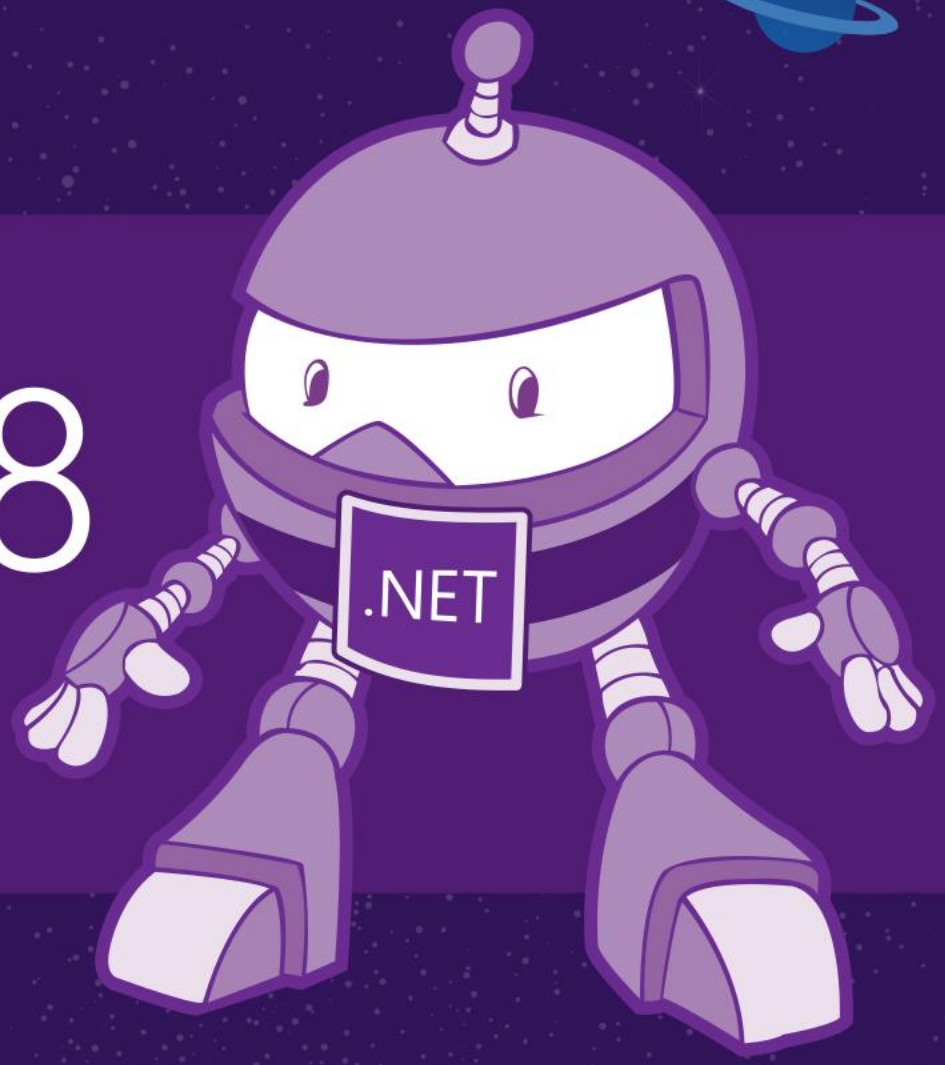
.NET Conf 2018

Discover the world of .NET



www.dotnetconf.net

Study4.TW

社團

fb.com/groups/2163125918226 35

Study4.TW

Study4.TW

粉絲團

fb.com/Study4.tw

STUDY4.TW
為 學 習 而 生

.NET Conf 2018

特別感謝

Microsoft MVP Most Valuable Professional

Microsoft

KKTIX

逢甲大學 Feng Chia University

Wishing-Soft

RevDeBug

.NET Taiwan

TAIWAN

91APP

R-Ladies Taipei

SEA 台灣軟體工程學會 Taiwan

STUDY4.TW 為學習而生

.NET Conf 2018

Web Development

Study4.TW Study4Love

ALM DevOps Agile Scrum

StudyHost .NET Conf

skychang.github.io

TechDay Channel9 MVA

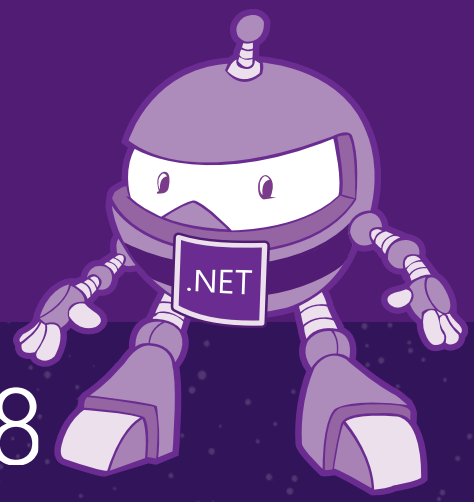
GitHub GitBook

Global Azure Bootcamp

Microsoft Azure MVP

網關攻略
淺談 Open Source API Gateway Ocelot

Sky Chang
天空的垃圾場



.NET Conf 2018

STUDY4.TW
為學習而生

注意事項

- 結束後會提供投影片
- 沒有甚麼是蠢問題!!
- 若有問題，可以利用結束或休息時間討論
- 軟體架構的世界，沒有絕對
- 沒有銀彈，沒有最好，只有最適合

Agenda

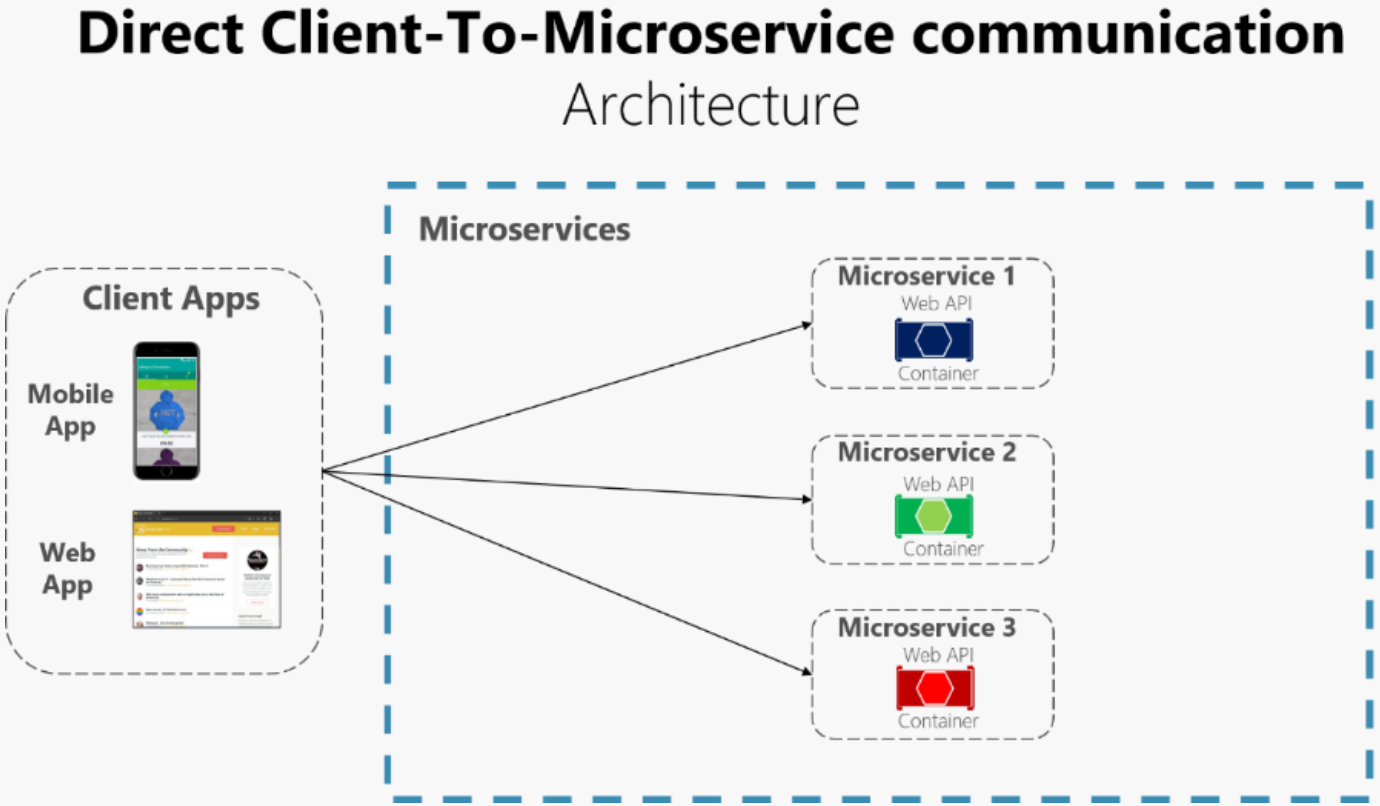
- 真實世界的問題!
- API Gateway Pattern
- Gateway Routing
- Request Aggregation
- Cross-cutting Concerns
- API Gateway 與開發技巧

網關攻略

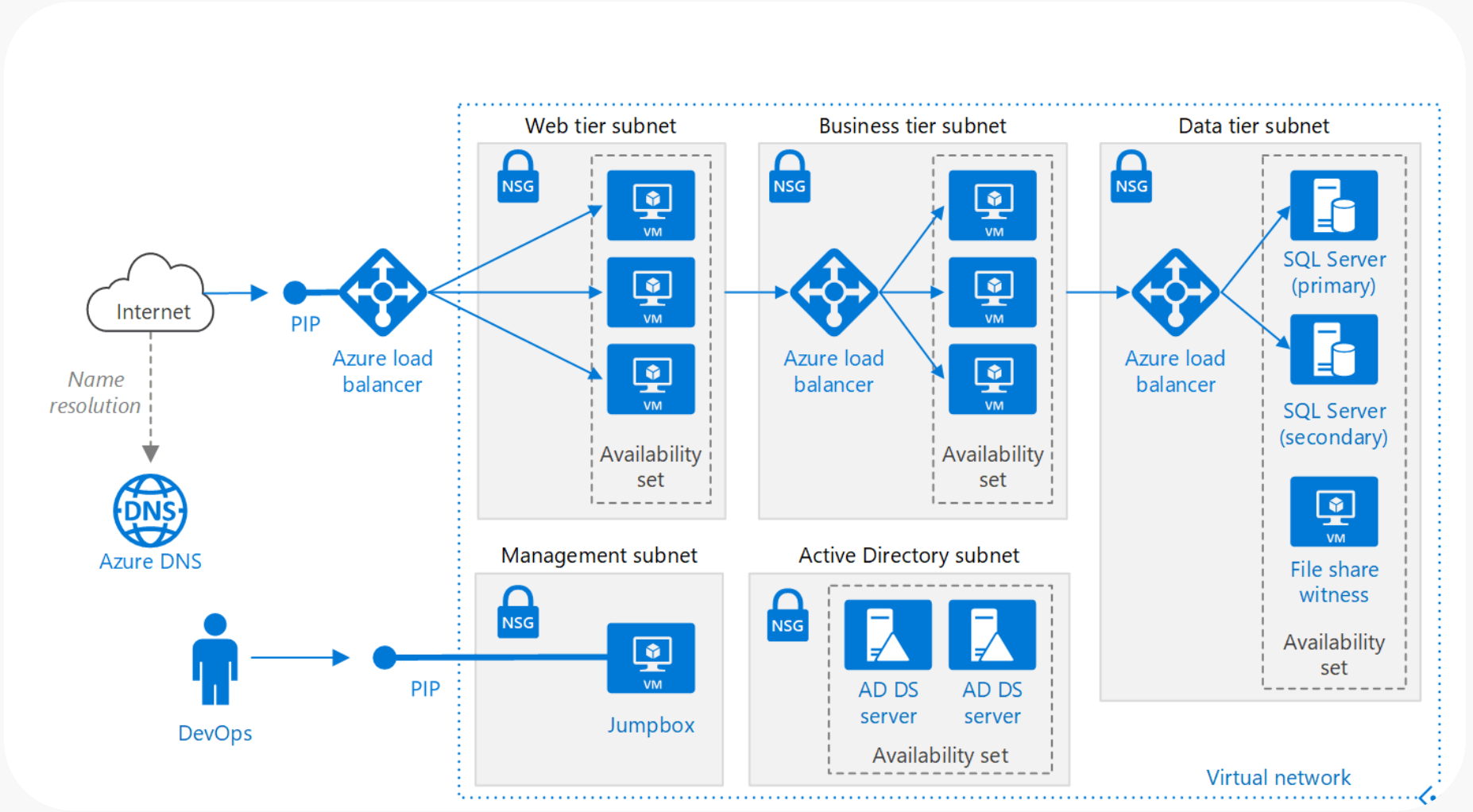
真實世界的問題!

微服務遭遇的問題

- 微服務需要大量的通訊
- 耦合問題
  - Client 與內部交互太過頻繁
  - Client 需要知道內部的結構與 API
  - 後端重構，Client 也需要大幅度修改
- 往返次數問題
  - 造成過多的網路浪費
- 安全問題
  - API 暴露於外面
- 橫切面問題
  - 針對安全、Log 等需要進行很多措施

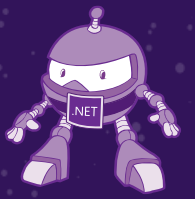


其實沒用微服務也一樣



API Gateway Pattern

.NET

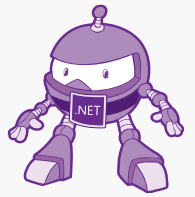


.NET Conf 2018

API Gateway Pattern

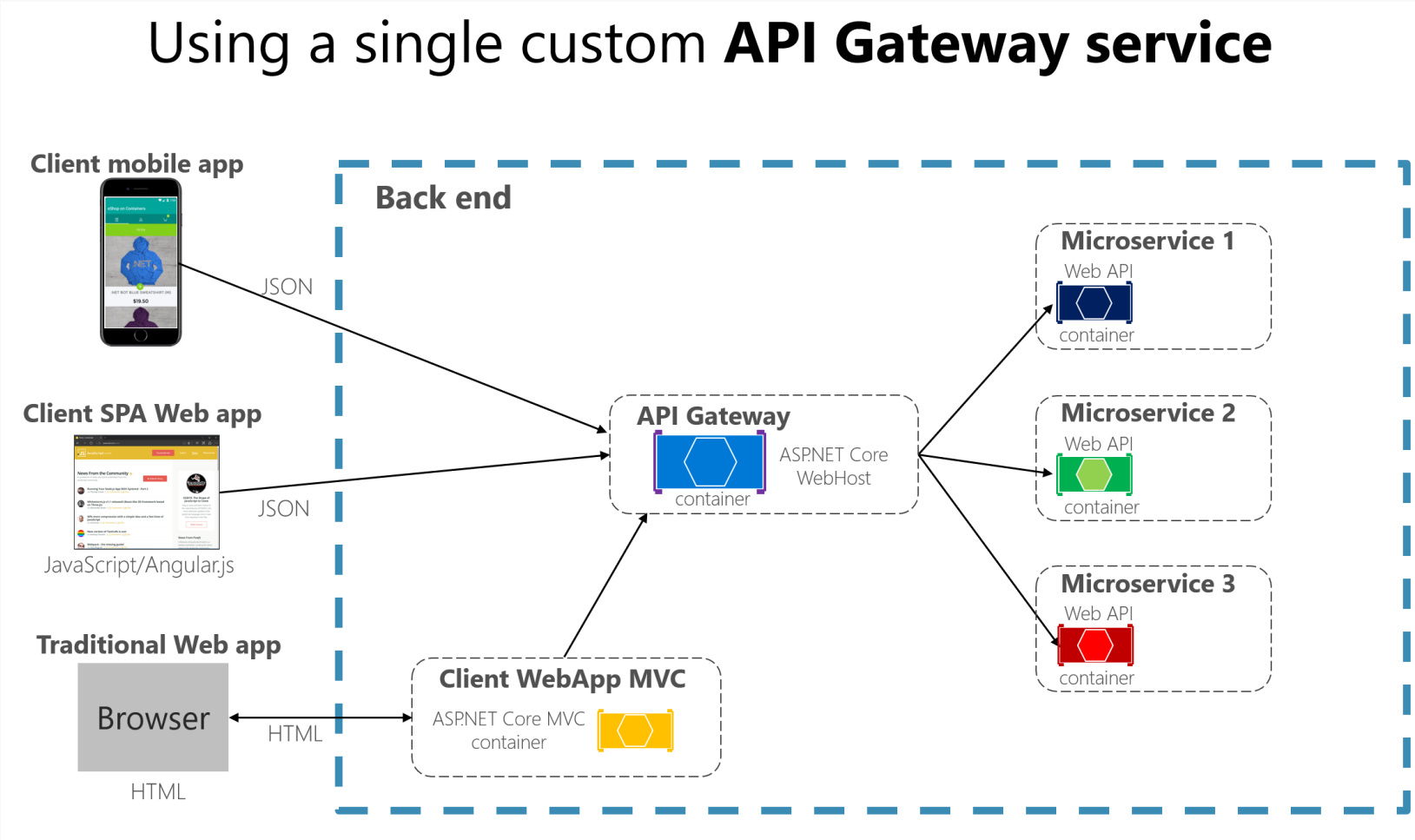
網關攻略

- 適合多 Client App 的應用情境
- 適合 microservice 情境
- 適合客戶一非常喜歡 N-Tier 的情慶
- 提供單一的入口給 Client 呼叫使用 (類似 Facade Pattern)
- 另外一種模式稱為 backend for frontend (BFF)
  - 提供不同的 Client ，建立不同的 Gateway
- 提供 Client 與 Service 的溝通橋樑
- 通常可能會包含身分驗證、SSL 終止、Cache、Log 等功能



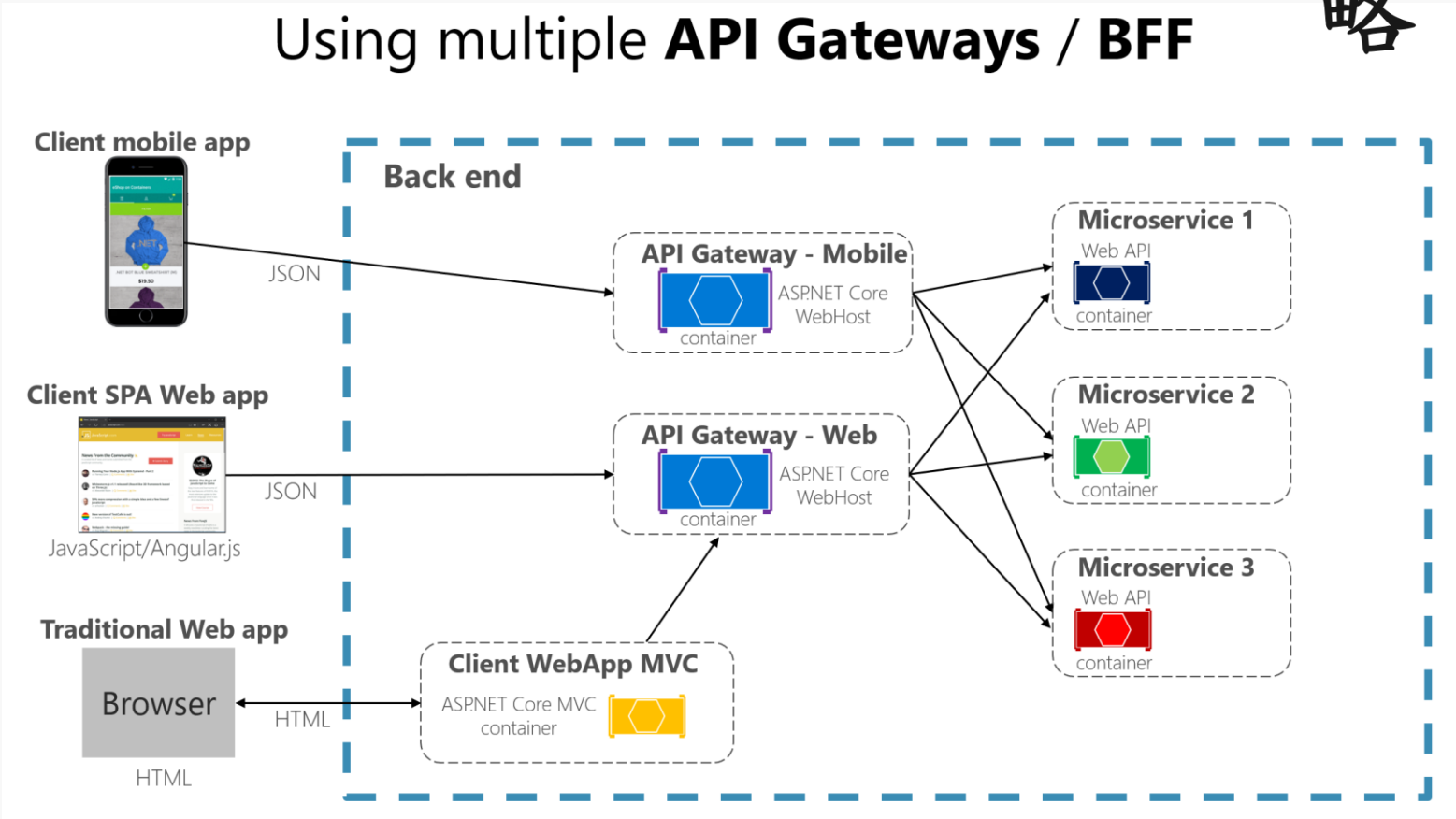
Single API Gateway

如果太多 Client ，可能會過於壅擠



Multiple API Gateways / BFF

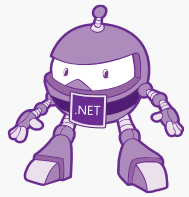
- 提供前端所需的 Service
- 可以簡化龐大
- 前端要甚麼，再給甚麼



API Gateway – 注意事項

網關攻略

- 確保 API Gateway 具有高可用性
  - 可通過運行多個 API Gateway 來避免單點故障
- 確保 API Gateway 的設計可滿足應用程式和端點的容量與擴展
  - 不會成為應用程式的瓶頸，並且具有足夠的可擴展性
- 對網關執行負載測試，以確保不會導致服務的串聯故障
- 如果需要追踪，請考慮為記錄目的與相關 ID
- 如有需要，監控 Request 和 Response 大小
- 考慮將 Cache 作為故障轉移策略來處理故障



Ocelot

網關攻略

- 輕量級 Open Source Gateway
- 輕巧、快速、簡單，由 .NET Core 撰寫而成
- 主要透過 ASP.NET Core Middleware 實現

Open Source .NET Core API Gateway

GET STARTED

https://github.com/ThreeMammals/Ocelot

Ocelot 安裝與設定

- NuGet - Install-Package Ocelot
- 建立設定檔 ocelot.json
- 調整 Program.cs
  - 設定多環境
- Startup.cs 設定

```
{
    "ReRoutes": [],
    "GlobalConfiguration": {
        "BaseUrl": "https://api.mybusiness.com"
    }
}
```

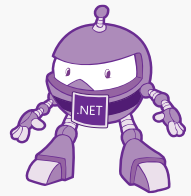
```
1 reference
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
    .ConfigureAppConfiguration((hostingContext, config) => {
        config
            .AddJsonFile("ocelot.json")
            .AddJsonFile($"configuration.{hostingContext.HostingEnvironment.EnvironmentName}.json");
    })
    .UseStartup<Startup>();
```

ConfigureServices

```
services.AddOcelot(Configuration);
```

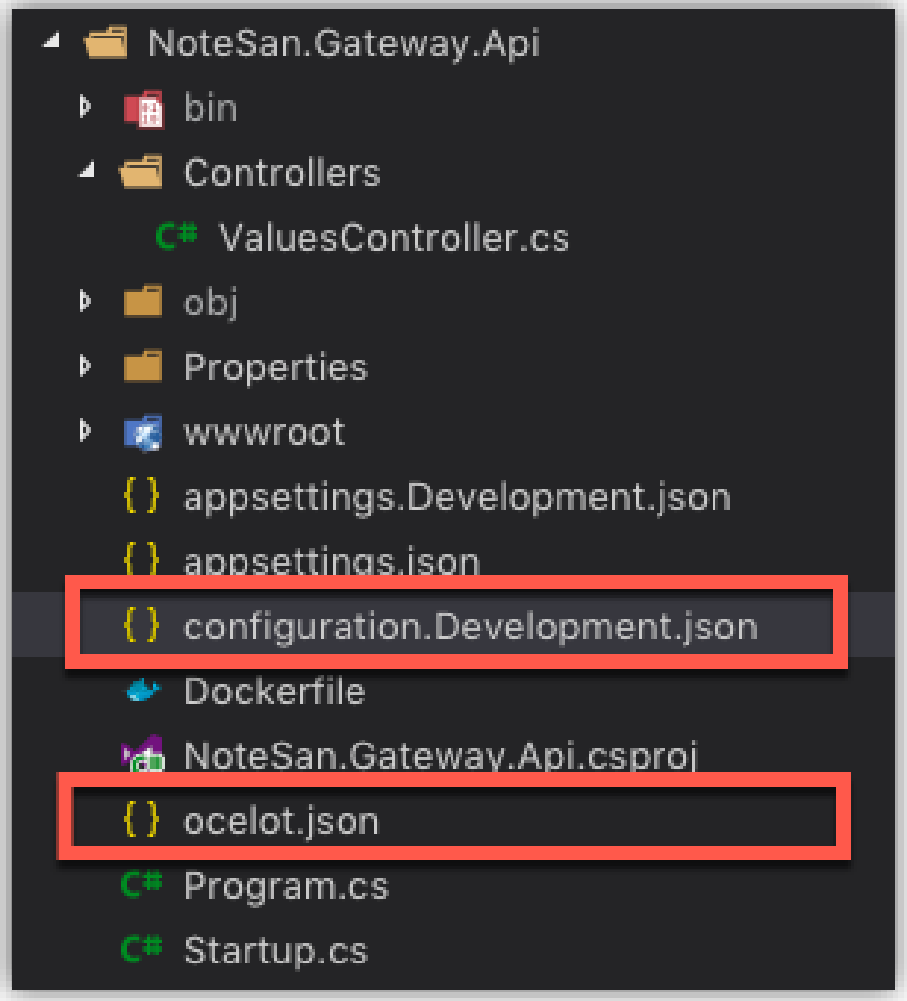
Configure

```
app.UseOcelot().Wait();
```

Ocelot 多環境設定

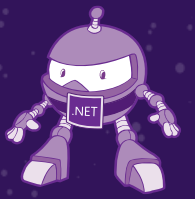
- 與 ASP.NET Core 多環境設定相同
- Configuration.Development.json
  - 表示 Development 環境
  - 根據 Program.cs 設定
- 若找不到環境 Config 則使用原始的
  - 根據 Program.cs 設定
- 也可以設定覆蓋



https://ocelot.readthedocs.io/en/latest/features/configuration.html

Reverse Proxy / Gateway Routing

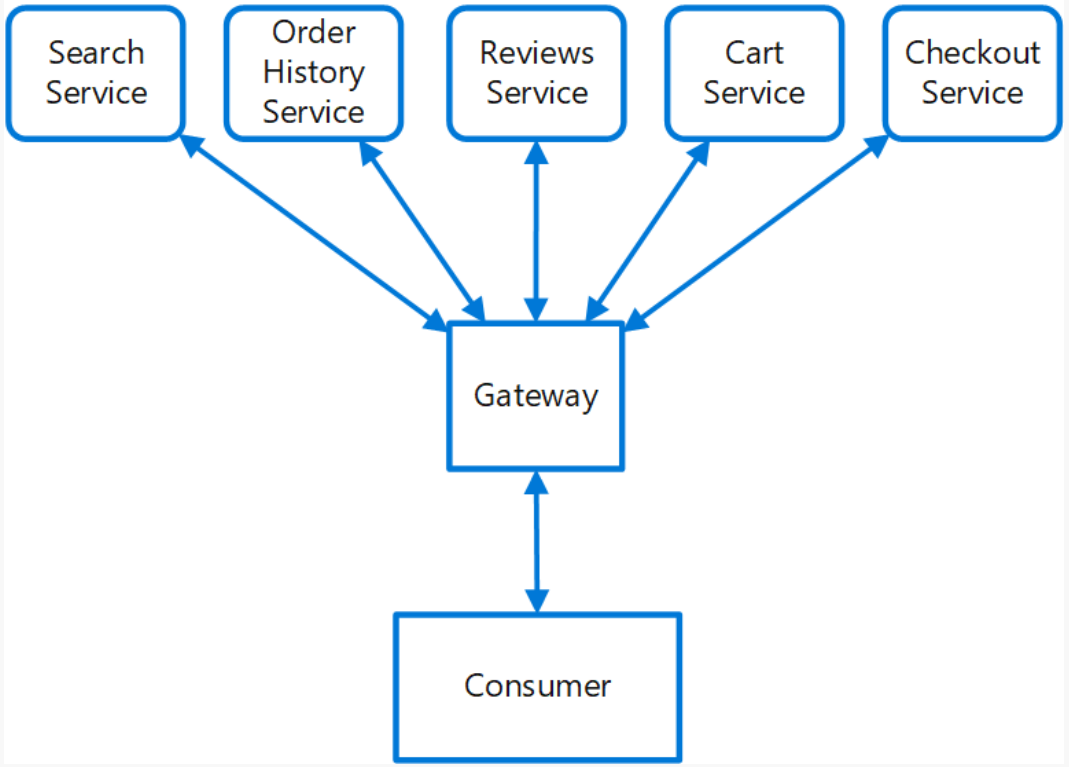
.NET



.NET Conf 2018

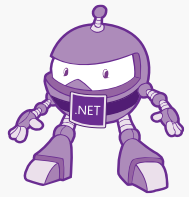
Reverse Proxy / Gateway Routing – 路由重導

- 通常為第七層（HTTP）
- 負責重新導向到後端
  - 可能是 URL 的重新調整
  - 可提供單點的 URL 給前端
  - 可調整不同的通訊協定
- 協助提供前後端的分離
  - 更改後面的 URL 不會影響到前端



Reverse Proxy / Gateway Routing – 注意事項

- 若後端只有一兩個 API，那可能不適用



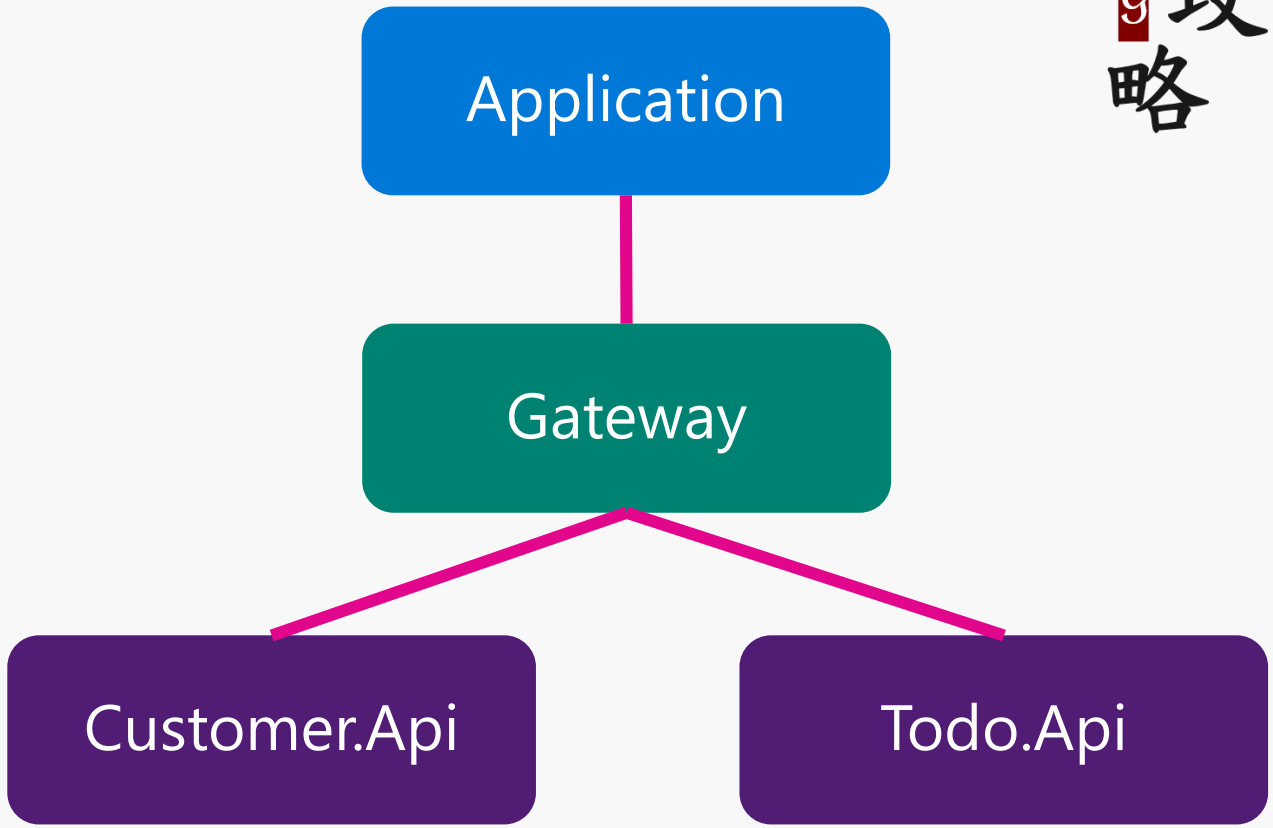
Ocelot - Routing – 路由

- 提供上下路由的關係與設定
- Priority
  - 優先級
- DownstreamPathTemplate
  - 下游，表示要導向的網址
- DownstreamScheme
  - http / https
- DownstreamHostAndPorts
  - Host 位置 & Port
- UpstreamPathTemplate
  - 上游，表示 Gateway 提供的網址
- UpstreamHttpMethod
  - 提供的 http 方法

```
{
    "ReRoutes": [
        {
            "Priority": 0,
            "DownstreamPathTemplate": "/api/{everything}",
            "DownstreamScheme": "https",
            "DownstreamHostAndPorts": [
                {
                    "Host": "localhost",
                    "Port": 7010
                }
            ],
            "UpstreamPathTemplate": "/customer/{everything}",
            "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
        }
    ],
    "GlobalConfiguration": {

    }
}
```

Ocelot - Routing – 路由

- UpstreamPathTemplate
  - /Customer/Customers
- UpstreamHttpMethod
  - Get , Post , Put , Delete
- DownstreamPathTemplate
  - /api/{everything}
- DownstreamScheme
  - https
- DownstreamHostAndPorts
  - localhost , 7010



網關攻略

Ocelot - Routing – 路由

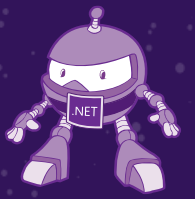
- Catch All – 全部捕獲
- Priority – 優先集
- Dynamic Routing – 動態路由
  - 可搭配 Service-discovery
- Query Strings – 查詢字串匹配

```
{
    "ReRoutes": [
        {
            "Priority": 0,
            "DownstreamPathTemplate": "/api/{everything}",
            "DownstreamScheme": "https",
            "DownstreamHostAndPorts": [
                {
                    "Host": "localhost",
                    "Port": 7010
                }
            ],
            "UpstreamPathTemplate": "/customer/{everything}",
            "UpstreamHttpMethod": [ "Get", "Post", "Put", "Delete" ]
        }
    ],
    "GlobalConfiguration": {

    }
}
```

Demo
Routing

Requests Aggregation

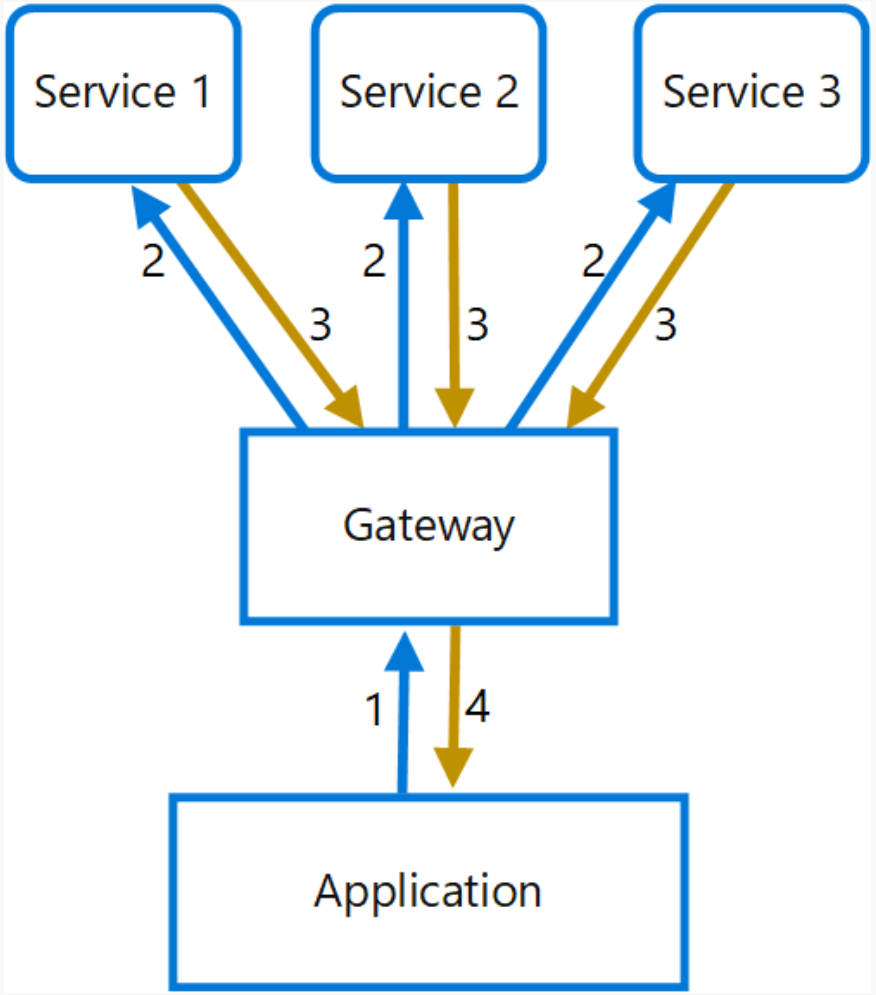
.NET



.NET Conf 2018

Requests Aggregation – 請求聚合

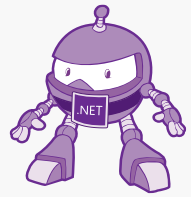
- 提供將多個呼叫，整合為一
- 減少 Client 與後端的干擾
- 若是 Server Site 的網頁處理，可能不會那麼明顯
  - ASP.NET MVC
- 通常 Gateway 產品會提供，但多半還是要自己刻 Code
  - 請放於 Gateway 之後



Requests Aggregation – 注意事項

網關攻略

- API Gateway 不應該引入服務耦合
- API Gateway 應位於後端服務附近，以減少延遲
- 內部呼叫使用隔離、斷路、重試、超時等處理技術
  - 如果一個或多個服務花費的時間太長，則可以考慮超時時返回部分數據集
- 使用 Async I / O 確保後端延遲不會導致應用程式出現性能問題
- 不要將聚合構建到 API Gateway，而是在 Gateway 後面放置聚合服務
  - 請求聚合可能具有與 Gateway 不同的資源要求，並且可能影響 Gateway 的路由和卸載功能

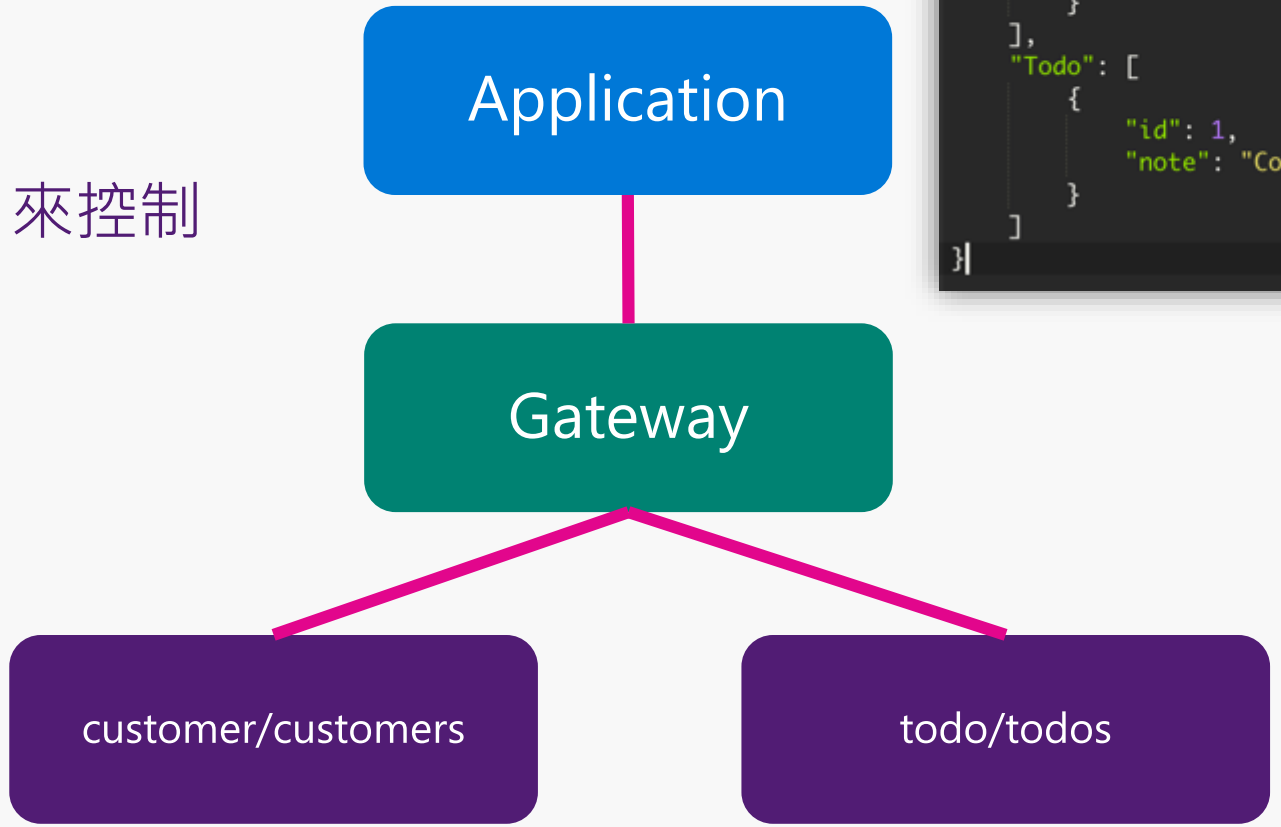


Ocelot - Requests Aggregation

- 將兩個請求合併
- 要指定明確的位置
- 透過 Key 來整合

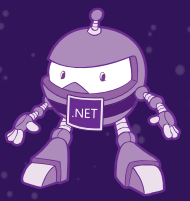
Ocelot - Requests Aggregation

- 聚合異常的時候，會沒有回應
- 回應的訊息為 JSON，
  就算是 404 也是 JSON
  - 無資料
- 可自行實作 IDefinedAggregator 來控制
  回應方法



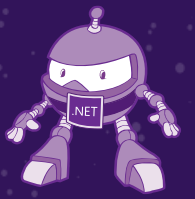
https://ocelot.readthedocs.io/en/latest/features/requestaggregation.html

Demo
Requests Aggregation



Cross-cutting Concerns / Gateway Offloading

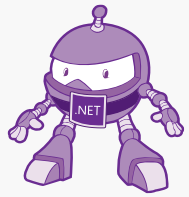
.NET



.NET Conf 2018

Cross-cutting Concerns / Gateway Offloading
橫切關注 / 集中功能

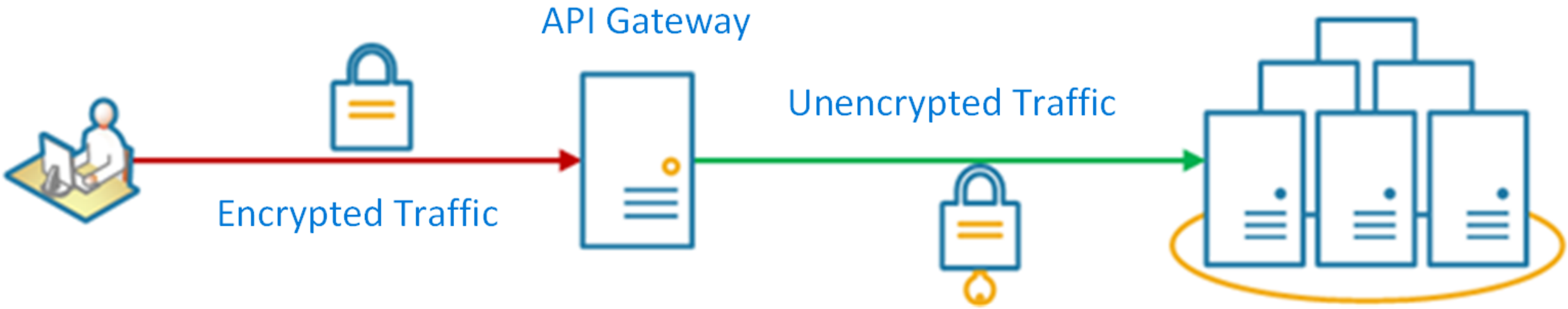
- 將原本在 API 層級的功能，轉移到 API Gateway
  - Authentication and authorization - 身份驗證和授權
  - Service discovery integration – 整合服務發現
  - Response caching - 響應緩存
  - Retry policies, circuit breaker and QoS - 重試策略，斷路器和QoS
  - Rate limiting and throttling - 速率限制和節流
  - Load balancing - 負載均衡
  - Logging, tracing, correlation - 記錄，跟踪，關聯
  - Headers, query strings and claims transformation - 標題、查詢字串、聲明轉換
  - IP whitelisting - IP白名單



Gateway Offloading - 優點

網關攻略

- 降低維護成本，簡化服務開發、配置與管理，提高升級容易度
  - Web服務器證書和安全網站配置
- 允許專門的團隊實施需要專業知識的功能
  - 安全性交給安全專家，內部人員專注於邏輯開發
- 為請求 / 回應記錄 / 監視提供一致性
  - 即使未正確檢測服務，也可以確保最低級別的監視和日誌記錄

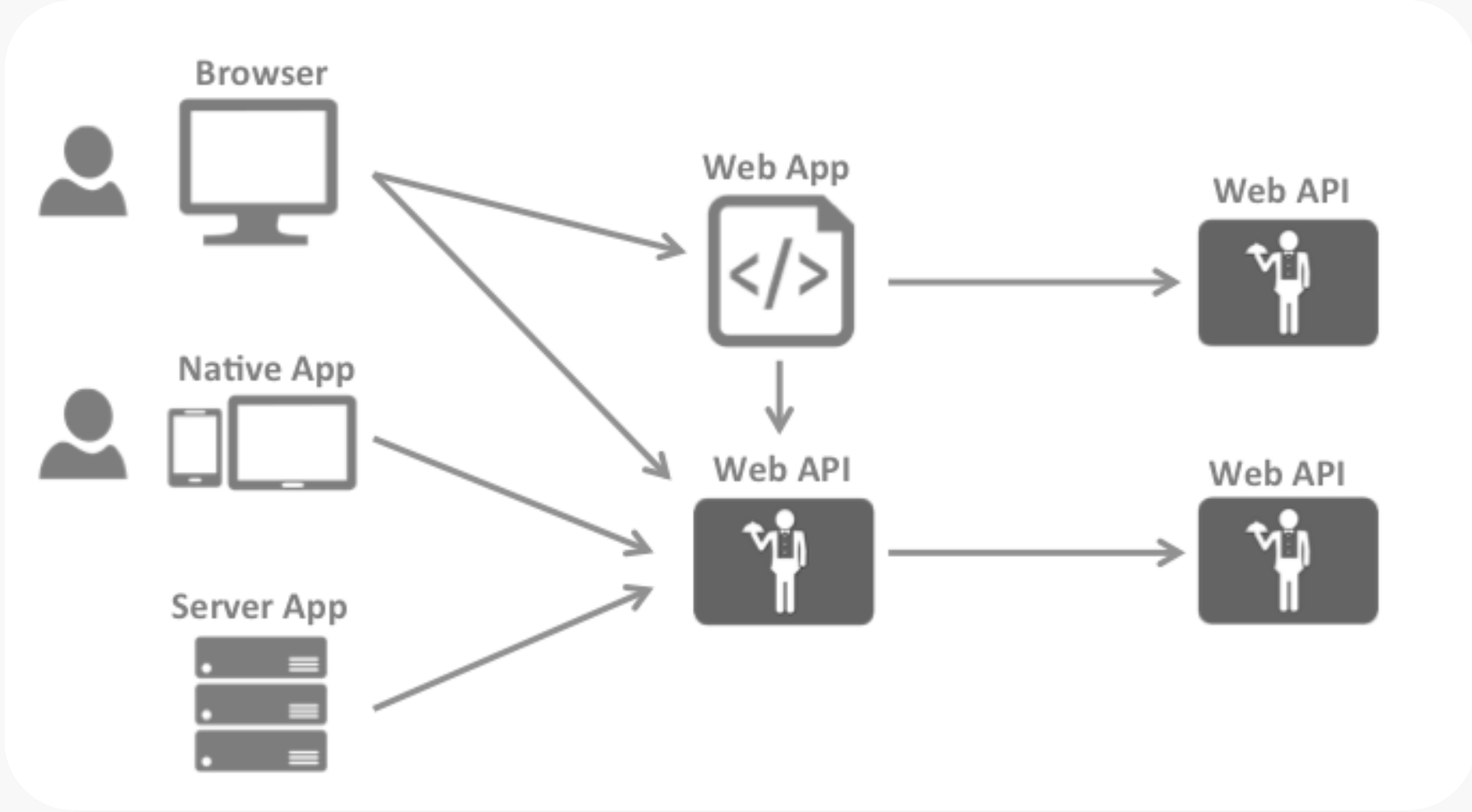


Gateway Offloading – 注意事項

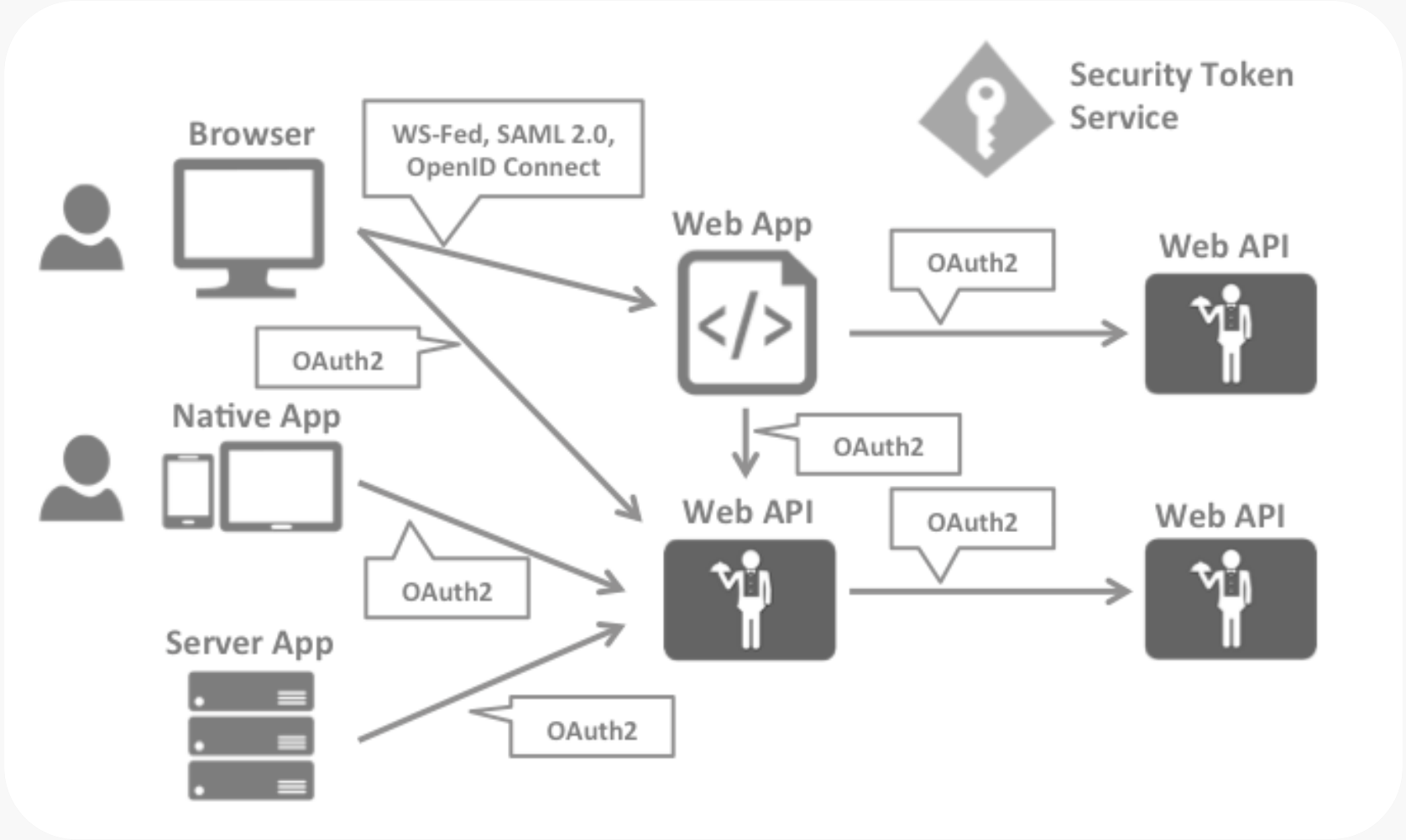
- 永遠不應將業務邏輯卸載到 API Gateway

網關攻略

一般應用程式情境

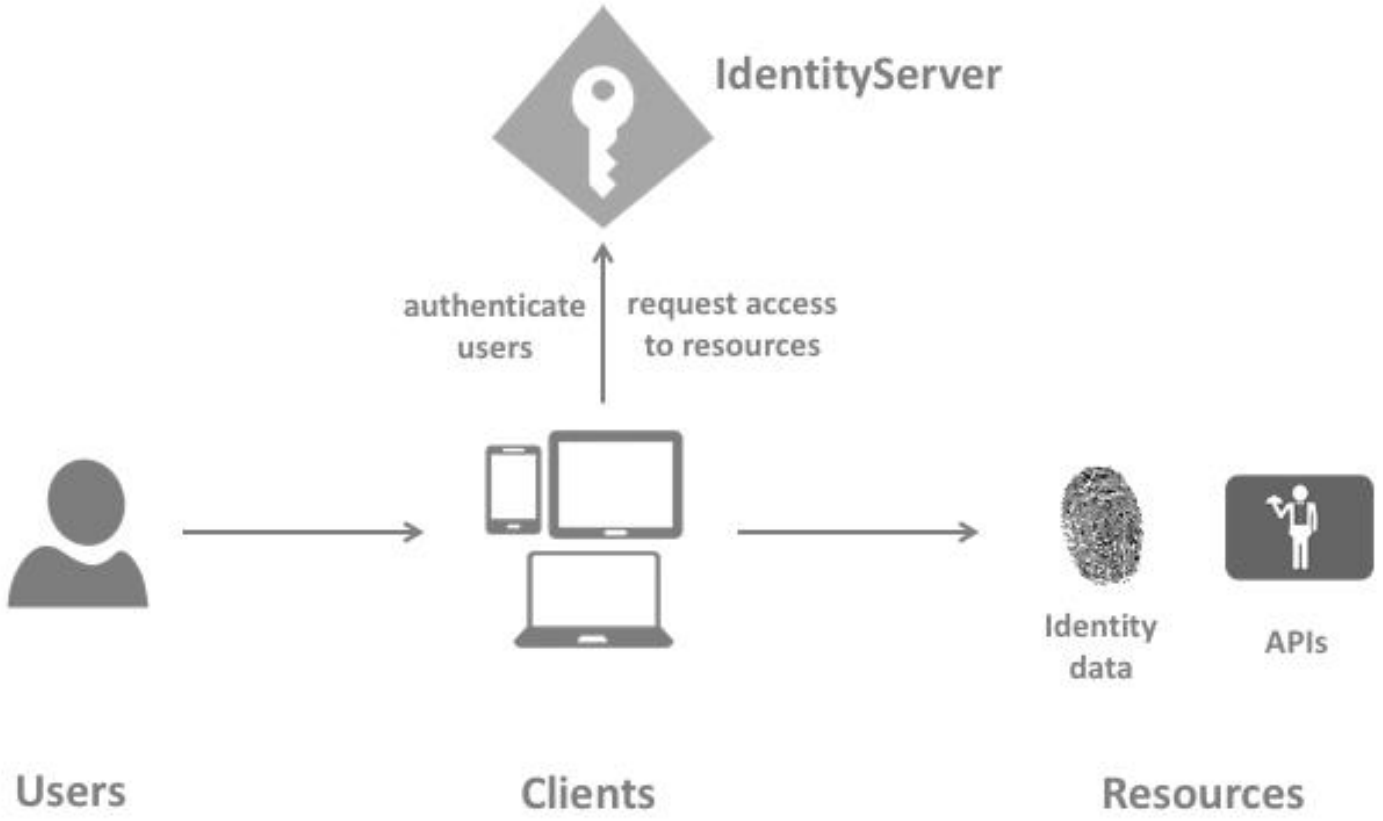


一般應用程式情境 – 加上授權

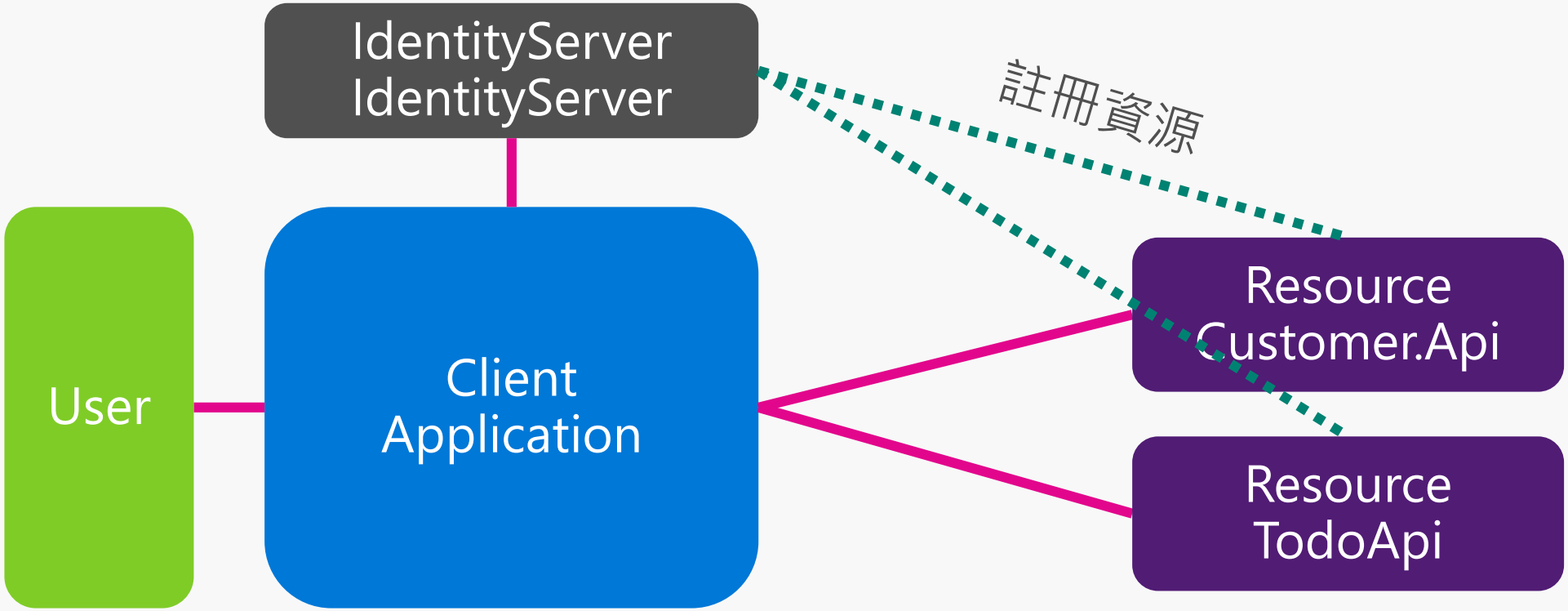


驗證授權架構

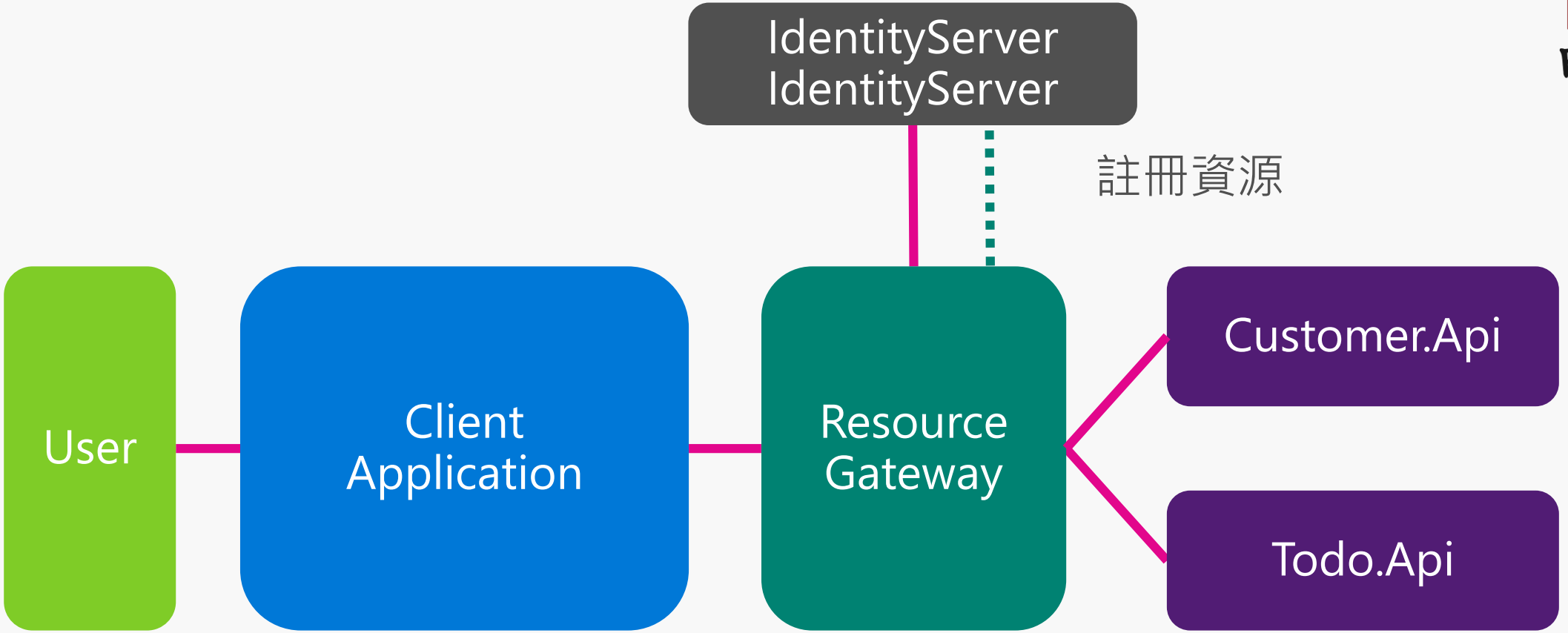
- IdentityServer
  - 驗證授權伺服器
  - 提供發放 Token
  - 註冊 Client
- Client
  - 應用程式
    - ASP.NET MVC 網站
    - Web API
    - SPA
- Resources
  - 被保護的資源
    - Web API



沒有 API Gateway 的時候..

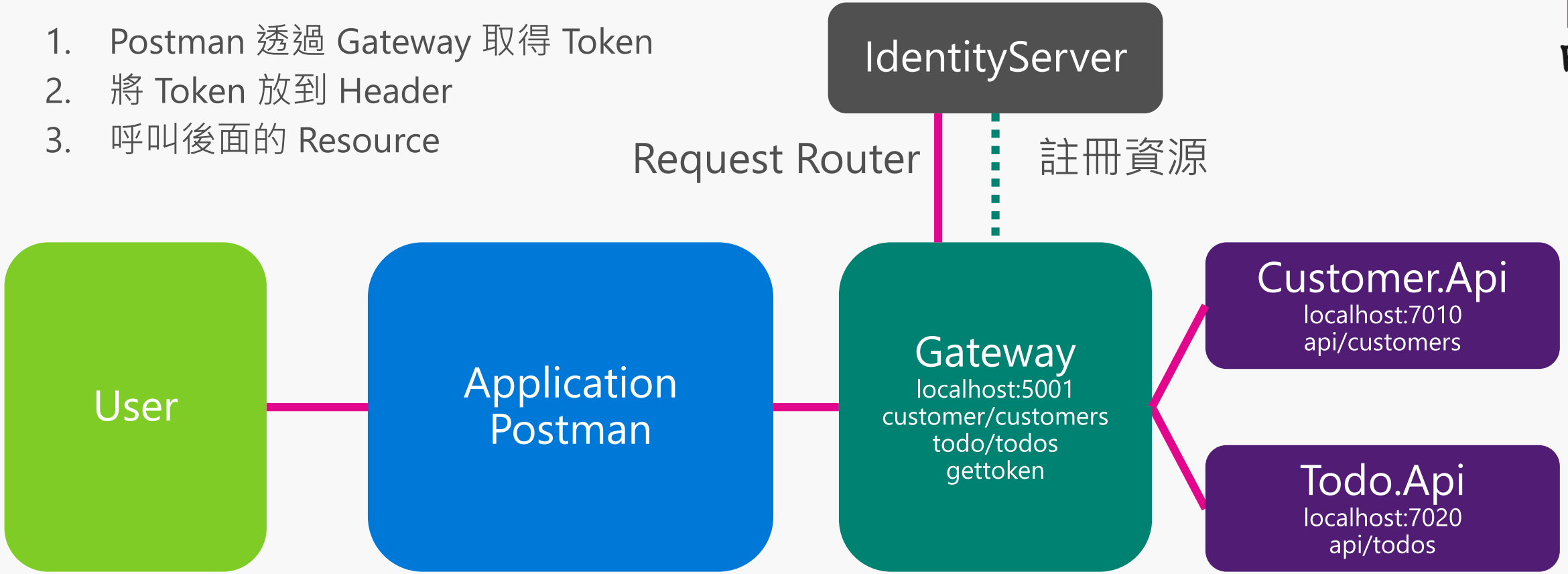


Ocelot + IdentityServer 方案一（不推薦）



Ocelot + IdentityServer Detail (不推薦)

1. Postman 透過 Gateway 取得 Token
2. 將 Token 放到 Header
3. 呼叫後面的 Resource



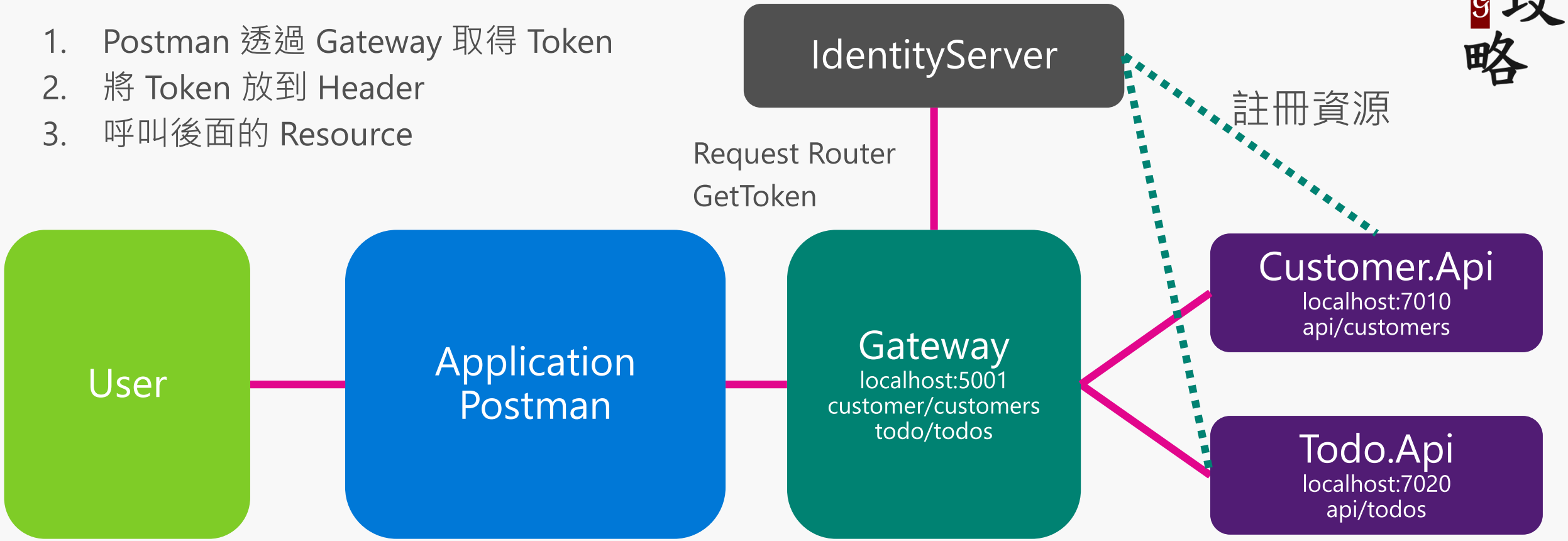
網關攻略

STUDY4.TW 為學習而生

.NET Conf 2018

Ocelot + IdentityServer 方案二 - Demo

1. Postman 透過 Gateway 取得 Token
2. 將 Token 放到 Header
3. 呼叫後面的 Resource

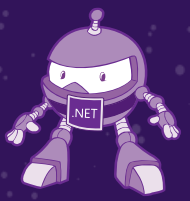


網關攻略

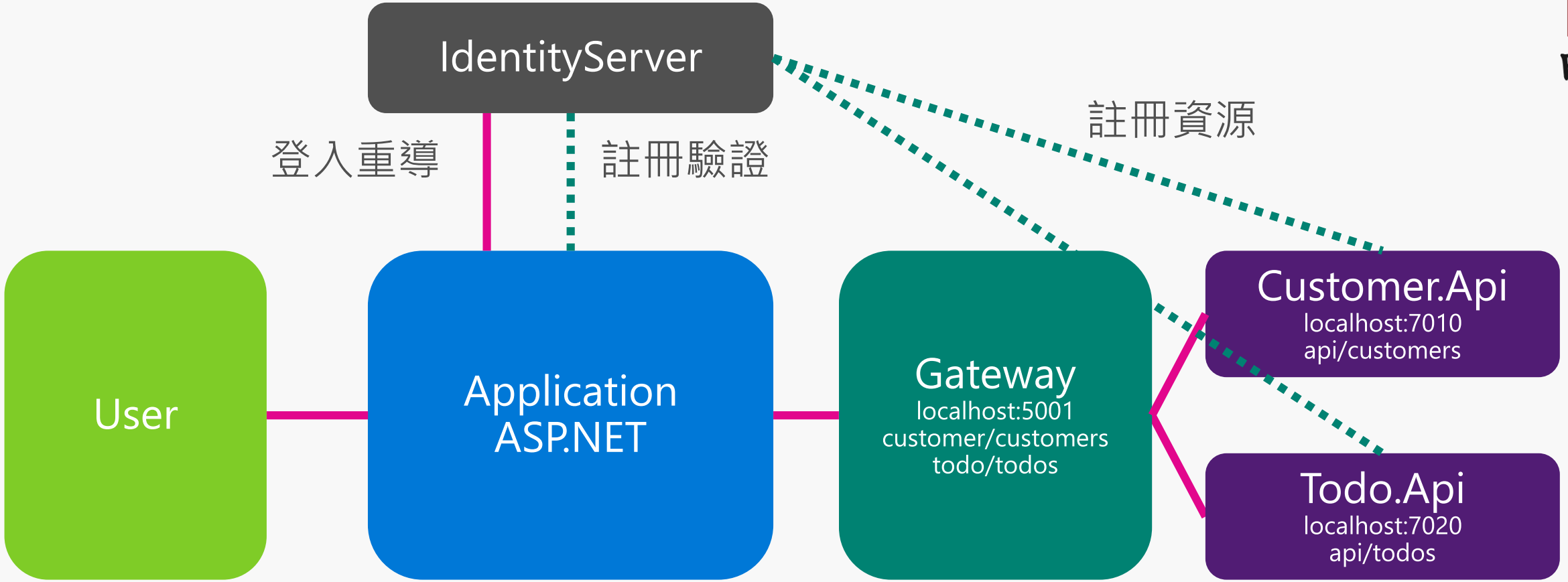
STUDY4.TW 為學習而生

.NET Conf 2018

Demo
Cross-cutting Concerns



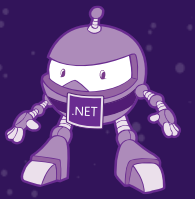
Ocelot + IdentityServer 方案二 for ASP.NET



1. ASP.NET 進行登入
2. 導向至 IdentityServer 進行登入
3. 導回 ASP.NET，並取得 Token
4. 將 Token 放到 Header
5. 呼叫後面的 Resource

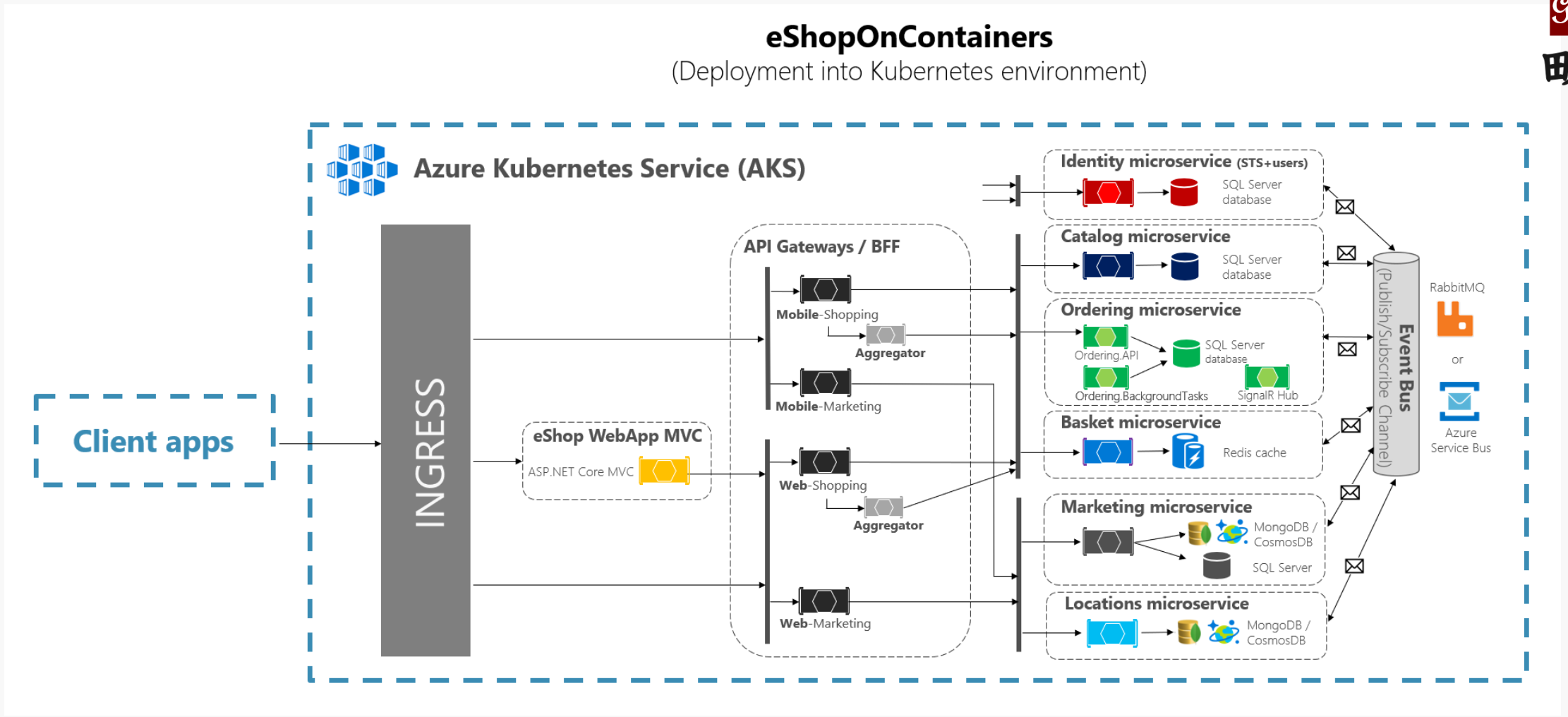
API Gateway 與 Microservice 開發

.NET



.NET Conf 2018

官方 Microservice 架構



API Gateway 與 Microservice 開發

- 透過 dockerfile 定義每一個站台

```
FROM microsoft/aspnetcore:2.0.5 AS base
WORKDIR /app
EXPOSE 80
```

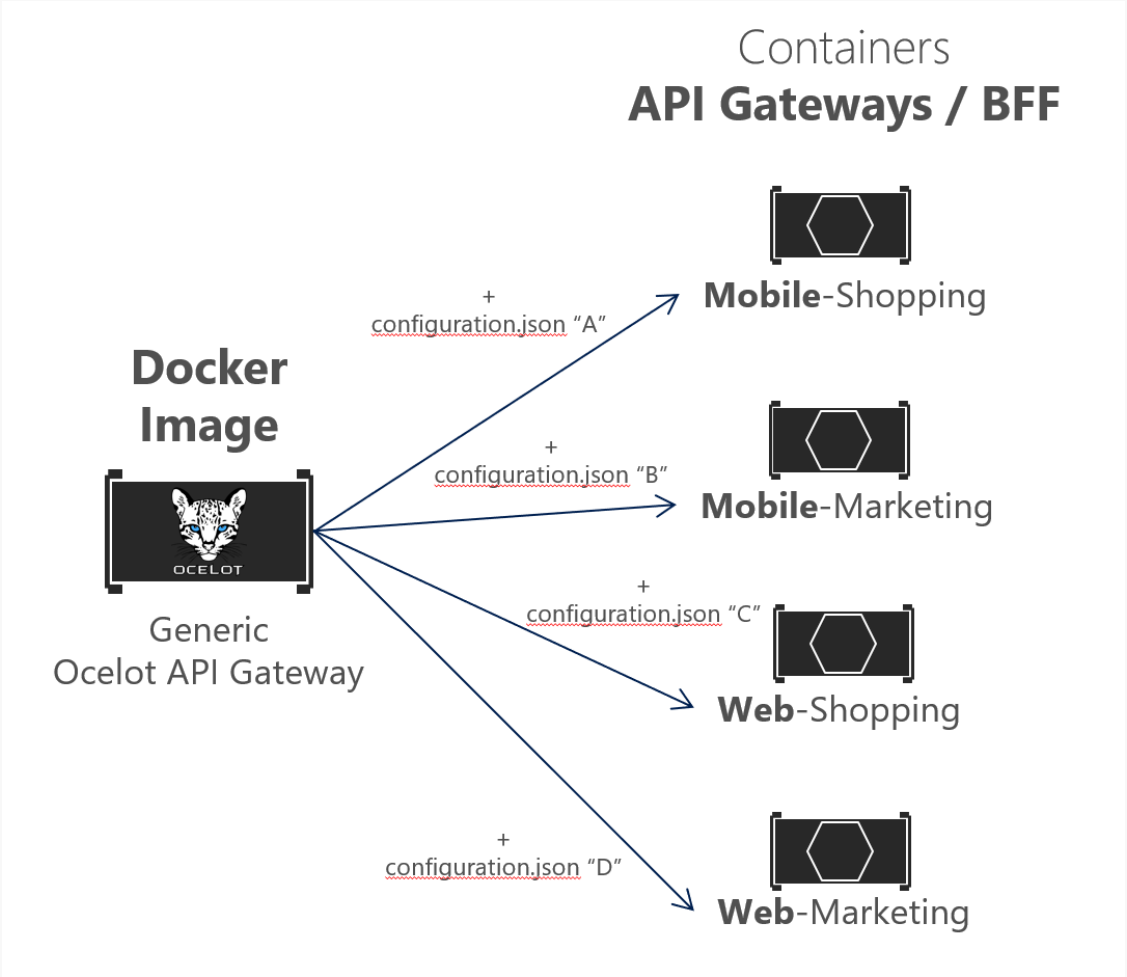
- 使用 docker-compose 定義開發時期對外的 Port

```
catalog.api:
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - ASPNETCORE_URLS=http://0.0.0.0:80
    - ConnectionString=YOUR_VALUE
    - ... Other Environment Variables
  ports:
    - "5101:80"   # Important: In a production environment you should remove the ex
                  # The API Gateway redirects and access through the internal port
```

- 正式環境透過 API Gateway 進行存取 (K8S)

API Gateway 與 Microservice 開發

- 使用一個 API Gateway Image
- 提供多個 API Gateway / BFF



API Gateway 與 Microservice 開發

- 指向不同的設定檔

```
mobileshoppingapigw:
  image: eshop/ocelotapigw:${TAG:-latest}
  build:
    context: .
    dockerfile: src/ApiGateways/ApiGw-Base/Dockerfile

mobilemarketingapigw:
  image: eshop/ocelotapigw:${TAG:-latest}
  build:
    context: .
    dockerfile: src/ApiGateways/ApiGw-Base/Dockerfile

webshoppingapigw:
  image: eshop/ocelotapigw:${TAG:-latest}
  build:
    context: .
    dockerfile: src/ApiGateways/ApiGw-Base/Dockerfile

webmarketingapigw:
  image: eshop/ocelotapigw:${TAG:-latest}
  build:
    context: .
    dockerfile: src/ApiGateways/ApiGw-Base/Dockerfile
```

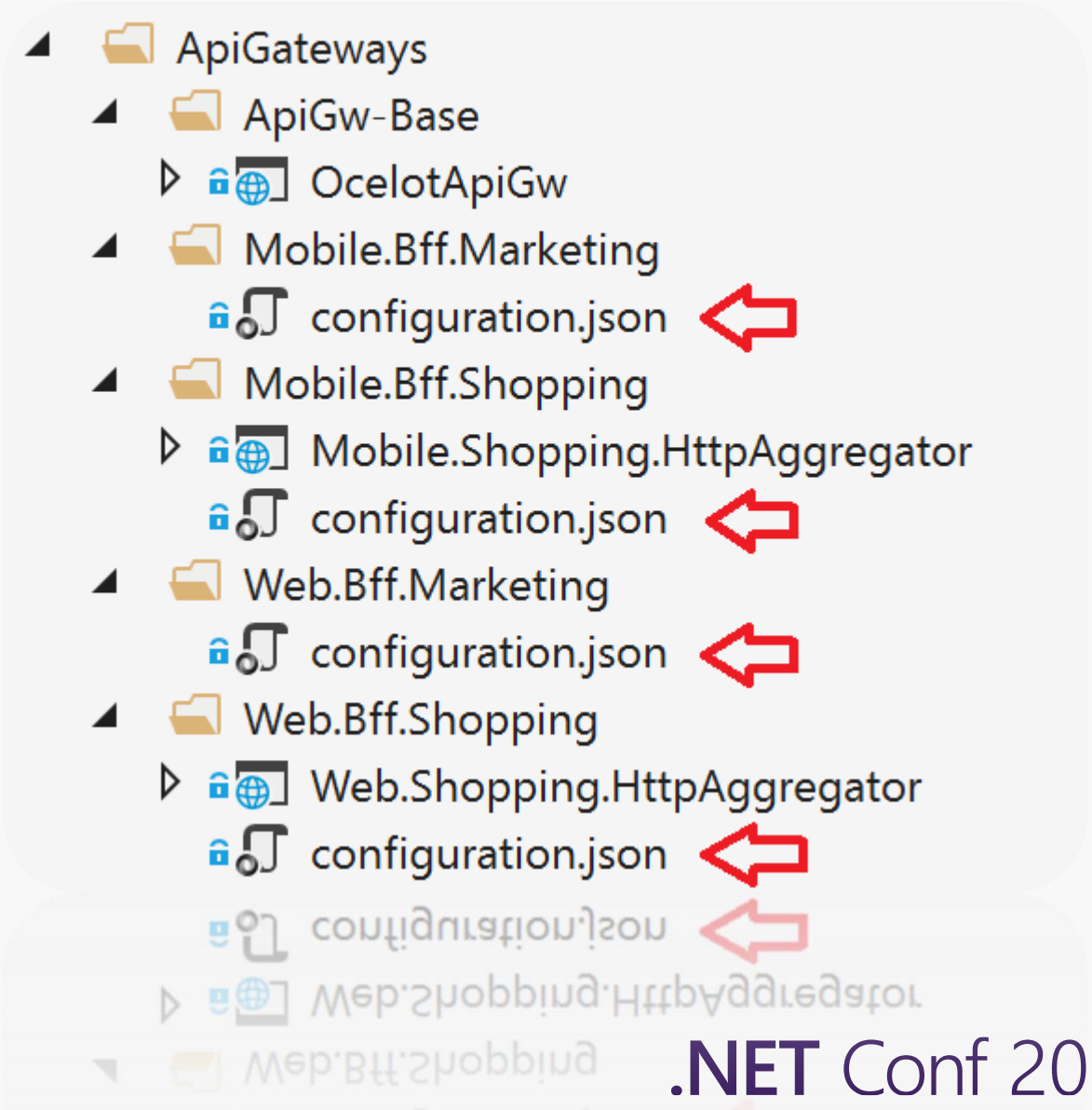
docker-compose.yml

```
mobileshoppingapigw:
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - IdentityUrl=http://identity.api
  ports:
    - "5200:80"
  volumes:
    - ./src/ApiGateways/Mobile.Bff.Shopping/apigw:/app/configuration

mobilemarketingapigw:
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - IdentityUrl=http://identity.api
  ports:
    - "5201:80"
  volumes:
    - ./src/ApiGateways/Mobile.Bff.Marketing/apigw:/app/configuration

webshoppingapigw:
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - IdentityUrl=http://identity.api
  ports:
    - "5202:80"
  volumes:
    - ./src/ApiGateways/Web.Bff.Shopping/apigw:/app/configuration

webmarketingapigw:
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - IdentityUrl=http://identity.api
  ports:
    - "5203:80"
  volumes:
    - ./src/ApiGateways/Web.Bff.Marketing/apigw:/app/configuration
```

docker-compose.override.yml

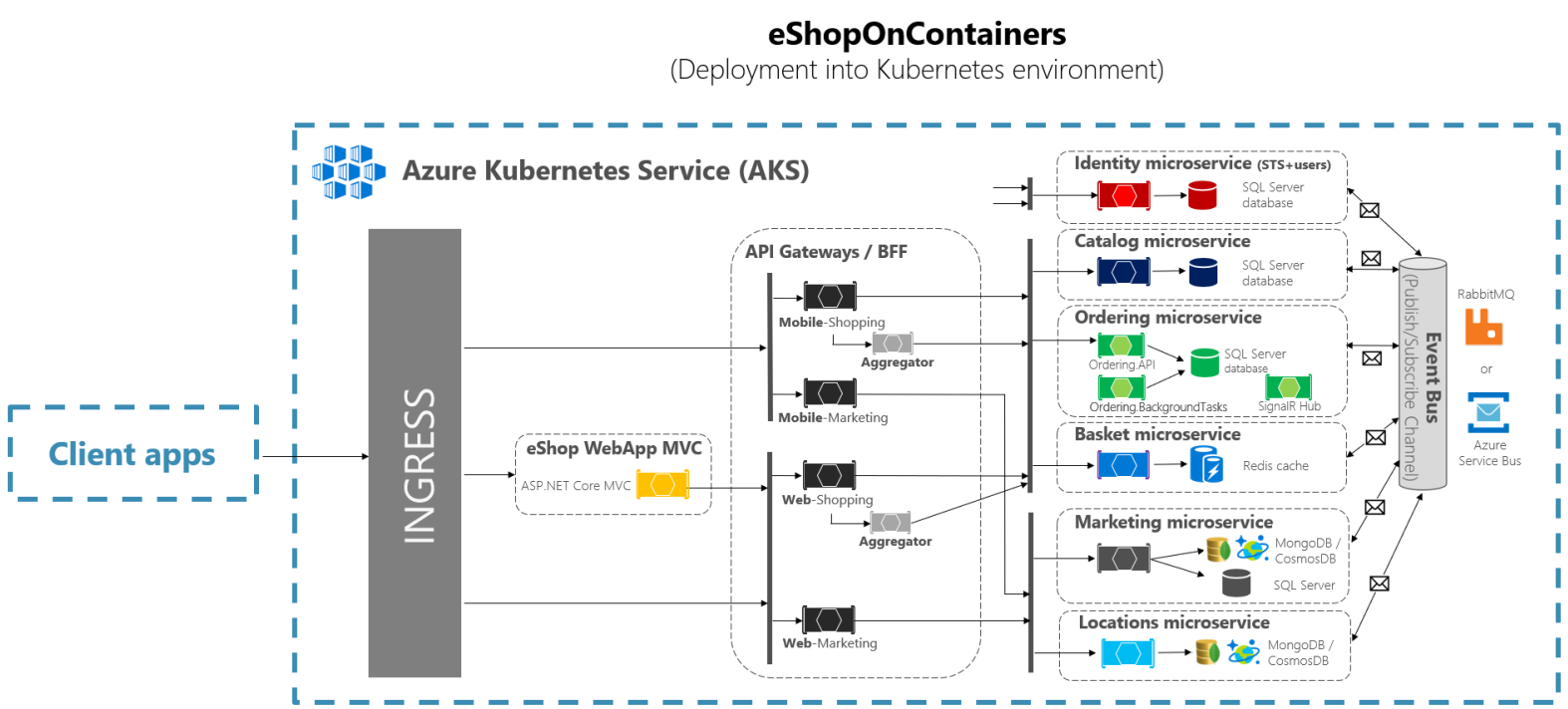
API Gateway 與 Microservice 開發

- 差異是設定檔



最終架構與回顧

- 搭配 K8S，可以方便快速的管理與擴展 Container
- 基礎設施架構，建議移出到高可用環境 (DB)
- API Gateway 可以分開或合併
  - 合併只須關注其中一個點
  - 分開可以依據商業邏輯自行發展
- 將部分功能轉移至 Gateway
- 透過 Queue 進行資料交換



More.

- 整合 Consul 服務探索
  - http://ocelot.readthedocs.io/en/latest/features/servicediscovery.html
- 快取
  - http://ocelot.readthedocs.io/en/latest/features/caching.html
- 記錄
  - http://ocelot.readthedocs.io/en/latest/features/logging.html
- 服務品質 (重試和斷路器)
  - http://ocelot.readthedocs.io/en/latest/features/qualityofservice.html
- 速率限制
  - http://ocelot.readthedocs.io/en/latest/features/ratelimiting.html

網關攻略

Thank You

.NET Conf 2018