Microsoft

Microservices Architecture on Azure

이정환
인피노브(주)

Agenda

Microservices – What, Why? (**마이크로서비스를 꼭 써야 하나**)

- **마이크로서비스 접근방식과 클라우드**
- **컨테이너 서비스와** MSA
- Azure **기반의 모범 사례**

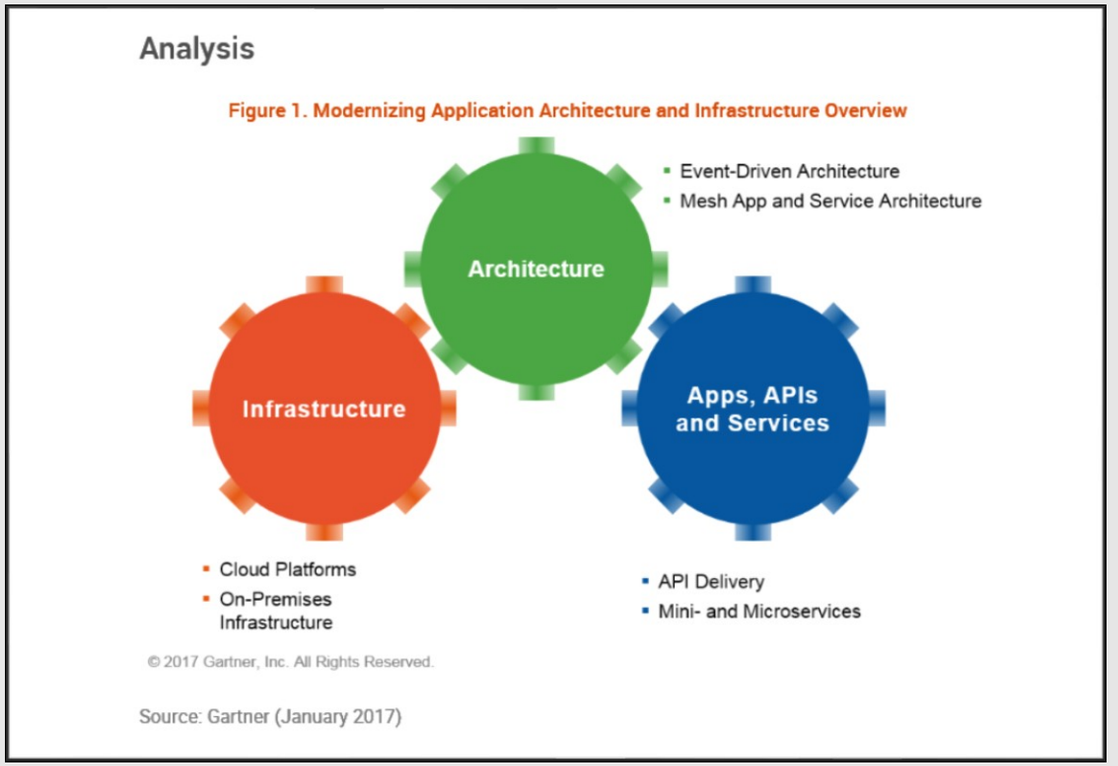
Agenda

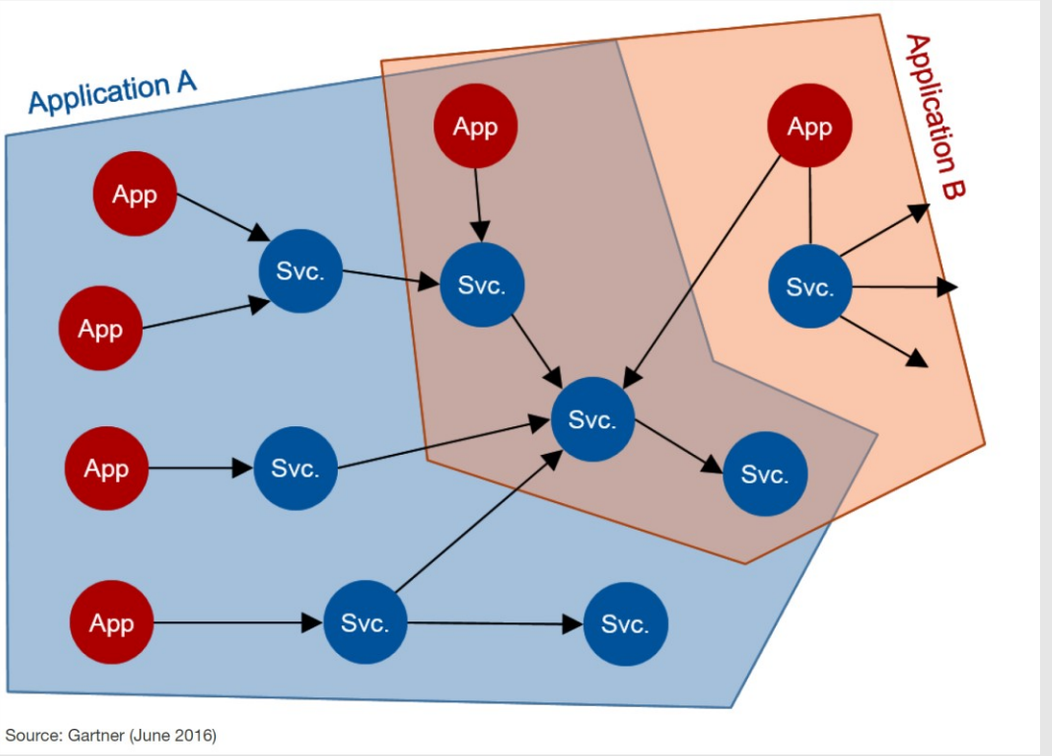
Microservices – What, Why? (마이크로서비스를 꼭 써야 하나)

- **마이크로서비스 접근방식과 클라우드**
- **컨테이너 서비스와** MSA
- Azure **기반의 모범 사례**

Microservices
What, why?

Why Microservices?





Not only a browser anymore (Retire the Three-Tier Application Architecture to Move Toward Digital Business)
MASA (Mesh App & Service Architecture) – TOP 10 Technology Trend by Gartner for 2017
Microservices – moving from monolythic design (deployment, scaling, maintenance)

*The mesh app and service architecture (MASA) is a multichannel solution architecture that **leverages cloud** and **serverless** computing, **containers** and **microservices** as well as **APIs** and events to deliver modular, flexible and dynamic solutions. Solutions ultimately support multiple users in multiple roles using multiple devices and communicating over multiple networks. However, MASA is a long term **architectural shift that requires significant changes** to development tooling and best practices. Gartner, Top 10 technology trends 2017*

New **patterns** and new **technologies**

Microservices

Autonomous
Bounded Context
Nomad & addressable services
Isolated
API Gateway
Decoupled
Events
Async. communication
Event Bus
Message Brokers
Service Discovery
Health Checks
Transient Failures Handling
Resiliency
Commands
Retries with Exponential Backoff
Circuit Breakers
Domain-Driven Design
CQRS simplified
Aggregates
Domain Entity
Domain Events
Mediator

Docker Containers

Linux Containers
Docker Image
Docker Host
Docker Registry
Windows Containers
Docker Hub
RabbitMQ
Hyper-V Containers
Azure Container Registry
Azure Service Bus
NServiceBus
MassTransit
Brighter
Polly

Orchestrators

Azure Service Fabric
Stateful Services
Actors
Azure Container Service
Kubernetes
Docker Swarm
Mesos DC/OS

Common Problems

일반목표: 퇴근을 제시간에 (또는 빨리) 하기 위해서...

- **개발이 복잡, 업무가 복잡**
- **테스트가 복잡. 절차가 많으면...**
- Deploy**가 오래 걸리면**
- **회의가 길어지면....**

Criteria

Maintainability

- How easy is it to maintain the code?
- How easy is it to fix a bug?

Monitoring

- How easy is it to monitor solution health?

Scalability

- How easy is it to add new computing power and handle heavier load?

Updates

- How easy is it to update solution to the newest version?

Onboarding

- How easy is it to get onboard a new team member?

What are Microservices?

비즈니스 > 서비스 > 마이크로서비스

- **서비스 : 유연한 비즈니스를 만들기 위한 분석의 결과물.**

- **마이크로서비스 아키텍처 : 이 결과물을 효과적으로 설계하기 위한 아키텍처적 접근**

What are Microservices?

- **서비스 : 유연한 비즈니스를 만들기 위한 분석의 결과물.**

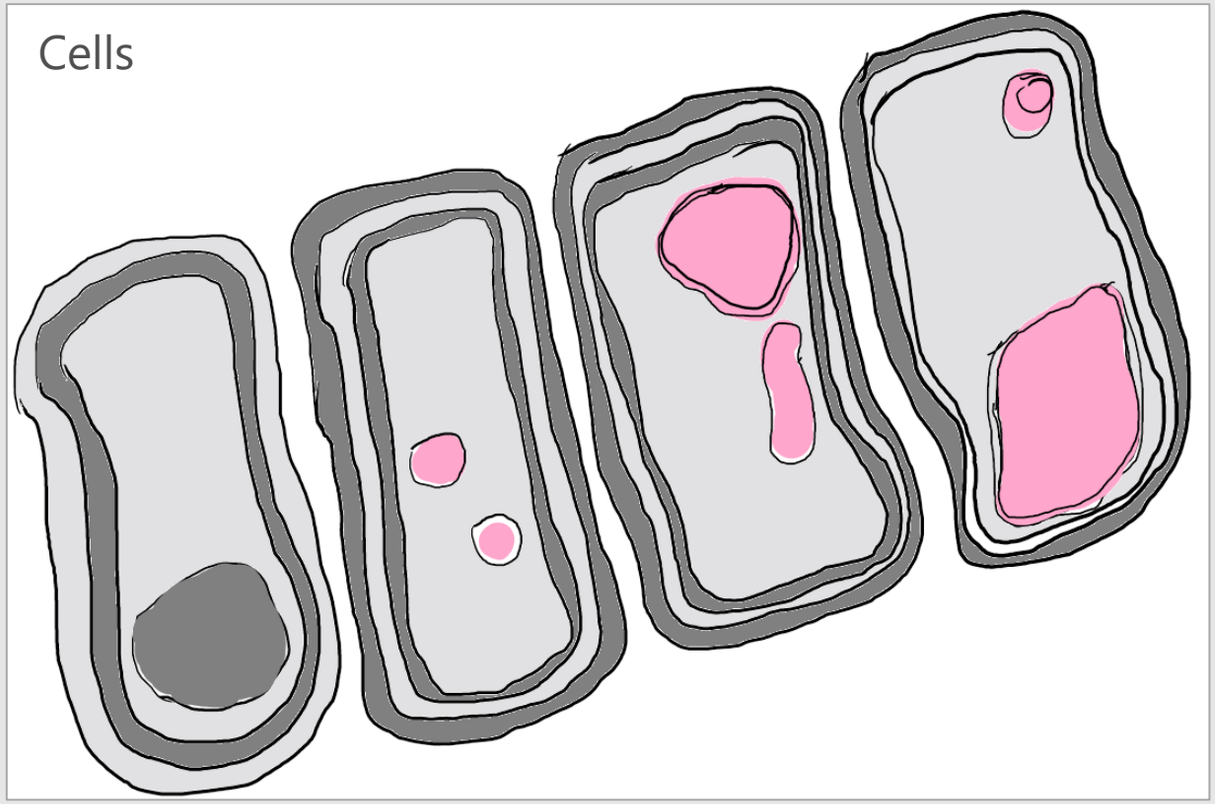
- **마이크로서비스 아키텍처 : 이 결과물을 효과적으로 설계하기 위한 아키텍처적 접근**

- People
- Process
- Technologies

Microservices

- Microservices is a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services.
- Microservices should be fine-grained and protocols should be lightweight.
- Application is easier to understand, develop and test.

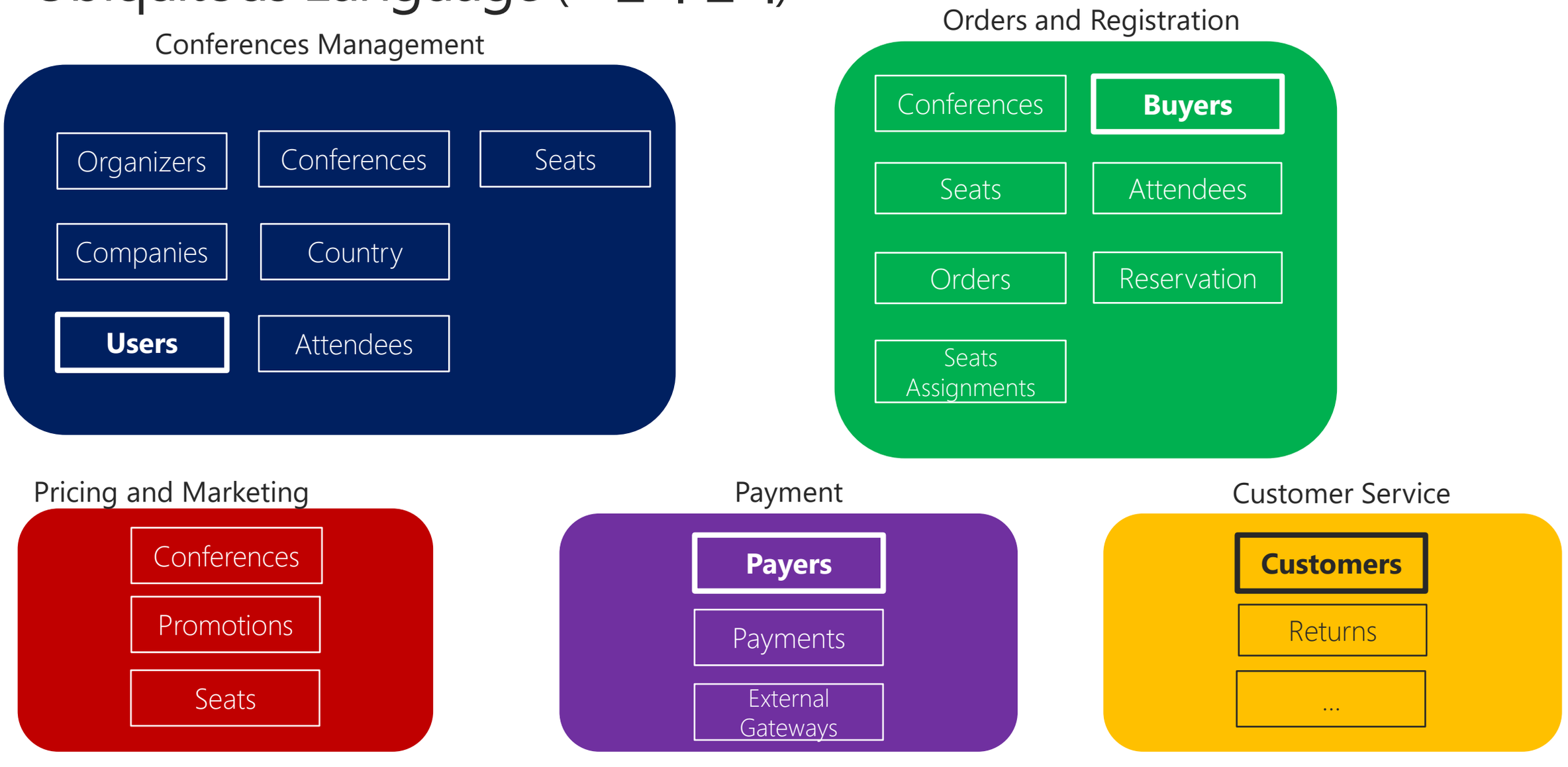
The Bounded Context pattern



Independent
Autonomous
Loosely coupled composition

"Cells can exist because their membranes define what is in and out and determine what can pass"
[Eric Evans]

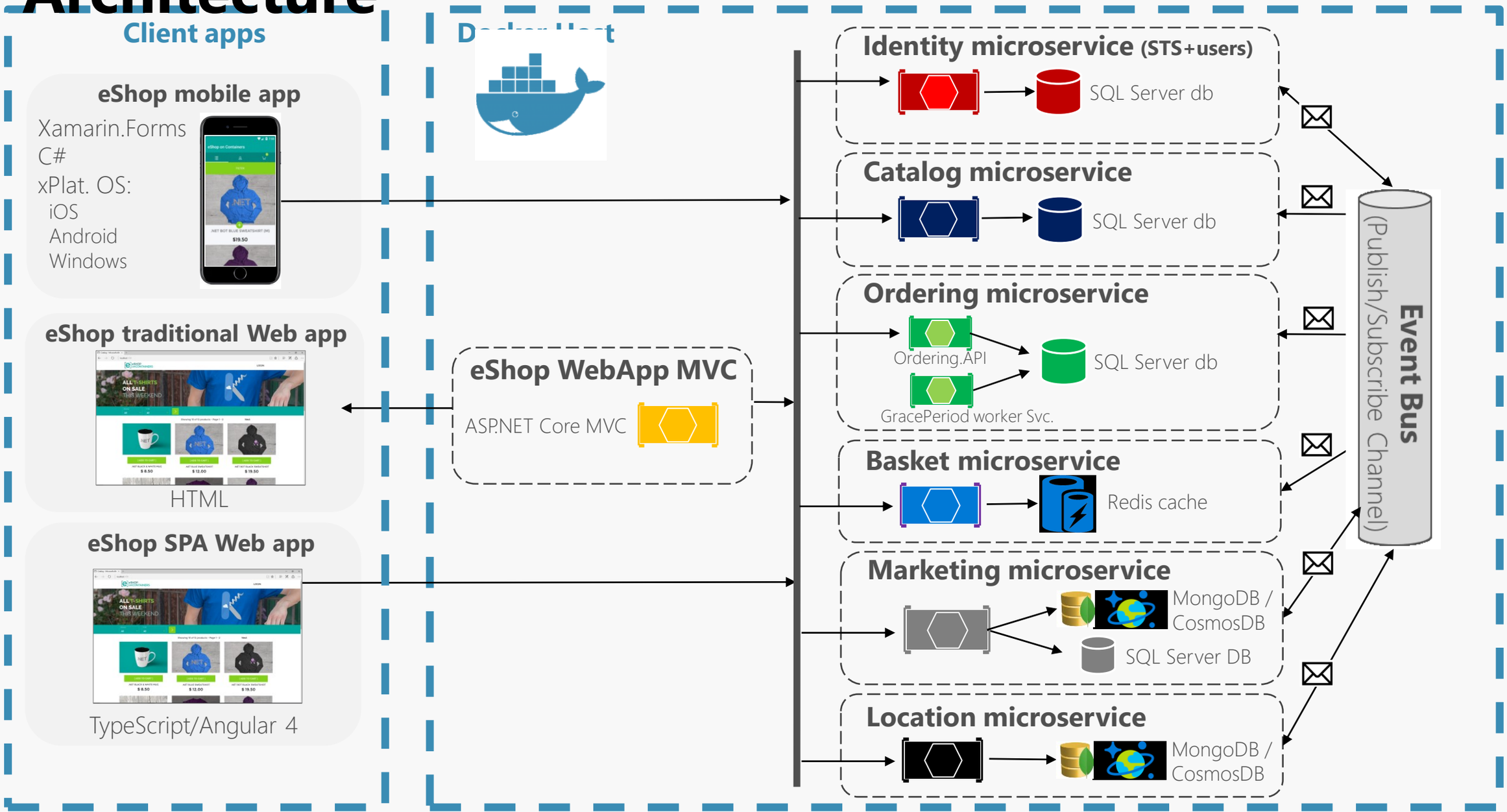
Identifying a Domain Model per Microservice/BoundedContext - Ubiquitous Language (보편적 언어)



Characteristics of a Microservice

- Services are small, independent, and loosely coupled.
- Each service is a separate codebase.
- Services can be deployed independently.
- Services are responsible for persisting their own data.
- Services communicate with each other by using well-defined APIs.
- Services don't need to share the same technology stack, libraries, or frameworks.

eShopOnContainers Reference Application - Architecture

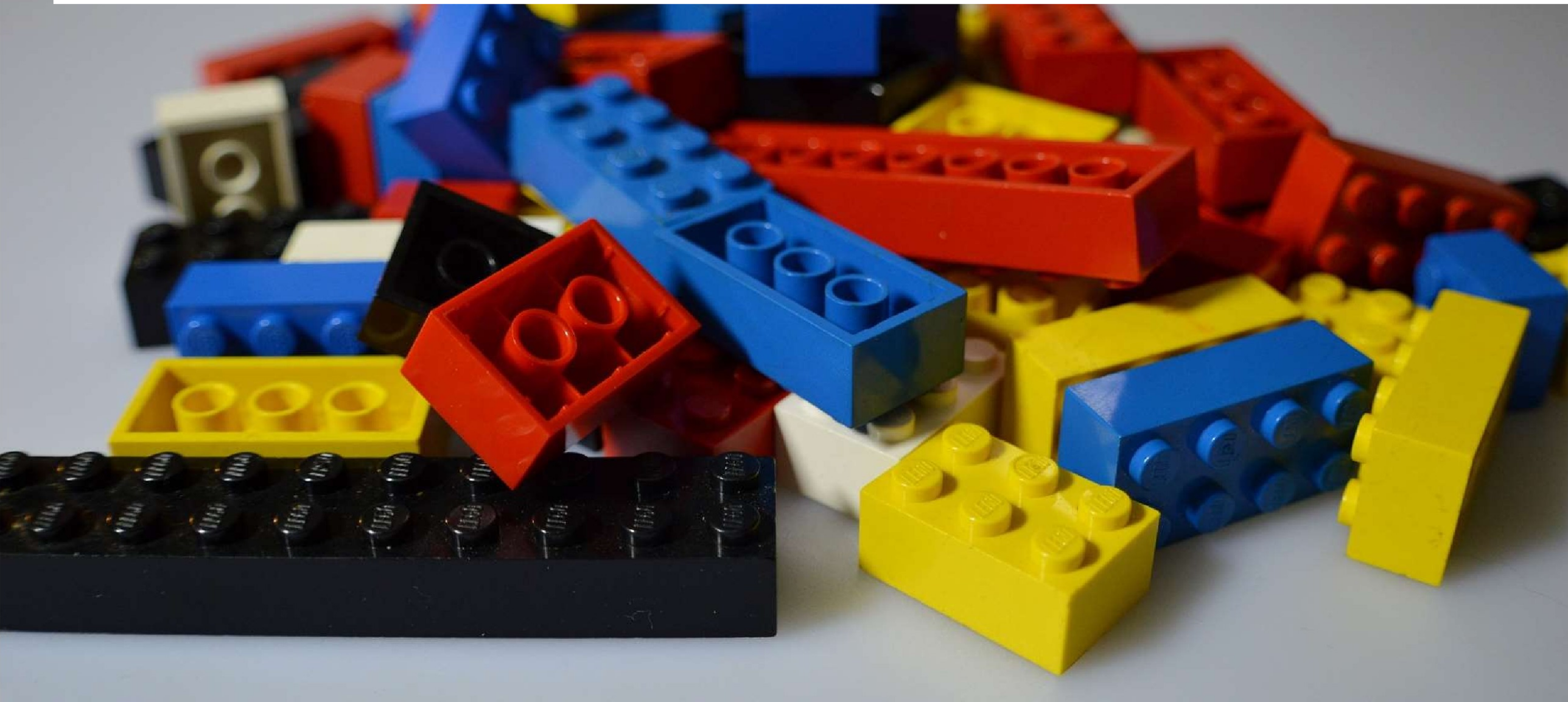


Agenda

Microservices – What, Why? (**마이크로서비스를 꼭 써야 하나**)

- **마이크로서비스 접근방식과 클라우드**
- **컨테이너 서비스와** MSA
- Azure **기반의 모범 사례**

They're building blocks.



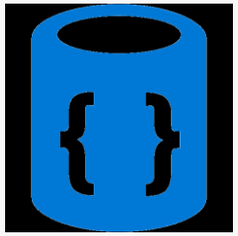
We can build simple structures from a few blocks,


or complex, integrated solutions from many
blocks
THE ANGLO
IRISH
IRISH

Services Powered by Service Fabric

SQL Database

2.1 million DBs



Cosmos DB

Billions transactions/day

IoT Hub

Millions of messages

Event Hubs

60bn events/day

Skype

Cortana

Intune

Dynamics

Power BI

Windows: GA
Linux: Preview

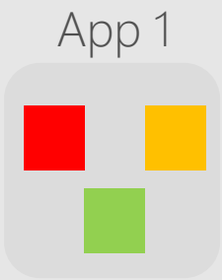
An n-Tier Web Application

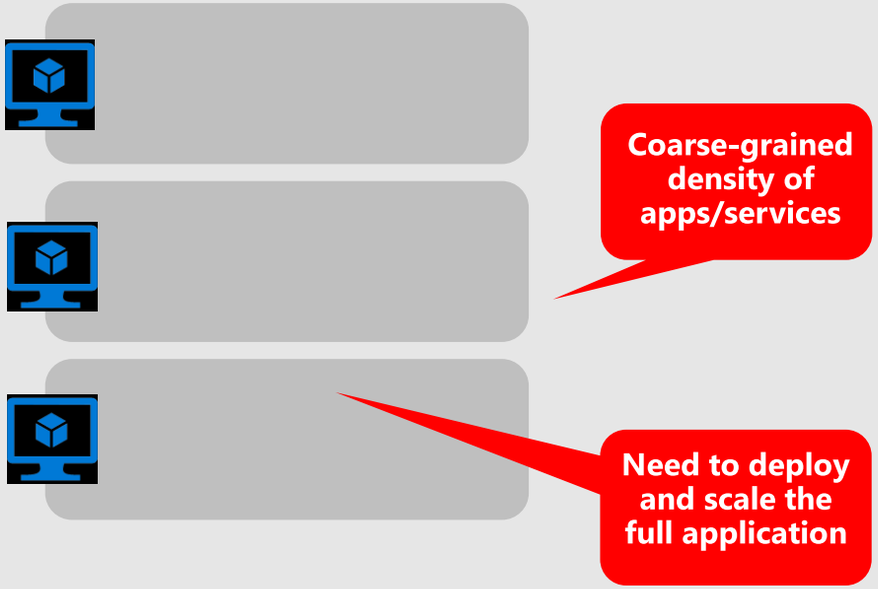
Why This Doesn't Work Well In The Cloud

1. Cloud-based network services are more abstract
2. Monoliths are difficult to maintain and scale
3. Requires lots of server/network configuration and admin
4. Doesn't make full use of cloud abstractions

Traditional application approach

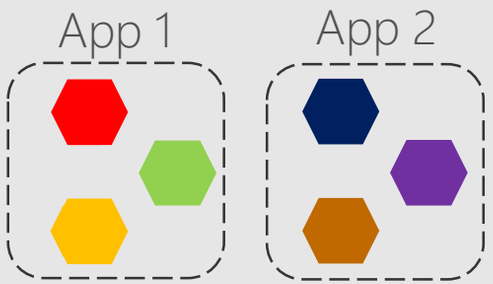
- A traditional application has most of its functionality within a few processes that are componentized with layers and libraries.
- Scales by cloning the app on multiple servers/VMs

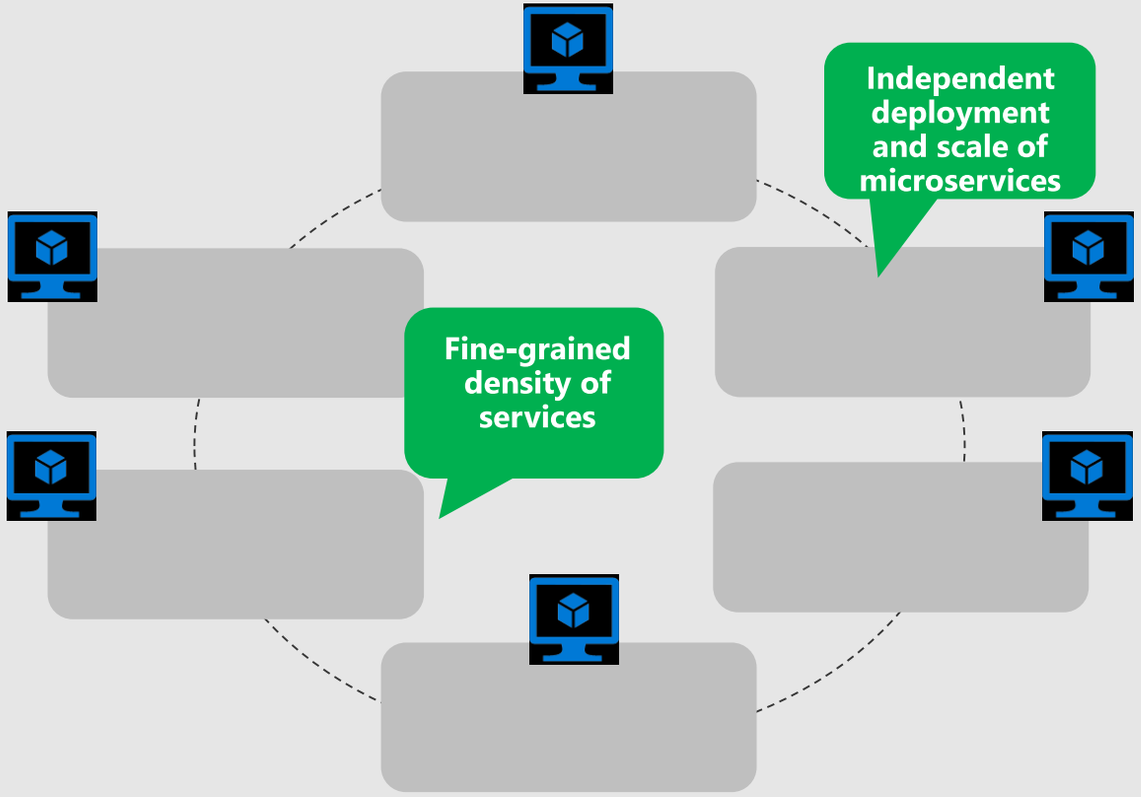




Microservices application approach

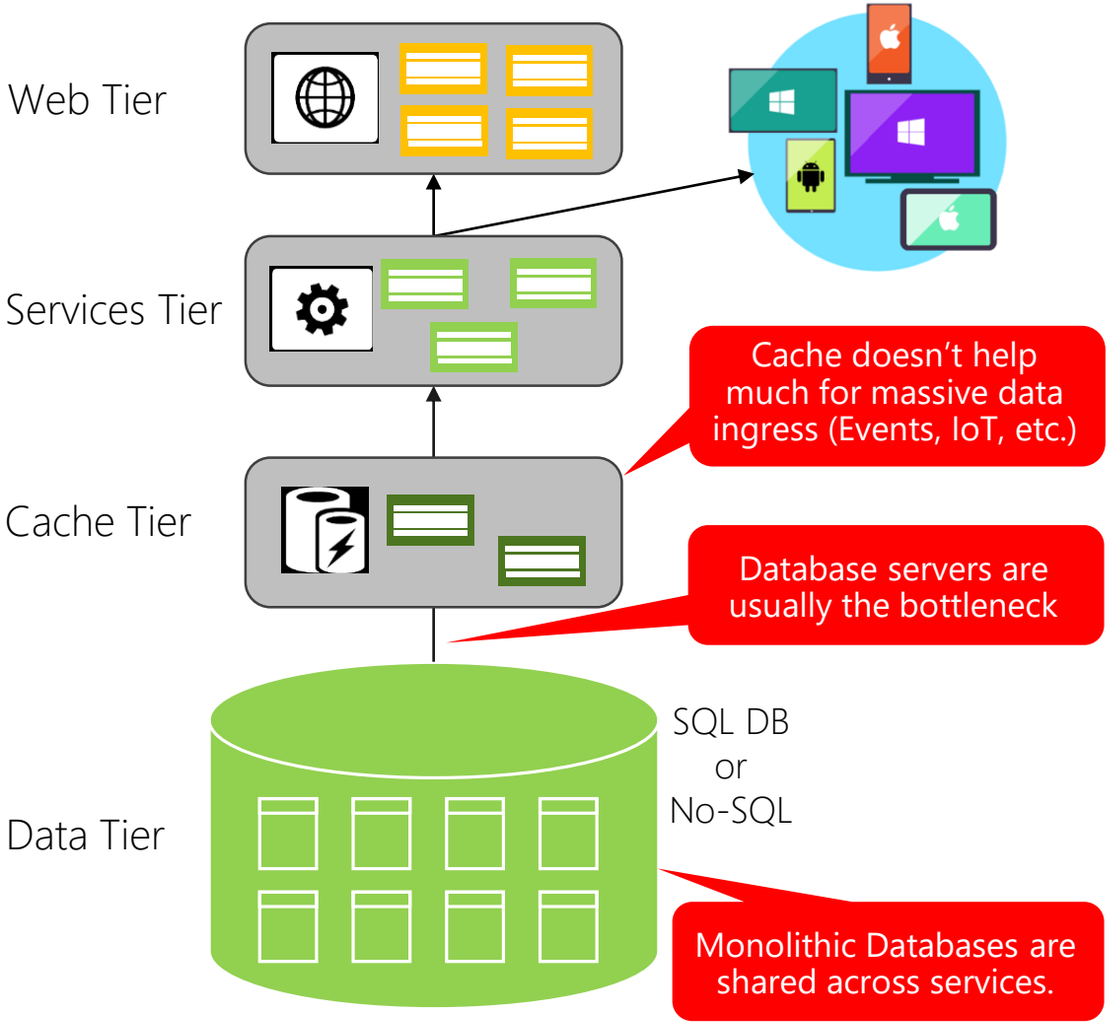
- A microservice application segregates functionality into separate smaller services.
- Scales out by **deploying each service independently** with multiple instances across servers/VMs





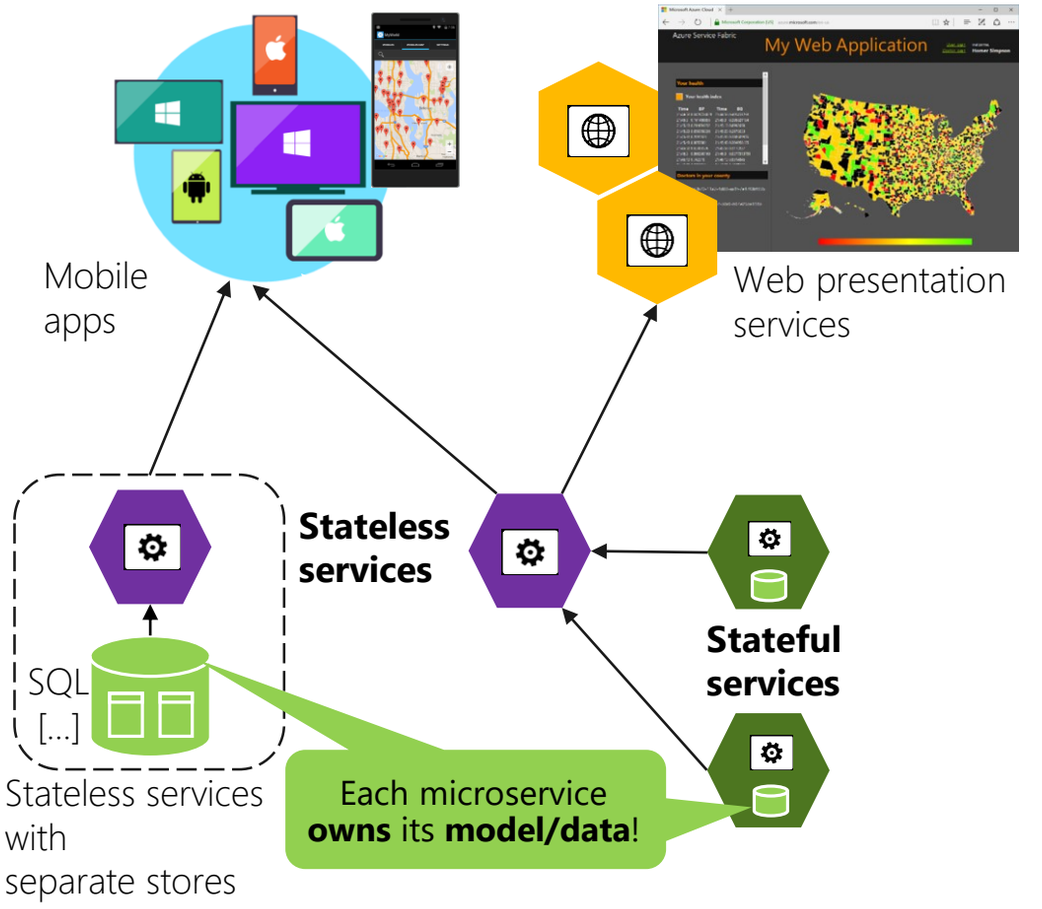
Data in Traditional approach

- Single monolithic database
- Tiers of specific technologies



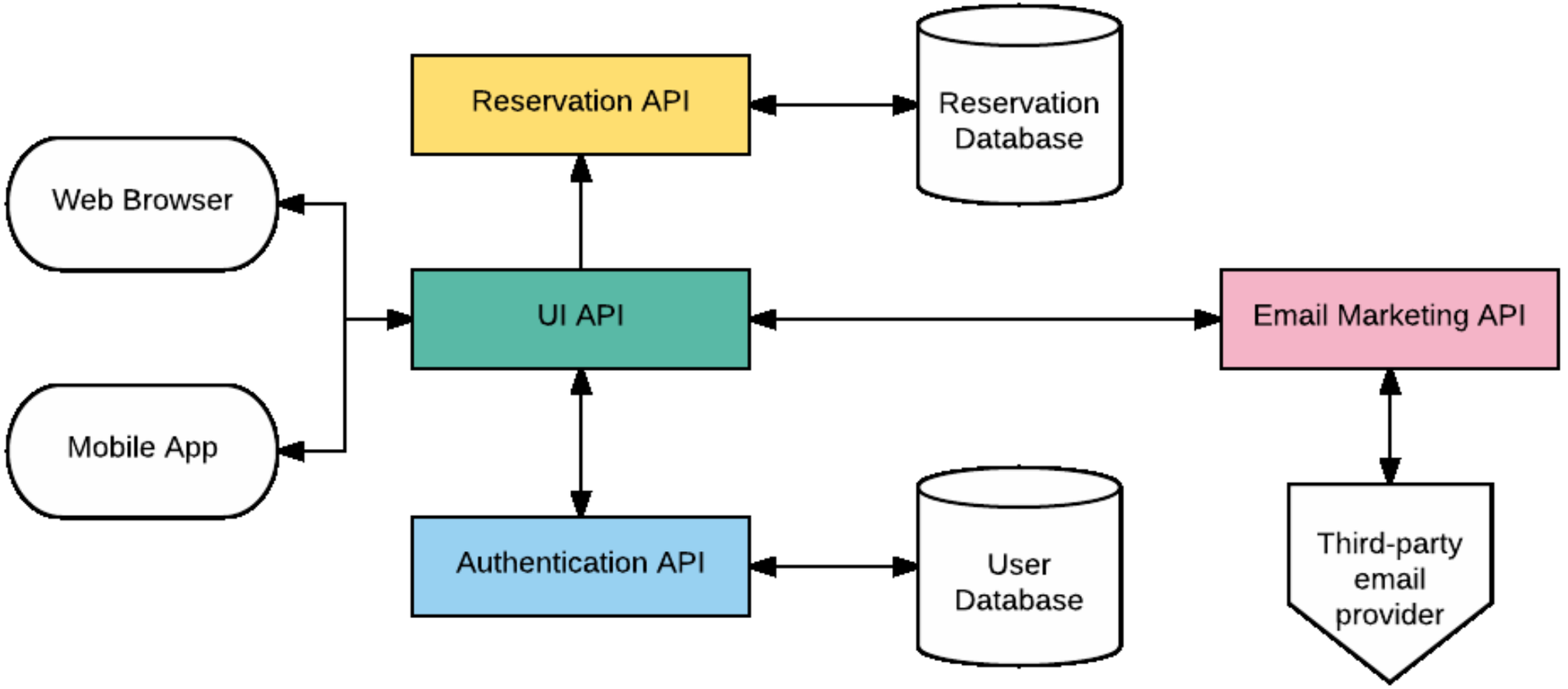
Data in Microservices approach

- Graph of interconnected microservices
- State typically scoped to the microservice
- Remote Storage for cold data

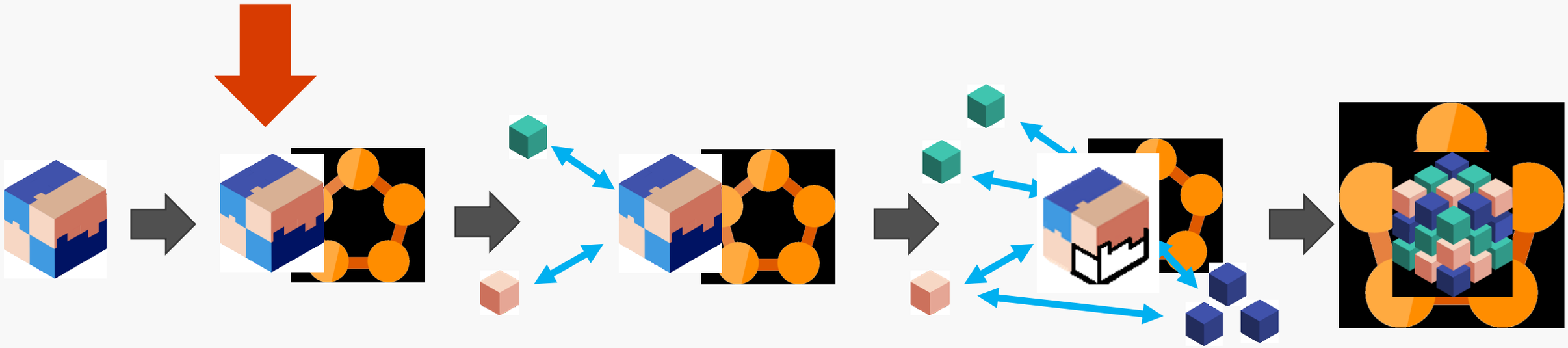


Microservices To The Rescue!

Microservices-Based Web Application



Migrating a traditional application to microservices



1) Traditional app
2) Hosted as guest executables or containers in Service Fabric
3) Simple modernization - new microservices added alongside
4) Deeper modernization - breaking app into microservices
5) Transformed into microservices

...You can stop at any stage

Service Fabric: A Microservices Platform

Build Applications with many Languages, Frameworks, & Runtimes

Service Fabric: Microservices Platform

Lifecycle Mgmt | Independent Scaling | Independent Updates | Always On Availability | Resource Efficient | Stateless/ Stateful

Public Cloud

On Premises
Private cloud

Other Clouds

Developer

Agenda

Microservices – What, Why? (**마이크로서비스를 꼭 써야 하나**)

- **마이크로서비스 접근방식과 클라우드**
- **컨테이너 서비스와 MSA**
- Azure **기반의 모범 사례**

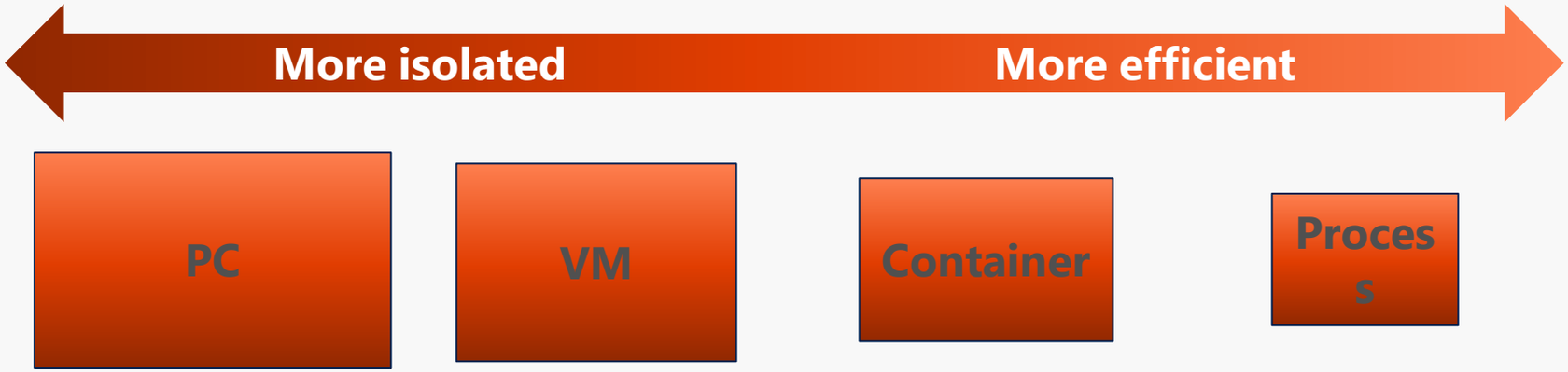
Microservices != Containers

But they are a great fit… ☺

Microservices and containers

“Microservices is an architectural design point; containers are an implementation detail that often helps.”

Density & Isolation levels

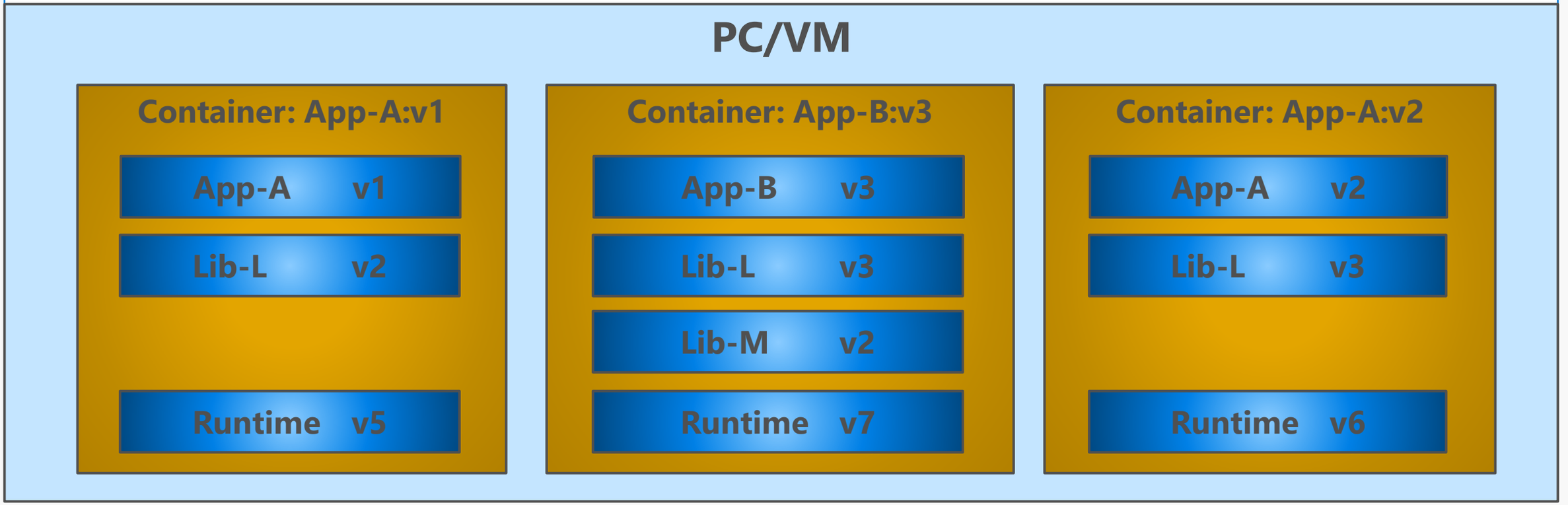


Hardware	Not shared	Shared	Shared	Shared
Kernel	Not shared	Not shared	Shared*	Shared
System Resources (ex: File System)	Not shared	Not shared	Not shared	Shared

* Windows Hyper-V containers do not share a kernel

What is a container

- Slices up the OS to run multiple apps on a single PC/VM.
- Every container gets its
  - Own root directory, network interface, own process Id
- Allows a versioned set of dependencies (components, runtimes, etc.) to run side-by-side with another set of dependencies in an isolated runtime environment on the same PC/VM



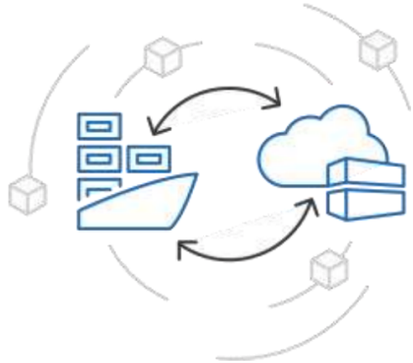
Azure Container Instances (ACI)

Serverless Containers

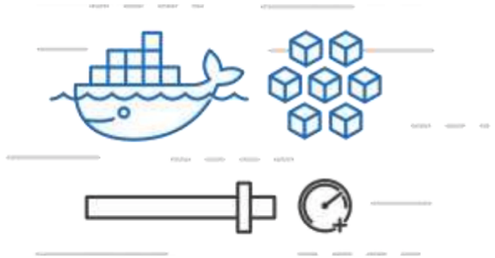
GA (Windows and Linux)
Preview of AKS and ACI

Benefits:

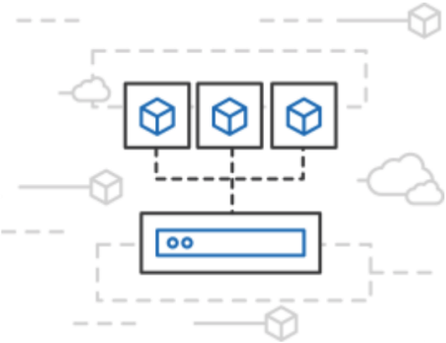
- Fastest and easiest way to run a container in the cloud
- No VM management
- Per-GB, Per-CPU, and Per-second billing
- Deploy images from DockerHub or Azure Container Registry
- Allows customers to opt-in to having the Azure Container Instances connector configured in their AKS cluster, without having to set it up themselves. The ACI connector enables customers to deploy additional container capacity for their AKS cluster using ACI.



Start using containers right away



Cloud-scale container capacity



Hyper-visor isolation

https://azure.microsoft.com/en-us/blog/azure-container-instances-now-generally-available/

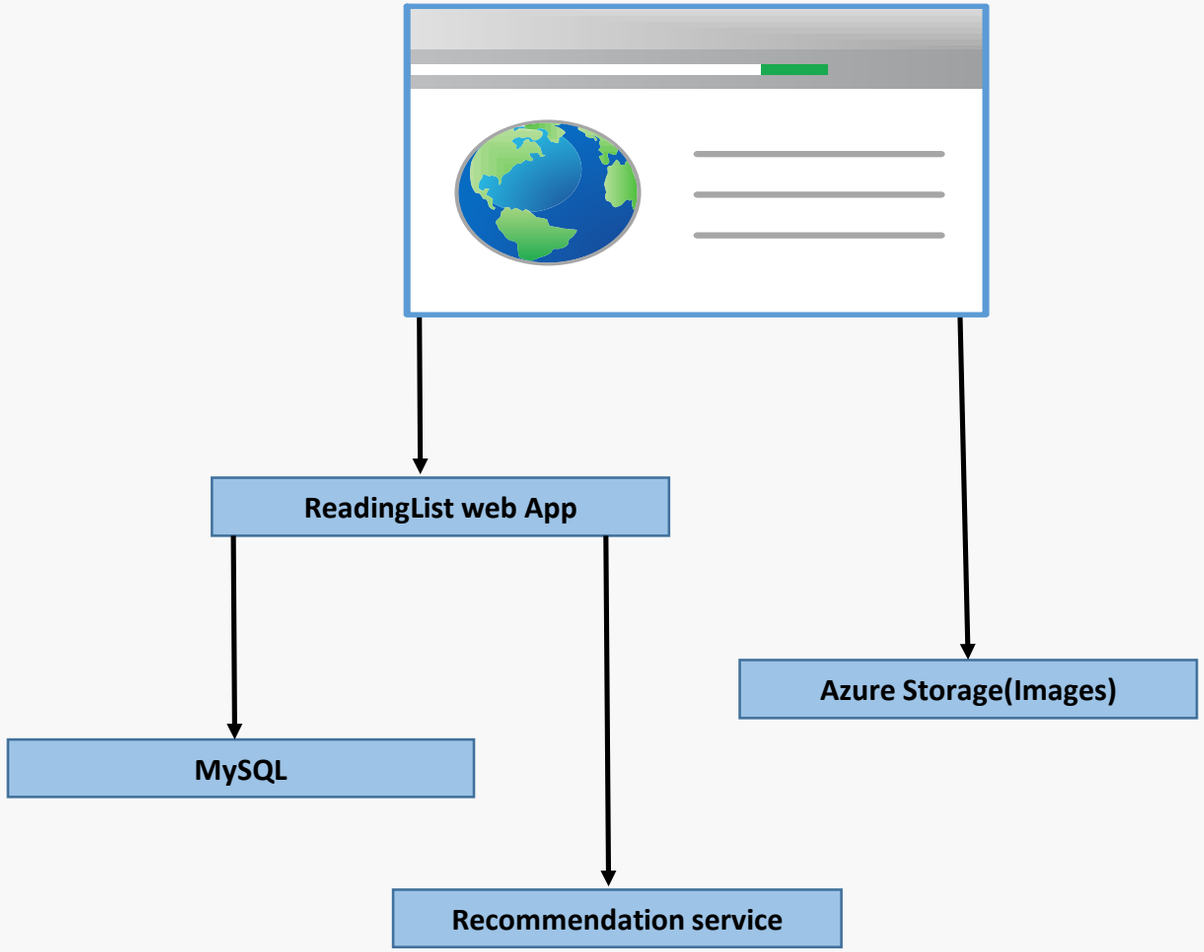
Azure Container Instances (ACI)

Get started easily

```
> az container create --name mycontainer --image microsoft/aci-helloworld --
resource-group myResourceGroup --ip-address public
  "ipAddress": {
    "ip": "52.168.86.133",
    "ports": [...]
  },
  "location": "eastus",
  "name": "mycontainer",
  "osType": "Linux",
  "provisioningState": "Succeeded",

> curl 52.168.86.133
<html>
<head>
  <title>Welcome to Azure Container Instances!</title>
</head>
```

But what if I need…

- Auto-scaling
- Rolling upgrades
- Service discovery
- Integrated load balancing
- Affinity/anti-affinity



Azure Kubernetes Service (AKS)

A fully managed Kubernetes cluster

Simplify the deployment, management, and operations of Kubernetes

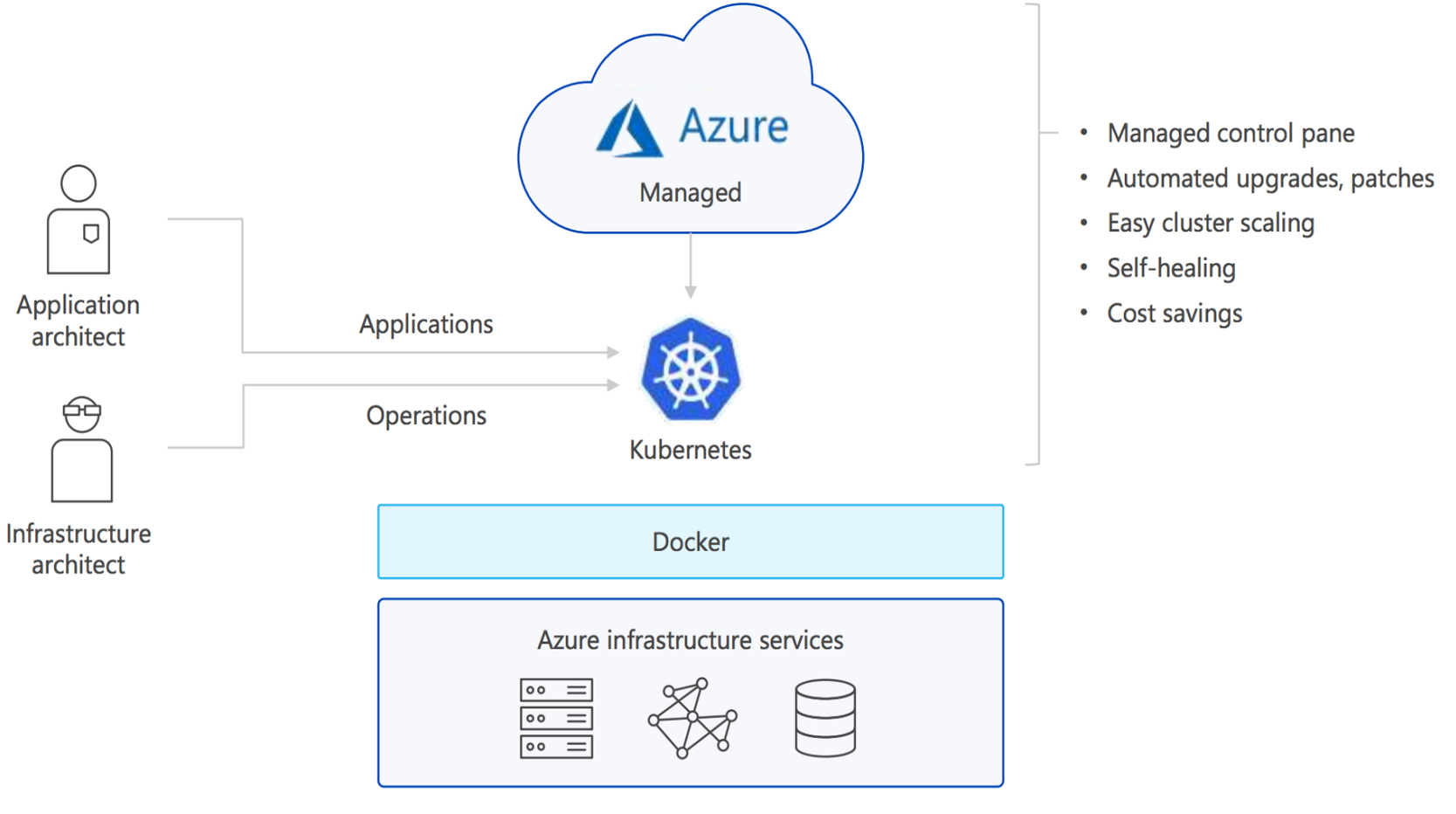
Easily manage clusters without container expertise

No per-cluster charge. Only pay for resources consumed.

Healing, auto-scaling, load balancing

Reliable, zero-downtime rollout of software versions

OSBA pre installed option enables customers to use the Open Service Broker for Azure with the Azure Container Service without having to first set it up.



Azure Kubernetes Service (AKS)

Get started easily

```
> az aks create -g myResourceGroup -n myCluster --generate-ssh-keys
\ Running ..

> az aks install-cli
Downloading client to /usr/local/bin/kubectl ..

> az aks get-credentials -g myResourceGroup -n myCluster
Merged "myCluster" as current context ..

> kubectl get nodes
NAME                          STATUS    AGE       VERSION
aks-mycluster-36851231-0      Ready     4m        v1.8.1
aks-mycluster-36851231-1      Ready     4m        v1.8.1
aks-mycluster-36851231-2      Ready     4m        v1.8.1
```

Azure Kubernetes Service (AKS)

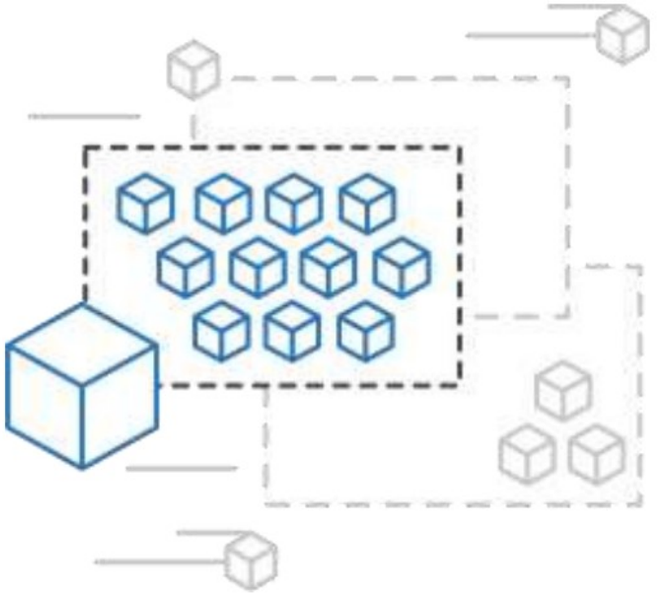
Manage an AKS cluster

```
> az aks list –o table
Name                Location    ResourceGroup     KubernetesRelease  ProvisioningState
------------------  ----------  --------------  -------------------  ------------------
myCluster           westus2     myResourceGroup                1.7.7  Succeeded

> az aks upgrade -g myResourceGroup -n myCluster –-kubernetes-version 1.8.1
\ Running ..

> kubectl get nodes
NAME                          STATUS    AGE       VERSION
aks-mycluster-36851231-0      Ready     12m       v1.8.1
aks-mycluster-36851231-1      Ready     8m        v1.8.1
aks-mycluster-36851231-2      Ready     3m        v1.8.1

> az aks scale -g myResourceGroup -n myCluster --agent-count 10
\ Running ..
```

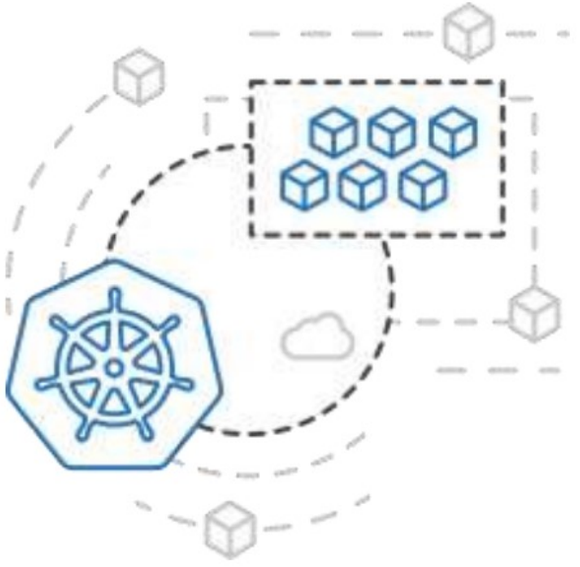
ACI Connector for Kubernetes

aka "Virtual Kubelet"

Kubernetes provides rich orchestration capabilities

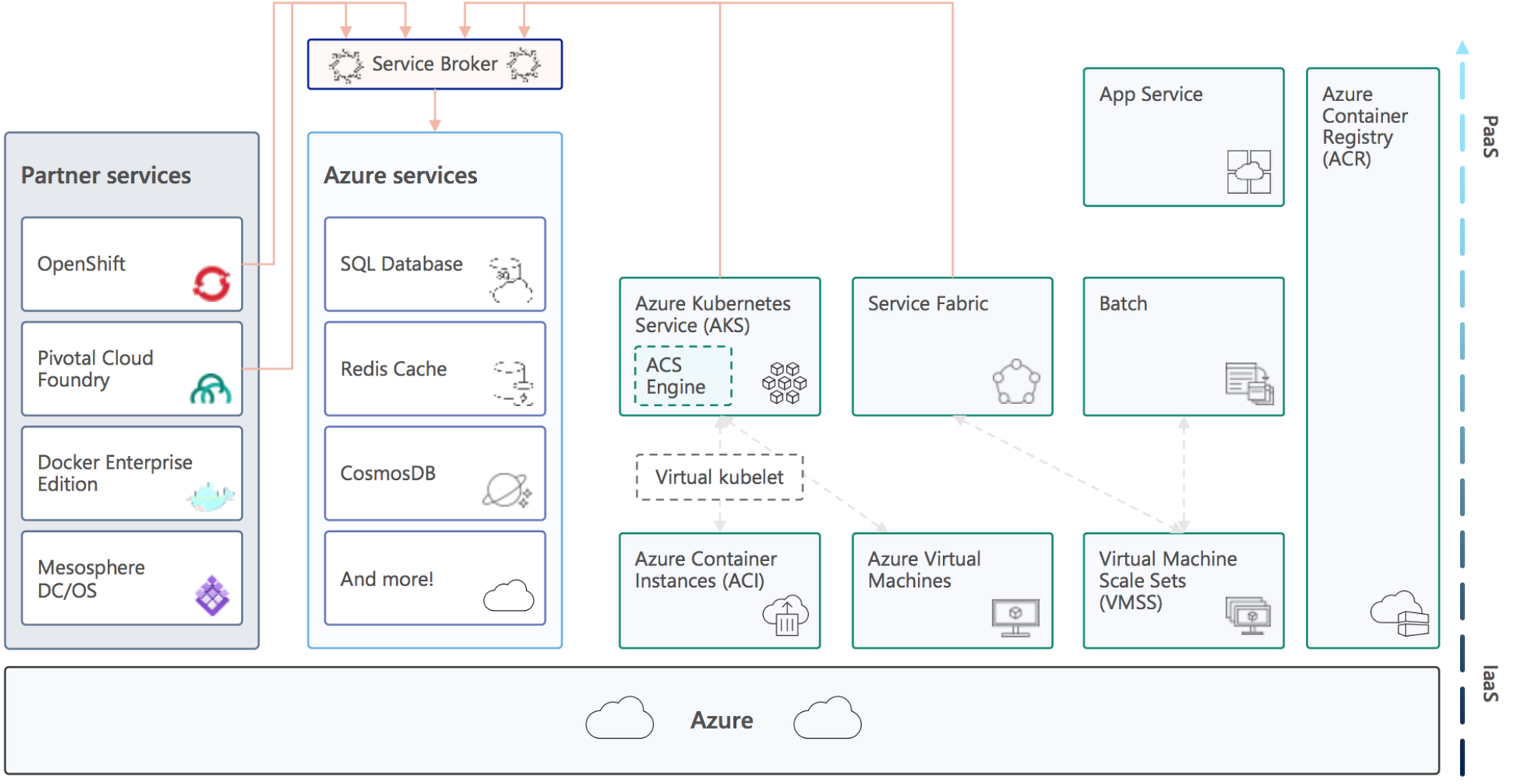


ACI provides infinite container-based scale



The ACI Connector for K8s brings them together

Container Deployment Options



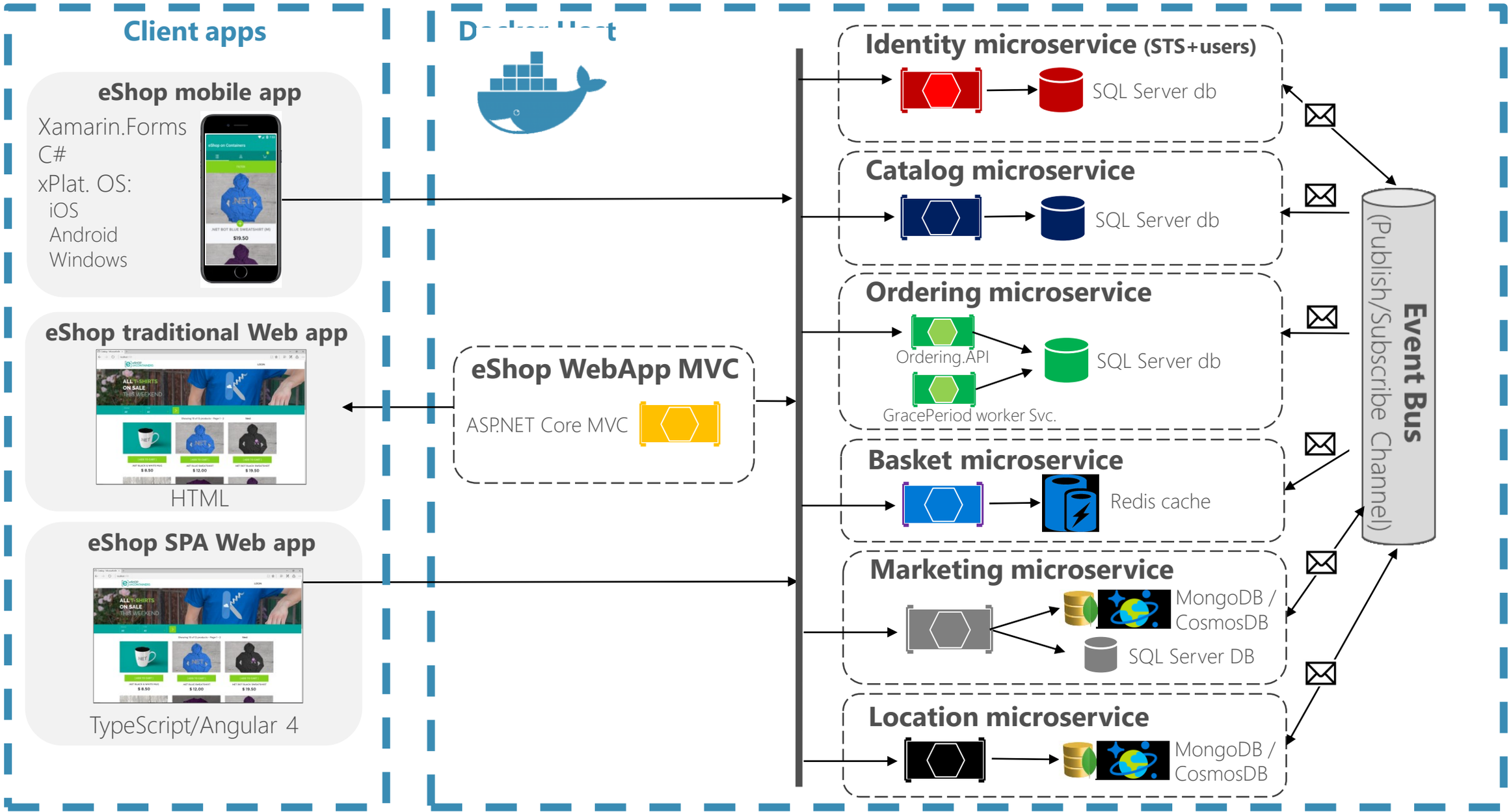
Agenda

Microservices – What, Why? (**마이크로서비스를 꼭 써야 하나**)

- **마이크로서비스 접근방식과 클라우드**
- **컨테이너 서비스와** MSA
- **Azure 기반의 모범 사례**

eShopOnContainers Reference Application - Architecture

aka.ms/MicroservicesArchitecture

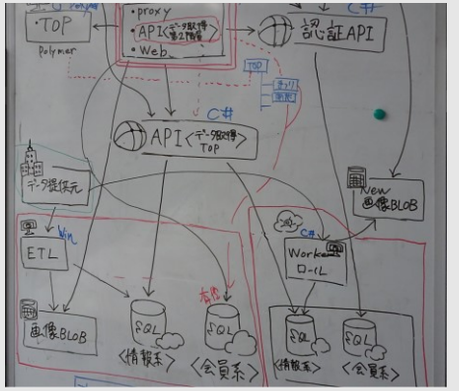


Microservices
Challenges & patterns

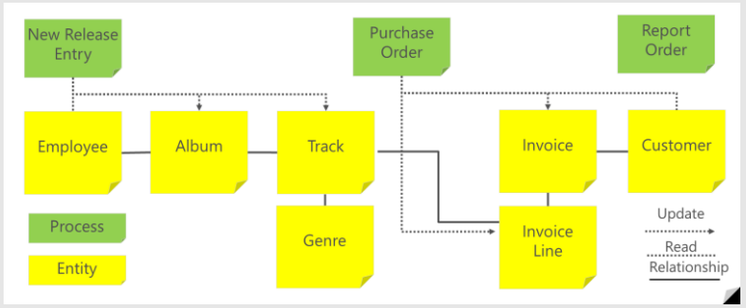
How to decompose Monolith to Microservices

1. Site Structure

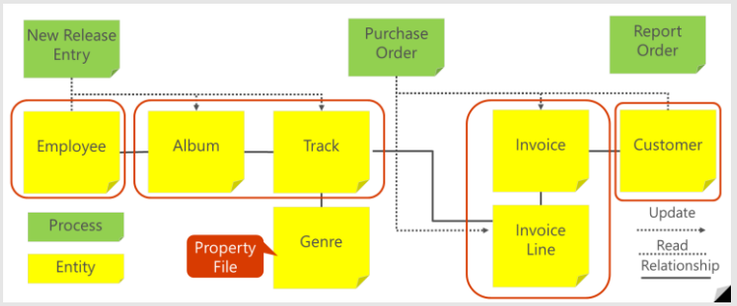
2. Domain Modeling



3. Architecture discussion

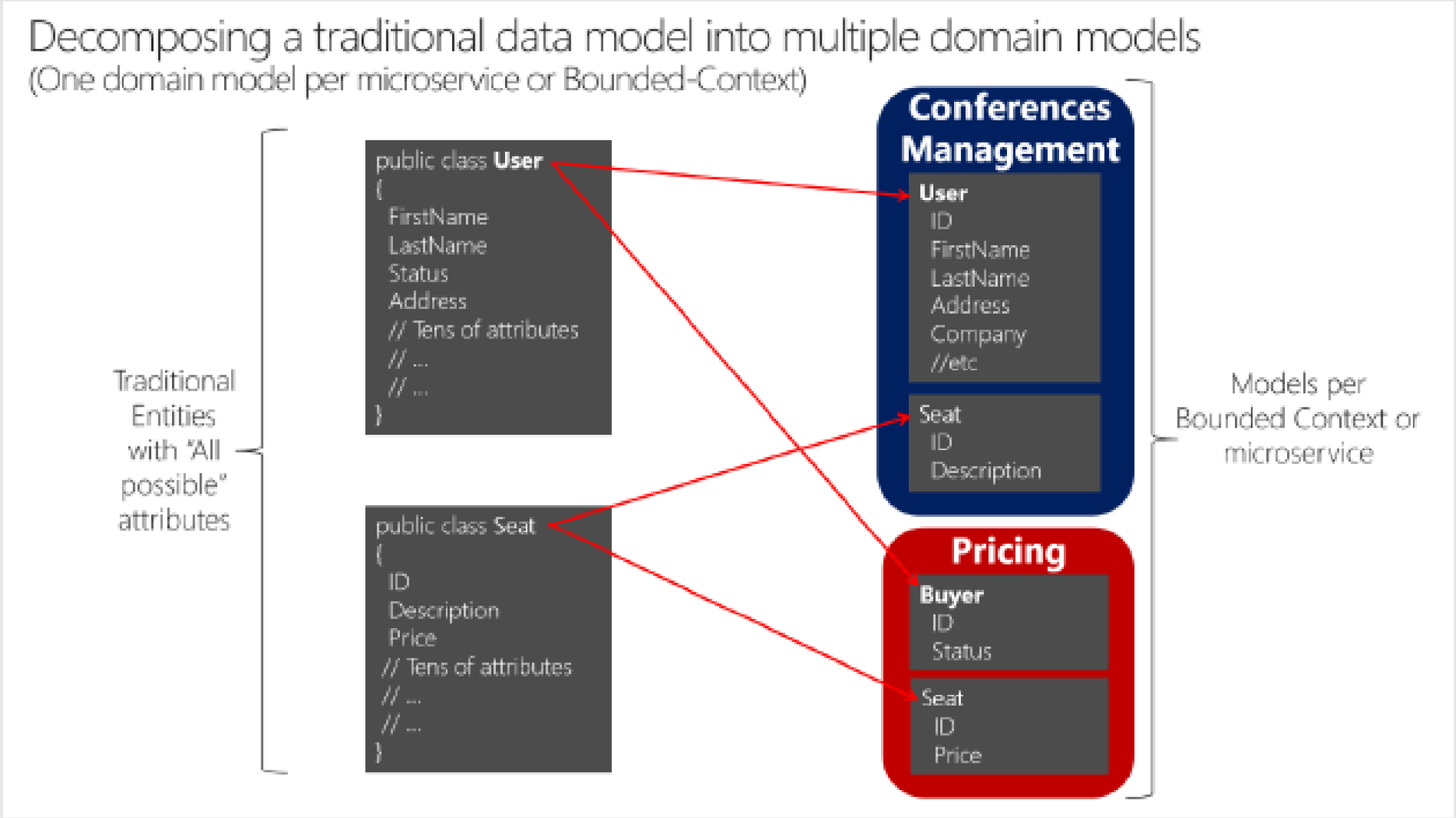


4. Draw dependency

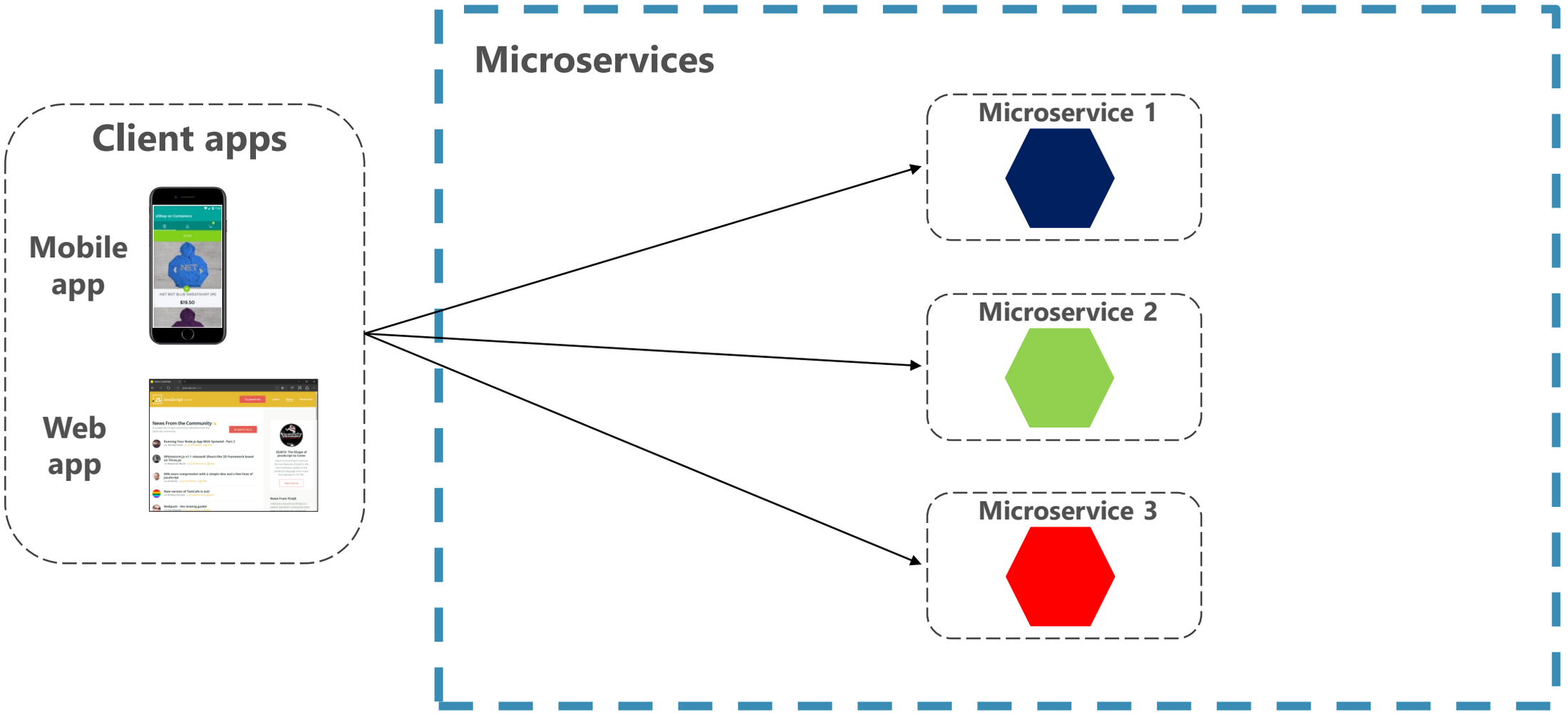


5. Set the boundary

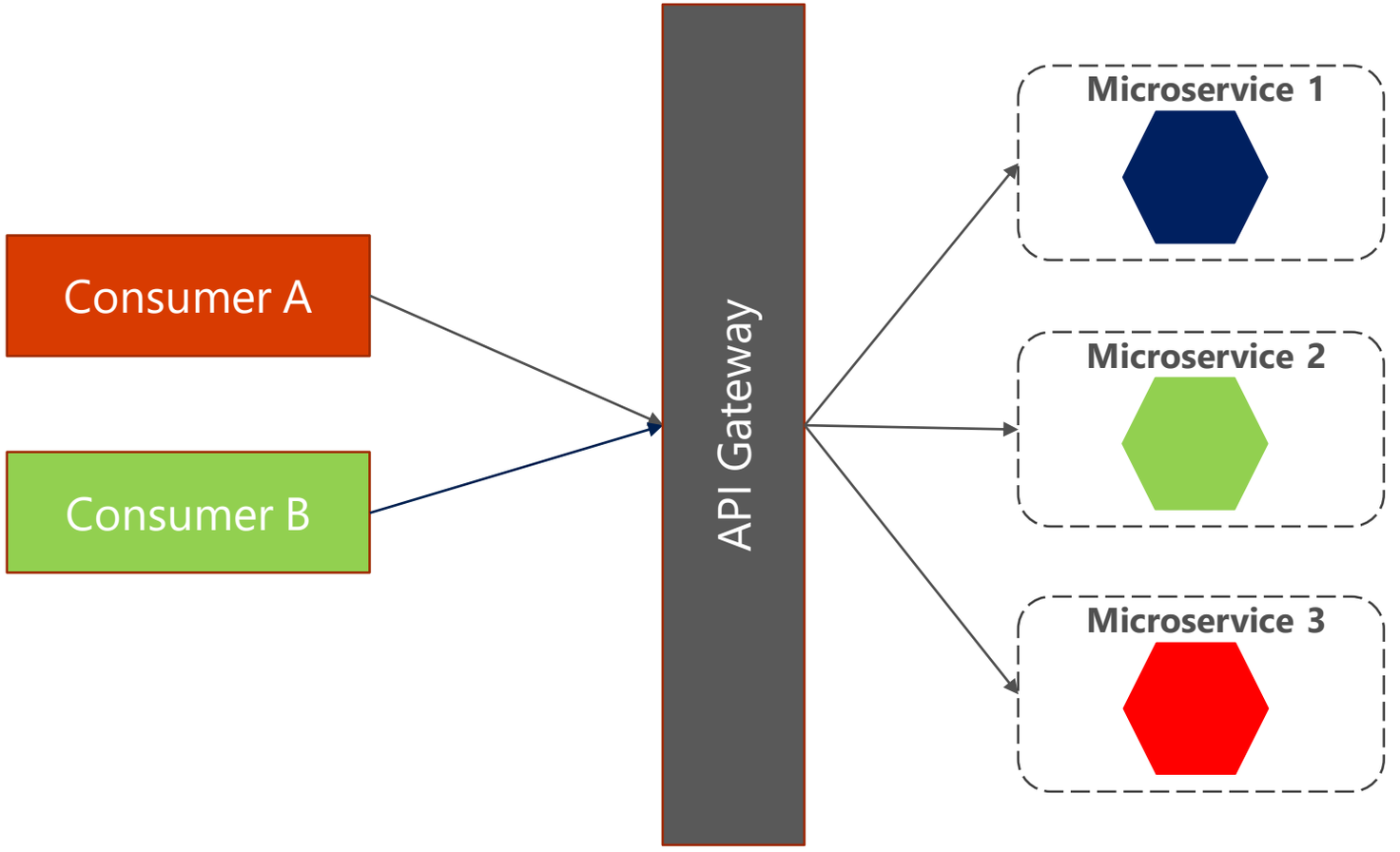
Decomposing a traditional data model



Direct Client-To-Microservice communication



API Gateway



Calls aggregation

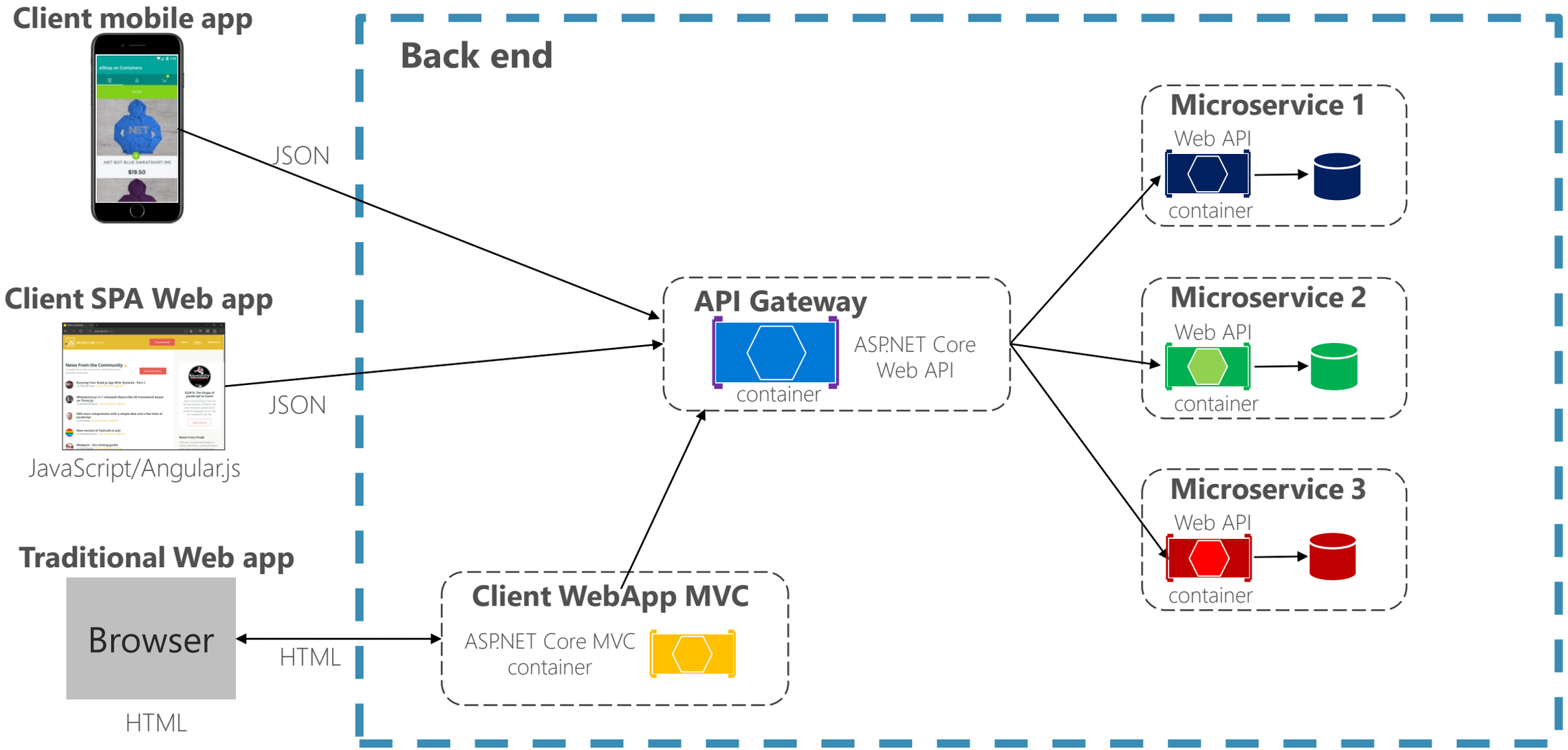
Who is consuming our services?

Who was consuming what?

What rate?

What time?

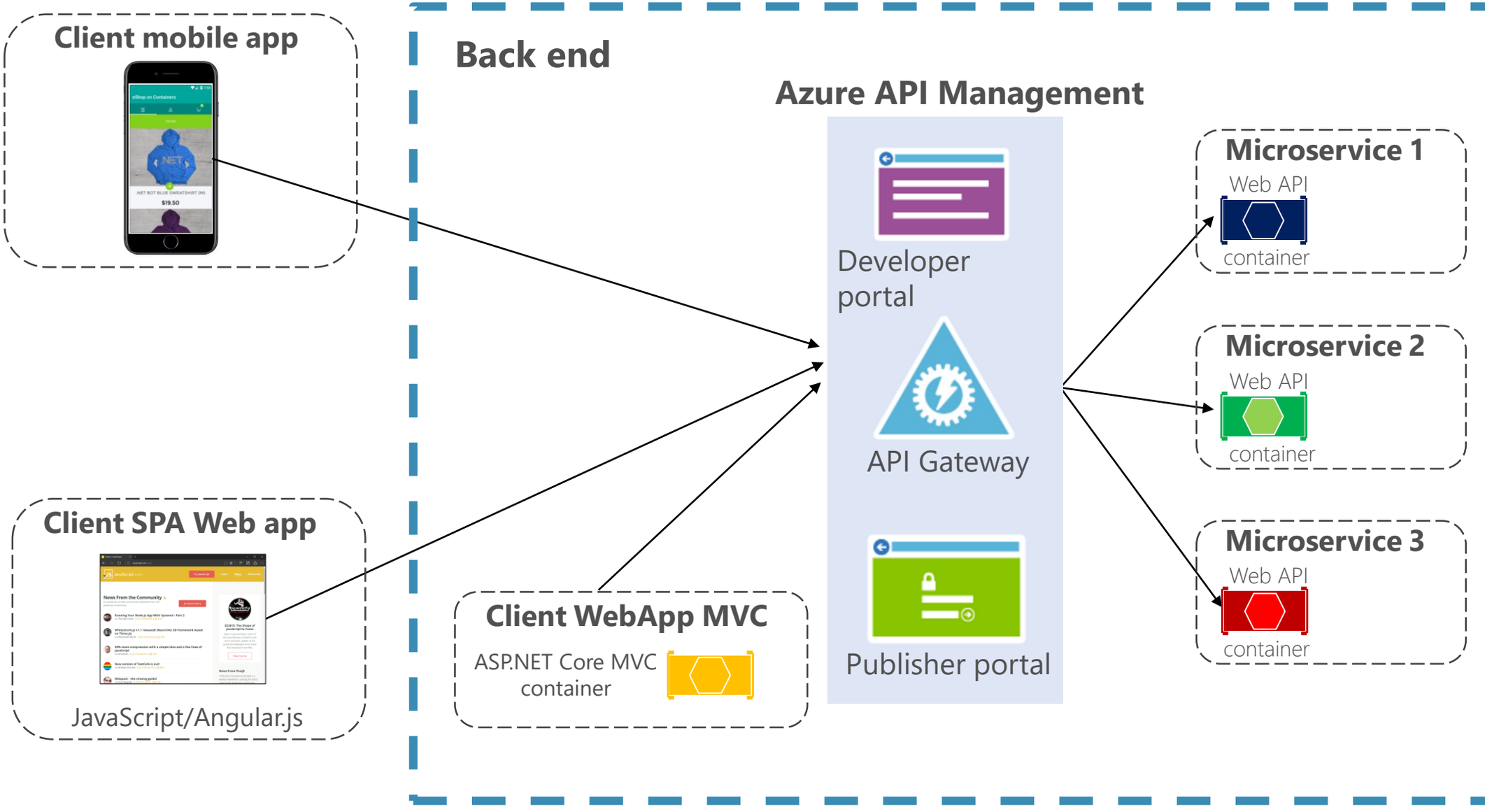
Using a **custom** API Gateway Service



API Gateway “as a service/product”

AZURE API MANAGEMENT

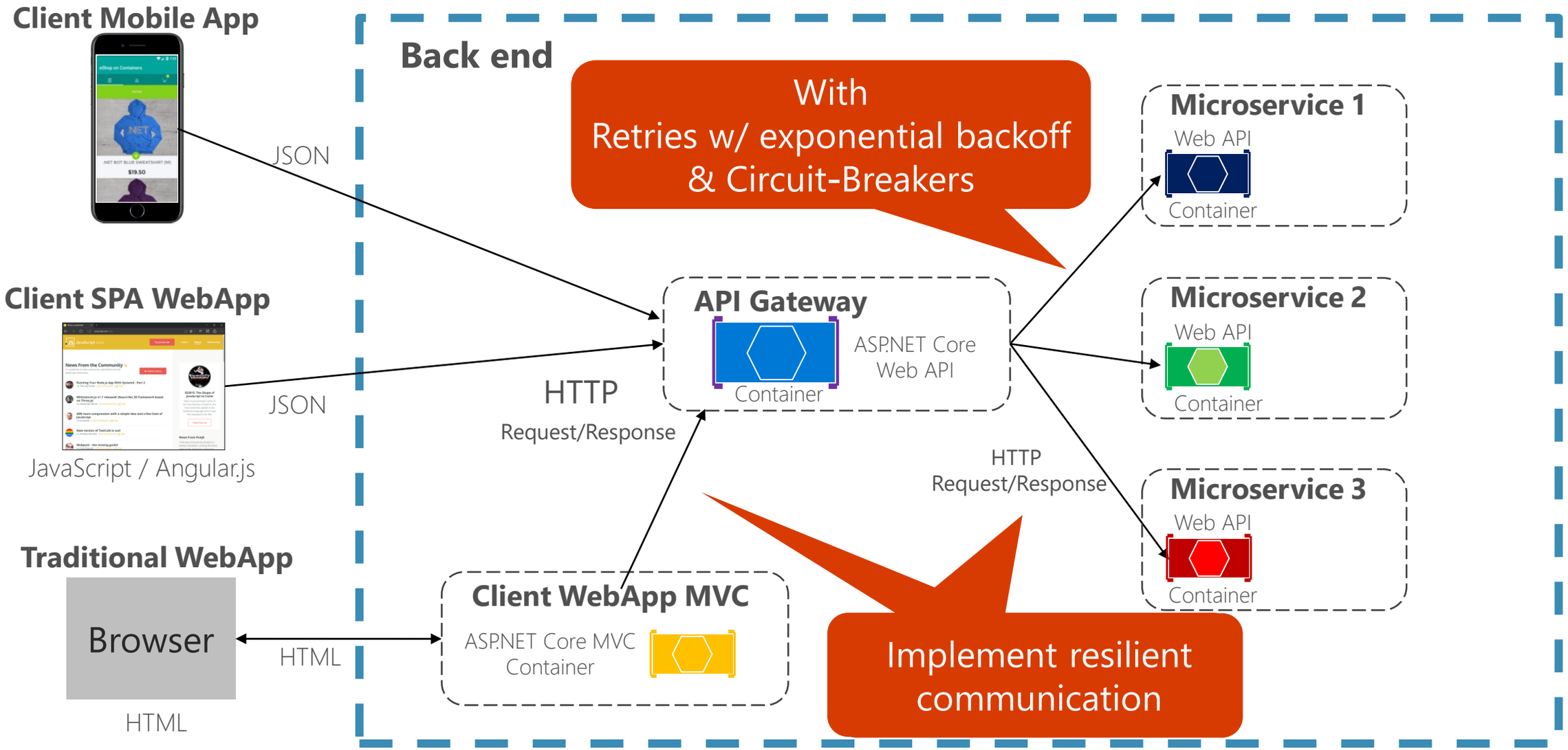
API Gateway with **Azure API Management** Architecture



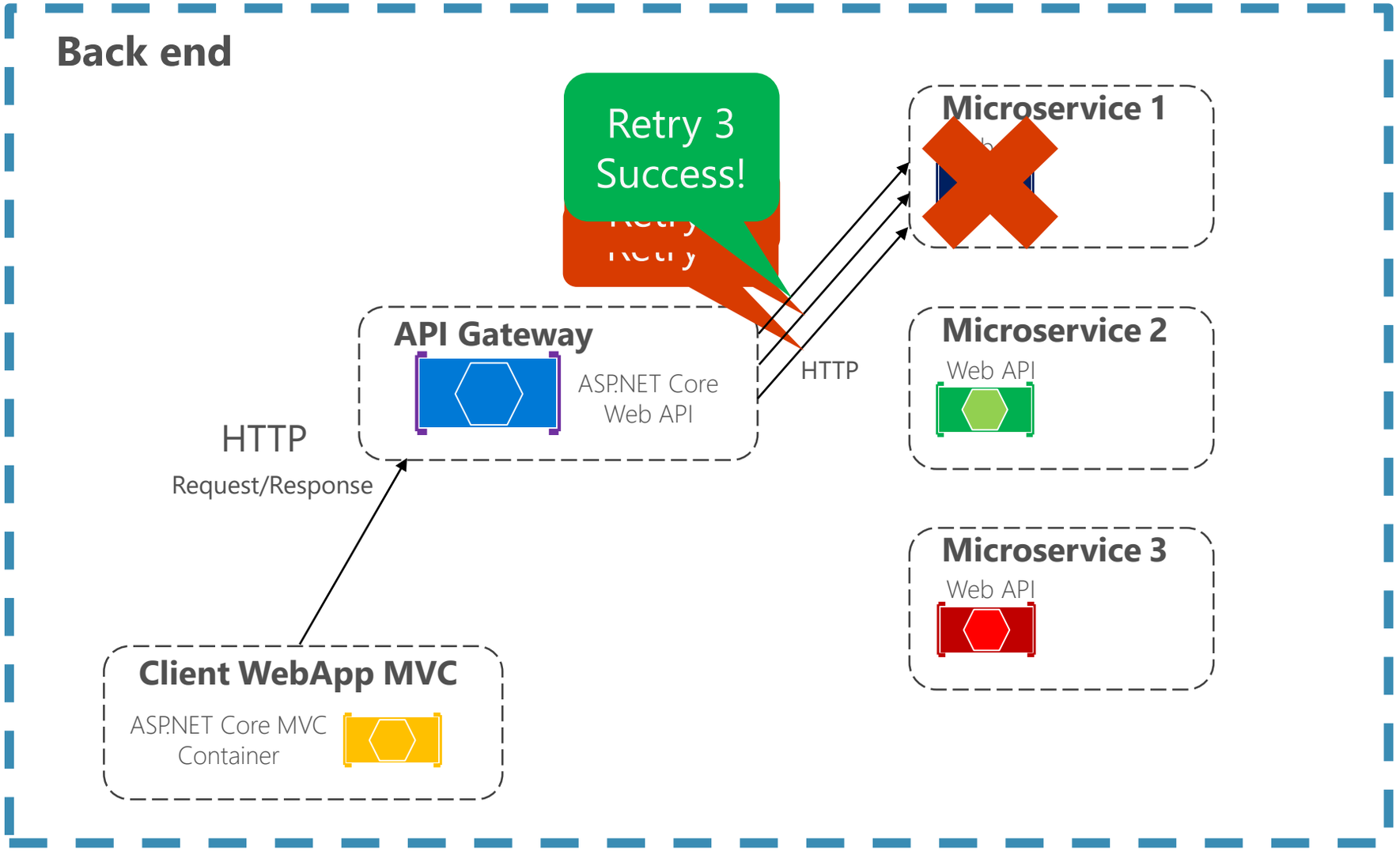
Communication

- Try to minimize internal communication
- Synchronous – HTTP/HTTPS
- Asynchronous – AMQP (Azure Service Bus, RabbitMQ; NServiceBus, MassTransit, Brighter)
- Design for failure

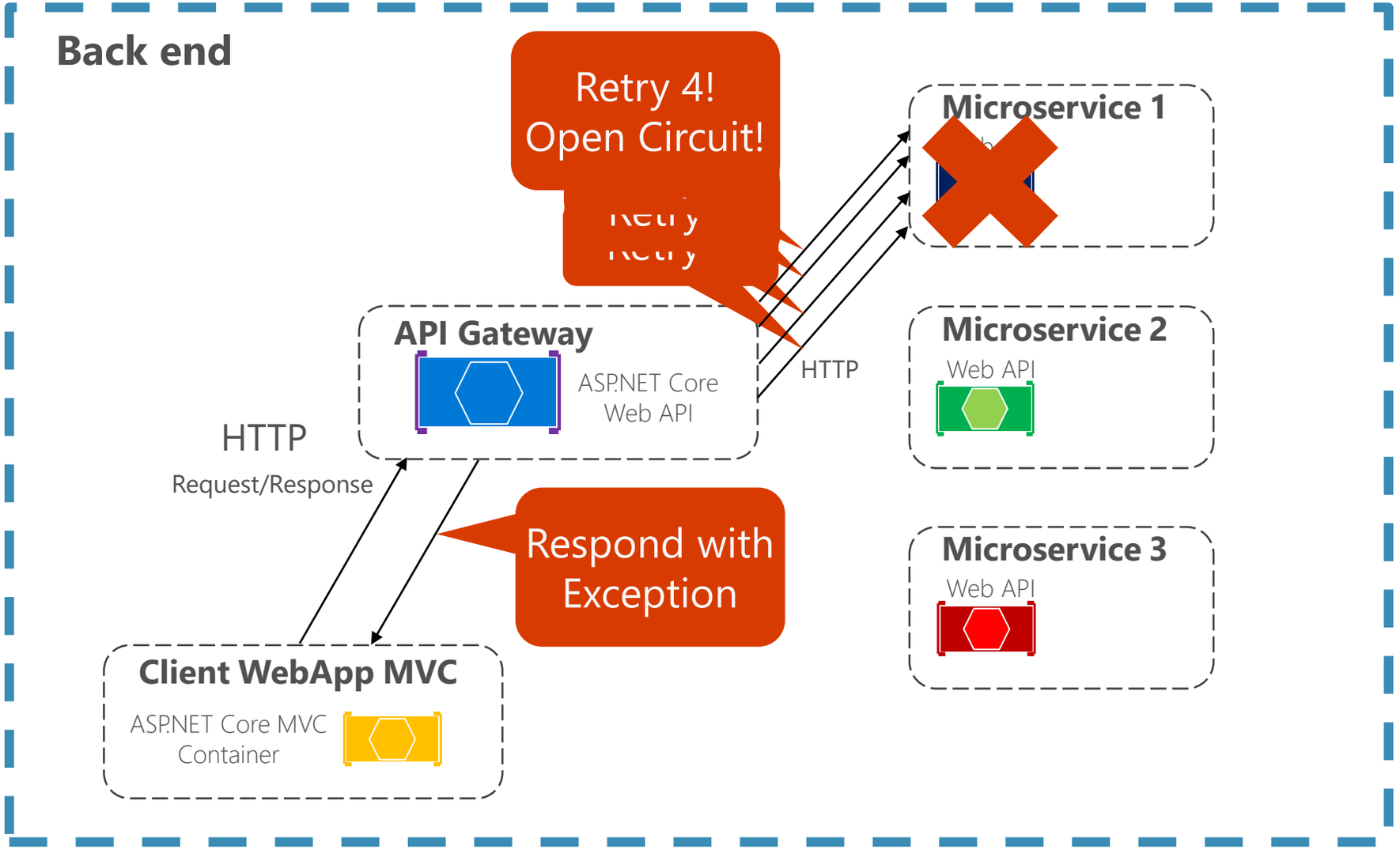
Building **resilient** cloud applications



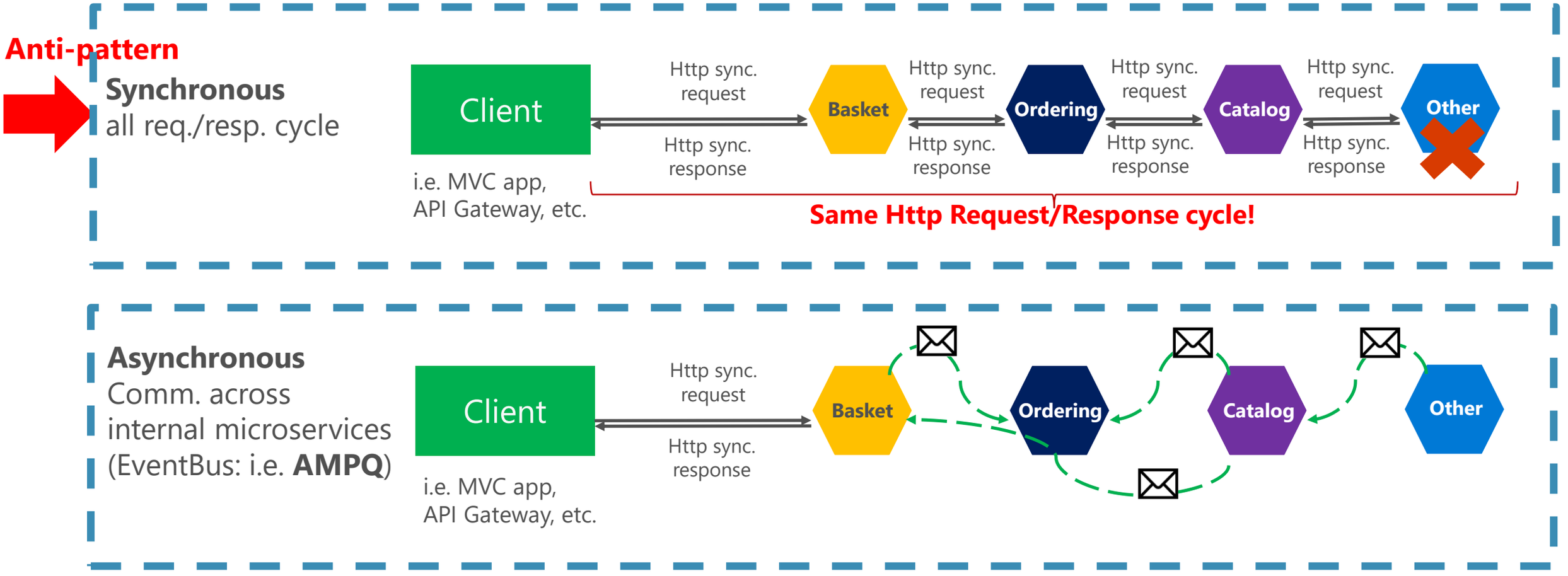
Retries with Exponential Backoff



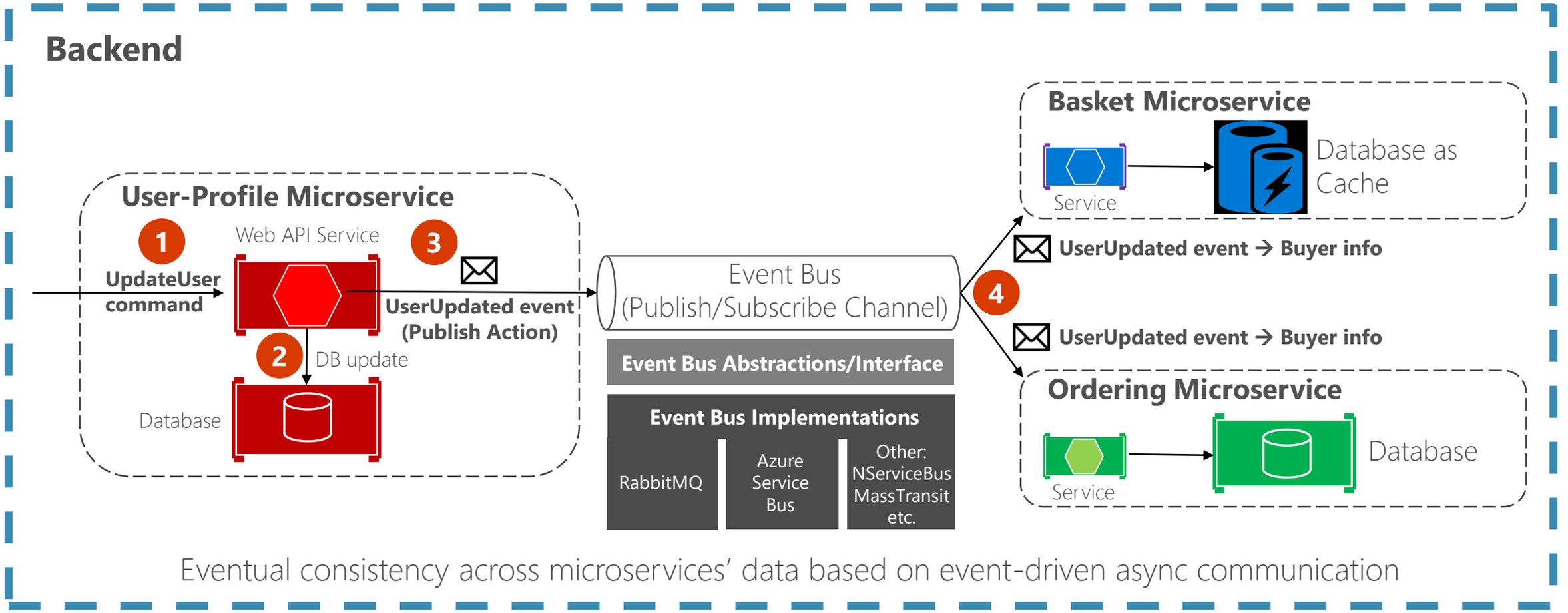
Retries with Exponential Backoff
+ Circuit Breaker



Synchronous vs. Async communication across Microservices

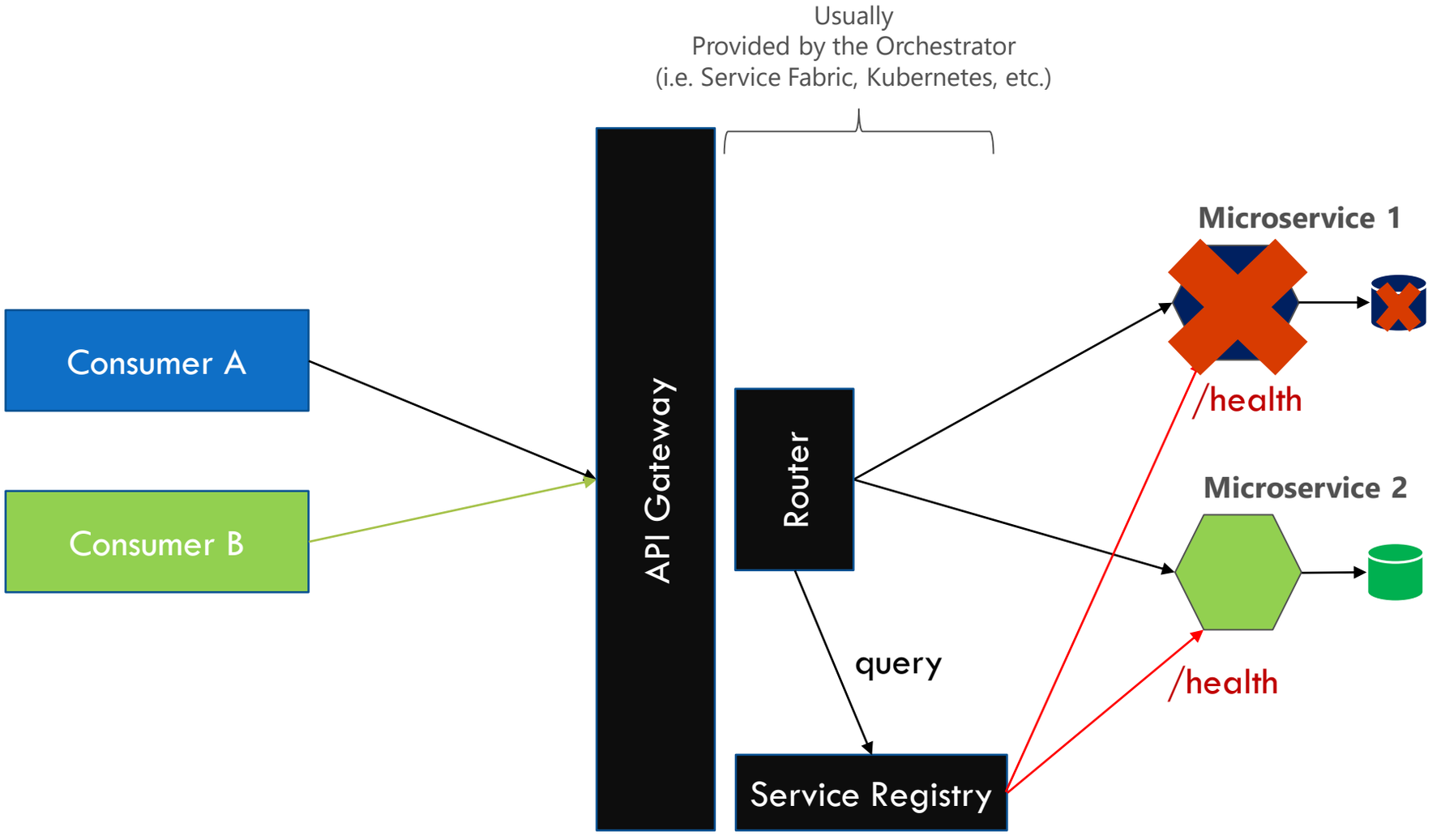


Asynchronous Event-Driven communication with an Event Bus



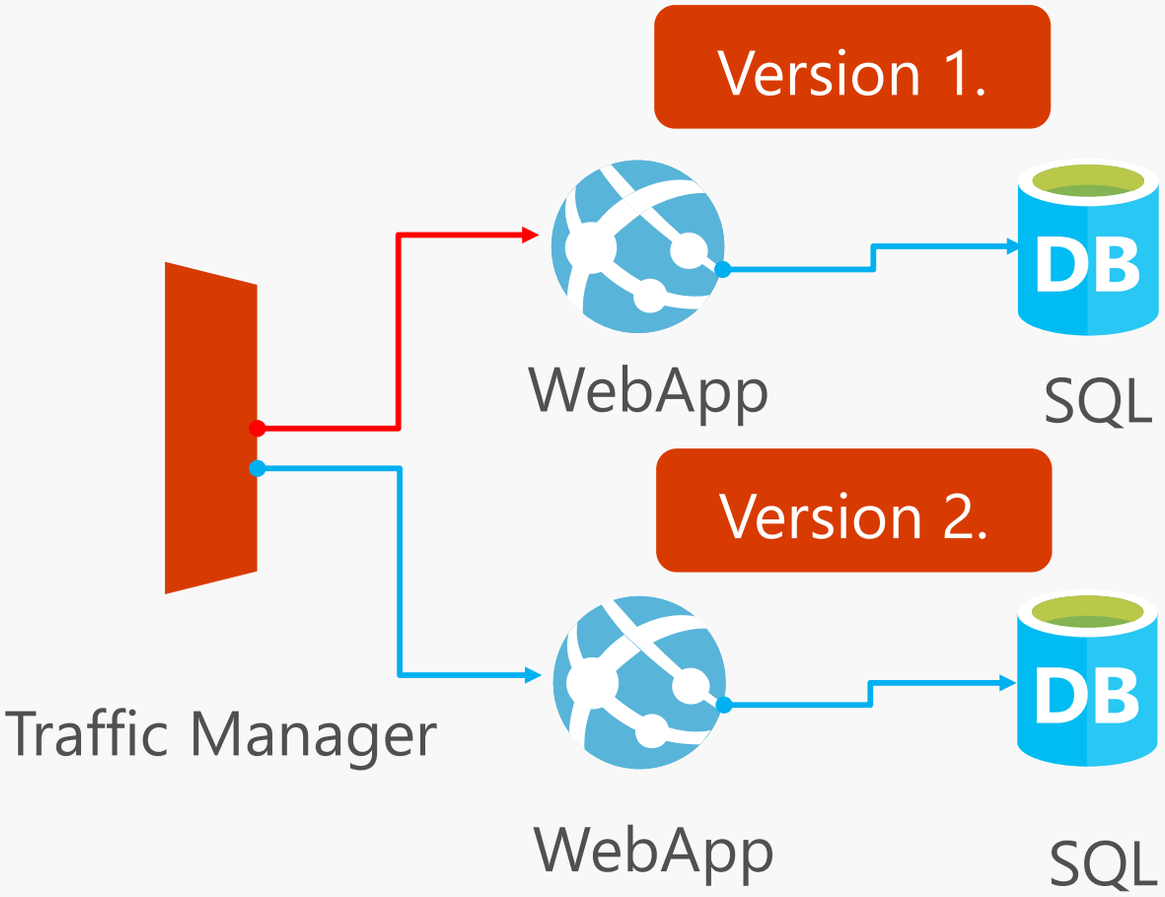
Indempotent subscribers, eventual consistency

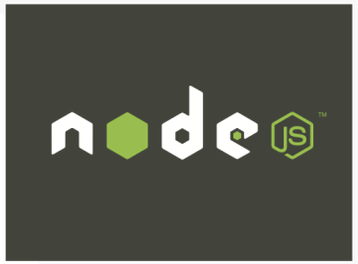
Health Checks API



Blue Green / Canary

It is easy for WebApp, but How to do it for Microservices

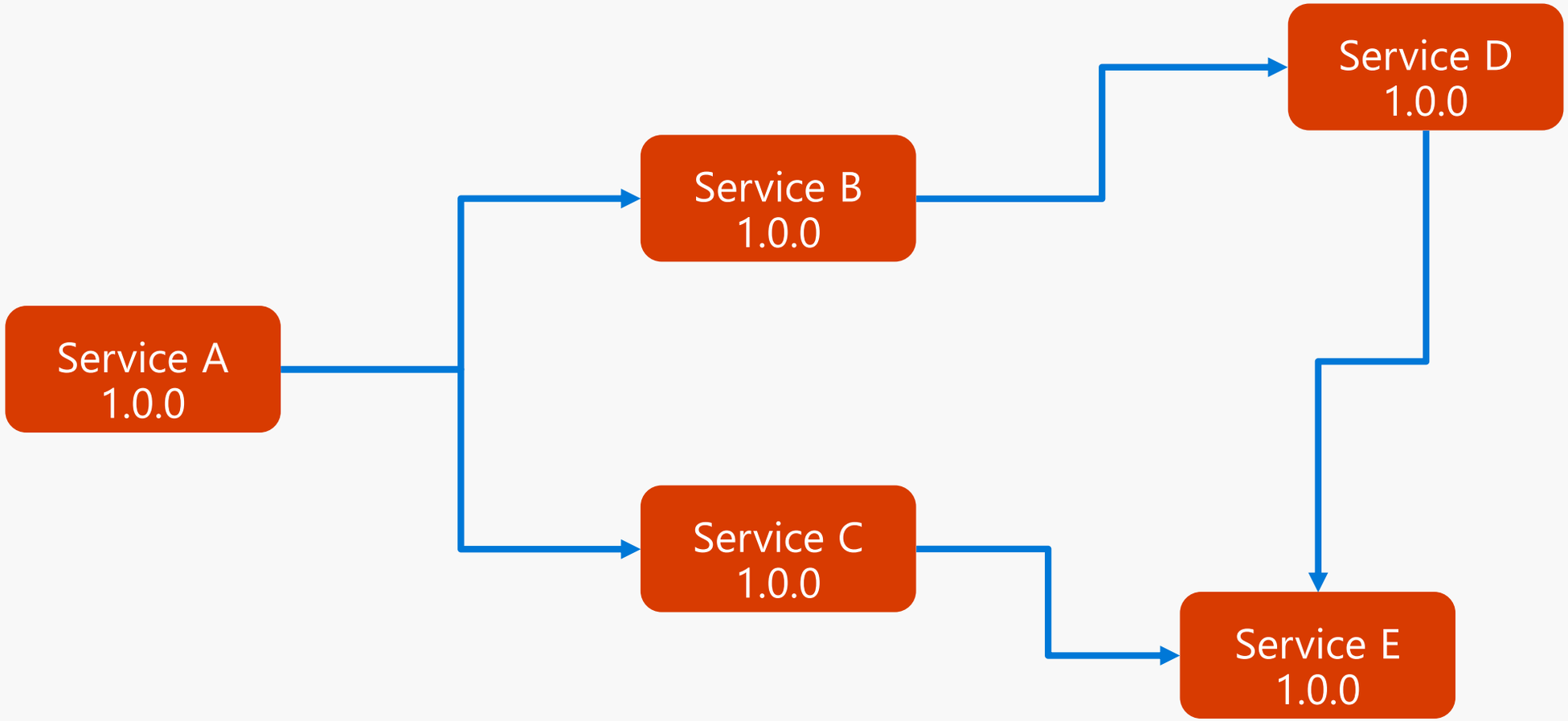




ASP.NET

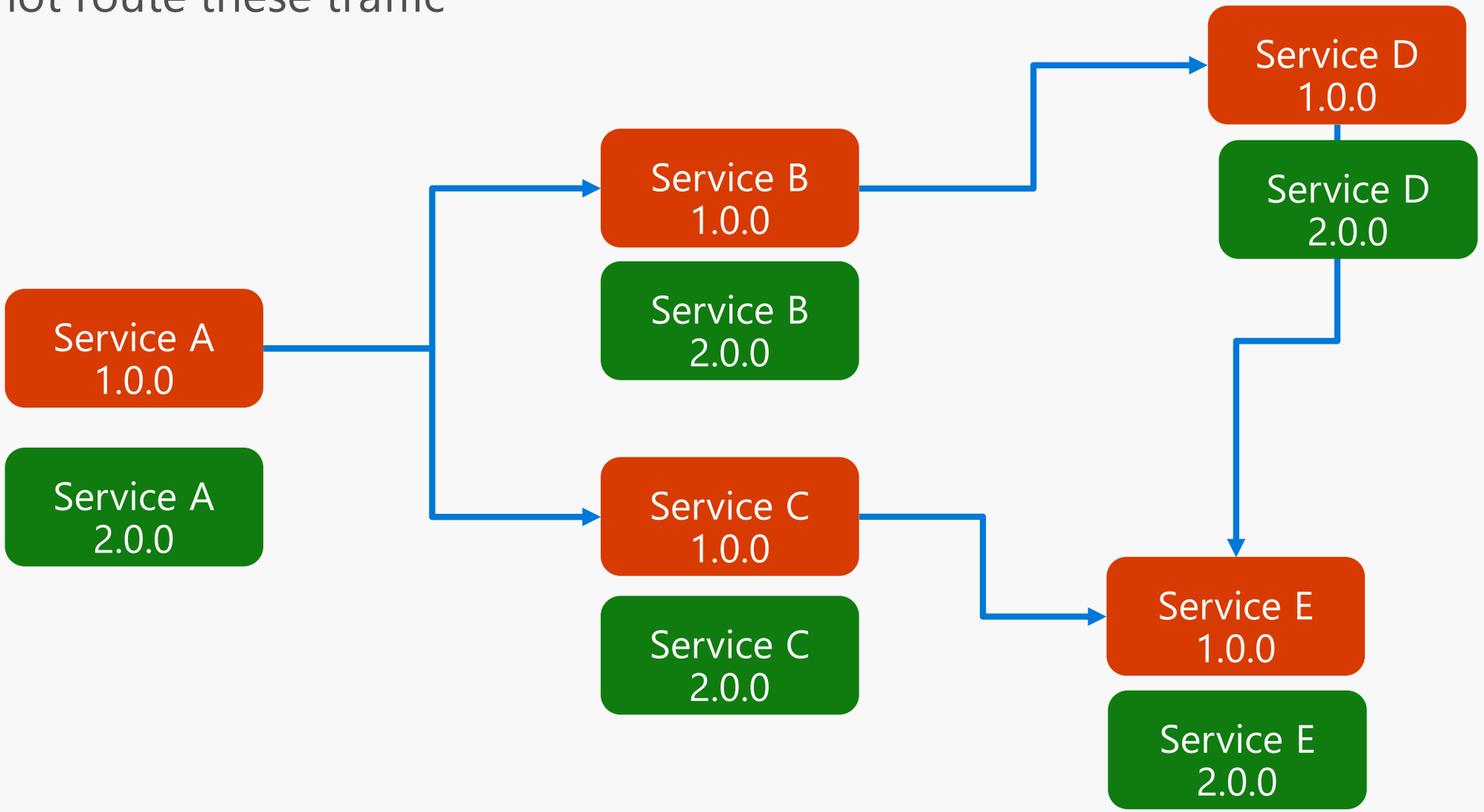
Web Apps

Blue Green Deployment with Microservices



Blue Green Deployment with Microservices

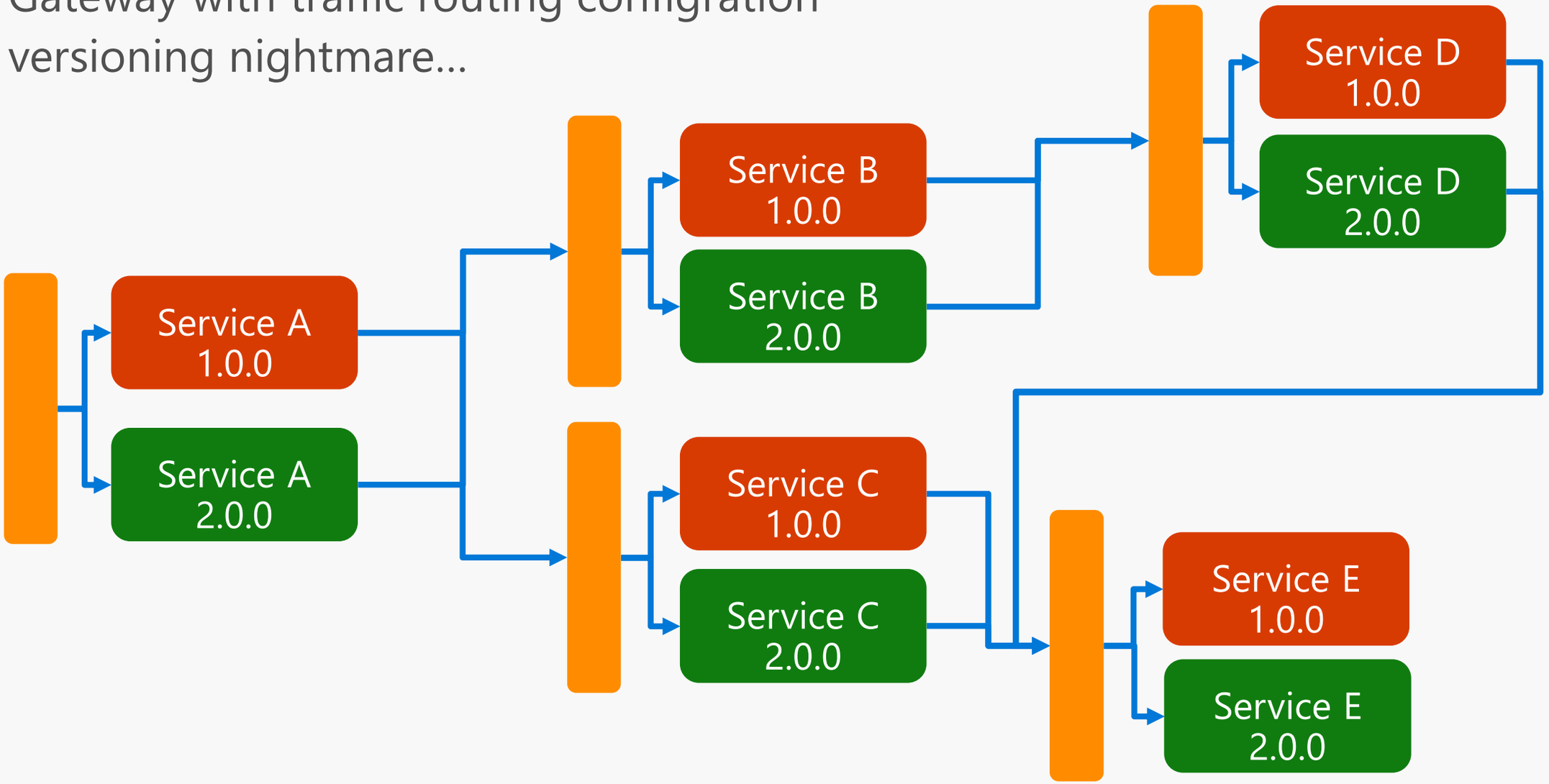
You can not route these traffic



Blue Green Deployment with Microservices

API Gateway with traffic routing configration

API versioning nightmare...



Summary

- Microservices allow to evolve, deploy and scale parts of the application independently
- Microservices offer great benefits but also new challenges
- Microservices are not suitable for all apps, but for large, scalable and long-term evolving applications with typically multiple autonomous development teams

Resources

Guide/eBook

eBook .PDF:

http://aka.ms/MicroservicesEbook

Online pages:

https://aka.ms/microservices-guide-online-msft-docs

Reference application

aka.ms/MicroservicesArchitecture