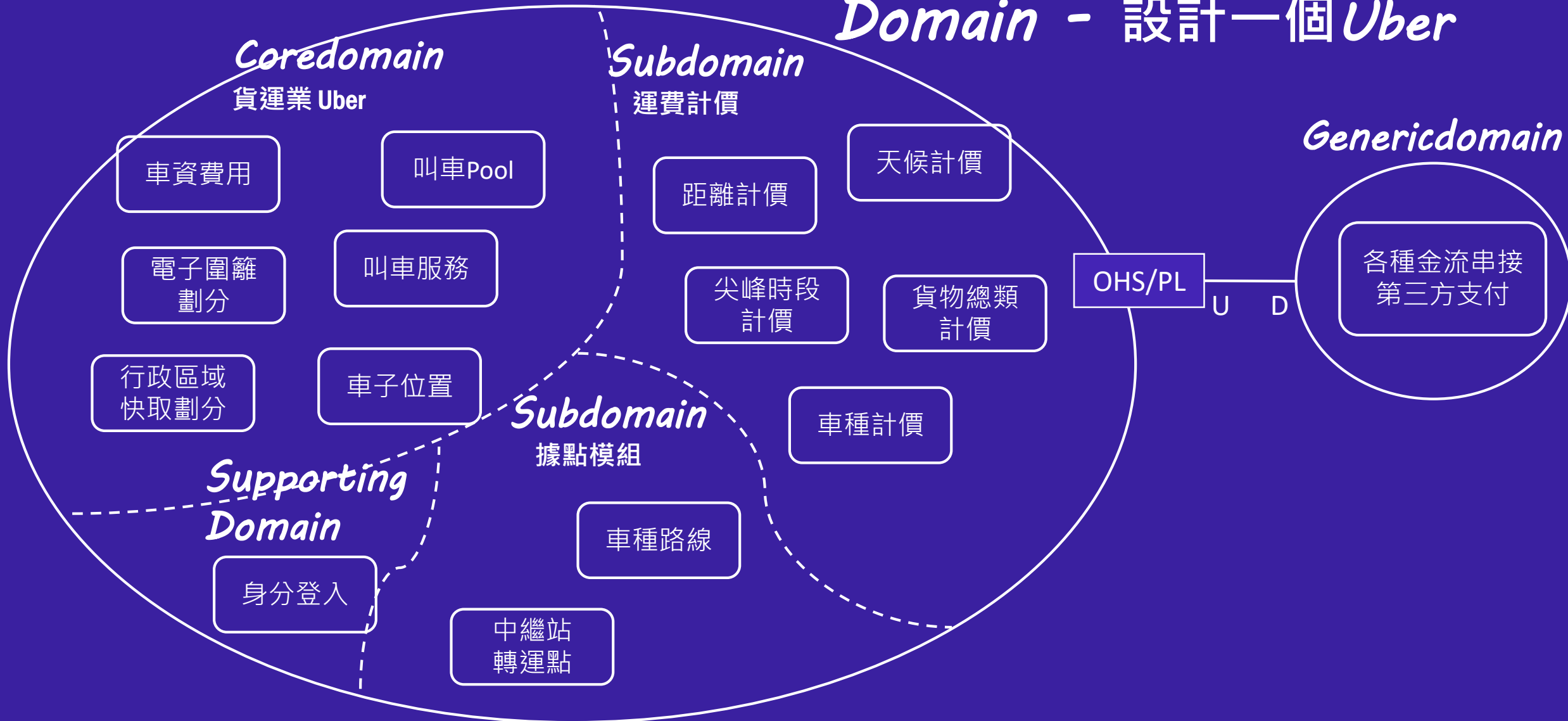


Payment



領域驅動找出系統邊界與服務單元

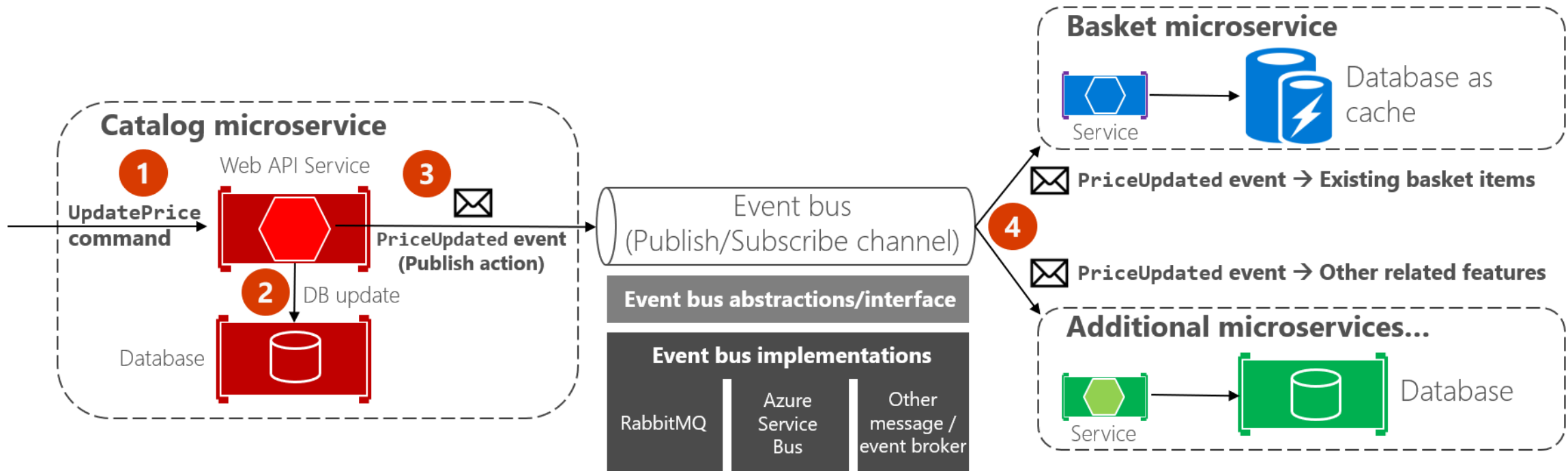
Domain - 設計一個Uber





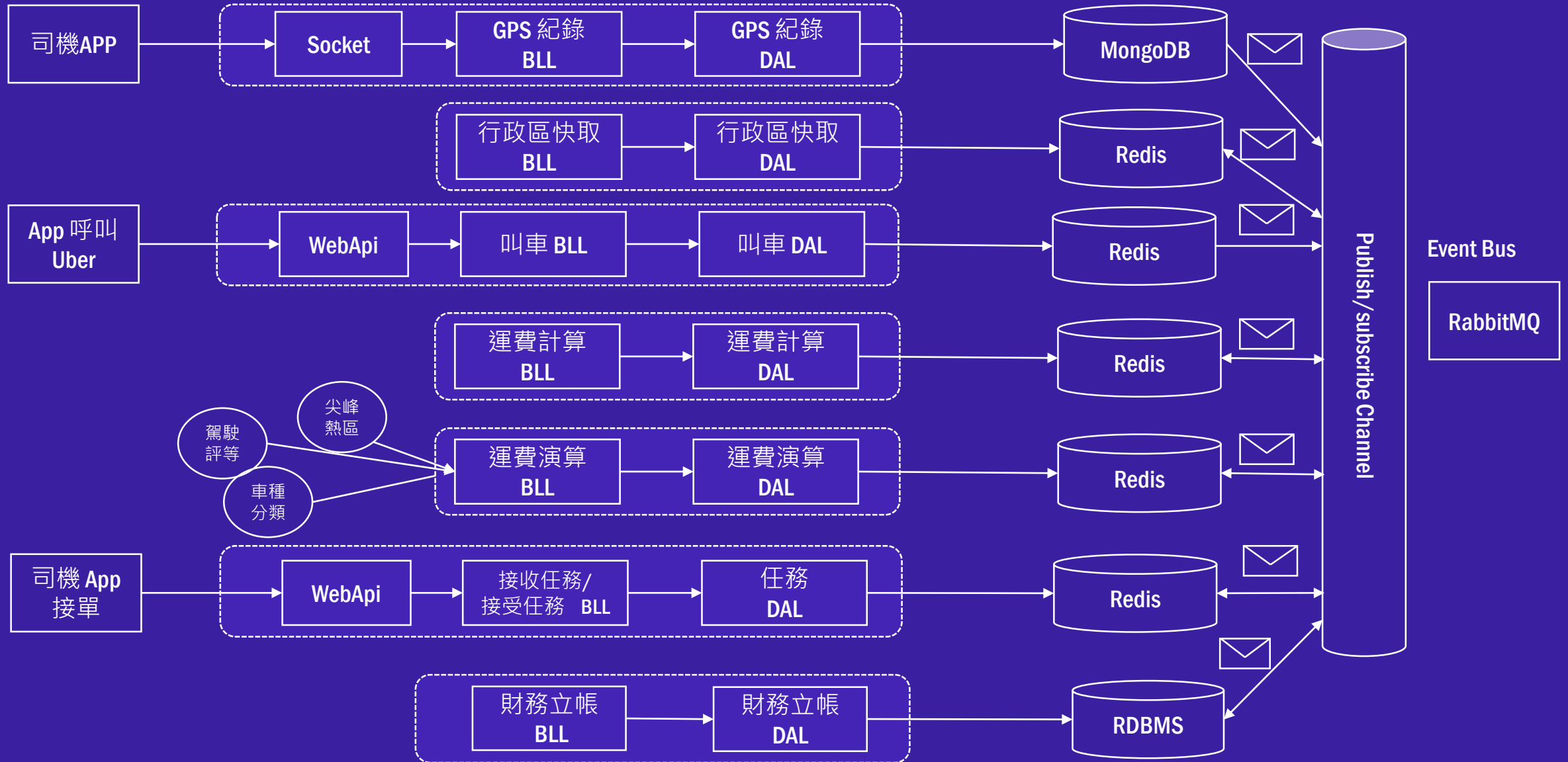
Implementing asynchronous event-driven communication with an event bus

Backend



Eventual consistency between microservices based on event-driven async communication

使用微服務替商業邏輯解耦

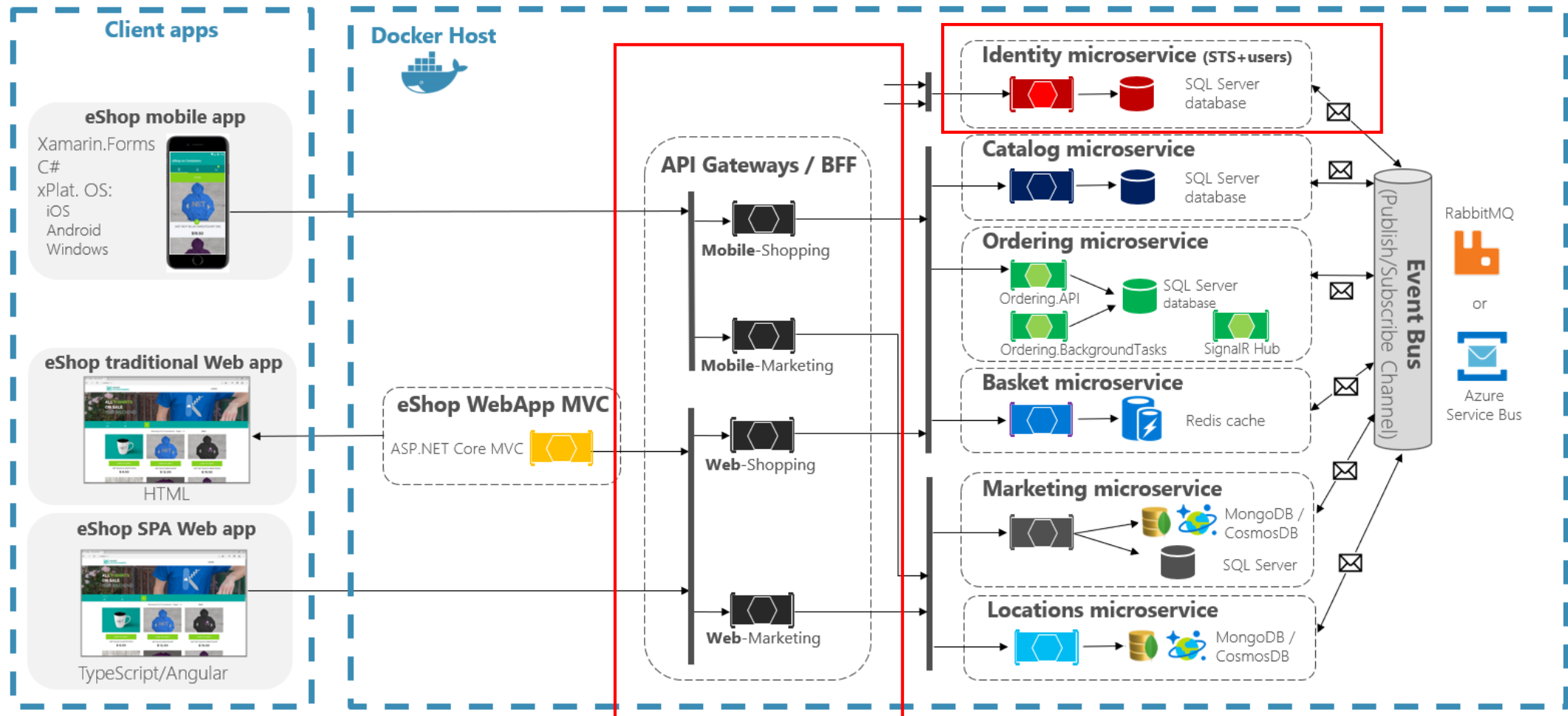


Demo

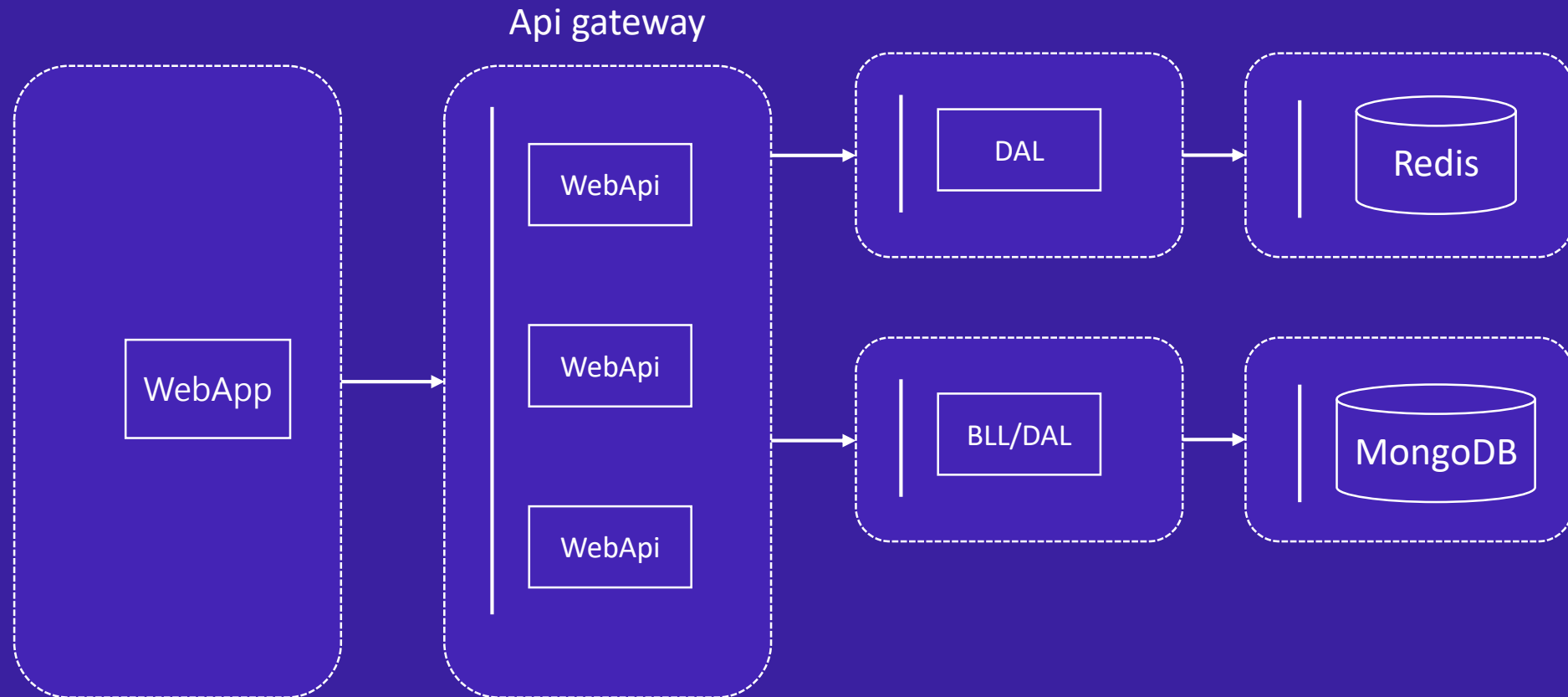


eShopOnContainers reference application

(Development environment architecture)



K8S 彈性擴展的第一關 - Serverless 的驗證



ASP.NET Core 中的金鑰儲存提供者

ASP.NET 語言 工作負載 API 資源

Docs / .NET / ASP.NET Core / 安全性及身分識別 / 資料保護 / 實作 / 金鑰儲存提供者

本主題有部分内容為機器翻譯。

版本
 ASP.NET Core 3.1

依標題排序

取得者 API
 設定
 擴充性 API
 實作
 總覽
 已驗證的加密詳細資料
 子機碼衍生和驗證的加密
 內容標頭
 金鑰管理
 金鑰儲存提供者
 待用時加密金鑰
 金鑰的不變性和設定
 金鑰儲存格式
 暫時資料保護提供者
 相容性
 秘密管理
 強制使用 HTTPS
 使用 HTTPS 執行 Docker
 使用 HTTPS 執行 Docker Compose
 EU 一般資料保護規定 (GDPR) 支援
 防止跨網站偽造要求 (XSRF/CSRF) 攻擊
 防止關聯重新導向攻擊

下載 PDF

ASP.NET Core 中的金鑰儲存提供者

2019/12/05

根據預設，資料保護系統會採用探索機制，來判斷該保存密碼編譯金鑰的位置。開發人員可以位置。

警告

如果您指定明確的金鑰持續性位置，資料保護系統會取消註冊靜態的預設金鑰加密機制，因此生產部署指定明確的金鑰加密機制。

檔案系統

若要設定以檔案系統為基礎的金鑰存放庫，請呼叫 `PersistKeysToFileSystem` 設定常式，如下所示，存放庫應儲存金鑰：

```

C#
public void ConfigureServices(IServiceCollection services)
{
    services.AddDataProtection()
        .PersistKeysToFileSystem(new DirectoryInfo(@"c:\temp-keys\"));
}
    
```

`DirectoryInfo` 可以指向本機電腦上的目錄，也可以指向網路共用上的資料夾，如果指向本機電腦的應用程式需要存取權才能使用此存放庫，請考慮在 Windows 上使用 `WINDOWS DPAPI` 加密的 `x.509` 憑證來加密待用的金鑰。

Azure 儲存體

`AspNetCore.DataProtection.blob` 封裝可讓您將資料保護金鑰儲存在 Azure Blob 儲存體中，您可以金鑰。應用程式可以 cookie 在多部伺服器之間共用驗證或 CSRF 保護。

Redis

`AspNetCore.DataProtection.StackExchangeRedis` 封裝可讓您將資料保護金鑰儲存在 Redis 快取中。您可以在 web 應用程式的數個實例之間共用金鑰。應用程式可以 cookie 在多部伺服器之間共用驗證或 CSRF 保護。

若要在 Redis 上設定，請呼叫其中一個 `PersistKeysToStackExchangeRedis` 多載：

```

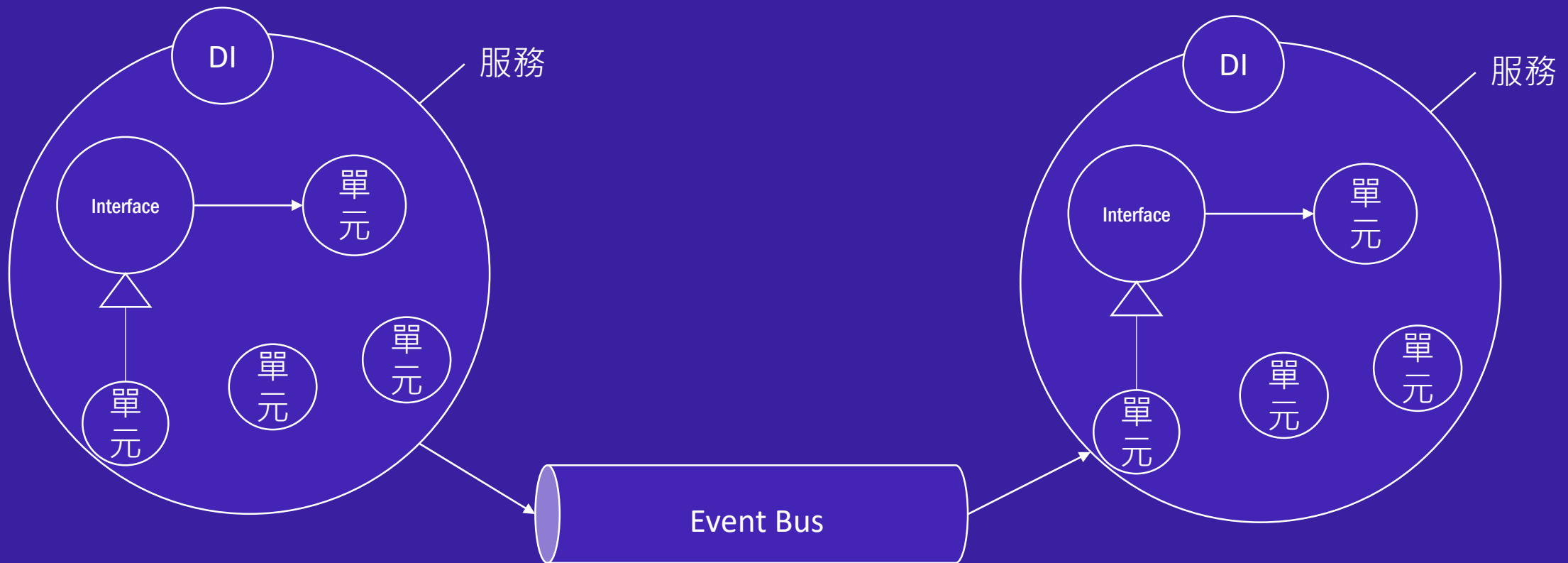
C#
public void ConfigureServices(IServiceCollection services)
{
    var redis = ConnectionMultiplexer.Connect("<URI>");
    services.AddDataProtection()
        .PersistKeysToStackExchangeRedis(redis, "DataProtection-Keys");
}
    
```

如需詳細資訊，請參閱下列主題：

- > [stackexchange.redis. Redis ConnectionMultiplexer](#)
- [Azure Redis 快取](#)
- [ASP.NET Core DataProtection 範例](#)

<https://docs.microsoft.com/zh-tw/aspnet/core/security/data-protection/implementation/key-storage-providers?view=aspnetcore-3.1&tabs=visual-studio>

元件之間靠 --> IOC/DI 解耦
商業邏輯靠 --> Event Bus 解耦



下一個挑戰?



微服務下的交易機制...



Thanks for joining!

Ask questions on Twitter using #dotNETConf



.NET Conf
2020

特別感謝

91APP
Technical Network

KK TIX



HackMD

MVP
Microsoft®
Most Valuable
Professional

Microsoft

Build School

STUDY4
為 學 習 而 生

以及各位參與活動的你們

