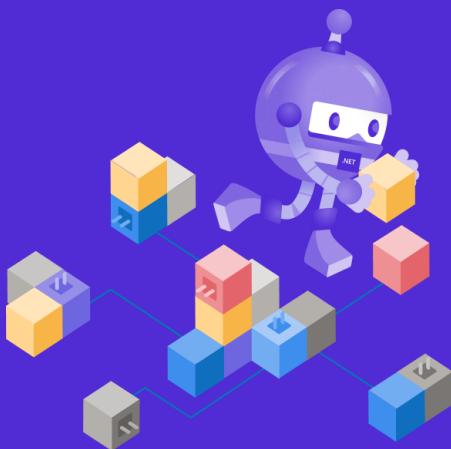


Microservices Architecture and Implementation on .Net

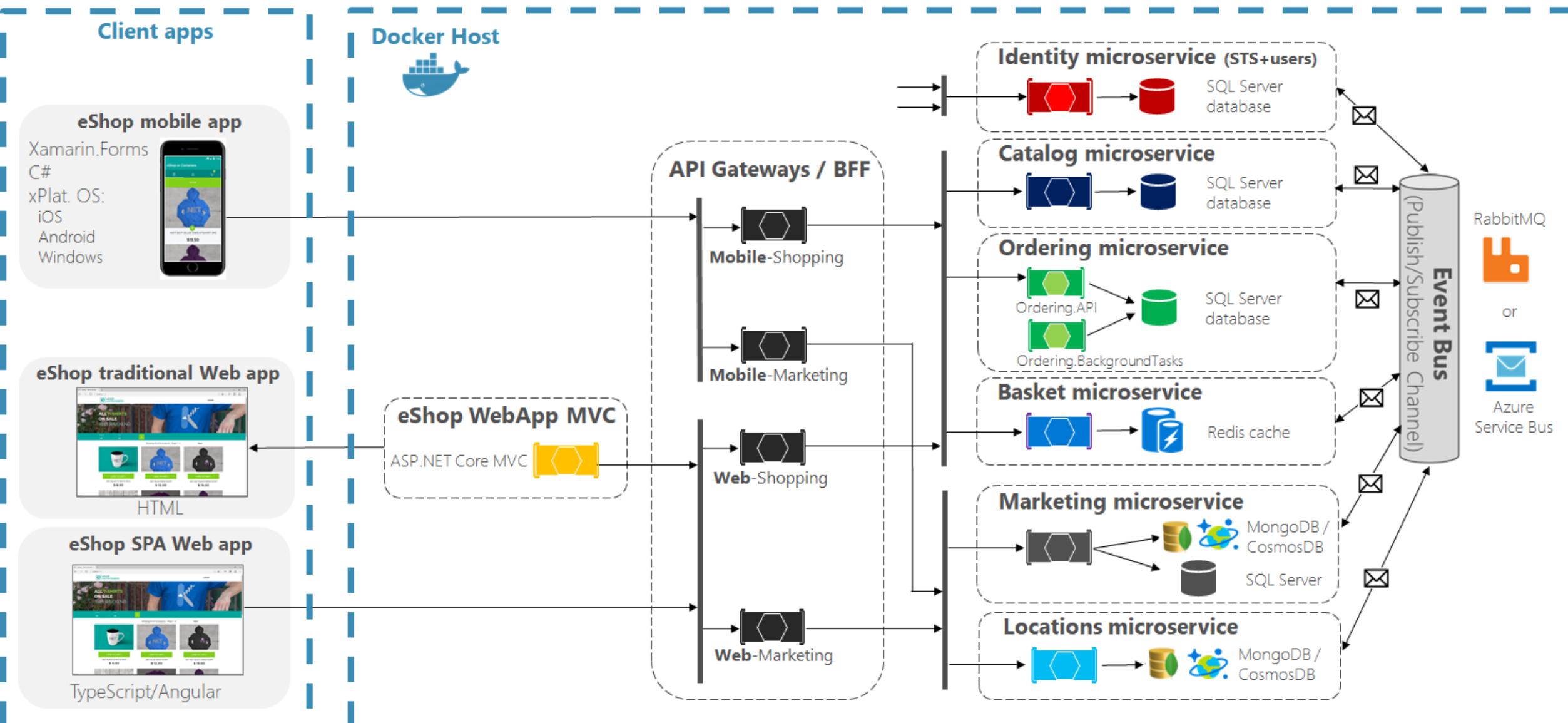


Mehmet Ozkaya
Software Architect, .NET
@ezozkme

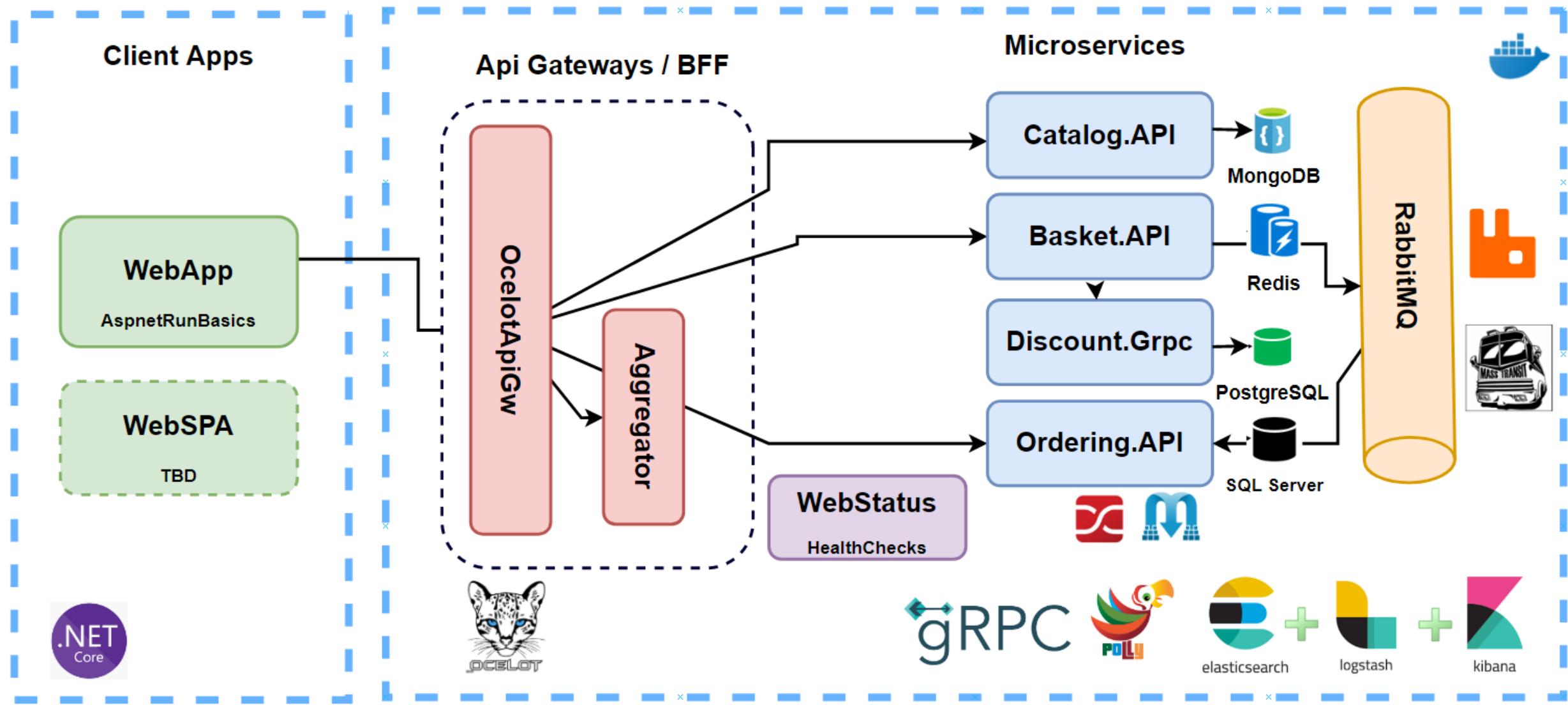
The screenshot shows a web application interface for an e-commerce platform named "AspnetRunBasics". The top navigation bar includes links for Home, Product, Cart, Order, and Contact, along with a search bar and a cart icon indicating three items. The main content area features a large banner for the "OPPO Find X" smartphone, highlighting its "Newone" model and "On Your First Choice" status. To the right of the banner is a "TOP PRODUCT" section featuring an iPhone X. Below these are four smaller product cards labeled "★ LAST PRODUCTS": iPhone X, Samsung 10, Huawei Plus, and Xiaomi Mi 9. Each card displays a thumbnail image of the phone, its name, and a brief description stating it is the company's biggest change to its flagship smartphone.

eShopOnContainers reference application

(Development environment architecture)

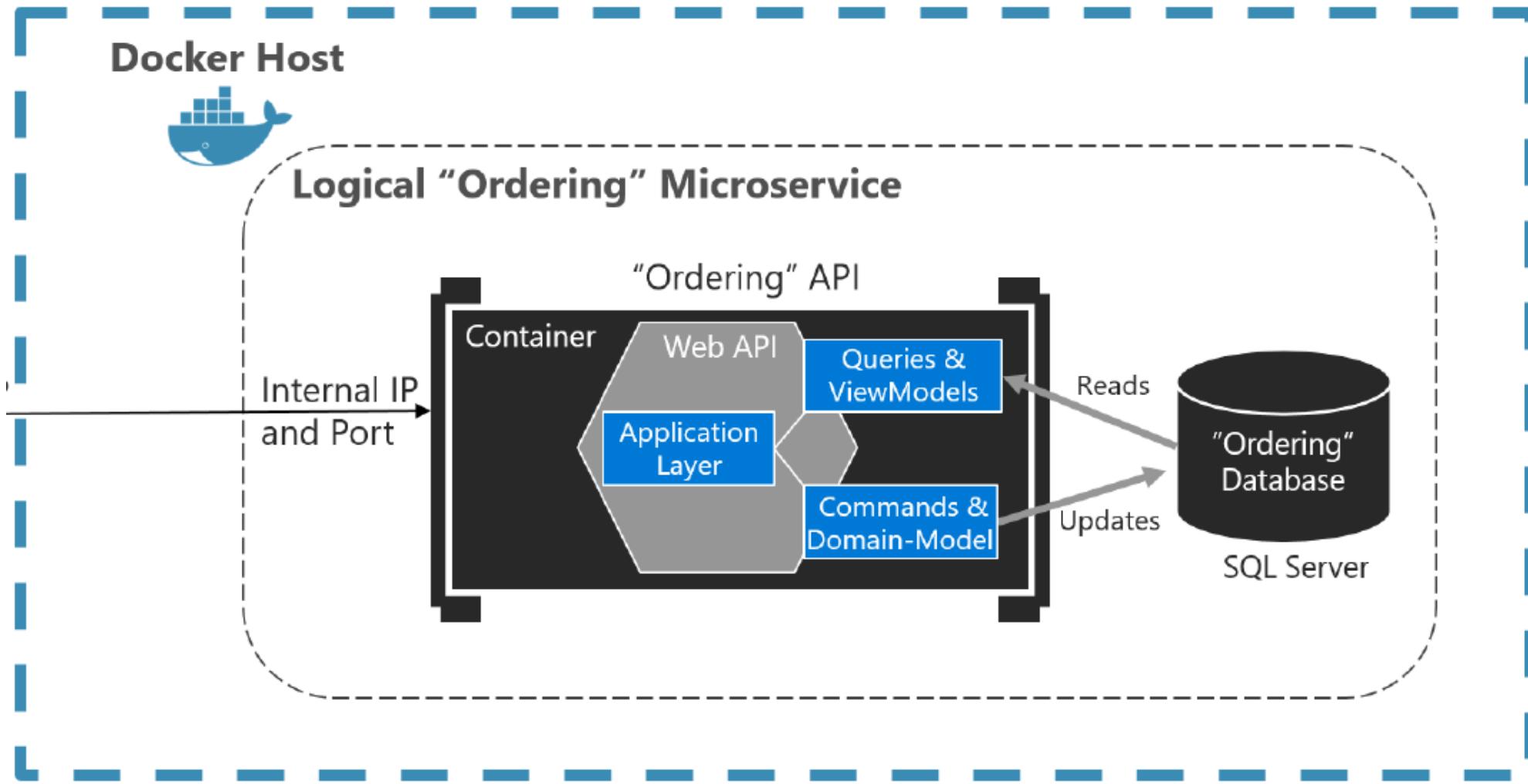


Big Picture



Simplified CQRS and DDD microservice

High level design





aspnetrun

The best path to leverage your aspnet skills. Onboarding to .Net Software Architect jobs. Download latest real world asp.net core microservices applications.

📍 Istanbul

🔗 <https://aspnetrun.azurewebsites.net>

✉ ezozkme@gmail.com

Repositories 13

Packages

People 1

Teams

Projects 1

Settings

Pinned repositories

Customize pinned repositories



learn
The best path to .Net Microservices Udemy Learning Path. .Net world evolving to the microservices and Cloud-native systems to provide rapid change, large scale, and resilience cutting-edge systems....

★ 2 ⚡ 1



run-aspnetcore-microservices

Microservices on .Net platforms which used Asp.Net Web API, Docker, RabbitMQ, MassTransit, Grpc, Ocelot API Gateway, MongoDB, Redis, PostgreSQL, SqlServer, Dapper, Entity Framework Core, CQRS and C...

● C# ★ 420 ⚡ 175



run-aspnetcore

Template

A starter kit for your next ASP.NET Core web application. Boilerplate for ASP.NET Core reference application, demonstrating a layered application architecture with applying Clean Architecture and D...

● C# ★ 228 ⚡ 57



run-aspnet-identityserver4

Secure microservices with using standalone Identity Server 4 and backing with Ocelot API Gateway. Protect our ASP.NET Web MVC and API applications with using OAuth 2 and OpenID Connect in IdentityS...

● C# ★ 39 ⚡ 18



run-aspnet-grpc

Using gRPC in Microservices for Building a high-performance Interservice Communication with .Net 5. See gRPC Microservices and Step by Step Implementation on .NET Course w/ discount->

● C# ★ 34 ⚡ 10



run-devops

Deploying .Net Microservices into Kubernetes, and moving deployments to the cloud Azure Kubernetes Services (AKS) with using Azure Container Registry (ACR) and how to Automating Deployments with Az...

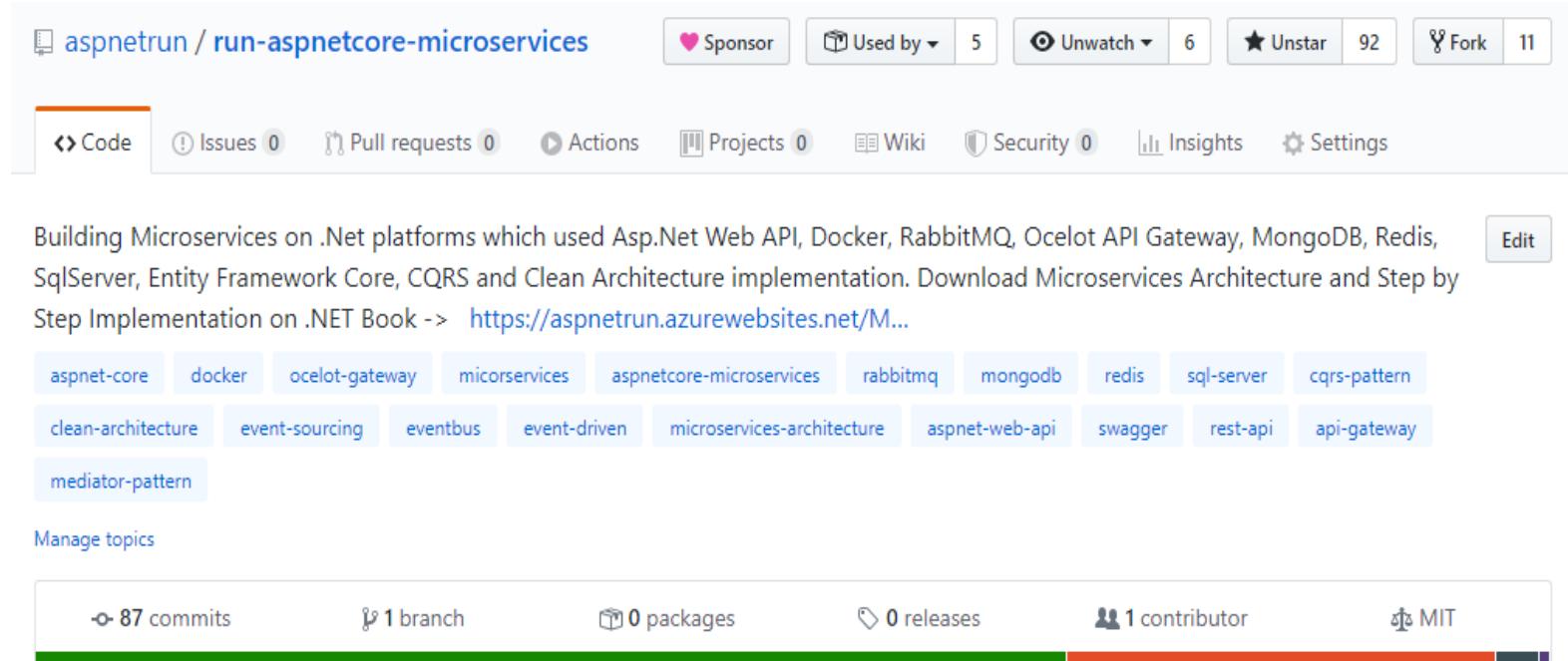
● C# ★ 4 ⚡ 8

Prerequisites

- Basics knowledge of C#
- Have knowledge of Asp.Net MVC and REST API's
- Docker Container knowledge

Source Code on Github

- Source code on Github
- Fork the repository
- Open issues
- Send Pull Requests



<https://github.com/aspnetrun/run-aspnetcore-microservices>

Tools that we will use



.NET 5.x or above SDK



Visual Studio 2019 v16.x or above



Docker Desktop



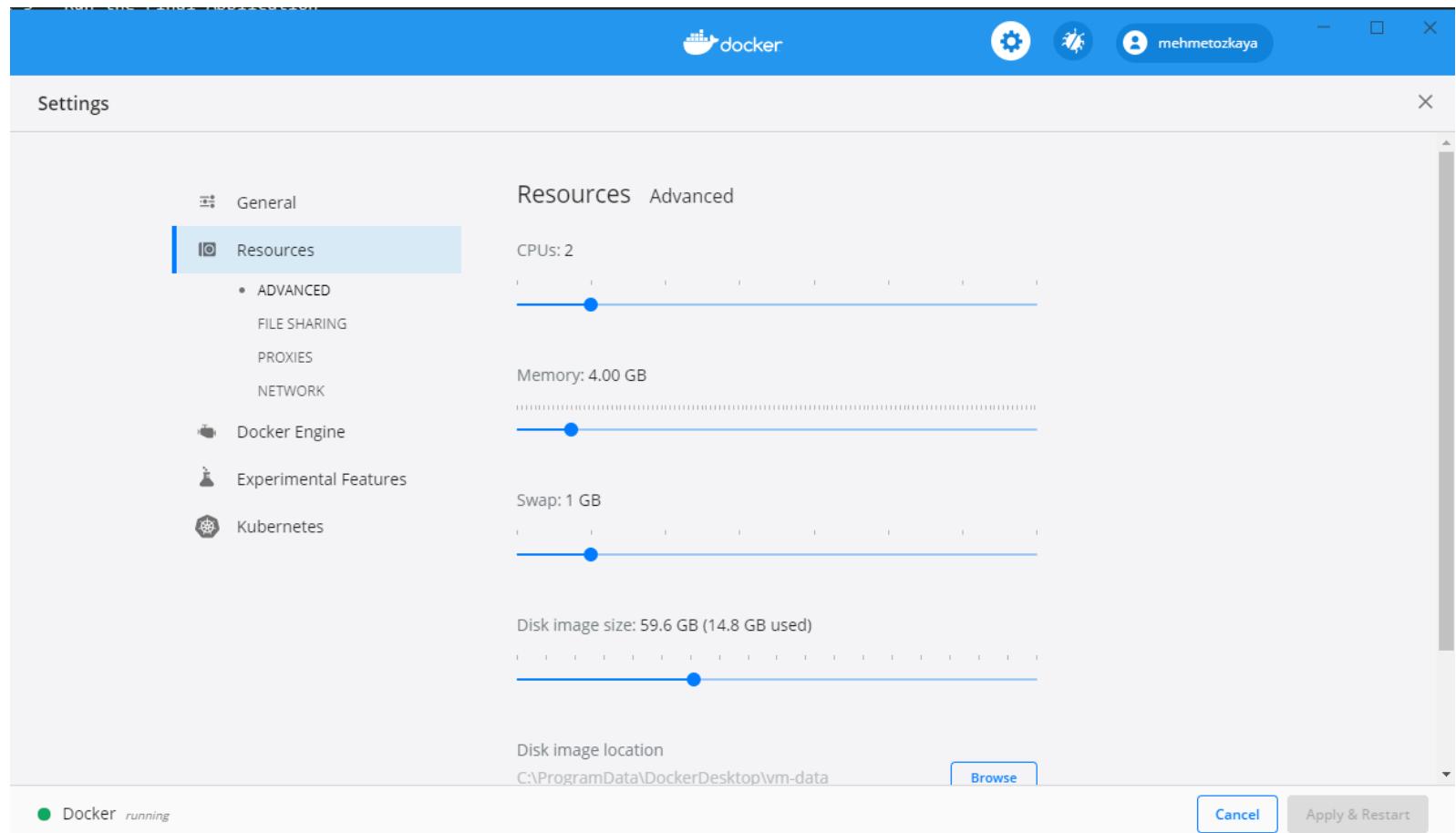
Postman



Configure Docker Desktop

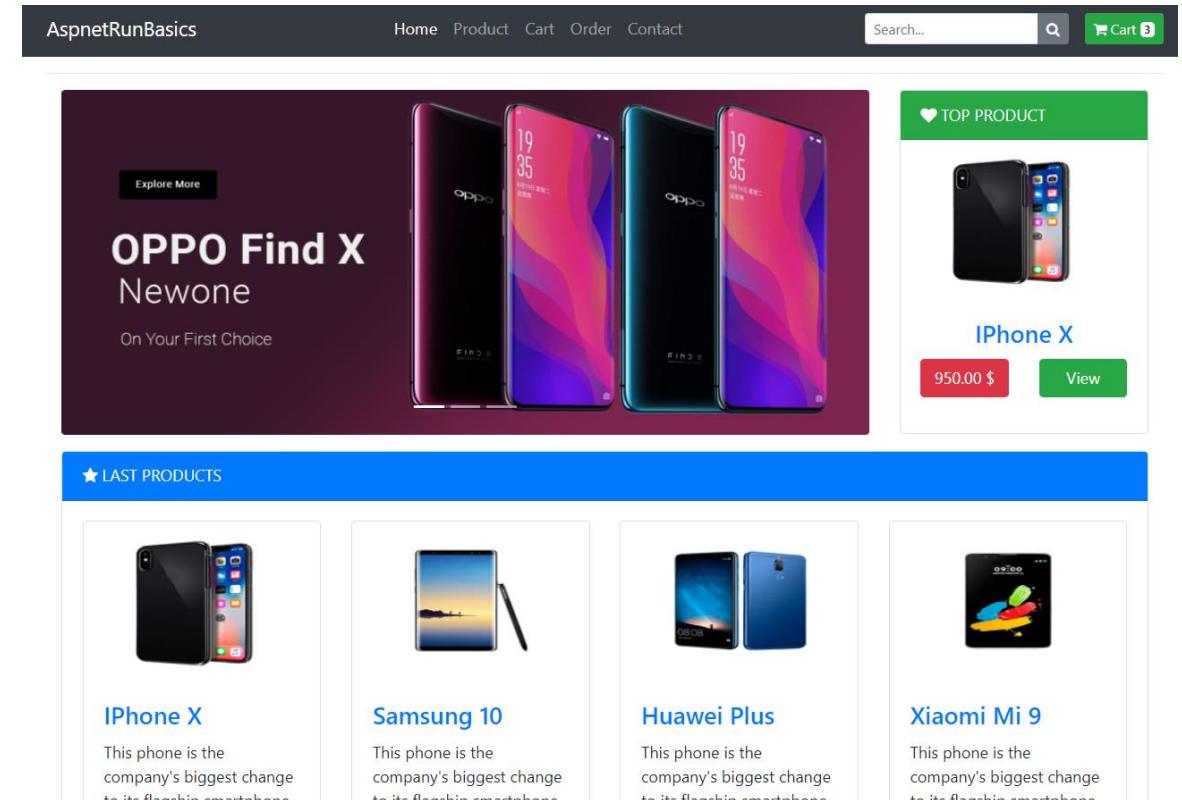
- Once Docker for Windows is installed, go to the **Settings > Advanced** option

- Memory - 4 GB
- CPU - 2



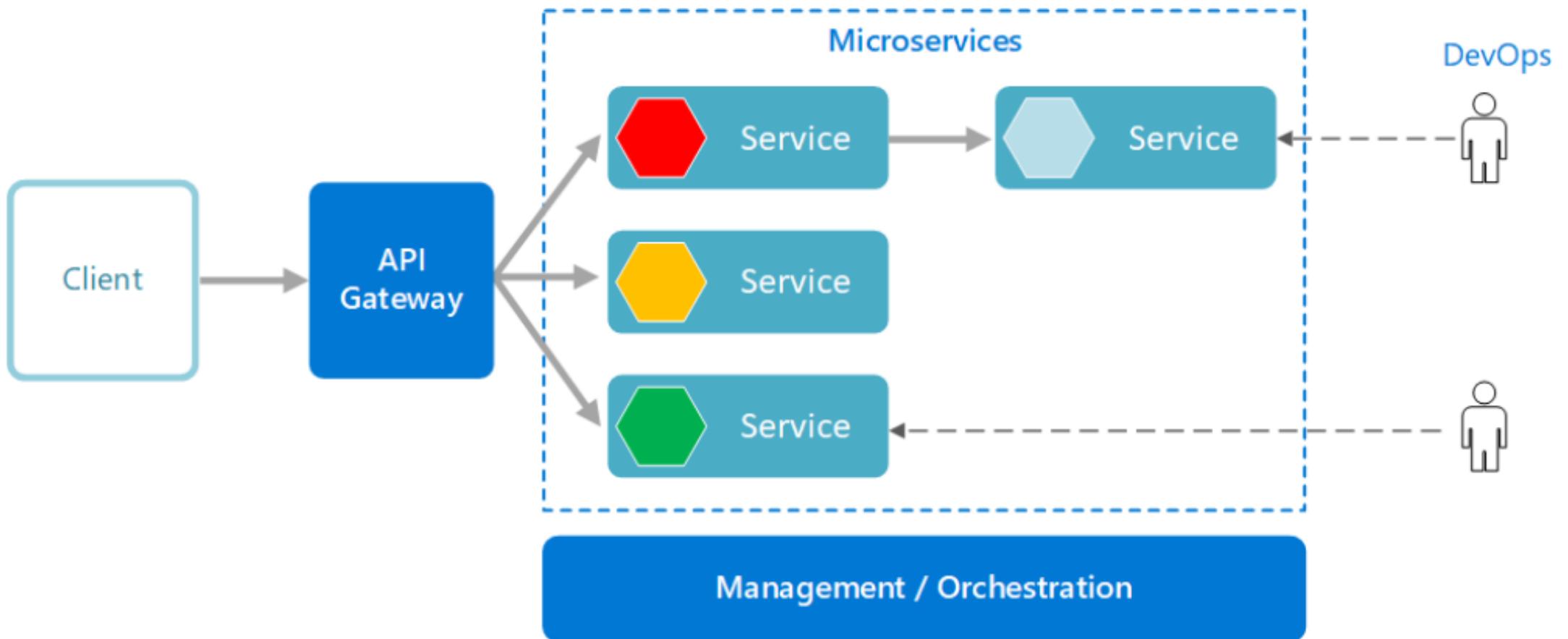
Run Final Application

- Clone Microservices Repository
- Docker-compose up
- Follow steps on github documentation
- Run the Project Section
- DEMO



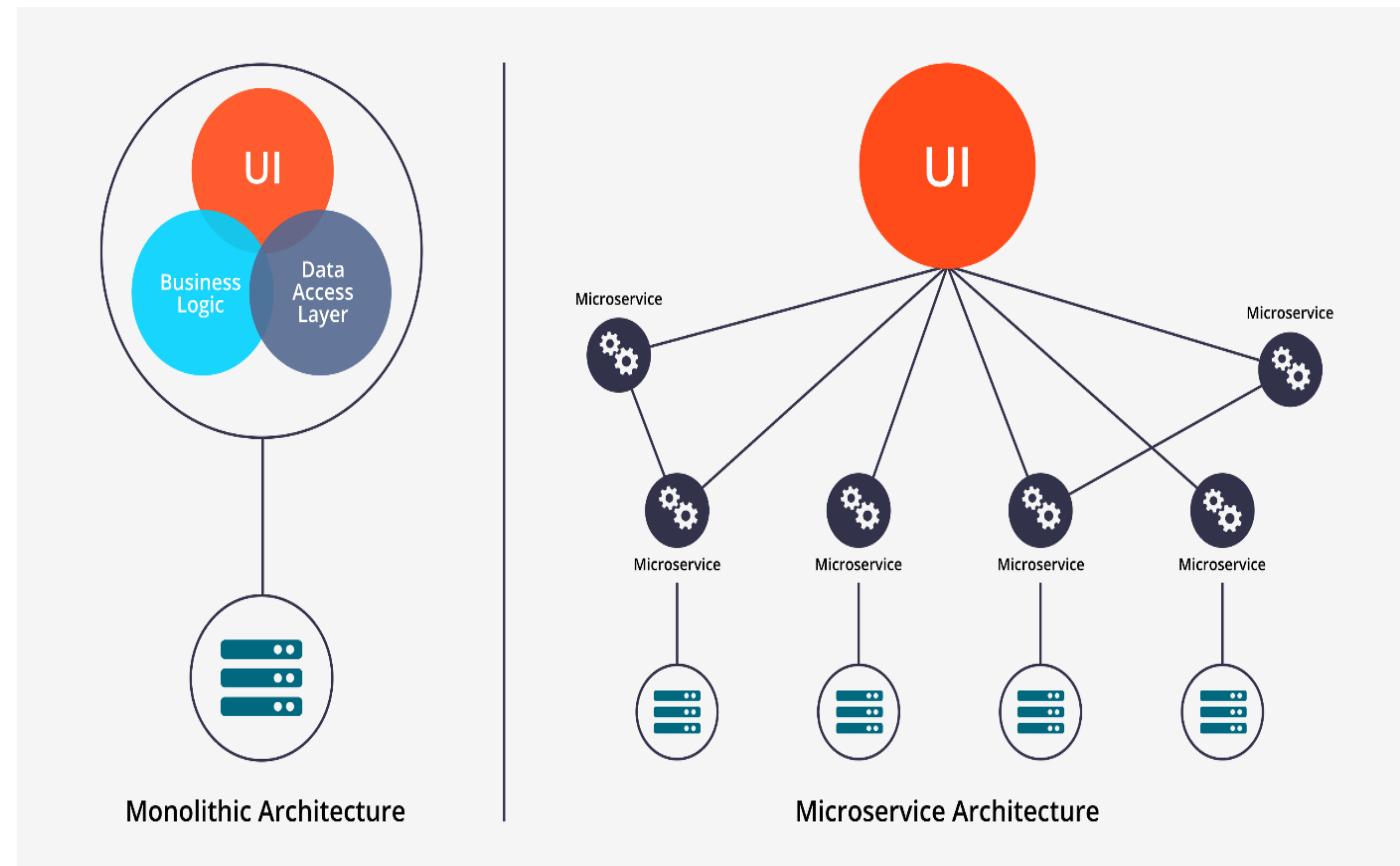
<https://github.com/aspnetrun/run-aspnetcore-microservices>

What are Microservices ?



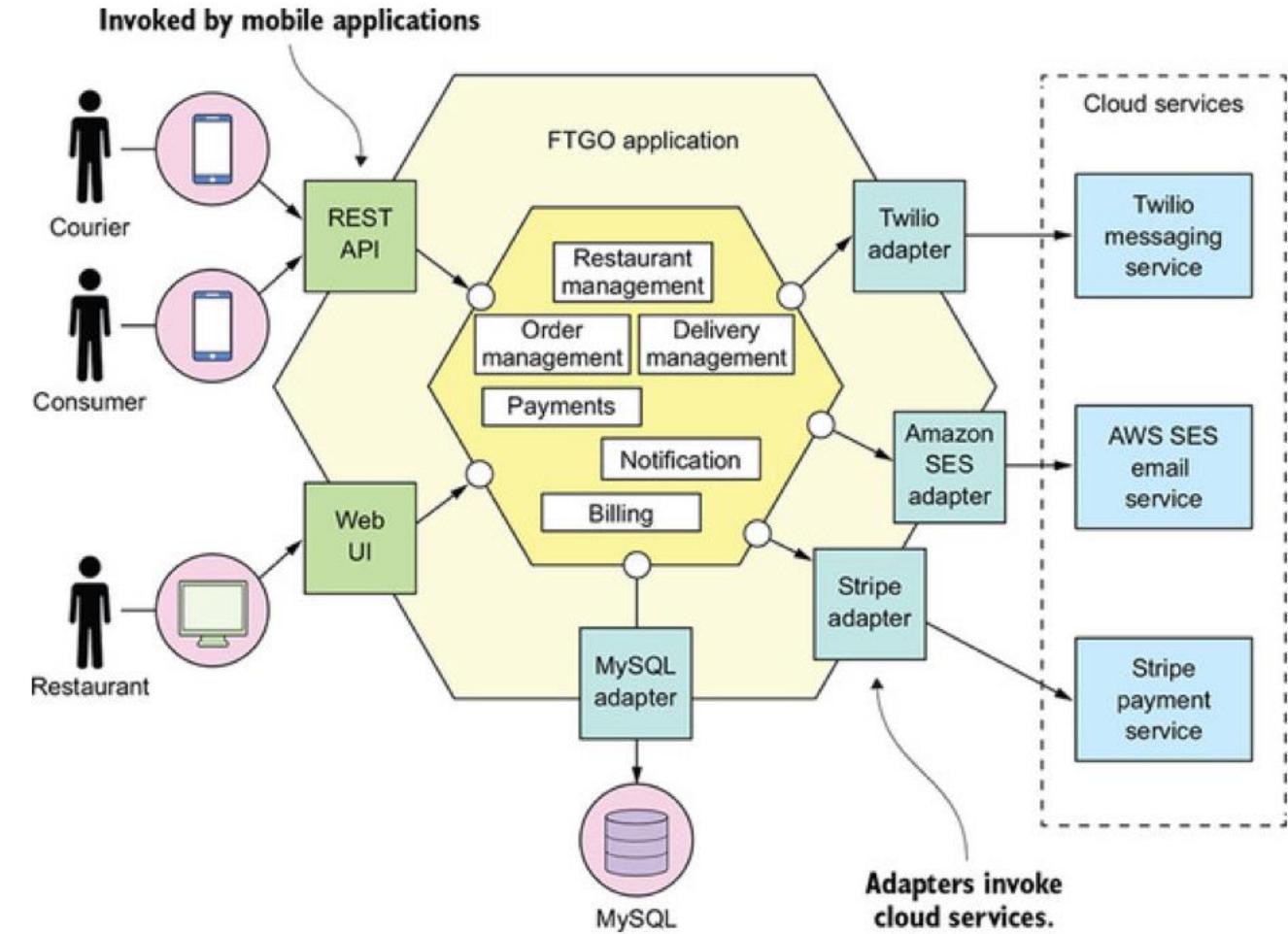
Microservices Characteristics

- Small, independent, and loosely coupled
- Separate codebase
- Deployed independently
- Persisting their own data
- Well-defined APIs
- Technology agnostic



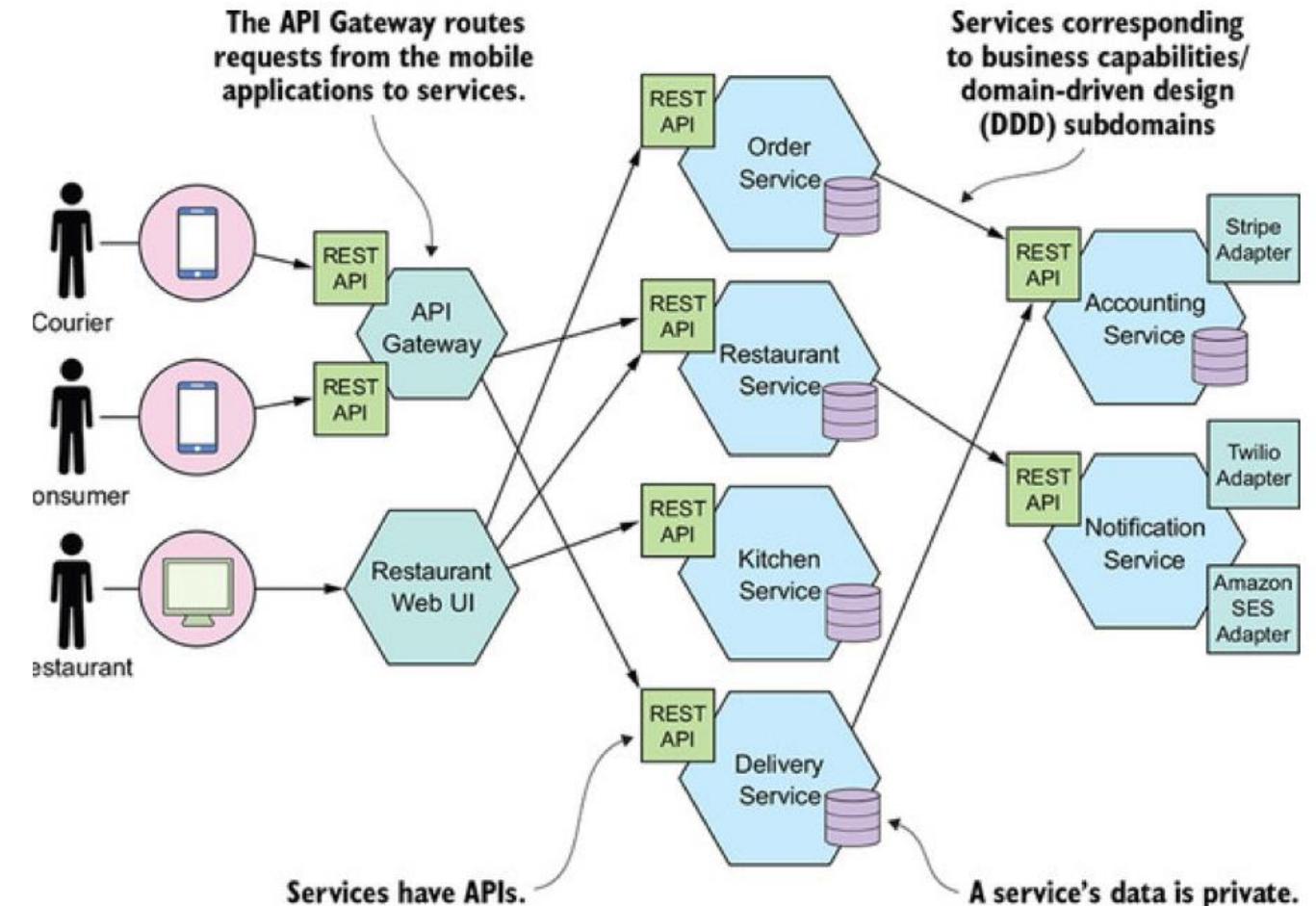
Monolithic Architecture Pros-Cons

- Single codebase
- Easy to develop, debug and deployments
- Complexity
- Hard to maintenance
- Challenge of making changes
- Inability to apply new technology

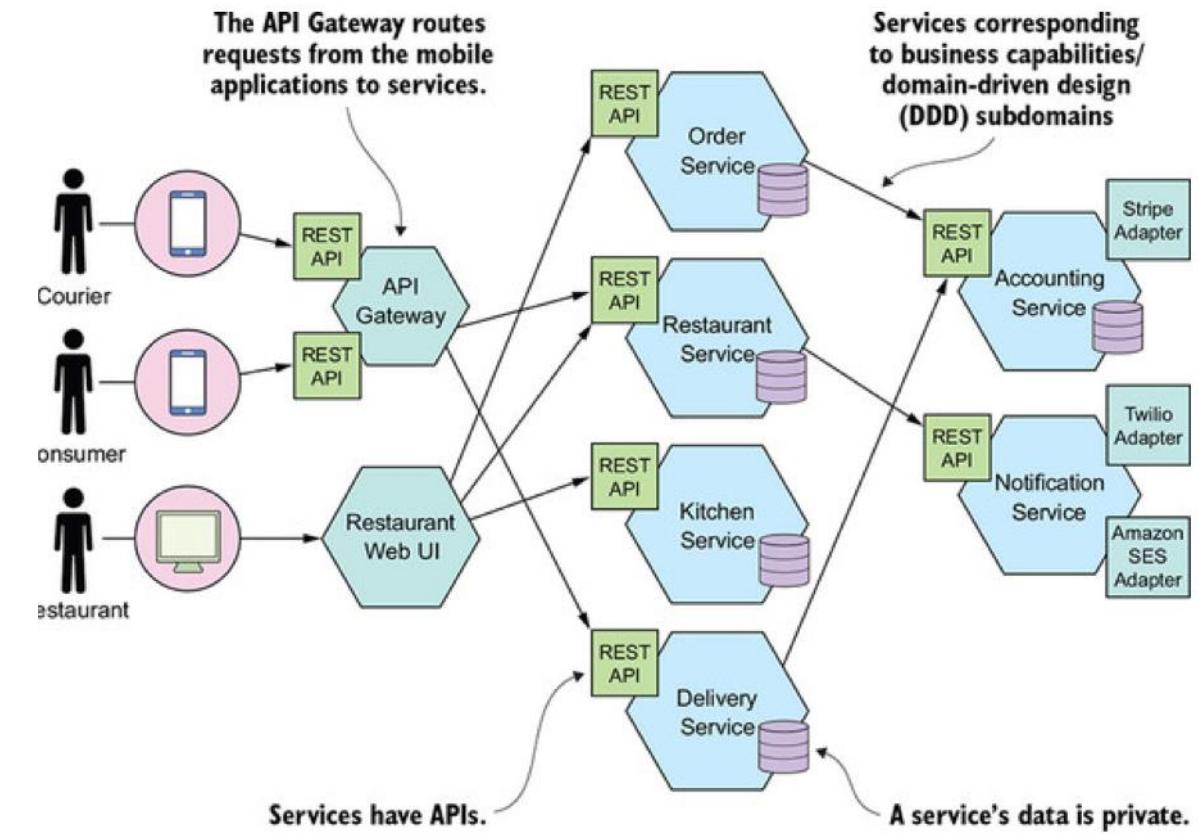
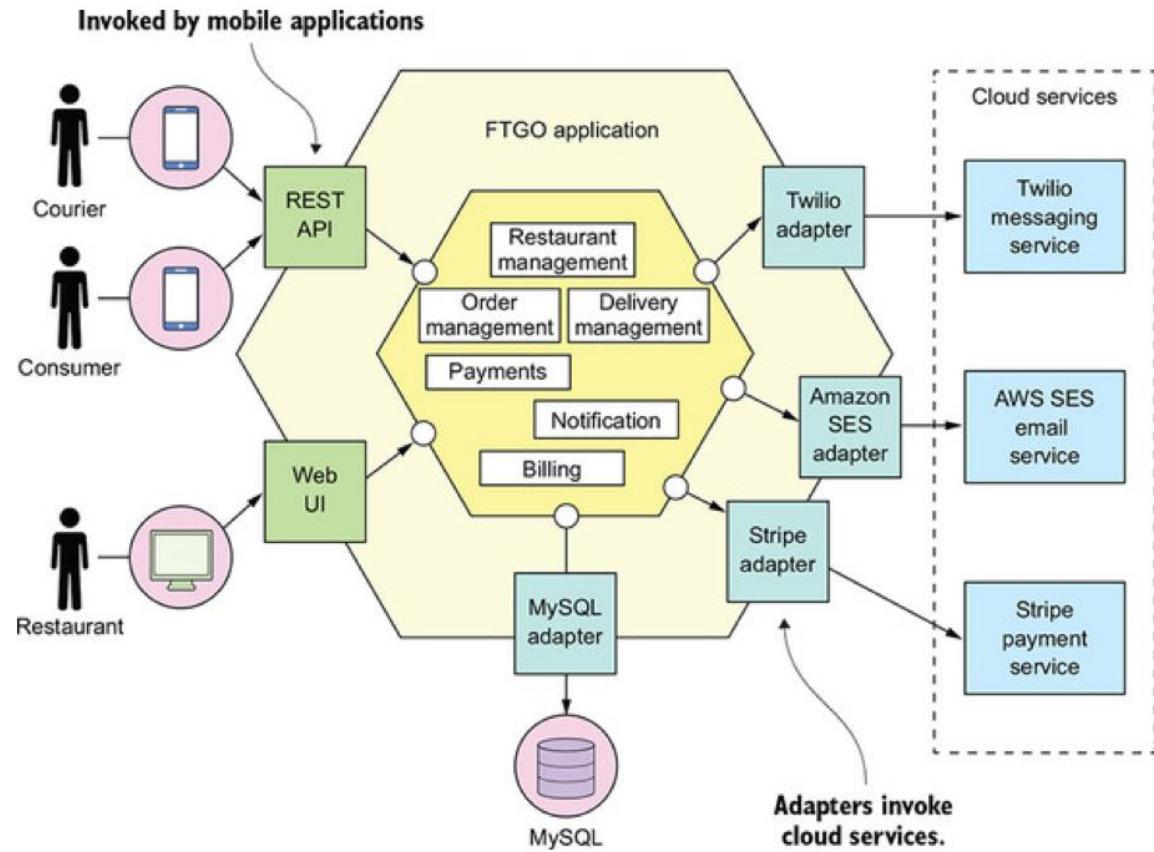


Microservices Architecture Pros-Cons

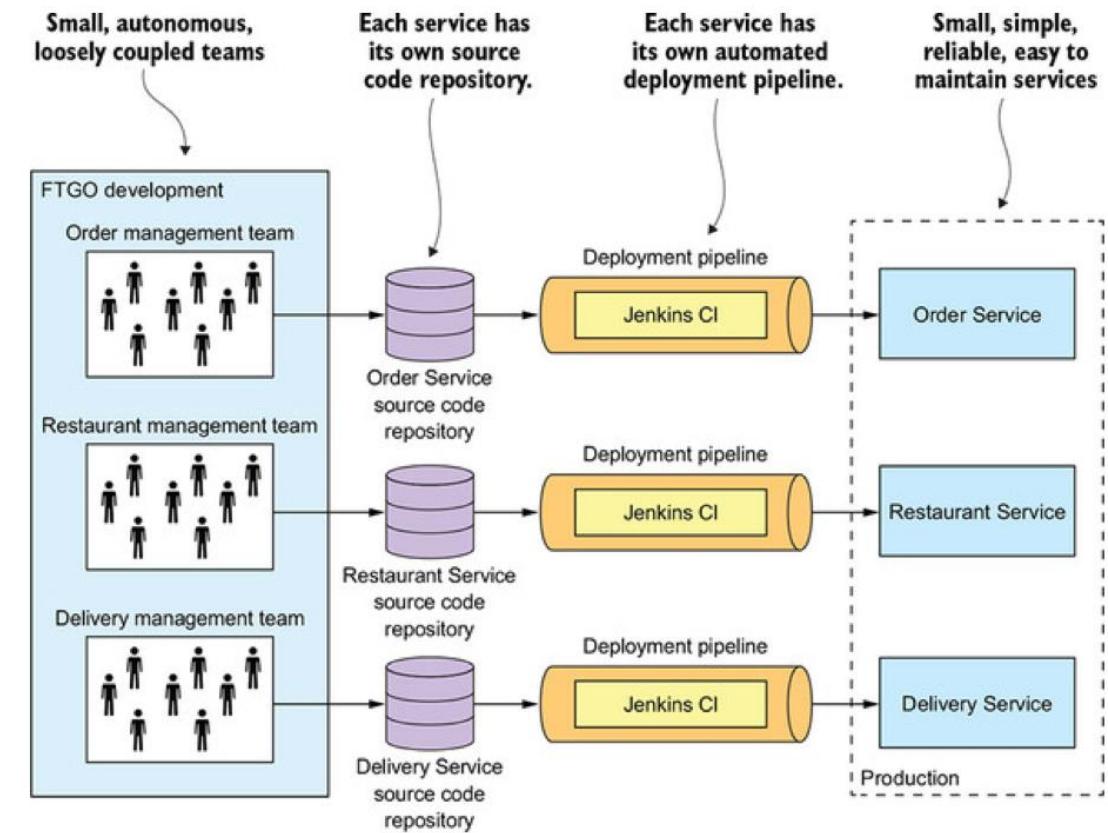
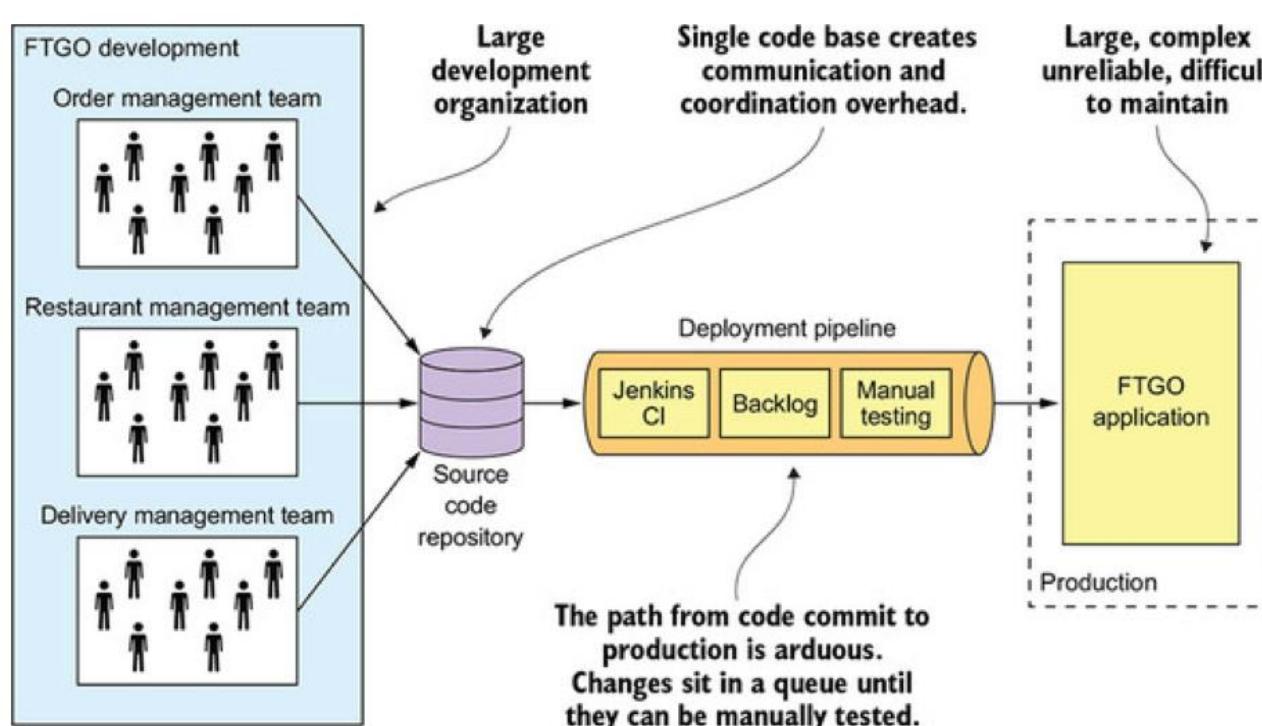
- Independent Services
- Better scalability
- Technology Diversity
- Agility
- Small, focused teams
- Challenge of management and traceability



Architecture Comparison

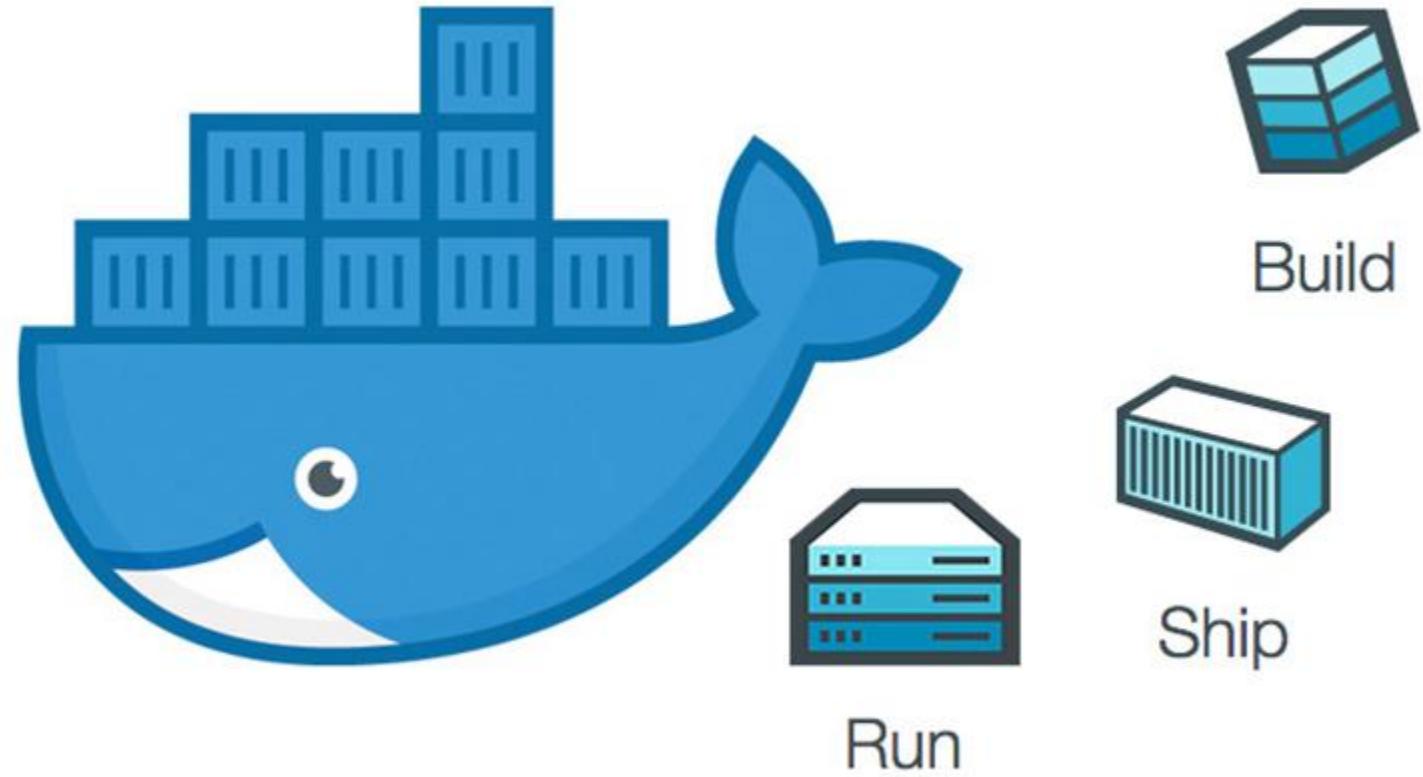


Deployment Comparison

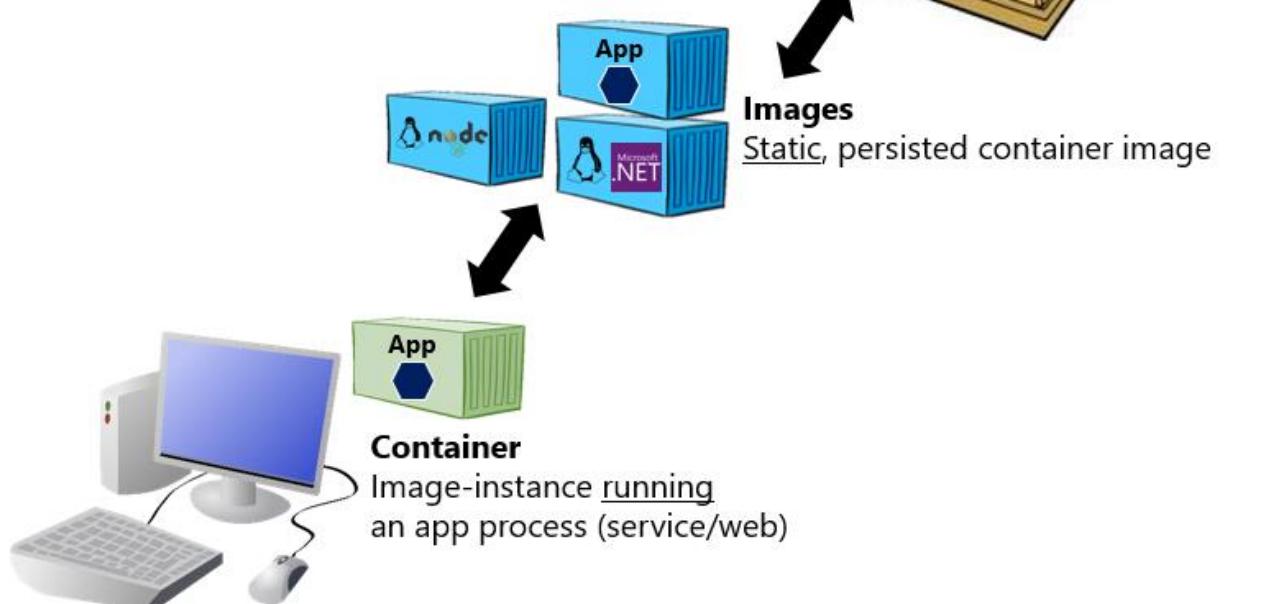
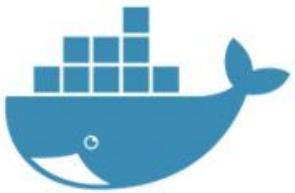


Docker and Container

- Developing, Shipping, and Running applications
- Reduce time to production
- Run anywhere
- Container packages code and dependencies



Basic taxonomy in Docker



Hosted Docker Registry

Docker Trusted Registry on-prem.

Docker Hub Registry

Docker Trusted Registry on-cloud

Azure Container Registry

AWS Container Registry

Google Container Registry

Quay Registry

Other Cloud

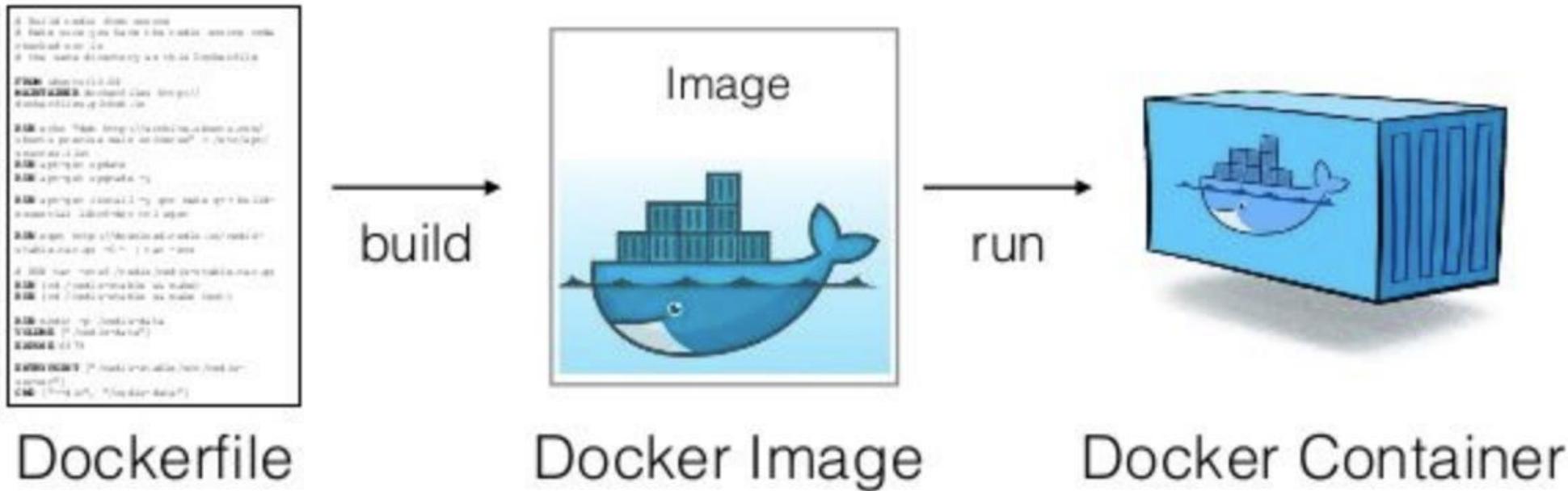
On-premises

('n' private organizations)

Public Cloud

(specific vendors)

Docker Application Containerization

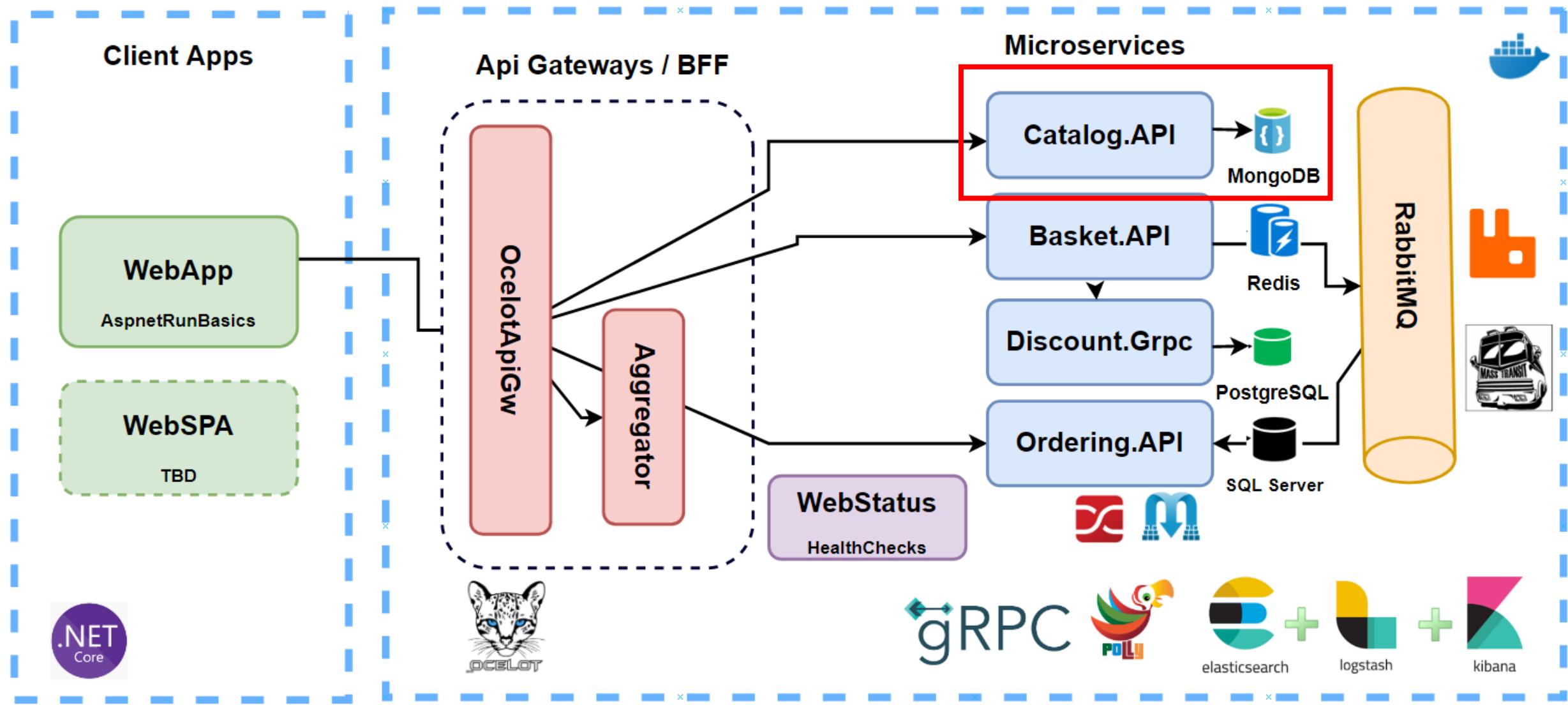


Section 2

Developing Your First Microservice

Catalog.API with MongoDB

Big Picture



MongoDb in Catalog Microservices

- Open-source NoSQL Database
- Cassandra, BigTable, Dynamo
- Json format and in collections as documents
- Document Database



Catalog API Microservices

- ASP.NET Core Web API application
- REST API principles, CRUD operations
- MongoDB database connection and containerization
- Repository Pattern Implementation
- Containerize Catalog Microservices with MongoDB using Docker Compose

The screenshot shows the Catalog API documentation generated by Swagger. At the top, it says "Catalog API v1 OAS3" and provides a link to "/swagger/v1/swagger.json". Below this, there's a section titled "Catalog" containing the following endpoints:

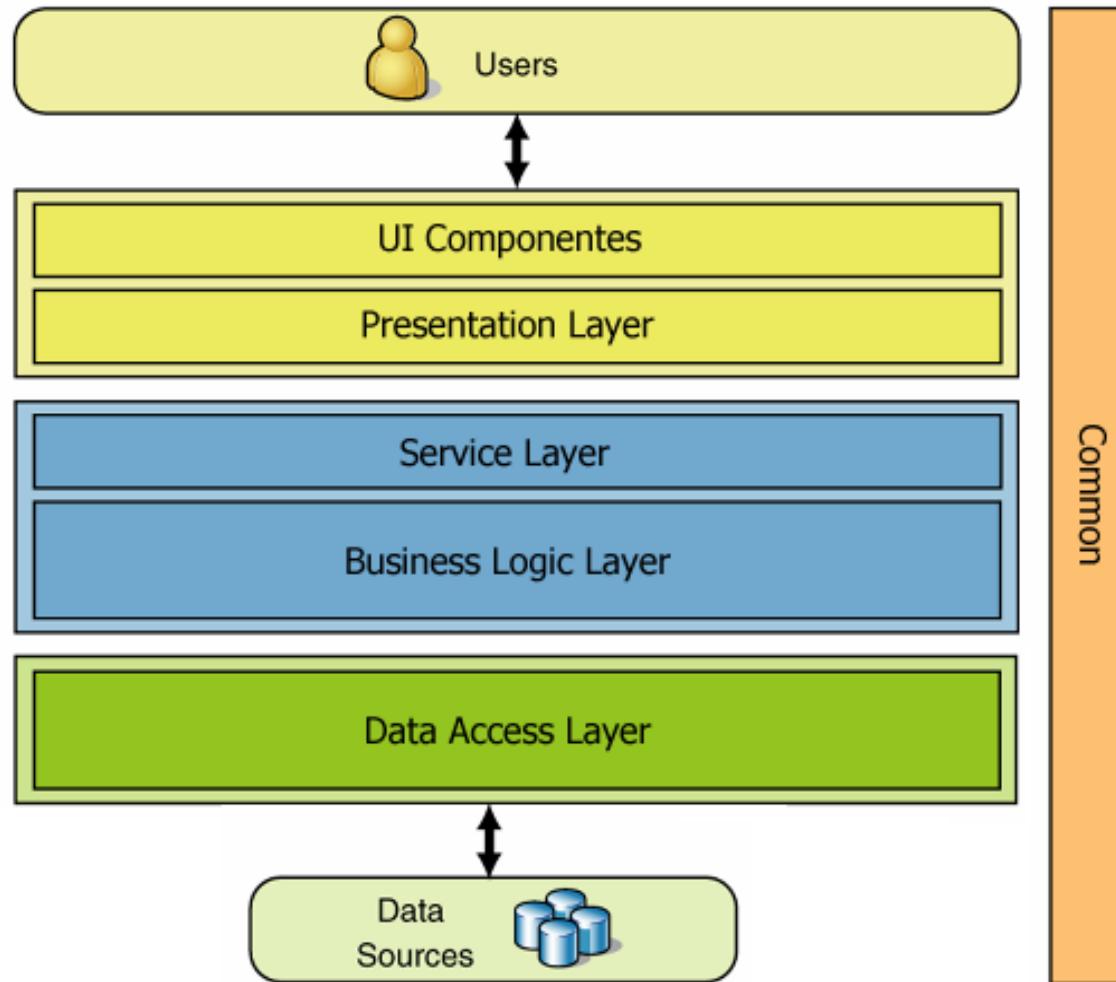
- GET /api/v1/Catalog** (blue button)
- POST /api/v1/Catalog** (green button)
- PUT /api/v1/Catalog** (orange button)
- GET /api/v1/Catalog/{id}** (blue button)
- DELETE /api/v1/Catalog/{id}** (red button)
- GET /api/v1/Catalog/GetProductByCategory/{category}** (blue button)

Catalog REST Apis

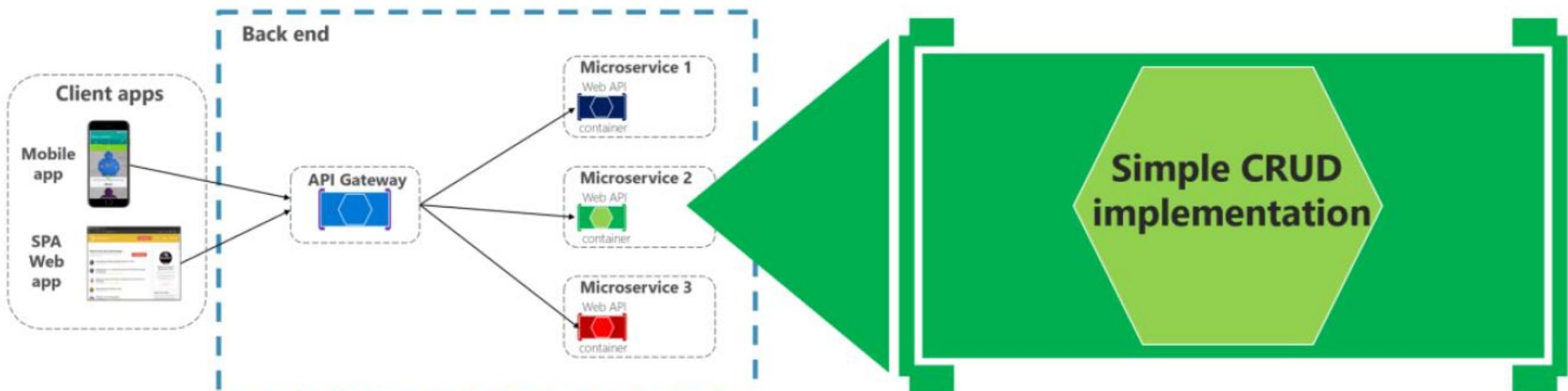
Method	Request URI	Use Case
GET	api/v1/Catalog	Listing Products and Categories
GET	api/v1/Catalog/{id}	Get Product with product Id
GET	api/v1/Catalog/ GetProductByCategory {category}	Get Products with category
POST	api/v1/Catalog	Create new Product
PUT	api/v1/Catalog	Update Product
DELETE	api/v1/Catalog/{id}	Delete Product

Catalog Layered Architecture

- Data Access Layer
- Business Logic Layer
- Presentation Layer



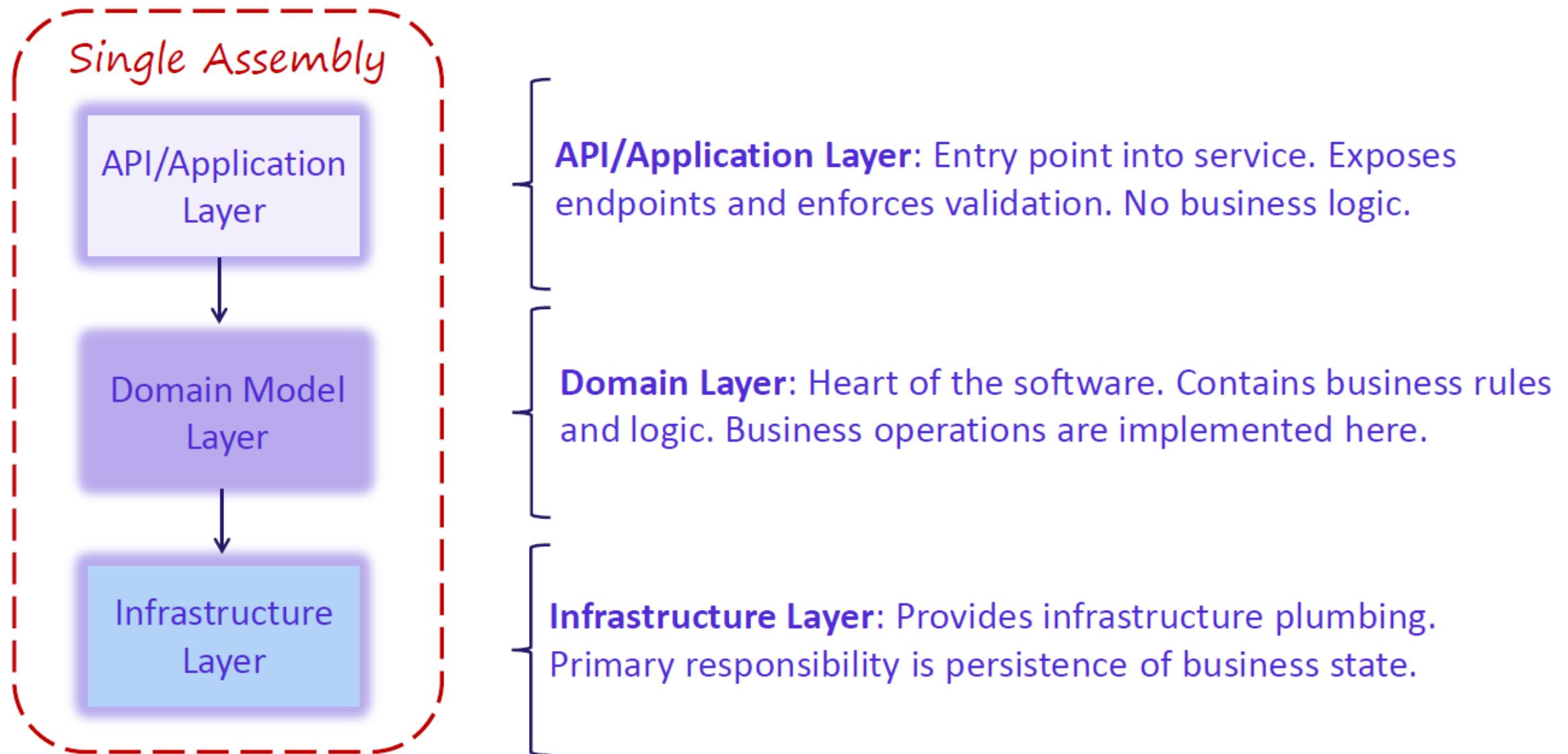
Catalog API – Simple CRUD Microservices



- External microservice patterns
- API Gateway
- Resilient communication
- Pub/Sub and event driven

Internal design patterns per microservice

Simple Data-Driven, CRUD Microservice



Catalog Microservice Nuget Packages

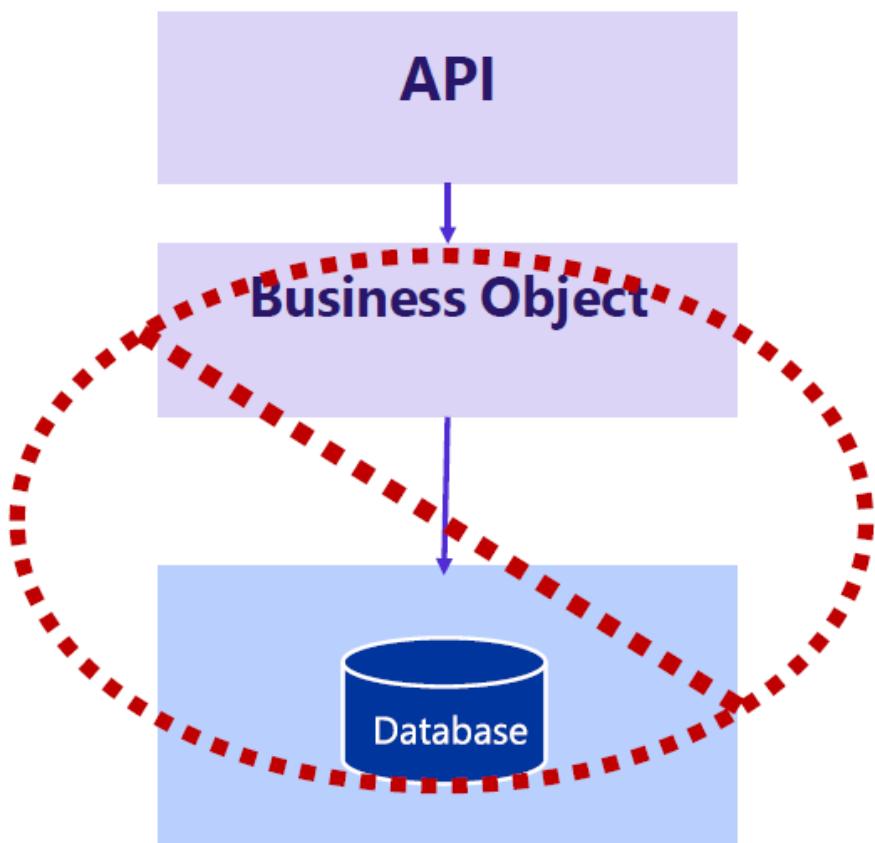
The screenshot shows a dark-themed catalog of NuGet packages. Two packages are listed:

- MongoDB.Driver** by vincentkam,dmitry_lukyanov,rstam,craigwilson
Official .NET driver for MongoDB.
- Swashbuckle.AspNetCore** by Swashbuckle.AspNetCore
Swagger tools for documenting APIs built on ASP.NET Core

Repository Pattern

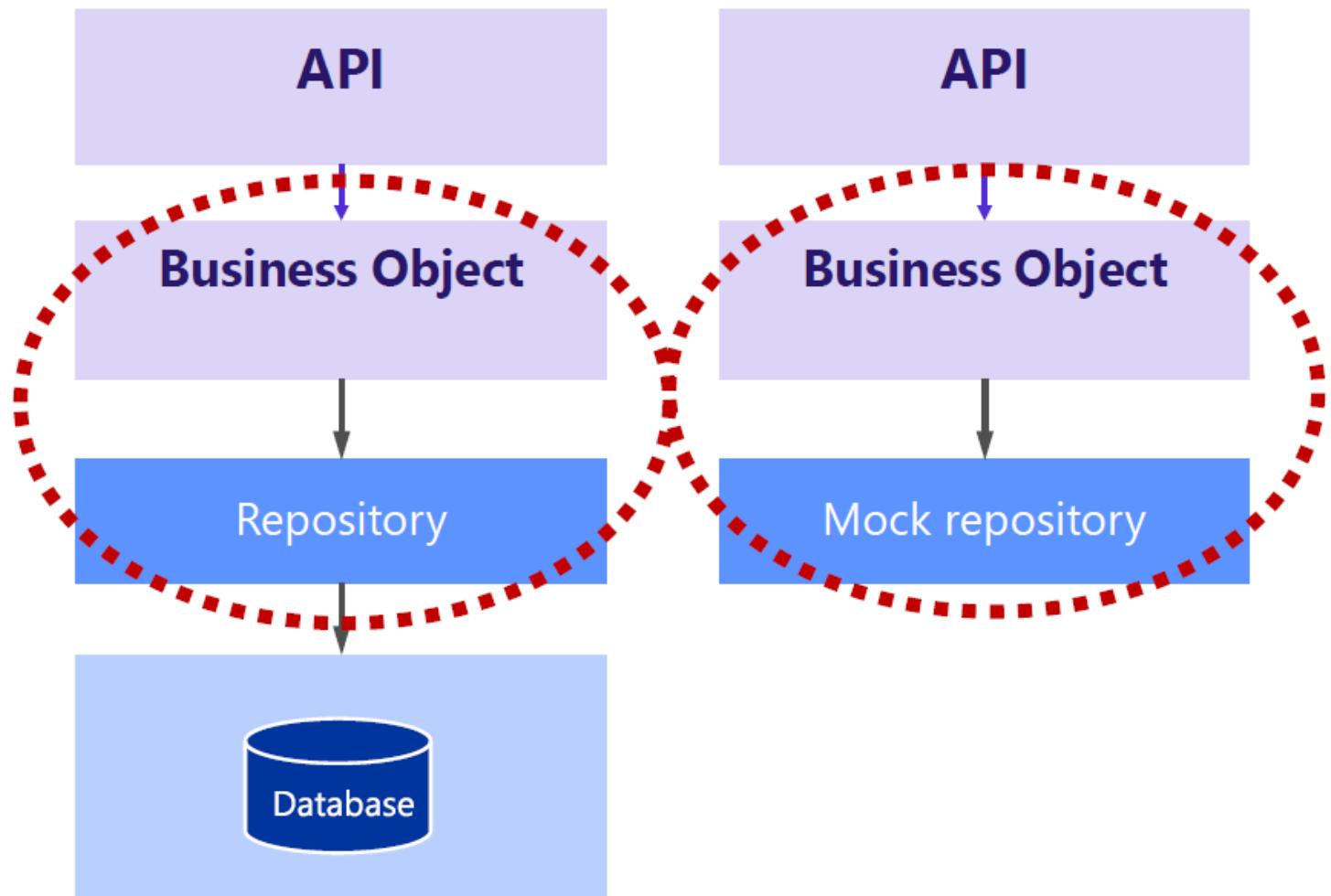
Without Repository

Direct access to database from controller/business layer



With Repository

Abstraction layer between business layer and database context.
Unit tests can mock data to facilitate testing.

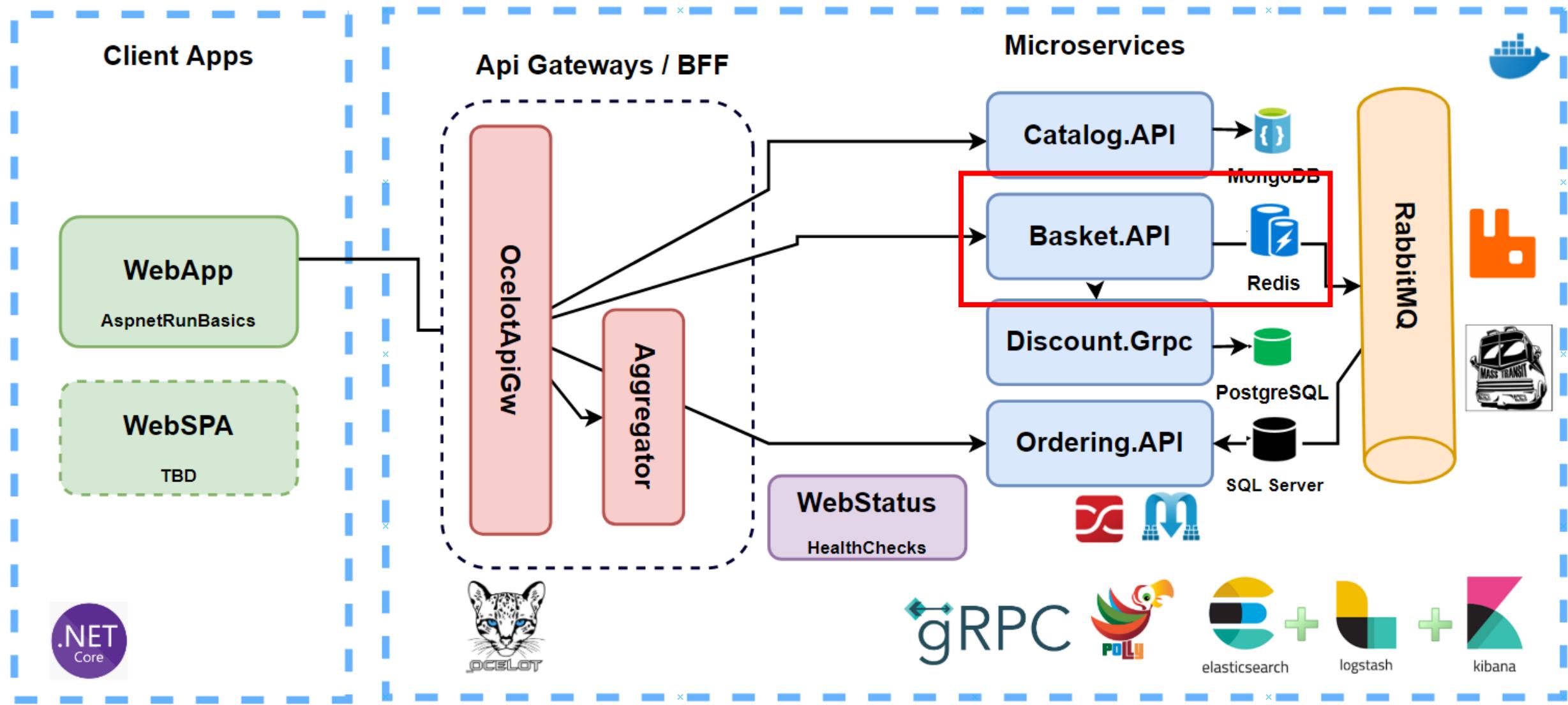


Section 3

Developing Basket.API Microservices with Redis

Basket.API with Redis

Big Picture



Redis in Basket Microservices

- Open-source NoSQL Database
- Remote Dictionary Server
- Key-value pairs
- Data Structure Server
- Extremely fast
- Save data both on RAM and on disk



Basket API Microservices

- ASP.NET Core Web API application
- REST API principles, CRUD operations
- Redis database connection and containerization
- Repository Pattern Implementation
- Containerize Basket Microservices with F# database using Docker Compose

The screenshot shows the Basket API's Swagger UI interface. At the top, it displays the title "Basket API v1 OAS3" and a link to "/swagger/v1/swagger.json". Below this, there is a section titled "Basket" containing four API endpoints:

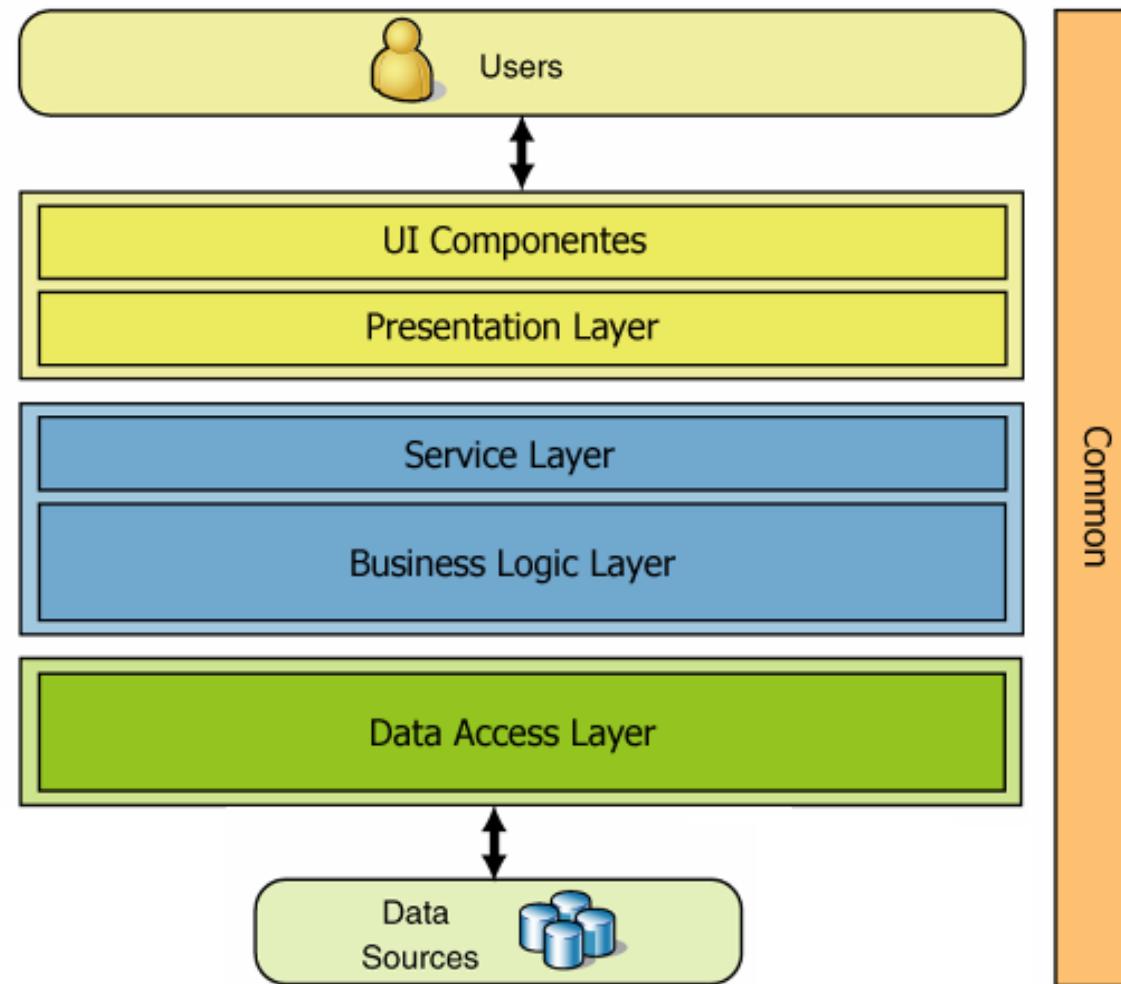
- A blue button labeled "GET /api/v1/Basket"
- A green button labeled "POST /api/v1/Basket"
- A red button labeled "DELETE /api/v1/Basket/{userName}"
- A green button labeled "POST /api/v1/Basket/Checkout"

Basket REST Apis

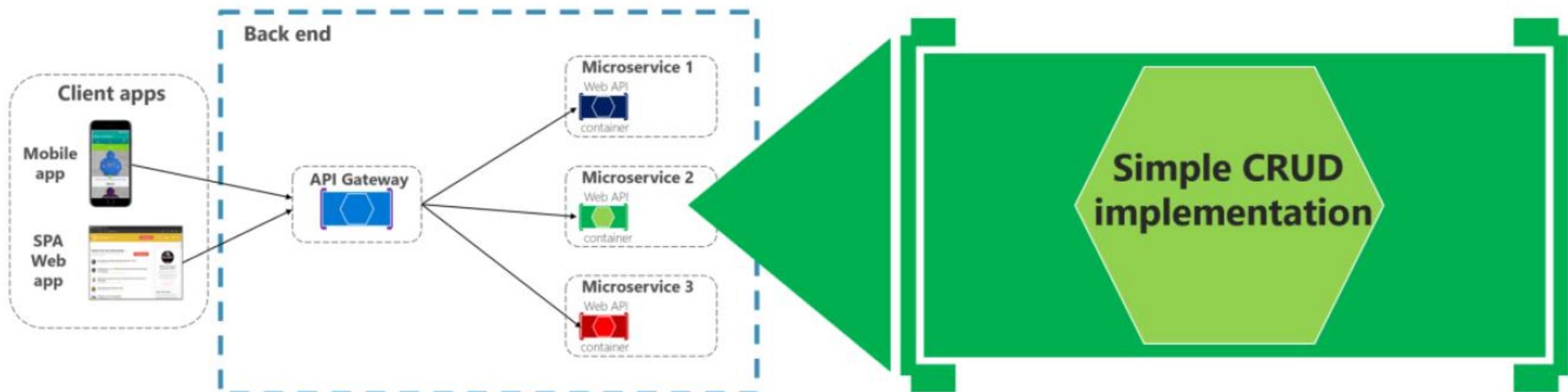
Method	Request URI	Use Case
GET	api/v1/Basket	Get Basket and Items with username
POST	api/v1/Basket	Update Basket and Items (add – remove item on basket)
DELETE	api/v1/Basket/{id}	Delete Basket
POST	api/v1/Basket/Checkout	Checkout Basket

Basket Layered Architecture

- Data Access Layer
- Business Logic Layer
- Presentation Layer



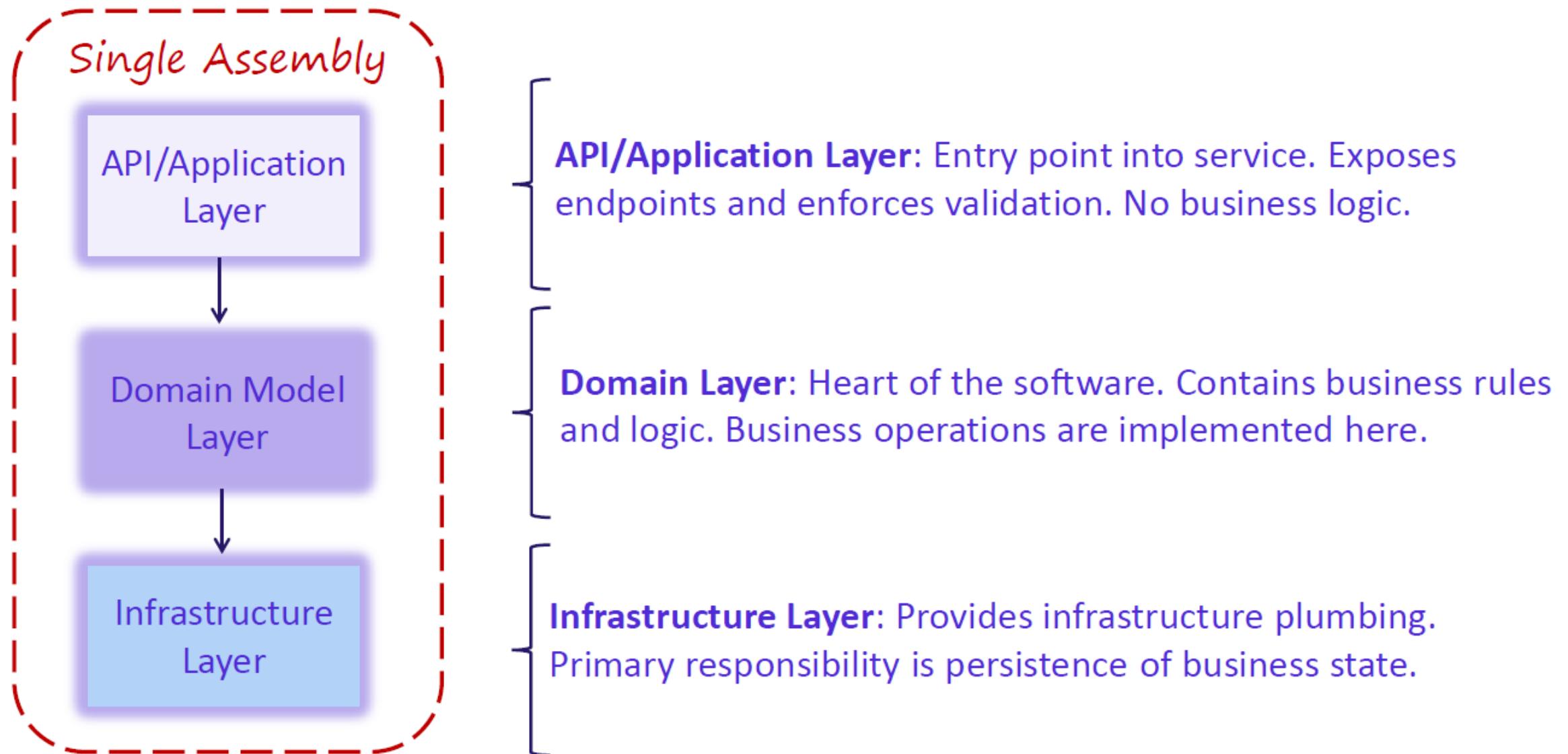
Basket API – Simple CRUD Microservices



- External microservice patterns
- API Gateway
- Resilient communication
- Pub/Sub and event driven

Internal design patterns per microservice

Simple Data-Driven, CRUD Microservice



Basket Microservice Nuget Packages

- Microsoft.Extensions.Caching.StackExchangeRedis
- Newtonsoft.Json
- Swashbuckle.AspNetCore

Container management with Portainer

- Open-source
- Managing container-based software application
- Kubernetes, Docker, Docker Swarm, Azure ACI and edge environments
- Manage environments, deploy applications, monitor app performance and triage problems



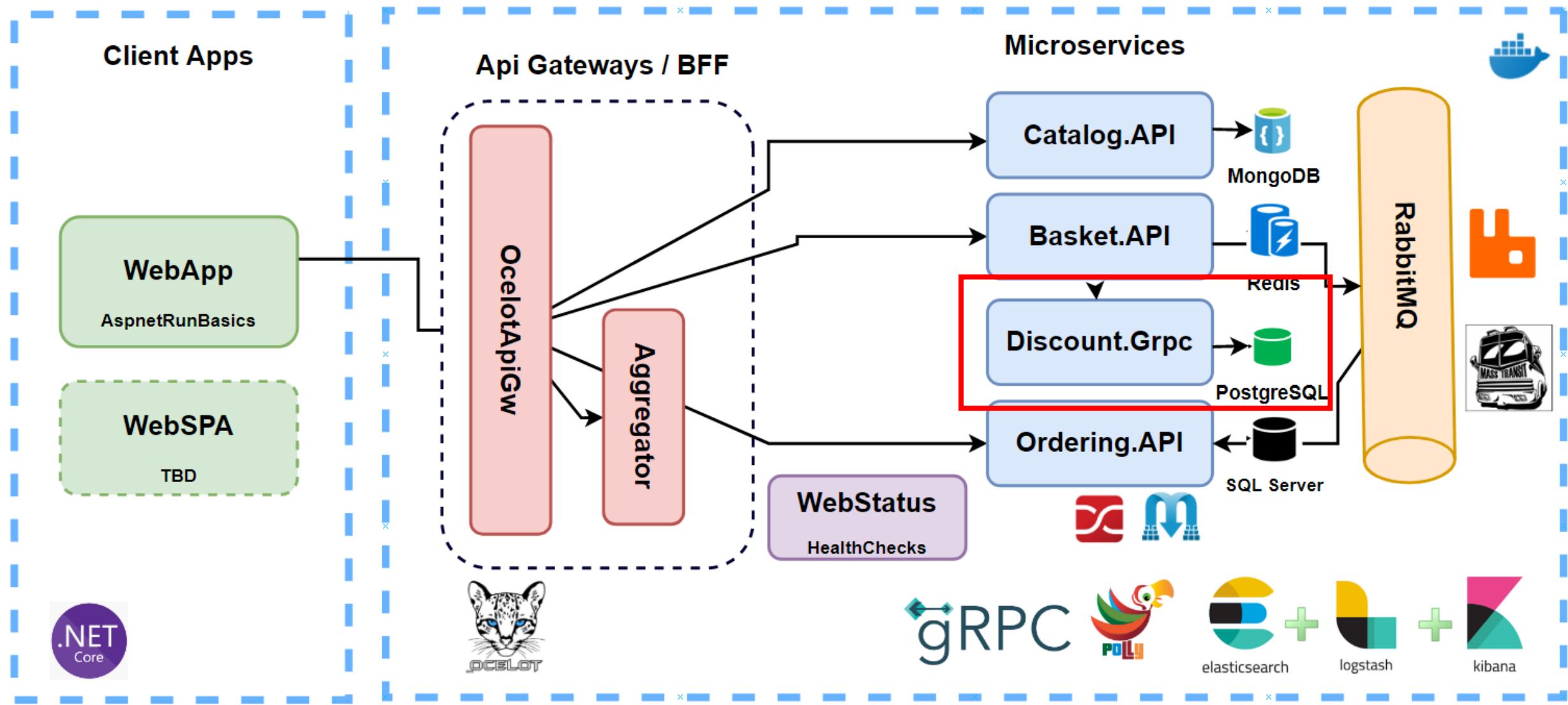
Section 4

Developing Discount.API

Microservices with PostgreSQL

Discount.API with PostgreSQL

Big Picture



PostgreSQL in Discount Microservices

- Open-source Relational Database
- Security, storability and scalability
- Reliability and flexibility
- Successful Performance



Discount API Microservices

- ASP.NET Core Web API application
- REST API principles, CRUD operations
- PostgreSQL database connection and containerization
- Repository Pattern Implementation
- Containerize Discount Microservices with PostgreSQL database using Docker Compose

Basket API v1 OAS3

</swagger/v1/swagger.json>

Basket

GET /api/v1/Basket

POST /api/v1/Basket

DELETE /api/v1/Basket/{userName}

POST /api/v1/Basket/Checkout

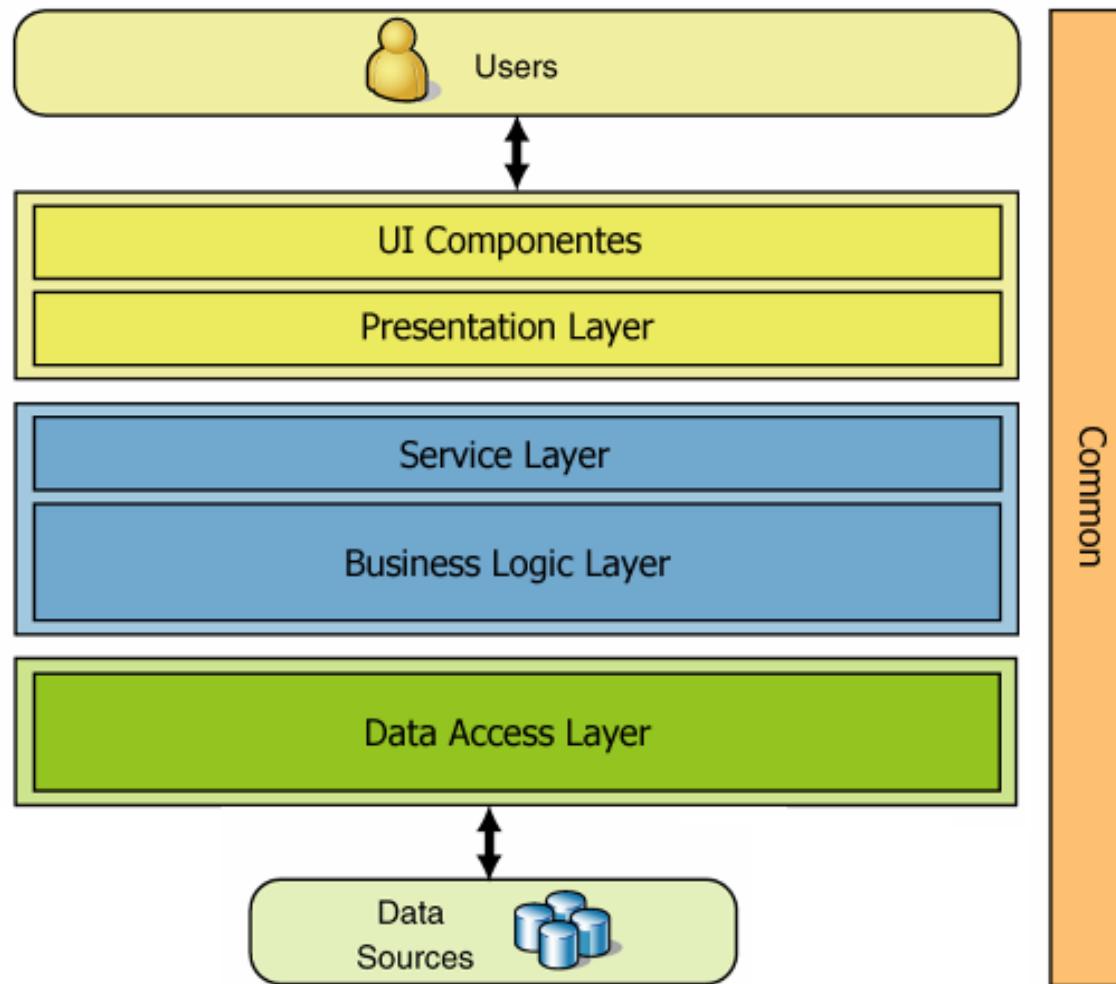
Discount REST APIs

Method	Request URL
GET	api/v1/
POST	api/v1/
DELETE	api/v1/
POST	api/v1/

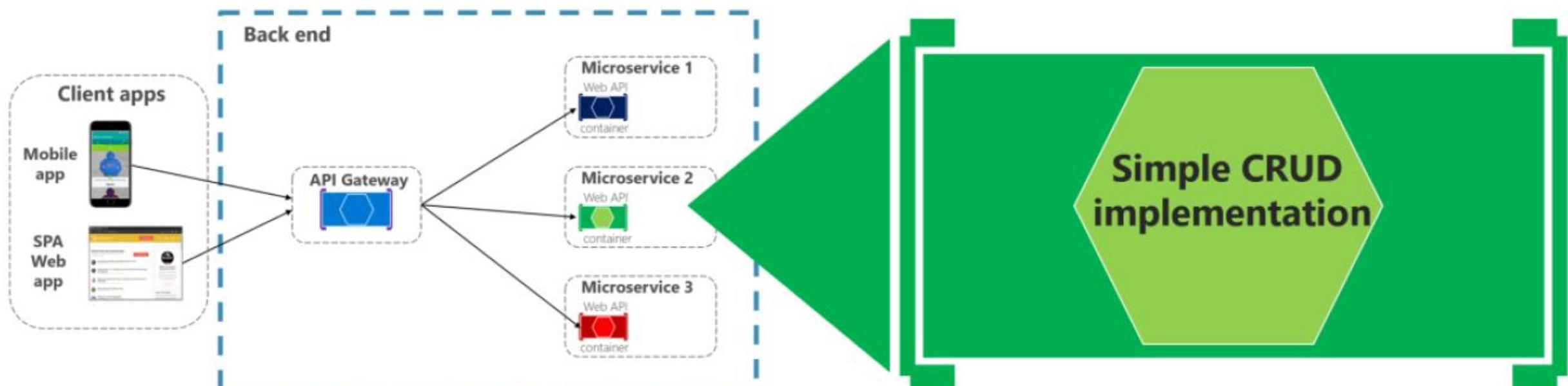


Discount Layered Architecture

- Data Access Layer
- Business Logic Layer
- Presentation Layer



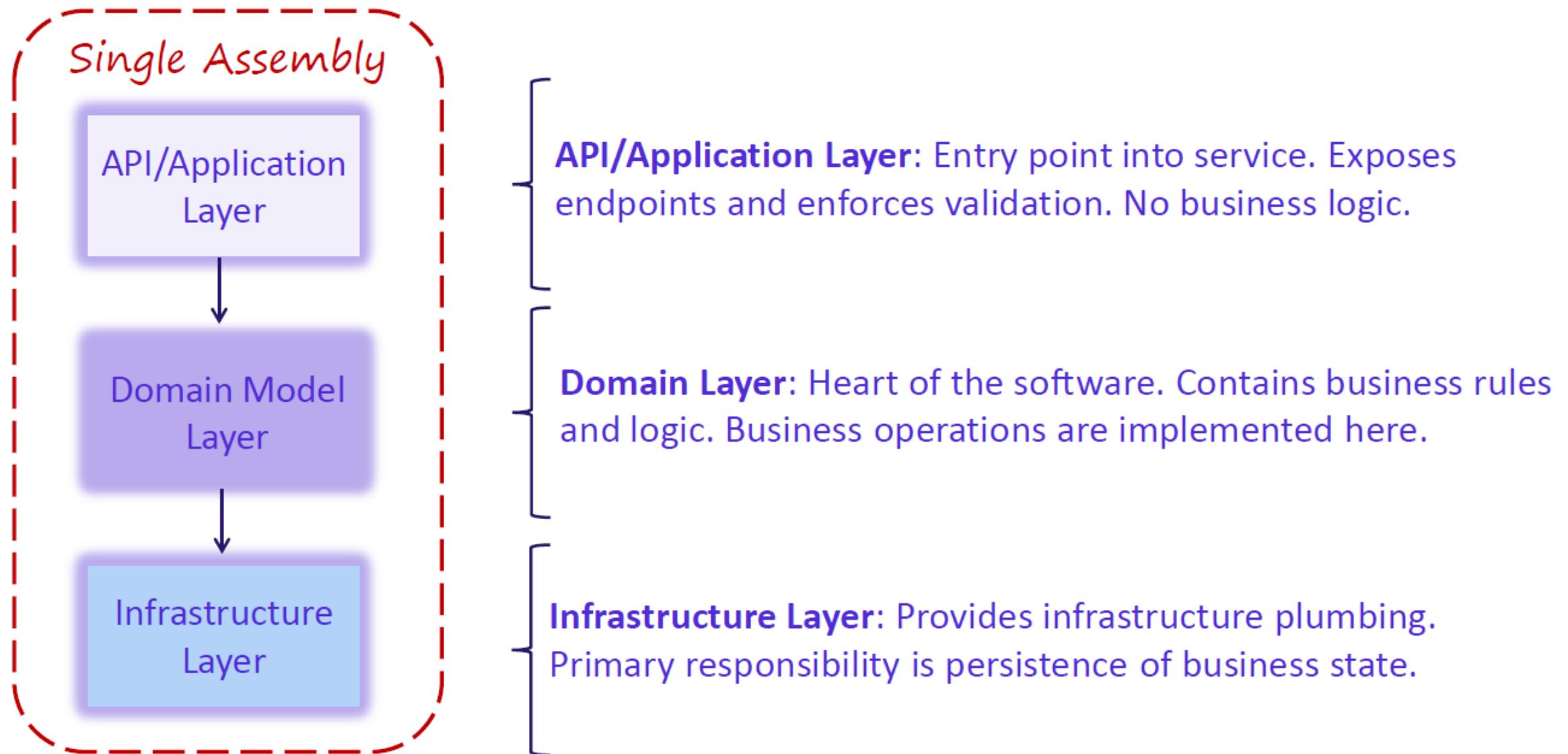
Discount API – Simple CRUD Microservices



- External microservice patterns
- API Gateway
- Resilient communication
- Pub/Sub and event driven

Internal design patterns per microservice

Simple Data-Driven, CRUD Microservice



Discount Microservice Nuget Packages

- Npgsql
- Dapper
- Swashbuckle.AspNetCore

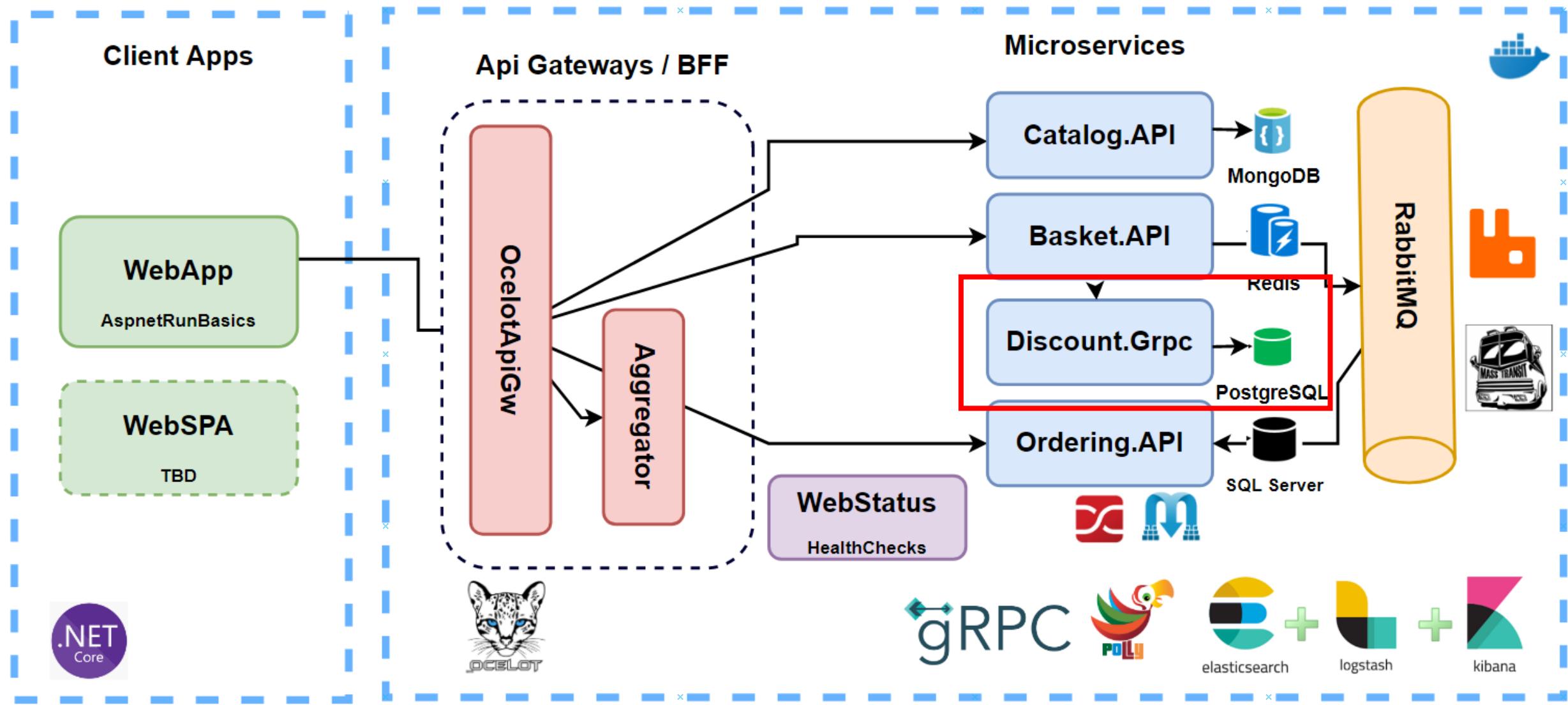


Section 5

Developing Discount.Grpc Microservices for Microservices Grpc Communication

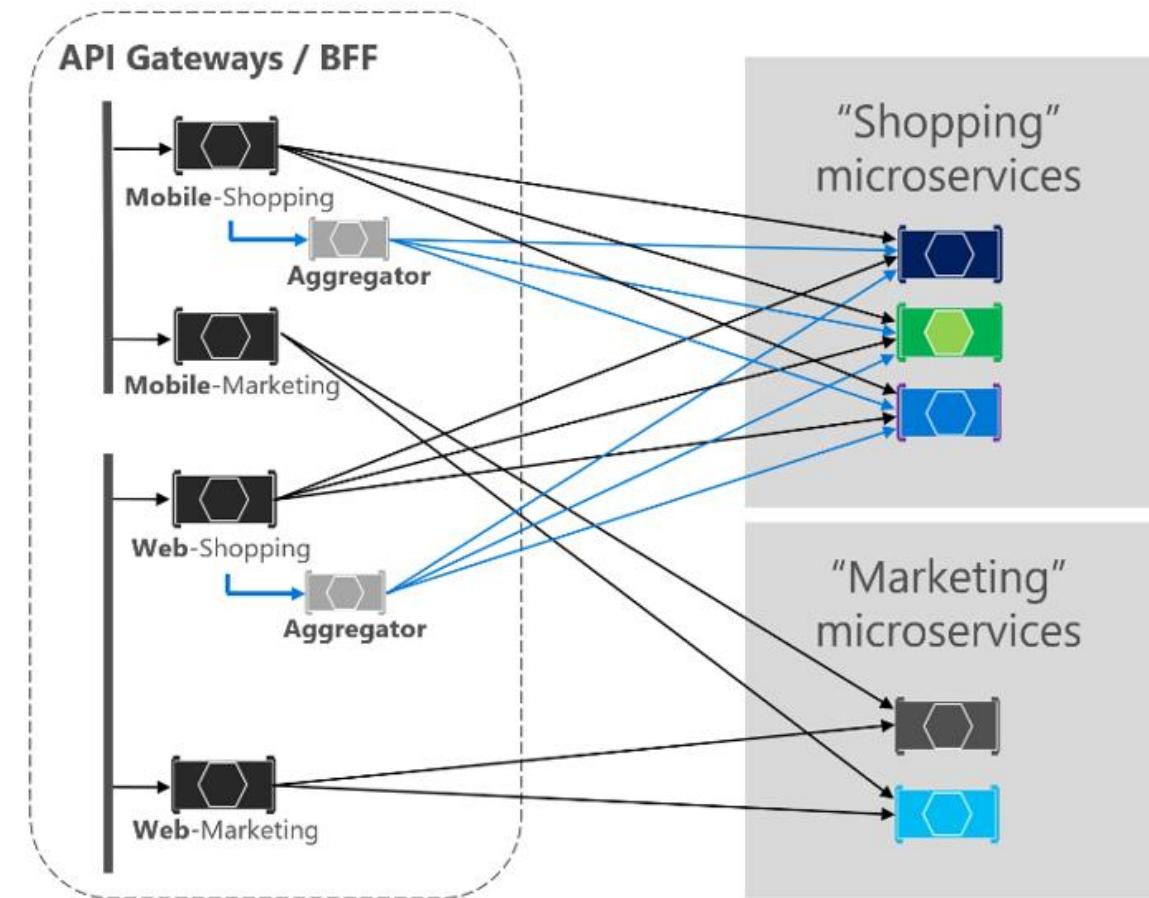
Discount.Grpc with PostgreSQL

Big Picture

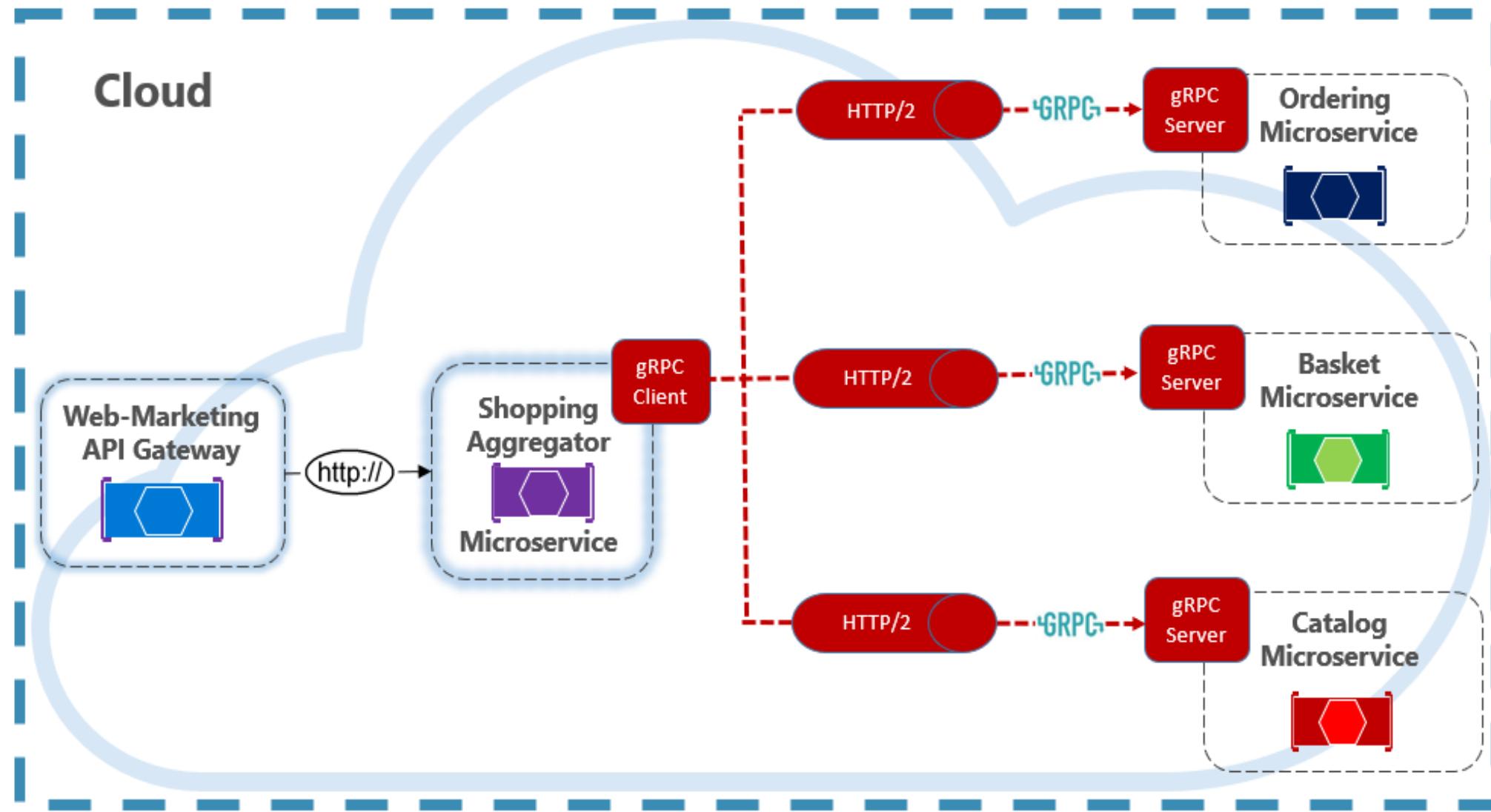


gRPC usage of Microservices Communication

- Synchronous backend microservice-to-microservice communication
- Polyglot environments
- Low latency and high throughput communication
- Point-to-point real-time communication
- Network constrained environments



Example of gRPC in Microservices





aspnetrun

The best path to leverage your aspnet skills. Onboarding to .Net Software Architect jobs. Download latest real world asp.net core microservices applications.

Istanbul

<https://aspnetrun.azurewebsites.net>

ezozkme@gmail.com

Repositories 12

Packages

People 1

Teams

Projects 1

Settings

Pinned repositories

Customize pinned repositories

run-aspnetcore

Template



A starter kit for your next ASP.NET Core web application. Boilerplate for ASP.NET Core reference application, demonstrating a layered application architecture with applying Clean Architecture and D...

C# 223 55

run-aspnetcore-microservices

Microservices on .Net platforms which used Asp.Net Web API, Docker, RabbitMQ, MassTransit, Grpc, Ocelot API Gateway, MongoDB, Redis, PostgreSQL, SqlServer, Dapper, Entity Framework Core, CQRS and C...

C# 388 164

run-aspnetcore-realworld

E-Commerce real world example of run-aspnetcore ASP.NET Core web application. Implemented e-commerce domain with clean architecture for ASP.NET Core reference application, demonstrating a layered a...

SCSS 206 75

run-aspnet-identityserver4



Secure microservices with using standalone Identity Server 4 and backing with Ocelot API Gateway. Protect our ASP.NET Web MVC and API applications with using OAuth 2 and OpenID Connect in IdentityS...

C# 35 17

run-aspnet-grpc

Using gRPC in Microservices for Building a high-performance Interservice Communication with .Net 5. See gRPC Microservices and Step by Step Implementation on .NET Course w/ discount->

C# 33 10

run-devops

Deploying .Net Microservices into Kubernetes, and moving deployments to the cloud Azure Kubernetes Services (AKS) with using Azure Container Registry (ACR) and how to Automating Deployments with Az...

C# 4 7

Using gRPC in Microservices for Building a high-performance

Using gRPC in Microservices Communication with .Net 5

Building a high-performance gRPC Inter-service Communication between backend microservices with .Net 5 and AspNet 5

4.4 ★★★★☆ (44 ratings) 508 students

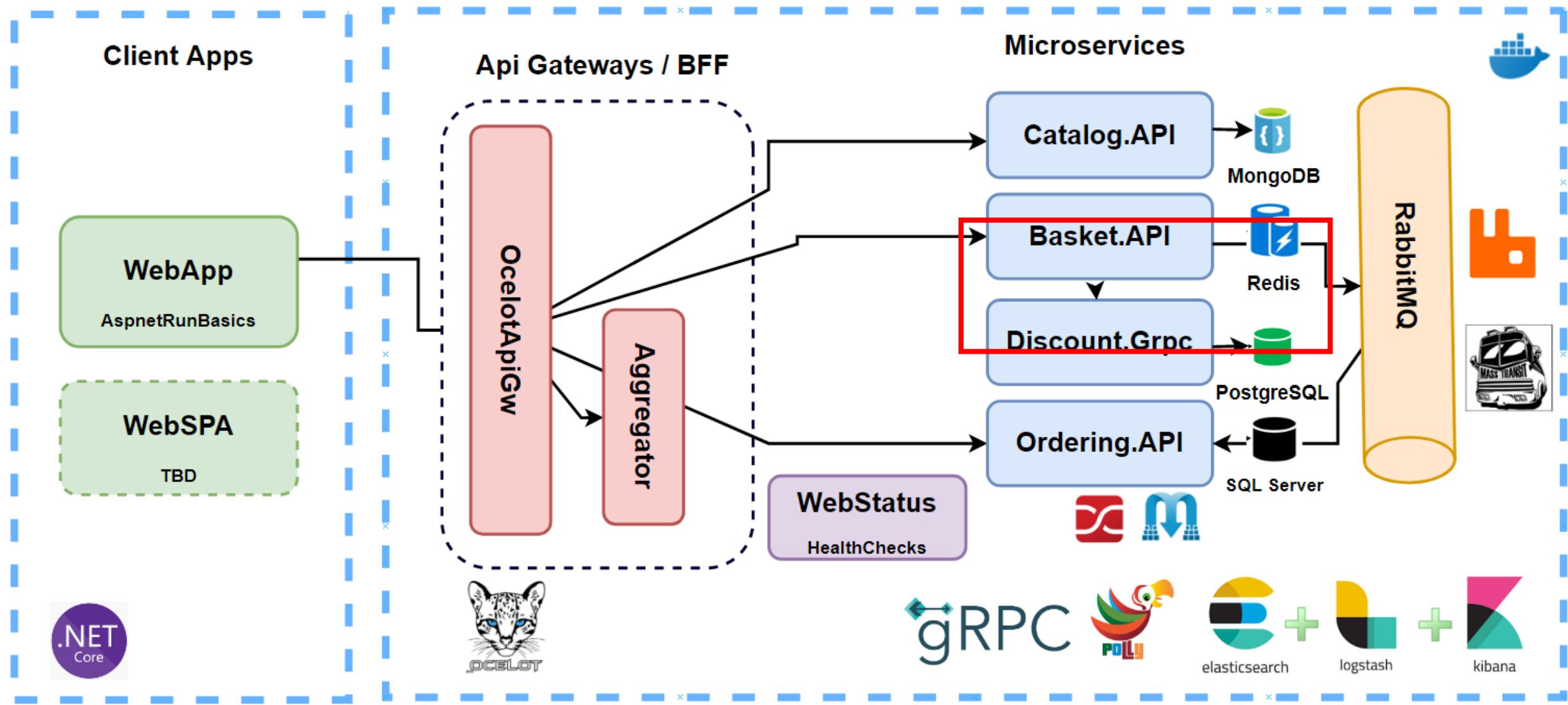
- Github Repository -> <https://github.com/aspnetrun/run-aspnet-grpc>

Section 6

Consuming Discount Grpc Service From Basket Microservice

When Adding Cart Item into Shopping Cart To Calculate Final Price

Big Picture



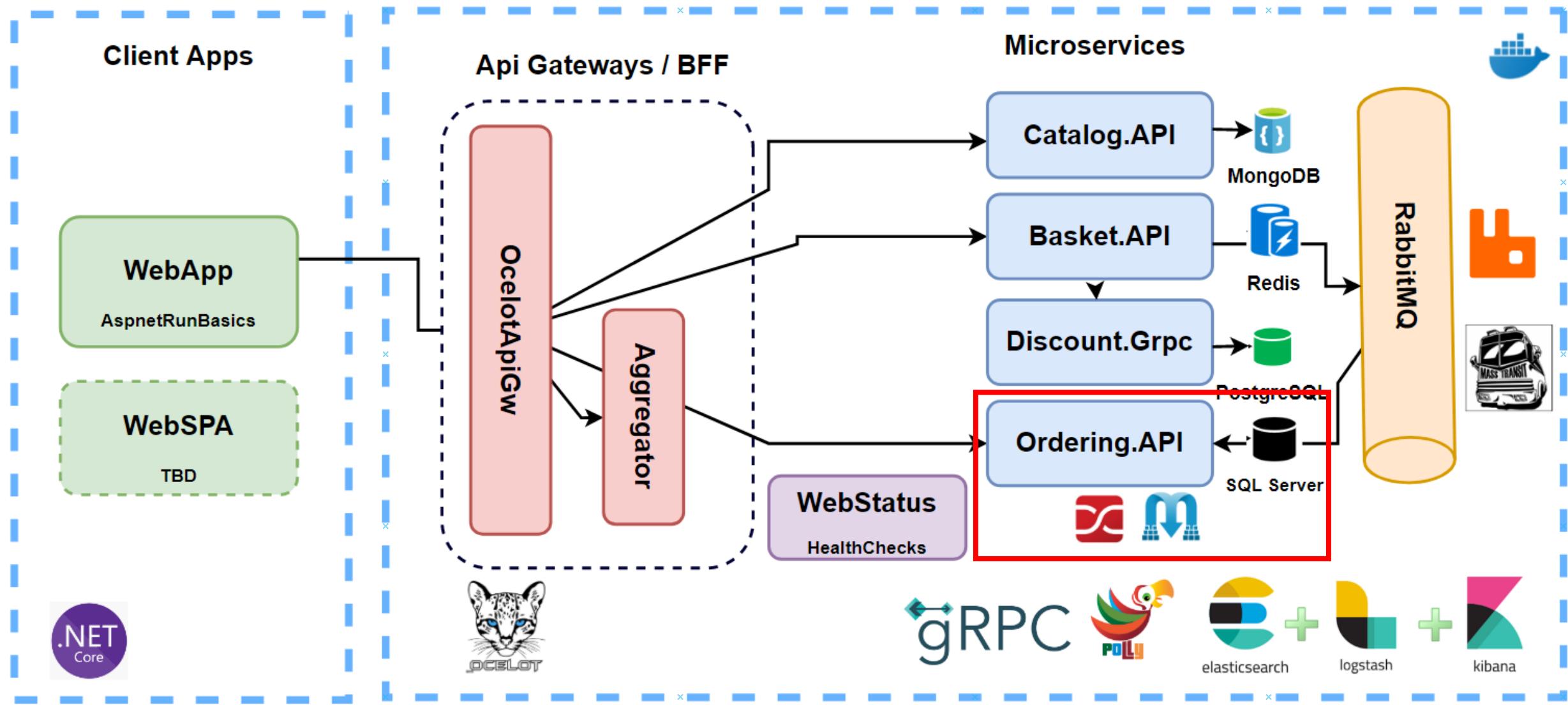
Section 7

Developing Ordering

Microservices

with Clean Architecture and CQRS Implementation

Big Picture



Ordering API Microservices

- ASP.NET Core Web API application
- REST API principles, CRUD operations
- Implementing DDD, CQRS, and Clean Architecture with using Best Practices applying SOLID principles
- Developing CQRS implementation on commands and queries with using MediatR, FluentValidation and AutoMapper packages
- Entity Framework Core Code-First Approach

The screenshot shows the Swagger UI for the Order API. At the top, it displays the API title "Order API" with version "v1" and "OAS3". Below the title, there is a link to the Swagger JSON file: "/swagger/v1/swagger.json". The main section is titled "Order". It contains two API endpoints: a GET endpoint for "/api/v1/Order" and a POST endpoint for "/api/v1/Order".

Ordering REST Apis

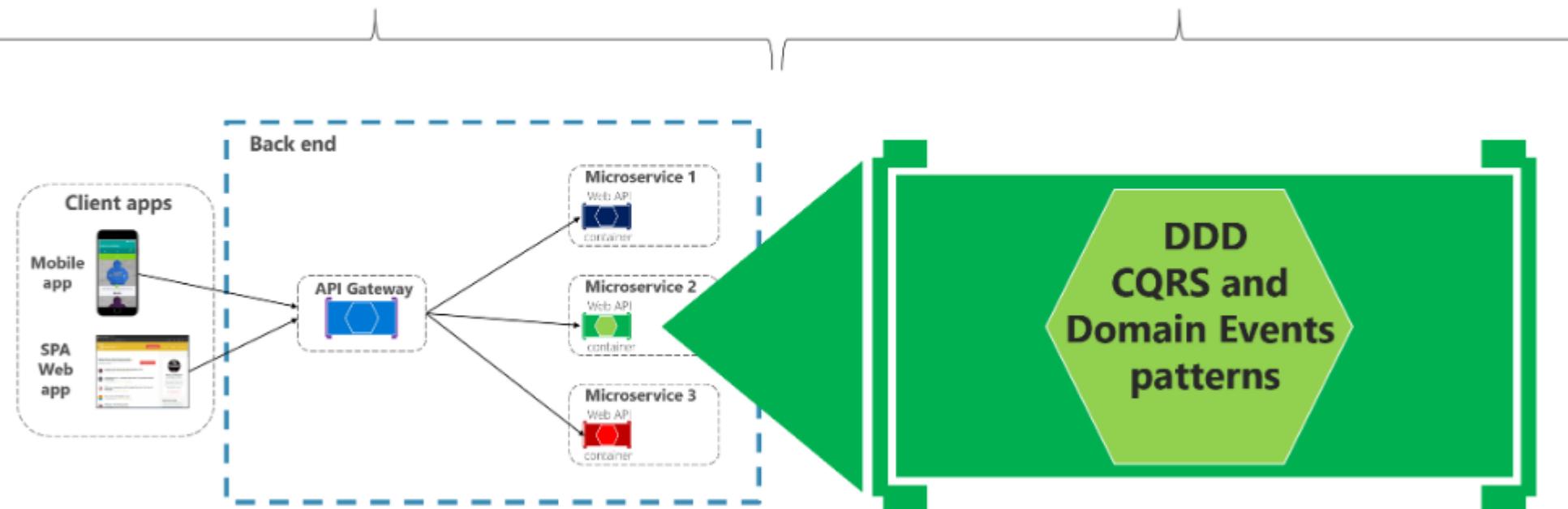
Method	Request URI	Use Case
GET	api/v1/Order	Get Orders with username

- Get Orders with username
- Consume basketCheckout event from RabbitMQ
- CQRS implementation with triggering OrderCommand to insert Order record

Ordering Architecture

External architecture
per application

Internal architecture
per microservice

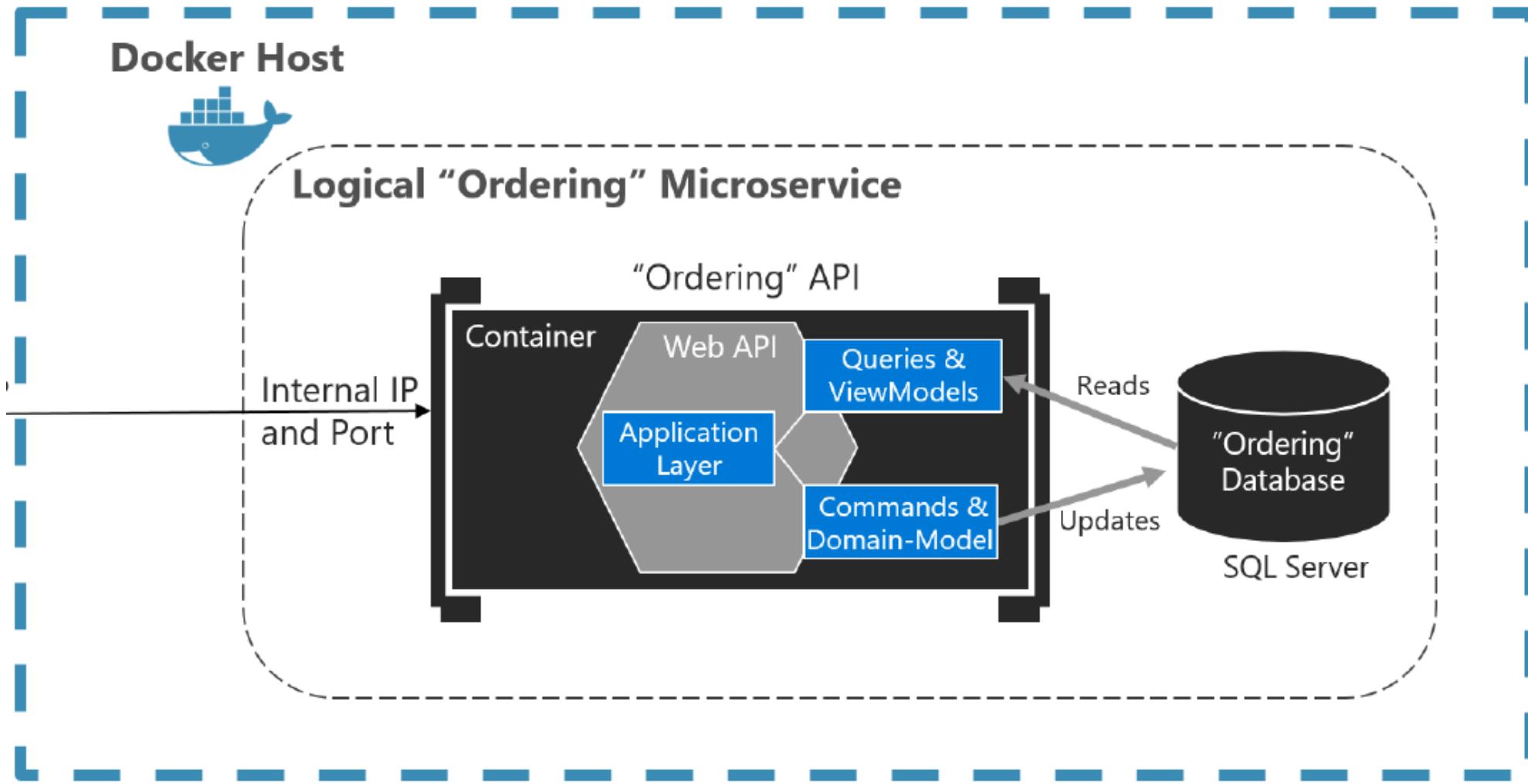


- External microservice patterns
- API Gateway
- Resilient communication
- Pub/Sub and event driven

Internal DDD patterns in addition to SOLID principles and Dependency Injection

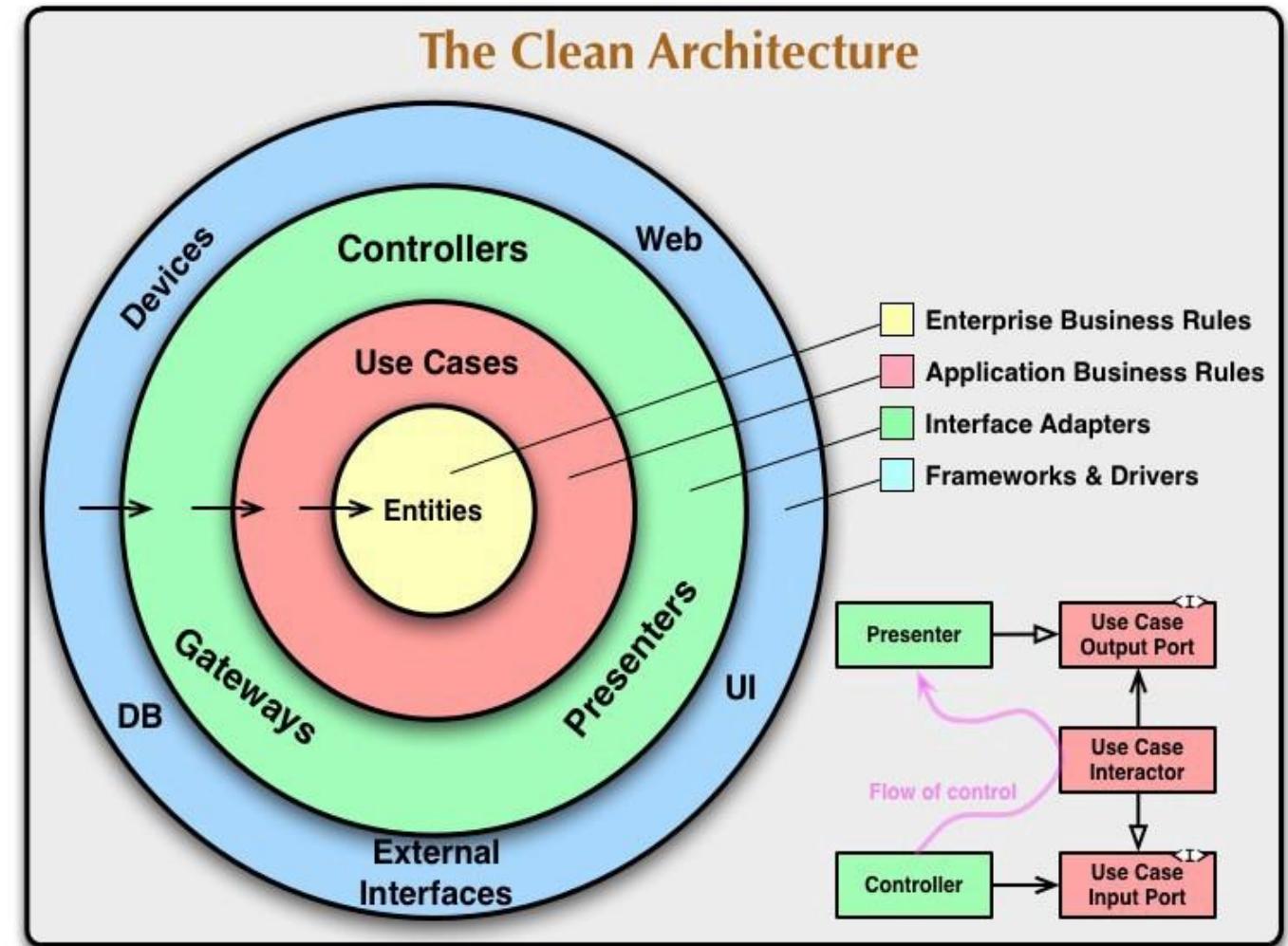
Simplified CQRS and DDD microservice

High level design

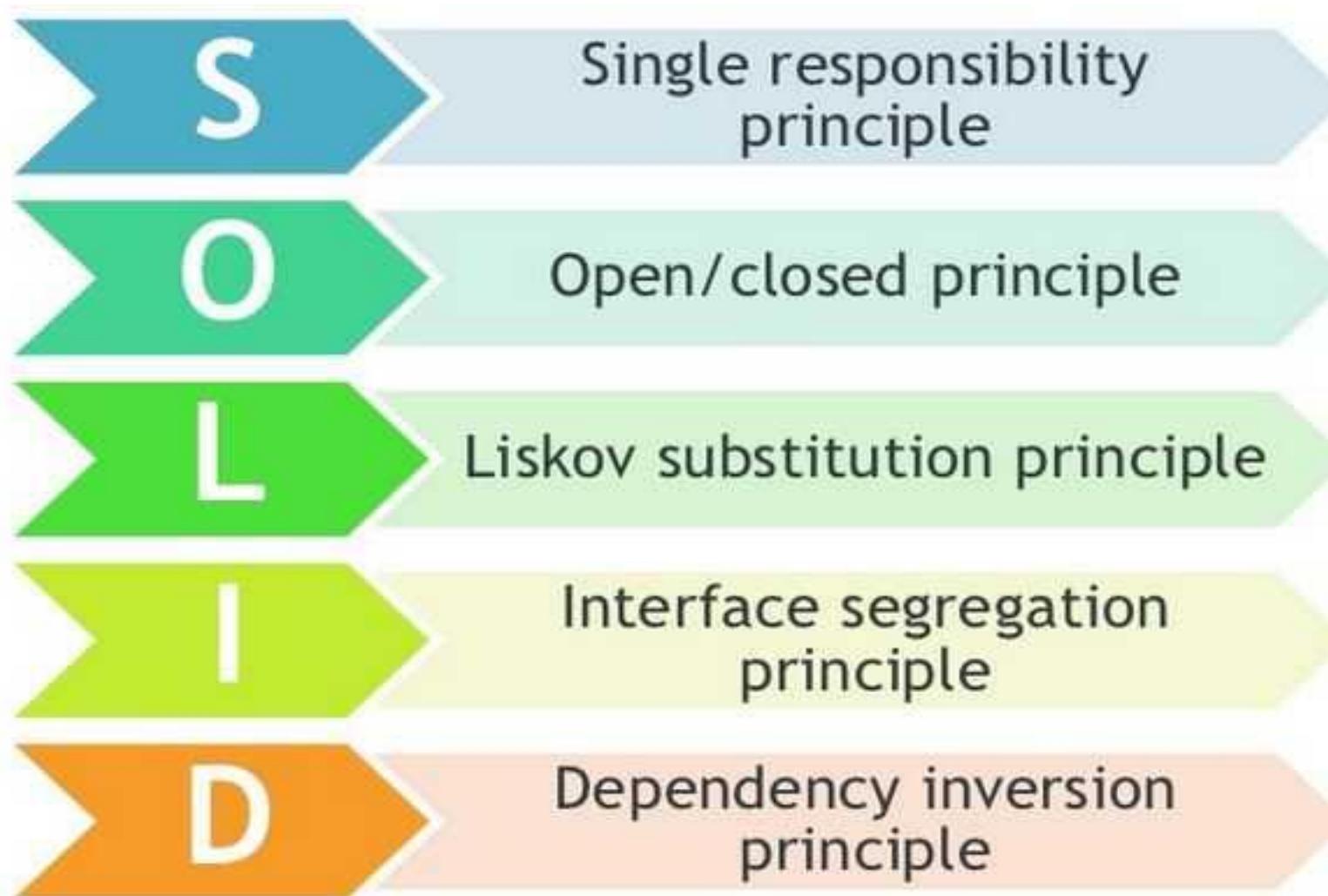


Architecture of Ordering Microservices

- SOLID Principles
- Domain Driven Design
- Clean Architecture
- Mediator Design Pattern
- CQRS Design Pattern
- CQRS & Event Sourcing

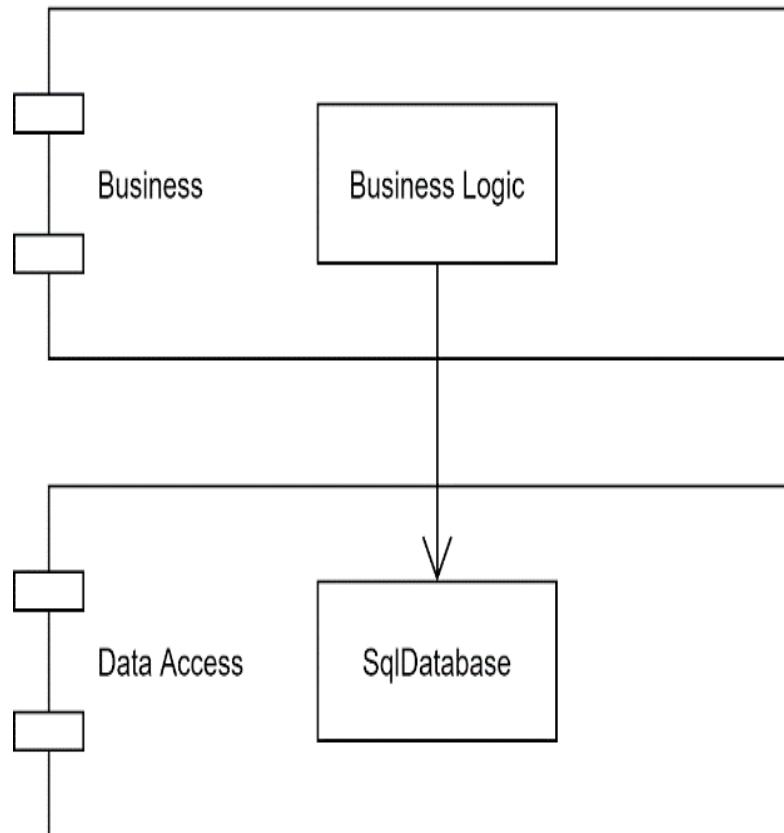


Design Principles - SOLID

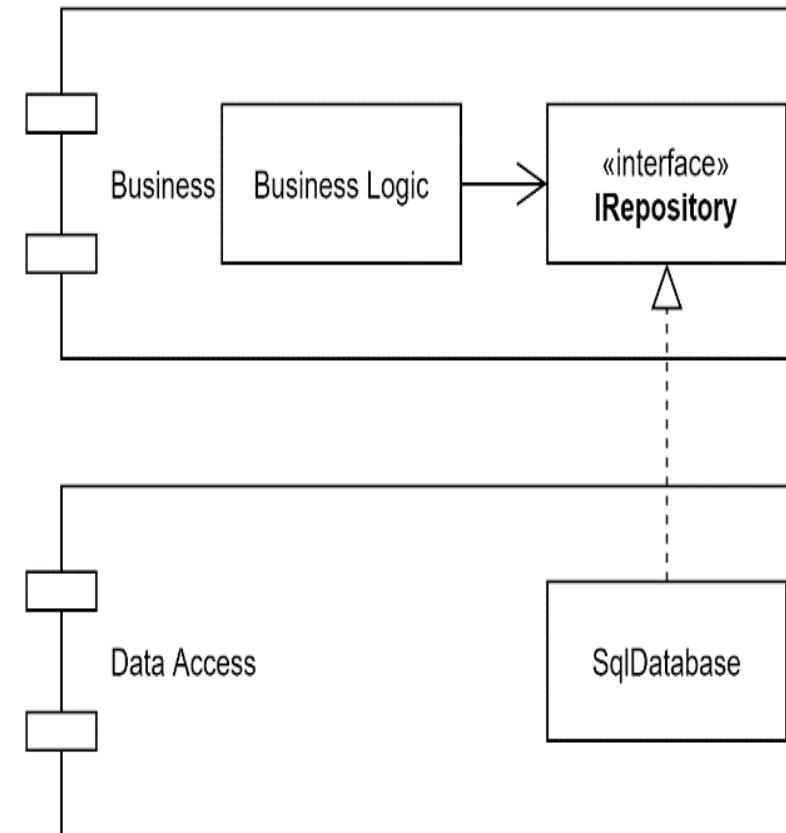


Dependency Inversion Principles (DIP)

Without Dependency Inversion



With Dependency Inversion



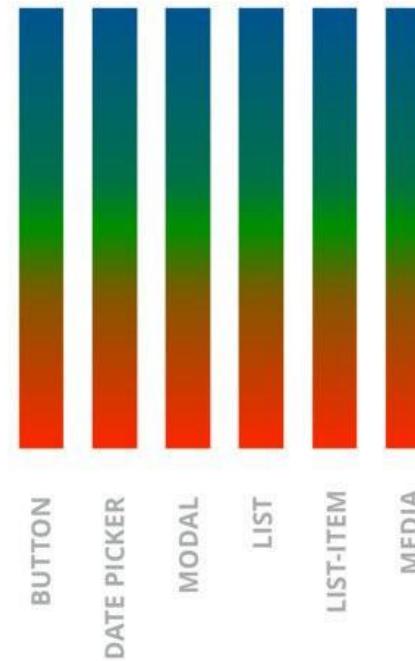
Separation of Concerns (SoC)

Separation of Concerns



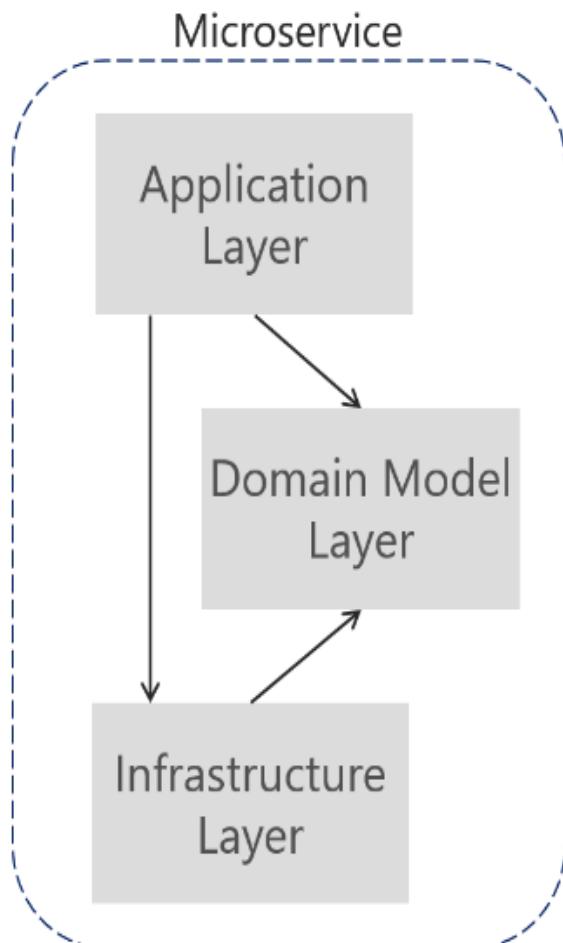
Separation of Concerns

(only, from a different point of view)



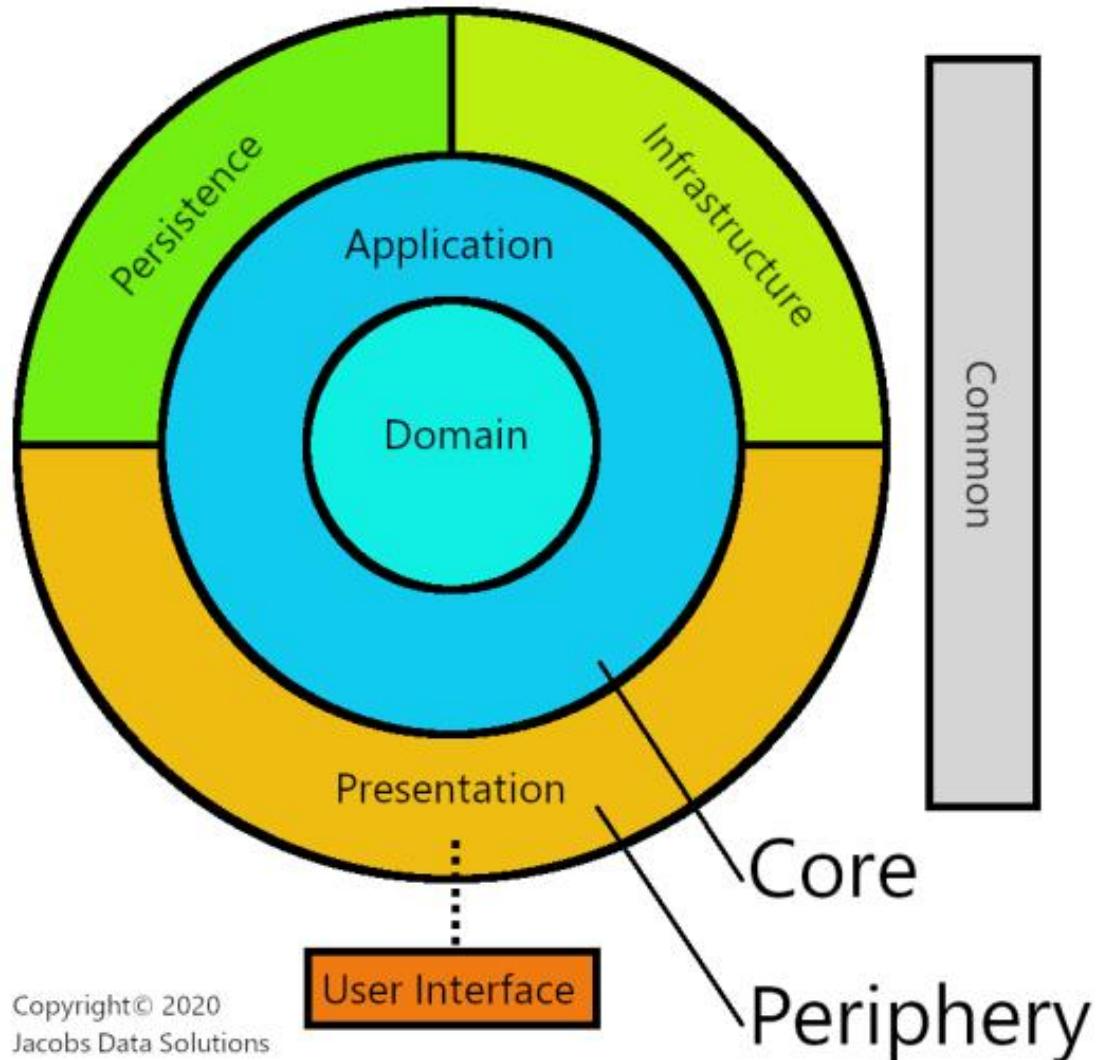
Domain Driven Design - DDD

Dependencies between Layers in a Domain-Driven Design service



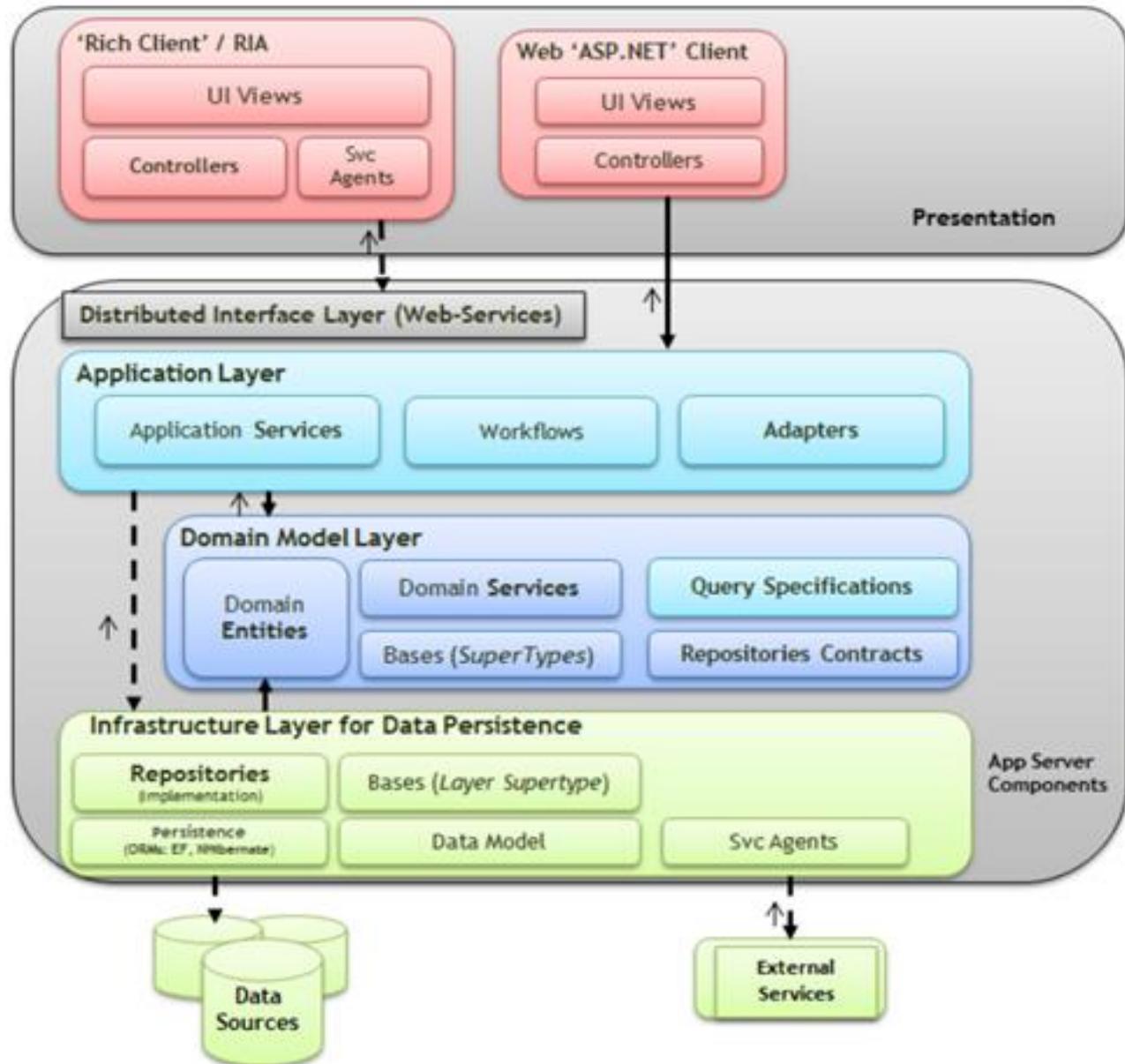
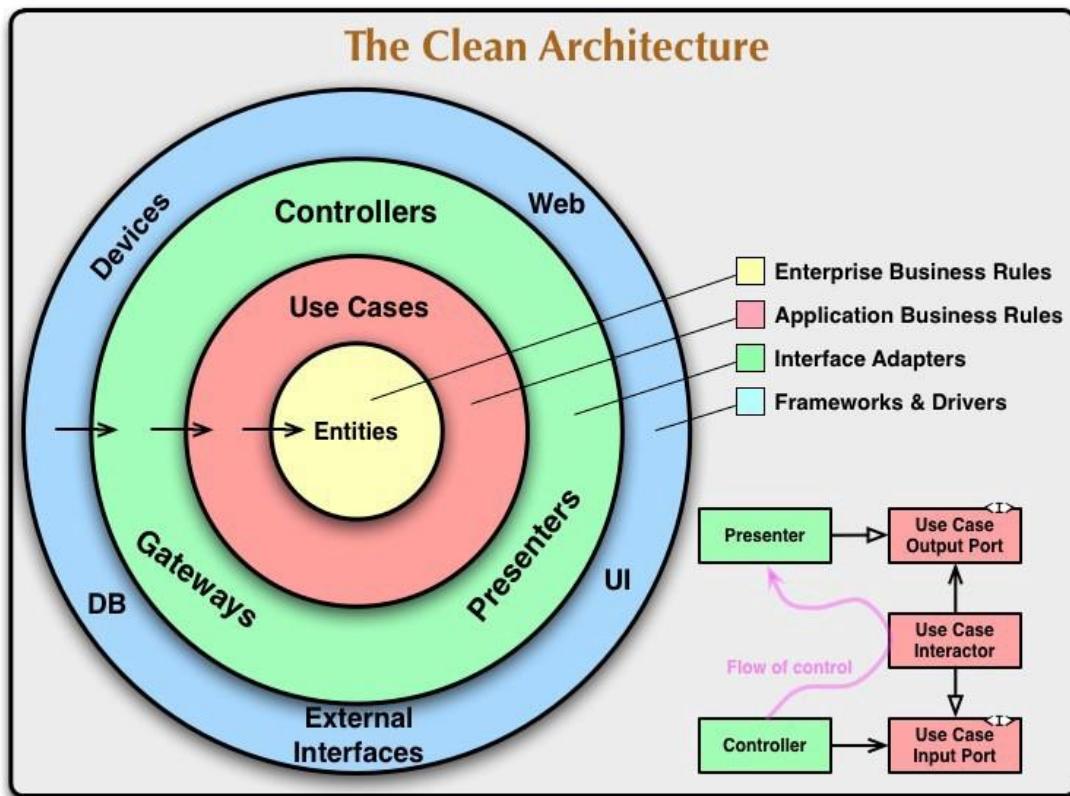
- Depends on the Domain-Model Layer so it can:
 - Use entity objects
 - Use Repository Interfaces/Contracts
- Depends on the Infrastructure Layer (thru DI) so it can:
 - Use Repository implementation classes, ideally through DI
- Ideally, it must NOT take dependency on any other layer
- It implements:
 - Domain Entities, Aggregate-Roots and Value-Objects
 - Repository Contracts/Interfaces (to be used in DI)
- Depends on the Domain-Model Layer so it can:
 - Use entity objects.
 - Like EF updating a database through mapped entities
- Direct dependency on infrastructure frameworks like EF Core or any other database, cache or infrastructure API

Clean Architecture



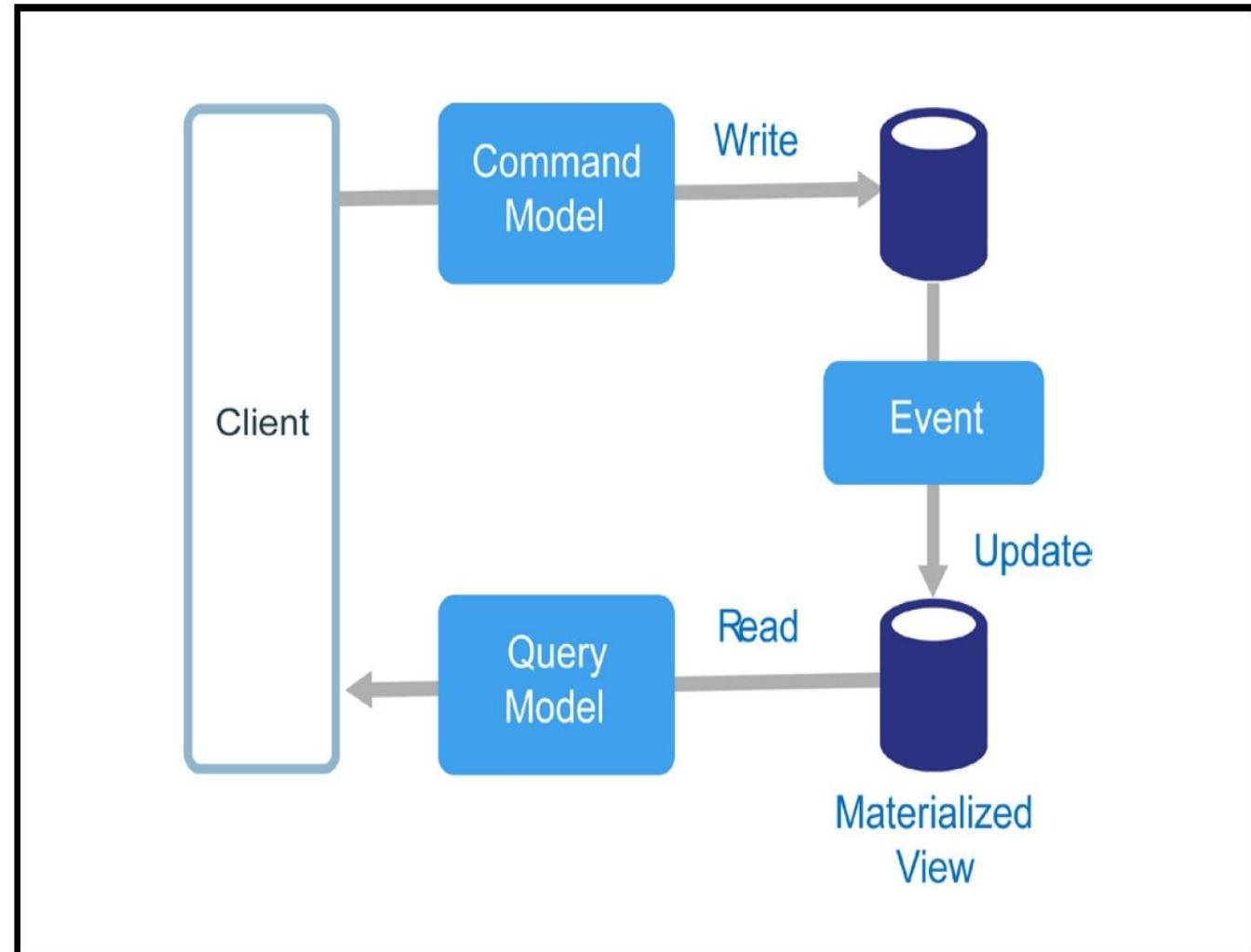
Copyright© 2020
Jacobs Data Solutions

Clean Architecture



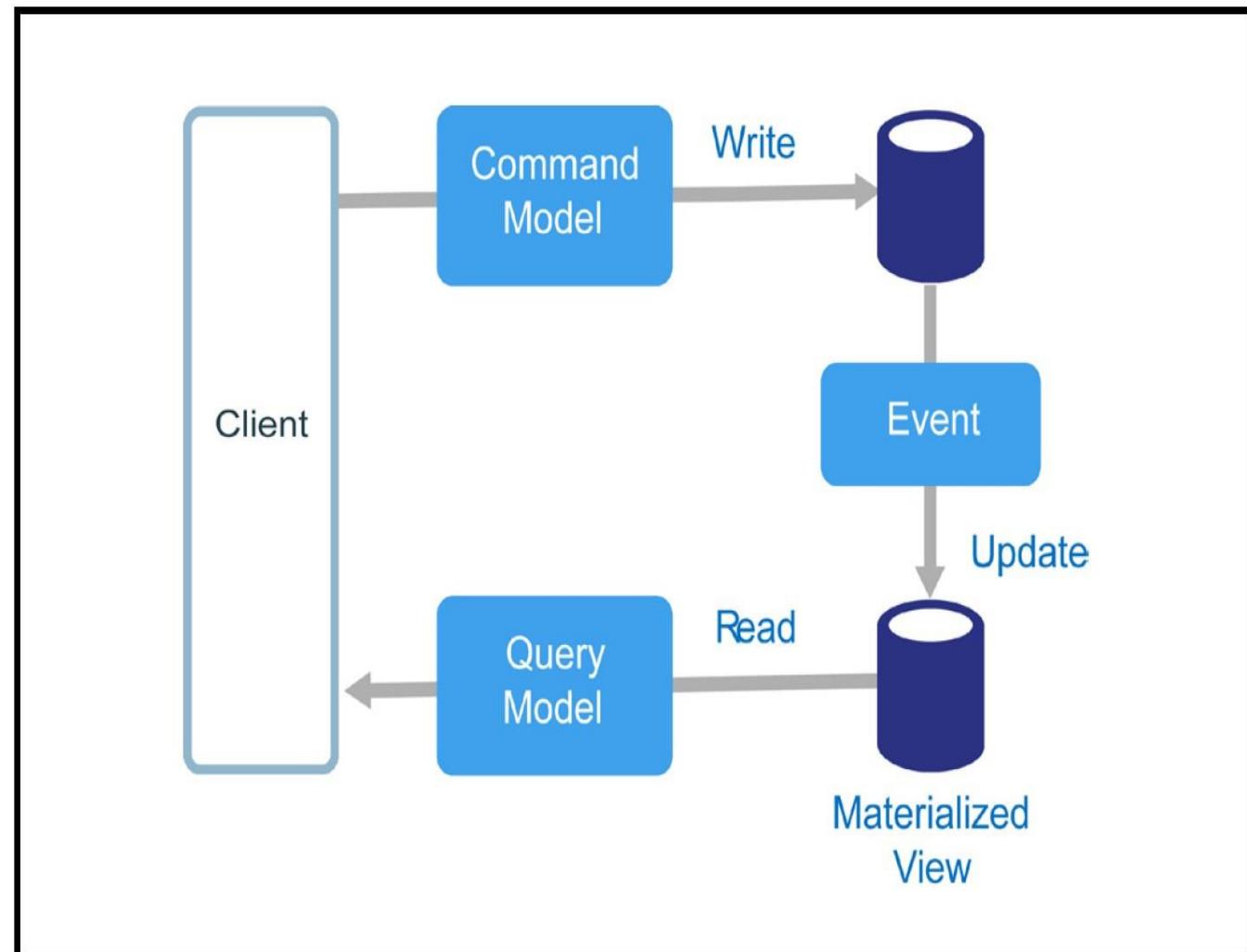
CQRS (Command Query Responsibility Segregation) Design Pattern

- Separation of Commands and Query Responsibility
- CQS (Command Query Separation)
- Commands – CommandHandlers
- Query – QueryHandlers



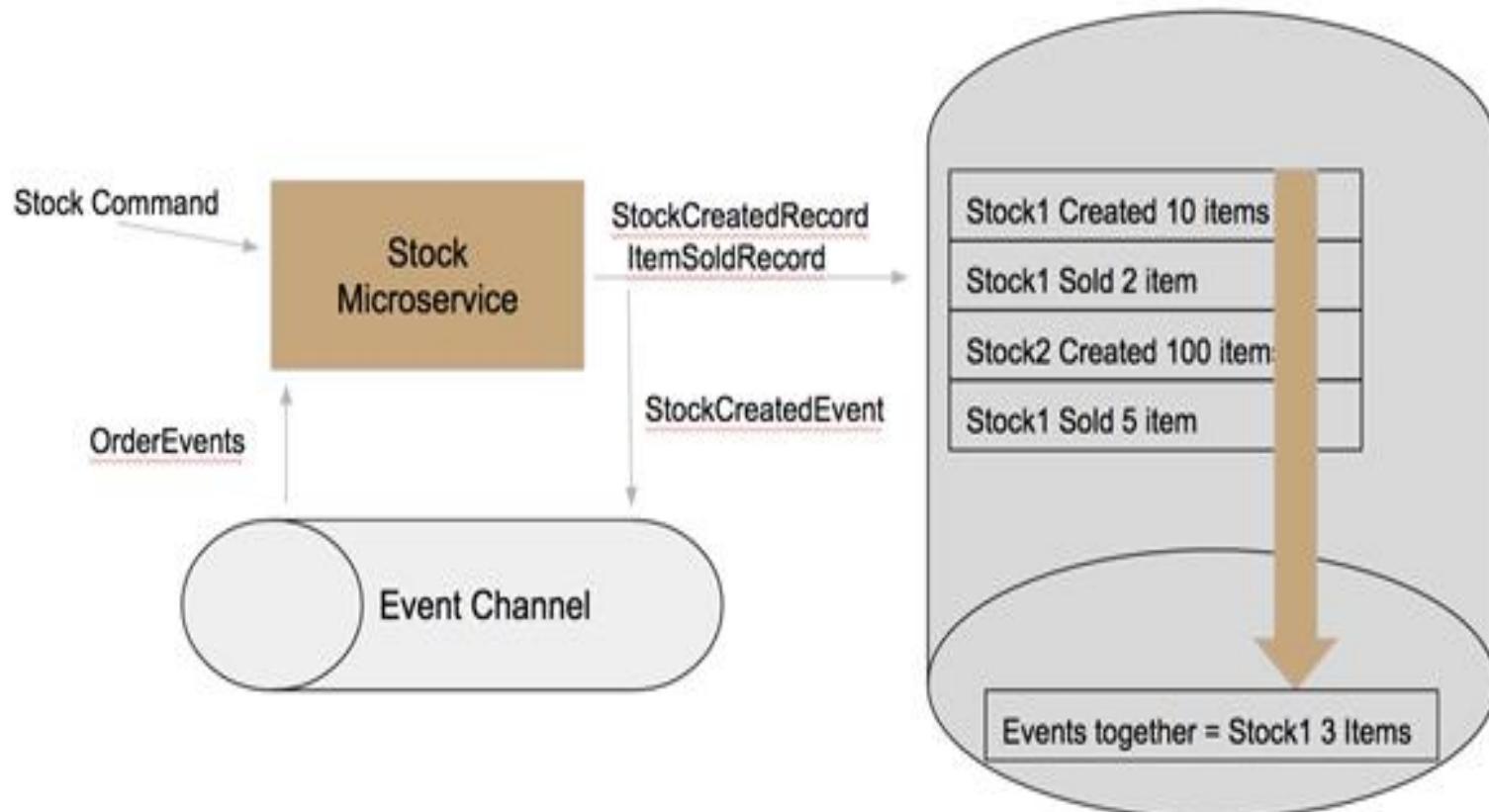
Eventual Consistent

- Inconsistent for a while
- CQRS software pattern
- Asynchronous processes
- No Transactional dependency



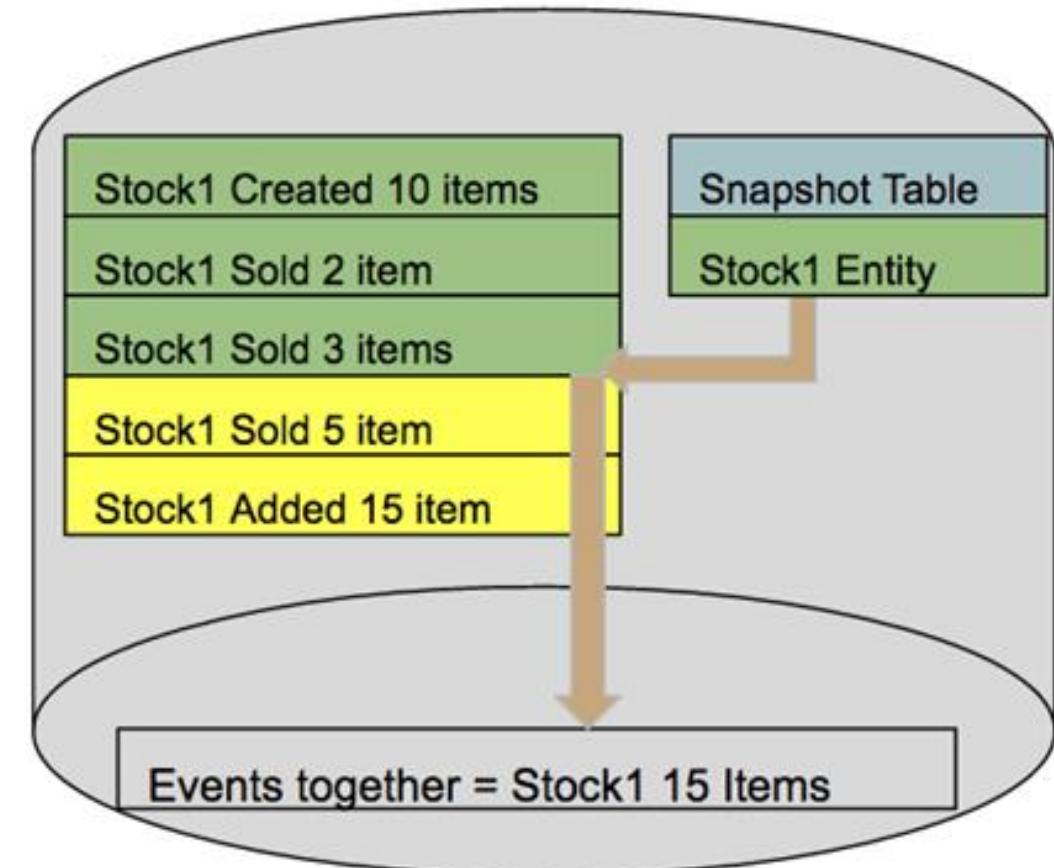
Event Sourcing

- Accumulating events
- Assets are not recorded
- Events Recording
- Generating state from the events



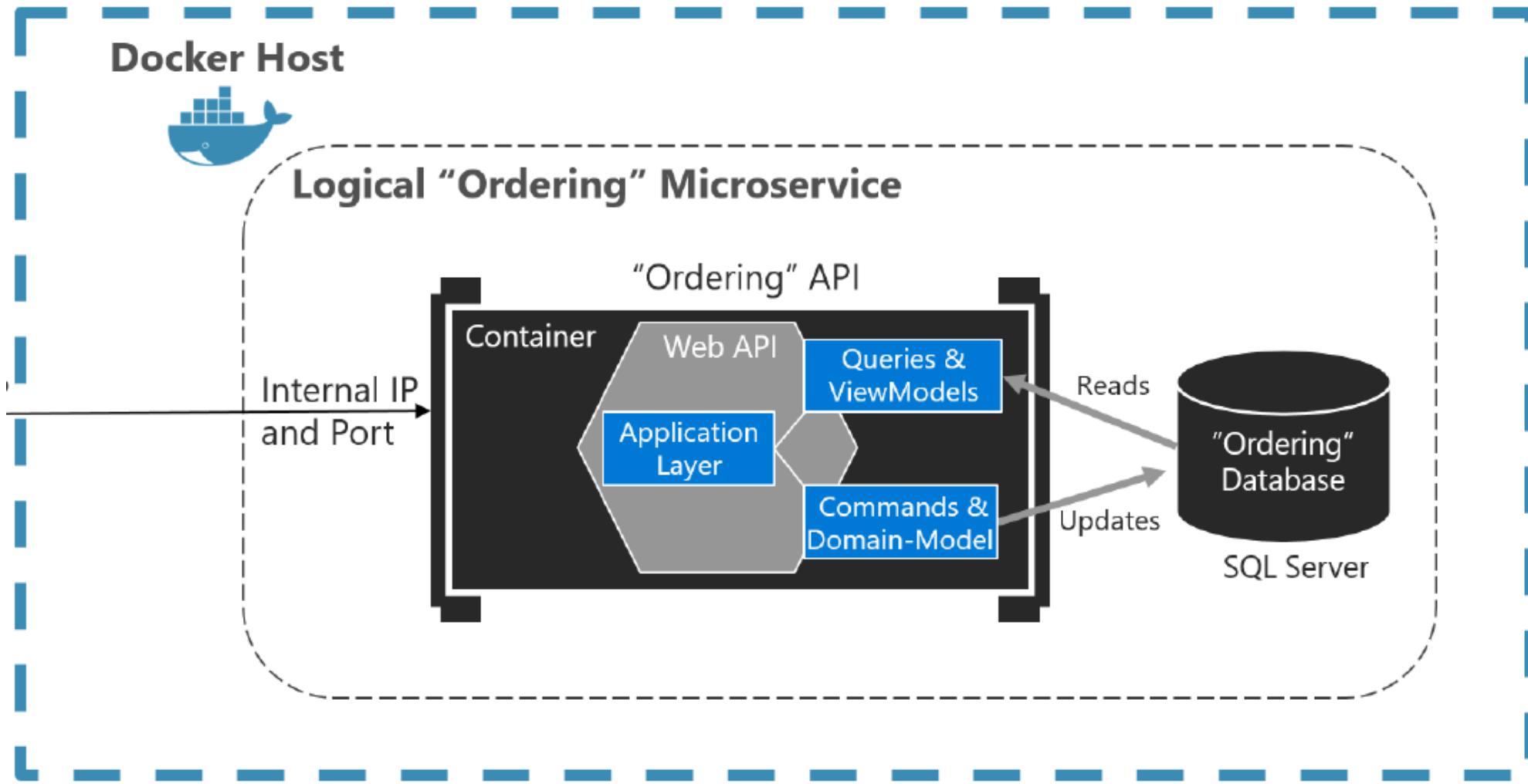
CQRS and Event Sourcing

- Writing models on the CQRS Design Pattern
- Final state of our existence
- Event information generated as a result
- Triggered and our reading model



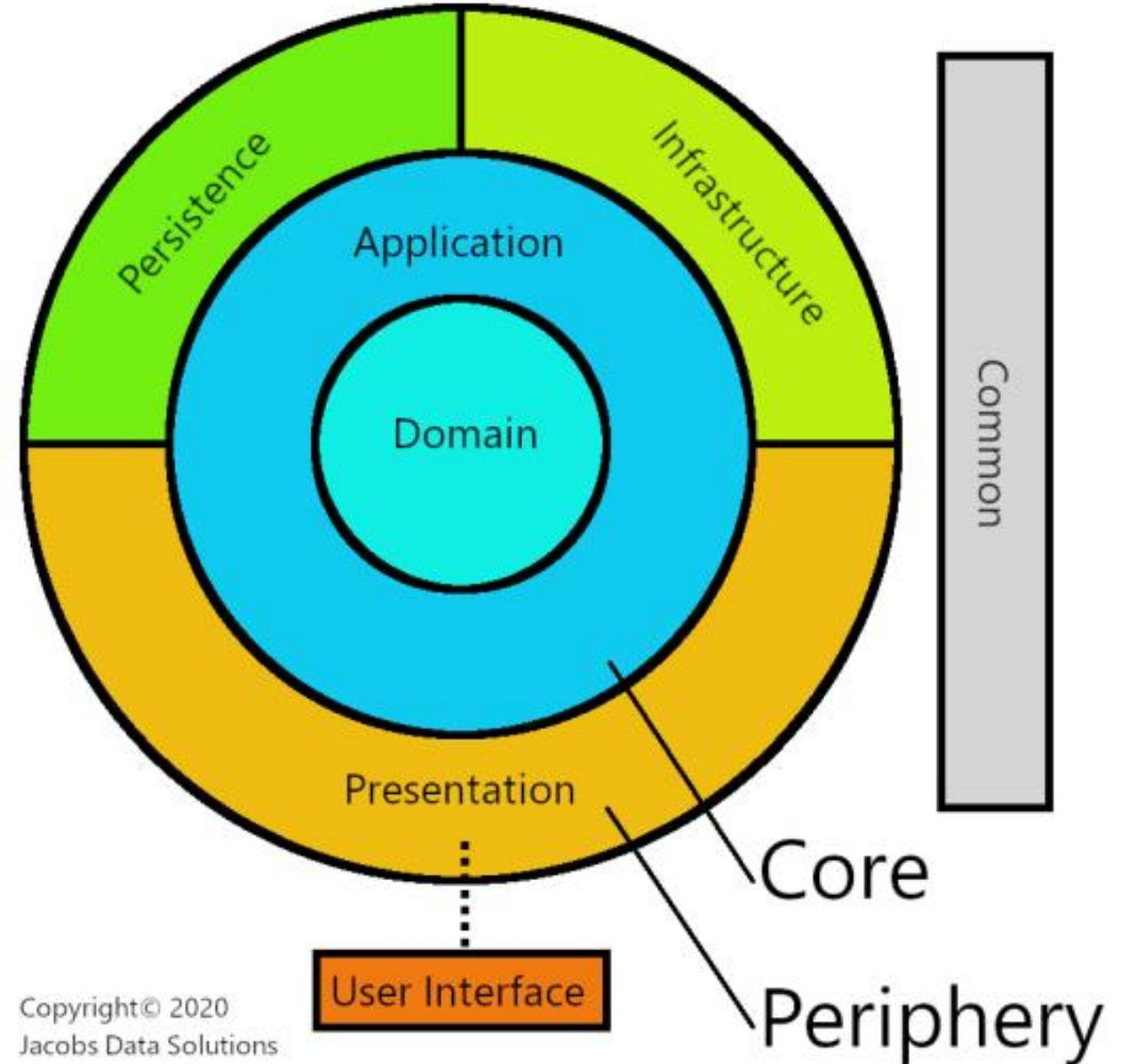
Simplified CQRS and DDD microservice

High level design

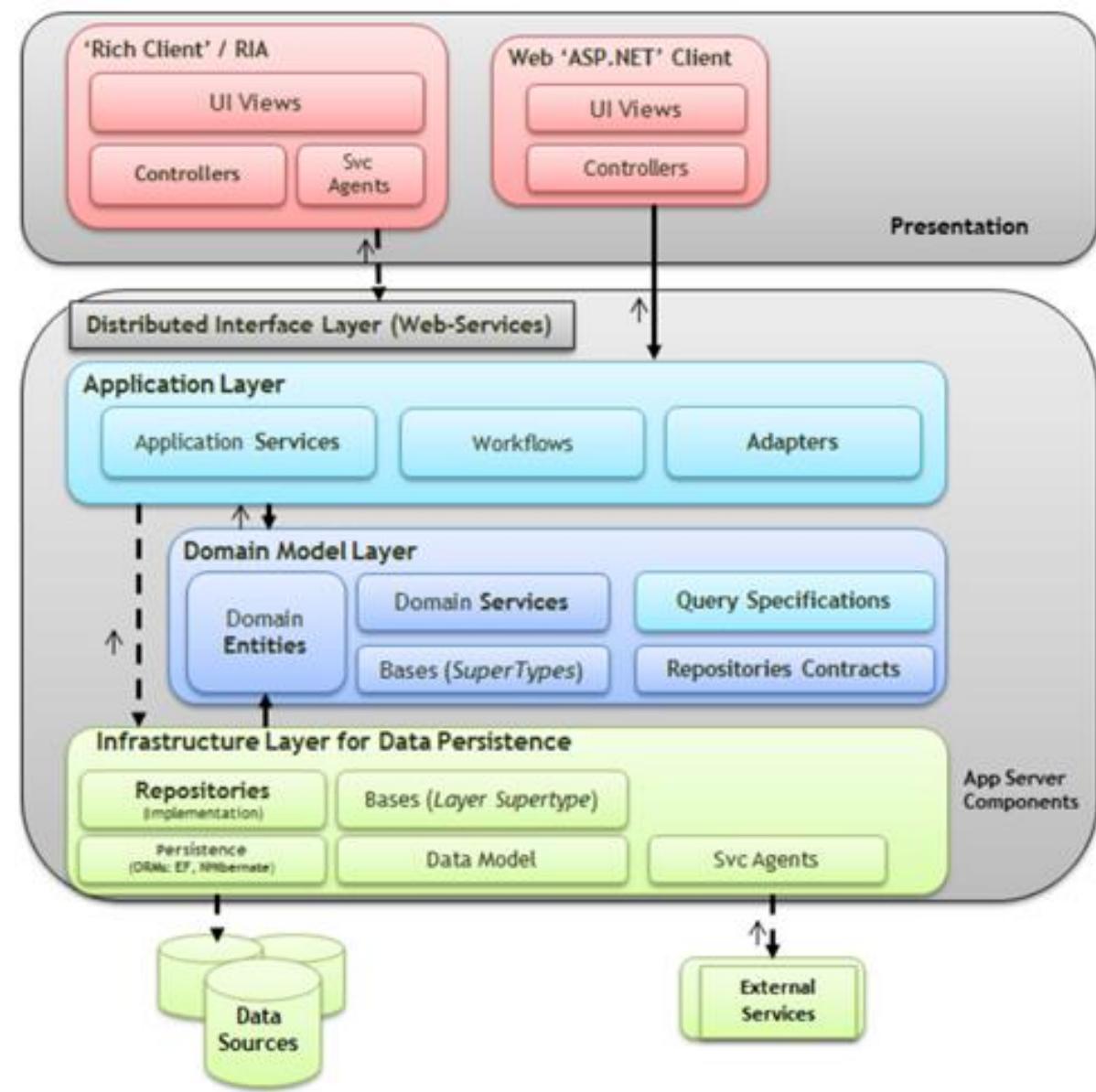
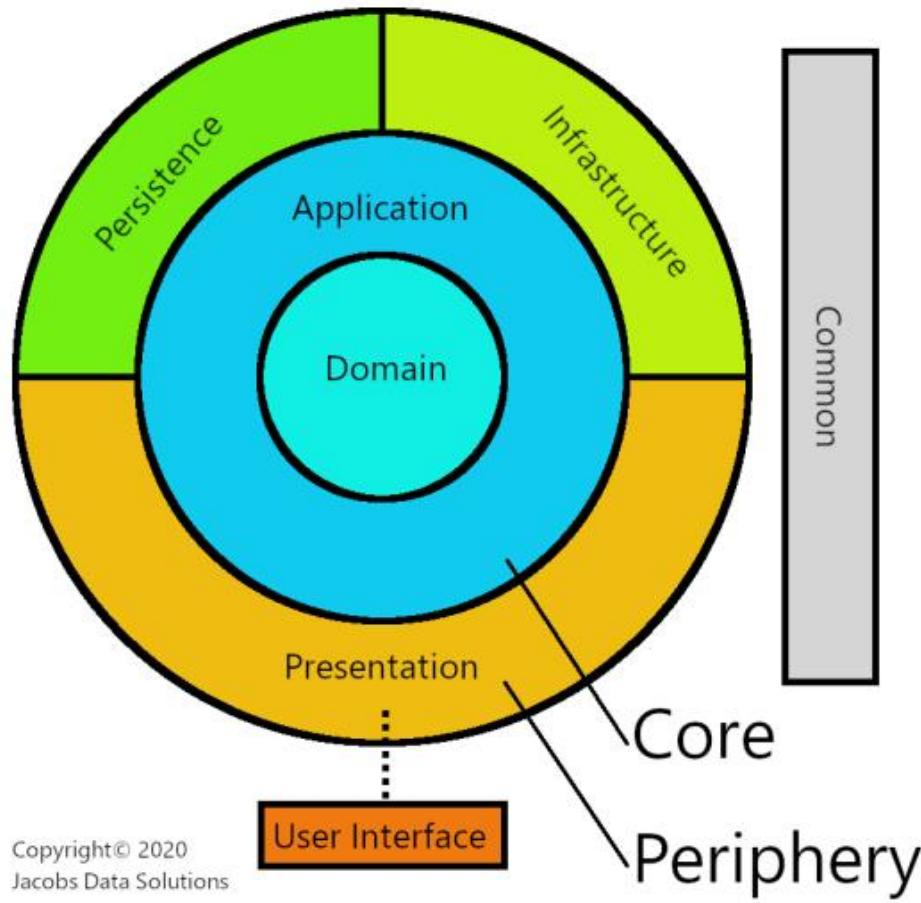


Code Structure of Ordering Microservices

- Ordering.Domain Layer
- Ordering.Application Layer
- Ordering.API Layer
- Ordering.Infrastructure Layer



Ordering Layers



Ordering.Domain Nuget Packages

- Ordering.Domain – No Dependency

Ordering. Application Nuget Packages

- MediatR.Extensions.Microsoft.DependencyInjection
- FluentValidation
- FluentValidation.DependencyInjectionExtensions
- AutoMapper
- AutoMapper.Extensions.Microsoft.DependencyInjection
- Microsoft.Extensions.Logging.Abstractions
- **ProjectReference - Ordering.Domain**

Ordering. Infrastructure Nuget Packages

- Microsoft.EntityFrameworkCore.SqlServer
- SendGrid
- **ProjectReference - Ordering. Application**

Ordering.API Nuget Packages

- Microsoft.EntityFrameworkCore.Tools
- **ProjectReference - Ordering. Application**
- **ProjectReference - Ordering. Infrastructure**



Jason Taylor
jasontaylordev

[Unfollow](#)

...

SSW Solution Architect

<https://github.com/jasontaylordev/CleanArchitecture>

<https://www.youtube.com/watch?app=desktop&v=5OtUm1BLmG0>



Gill Cleeren
GillCleeren

[Unfollow](#)

...

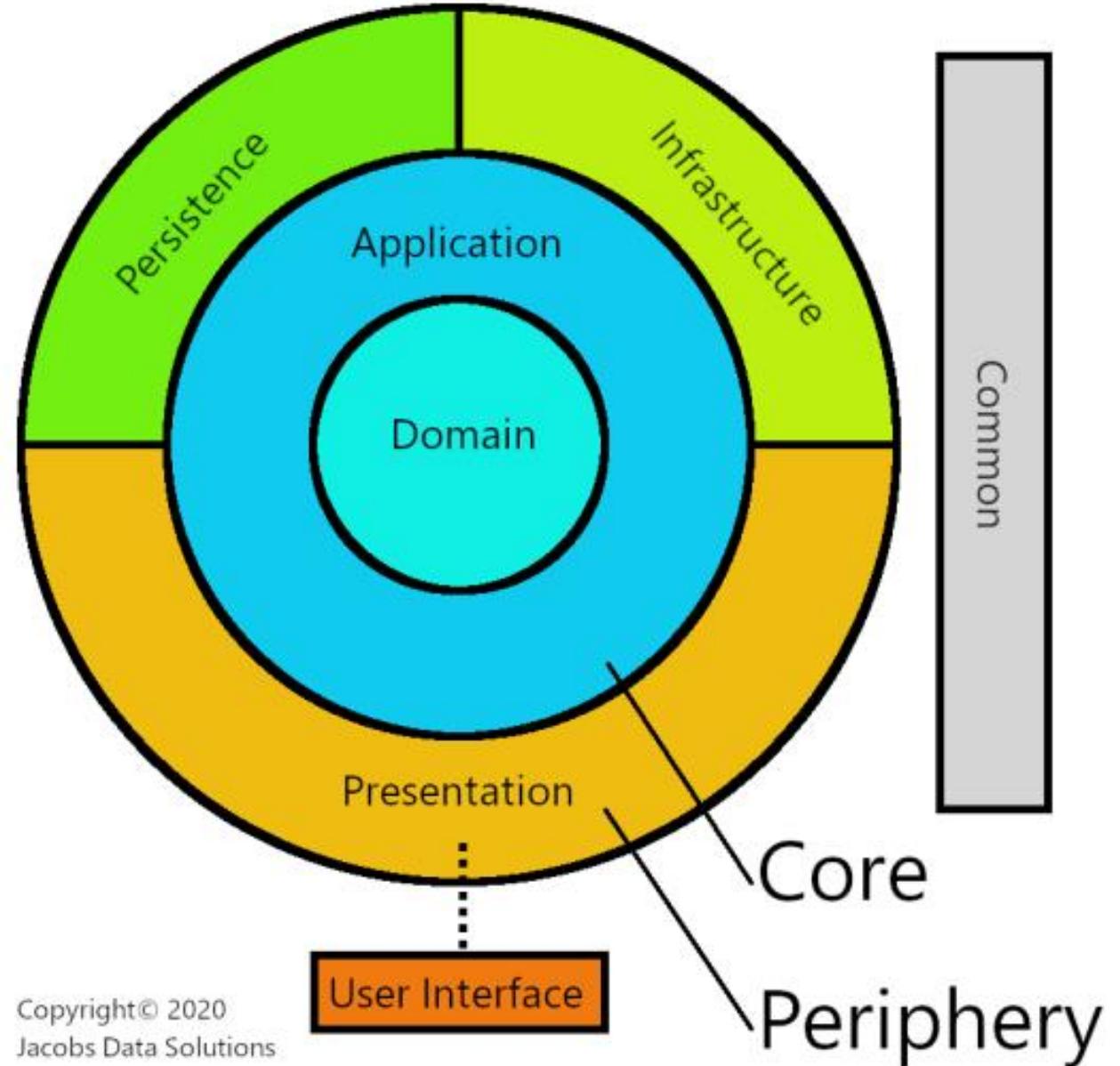
Gill focuses on web and mobile

<https://twitter.com/gillcleeren/status/1367863587163766>

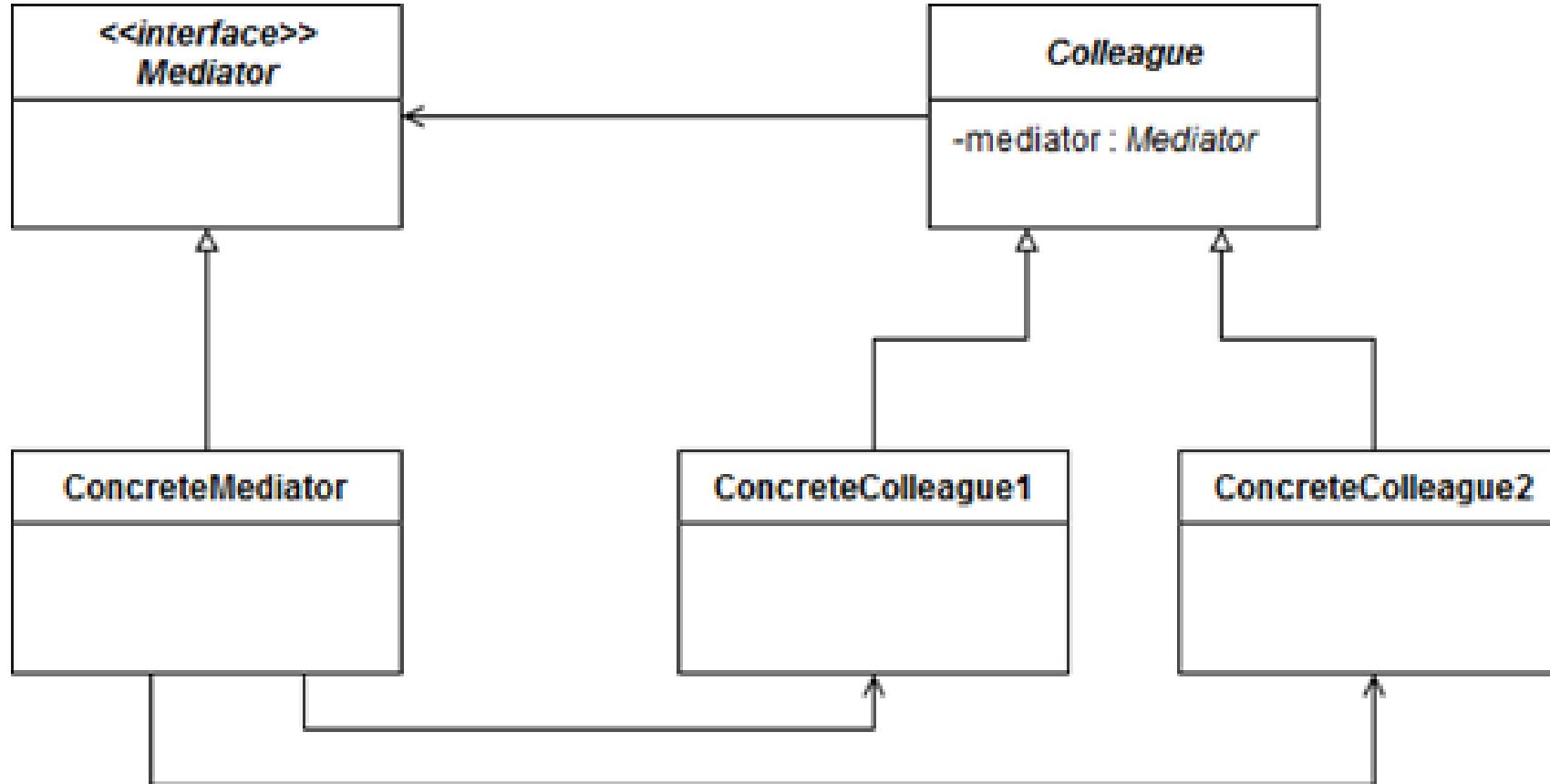
<https://www.youtube.com/watch?v=BxtHt7tsX-c>

Code Structure of Ordering Microservices

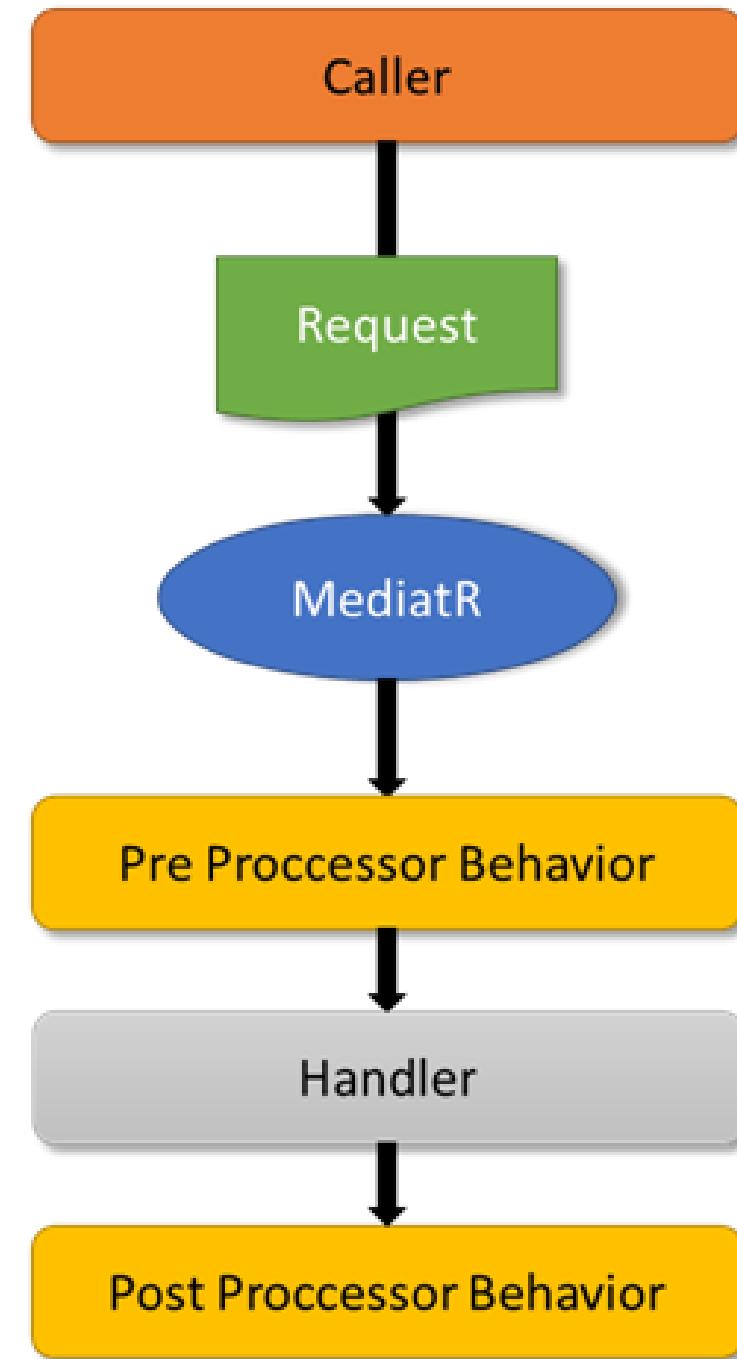
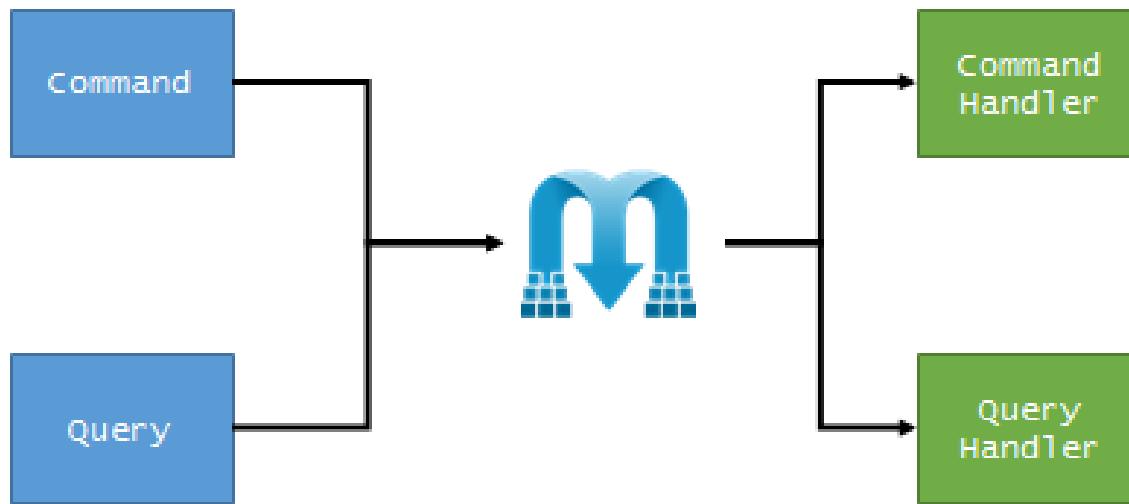
- Ordering.Domain Layer
- Ordering.Application Layer
- Ordering.API Layer
- Ordering.Infrastructure Layer



Mediator Design Pattern

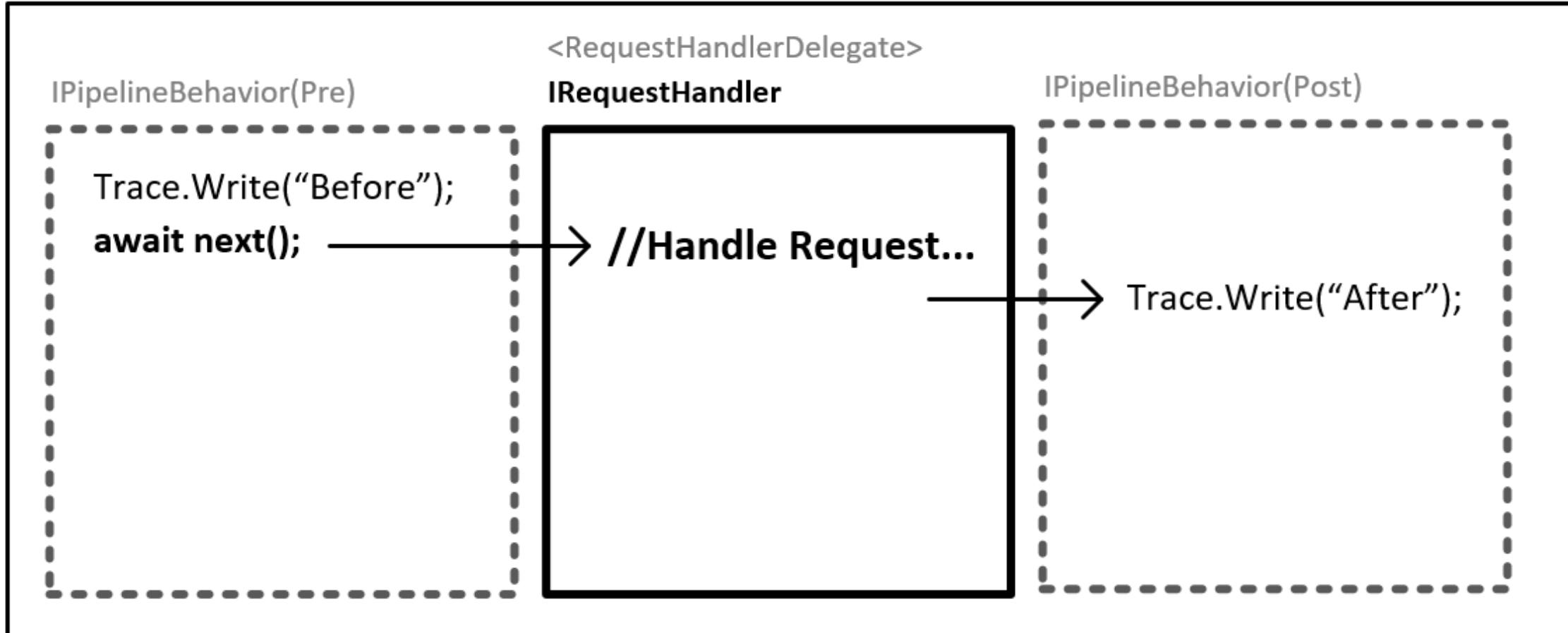


MediatR Nuget Package

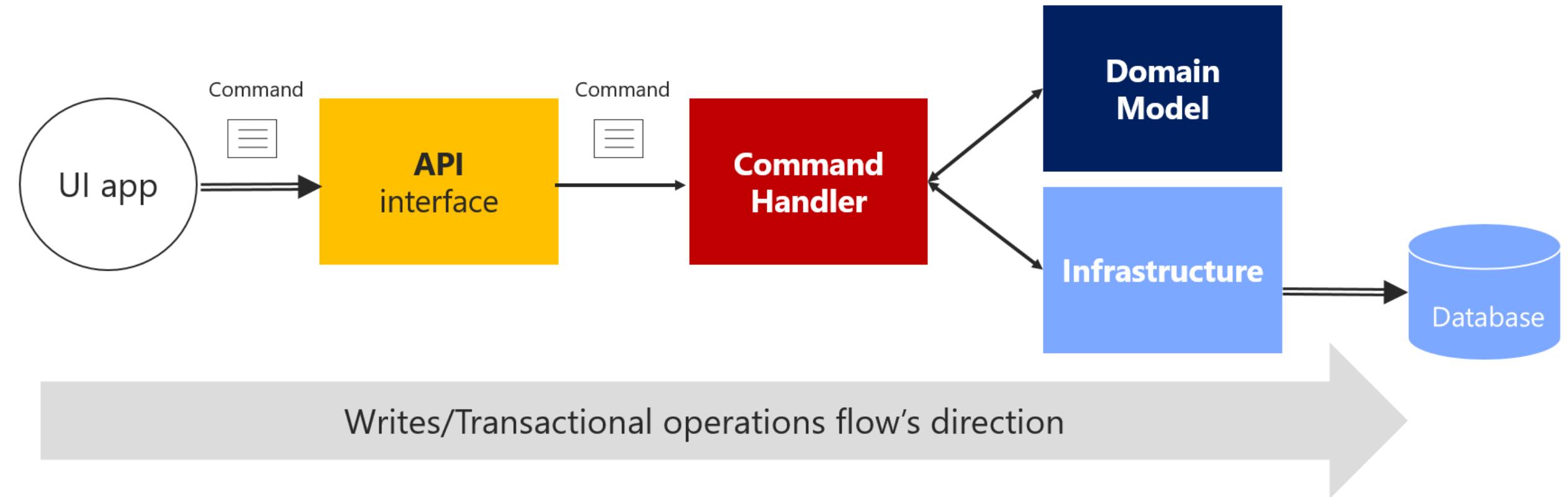


MediatR Pipeline Behavior

TracingBehavior



High level “Writes-side” in CQRS



Section 8

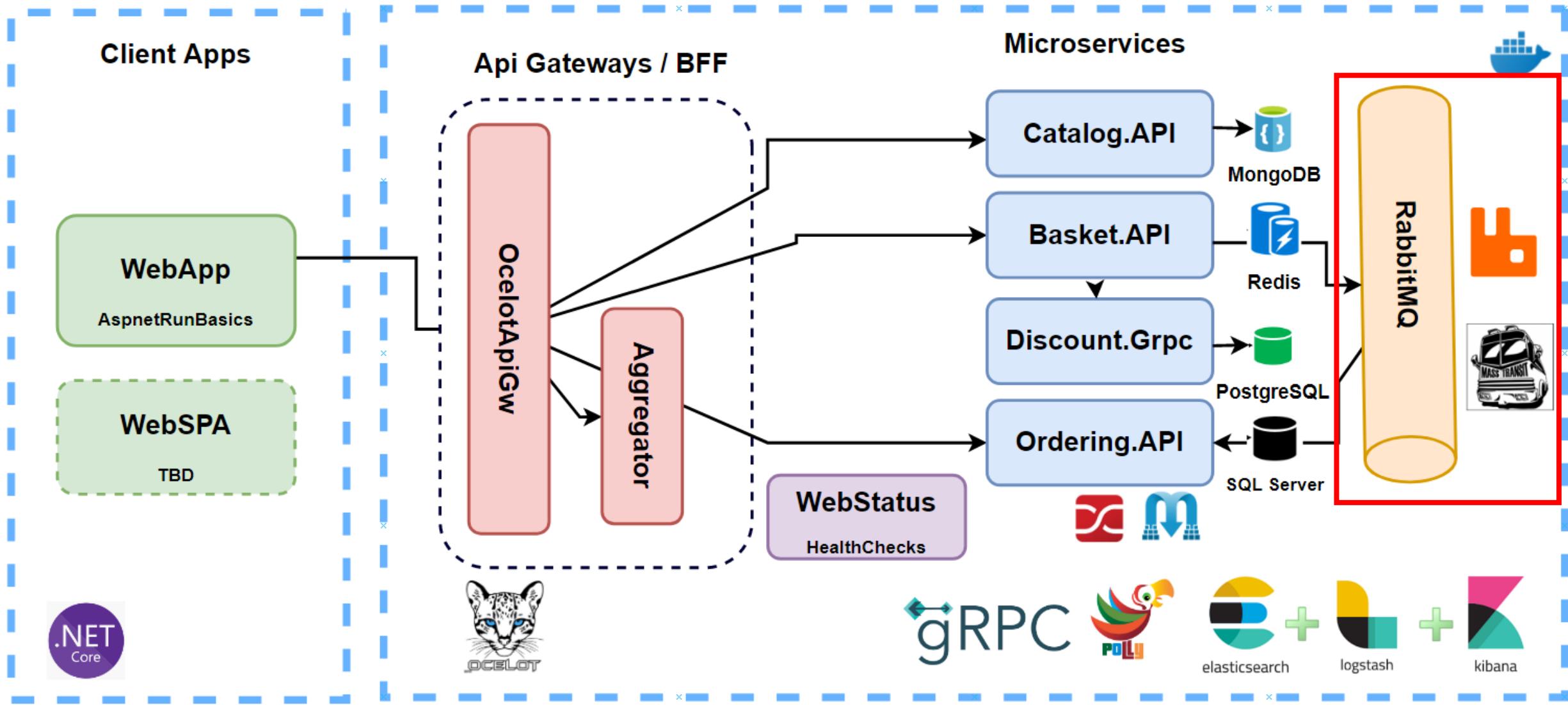
Microservices Async

Communication w/ RabbitMQ &

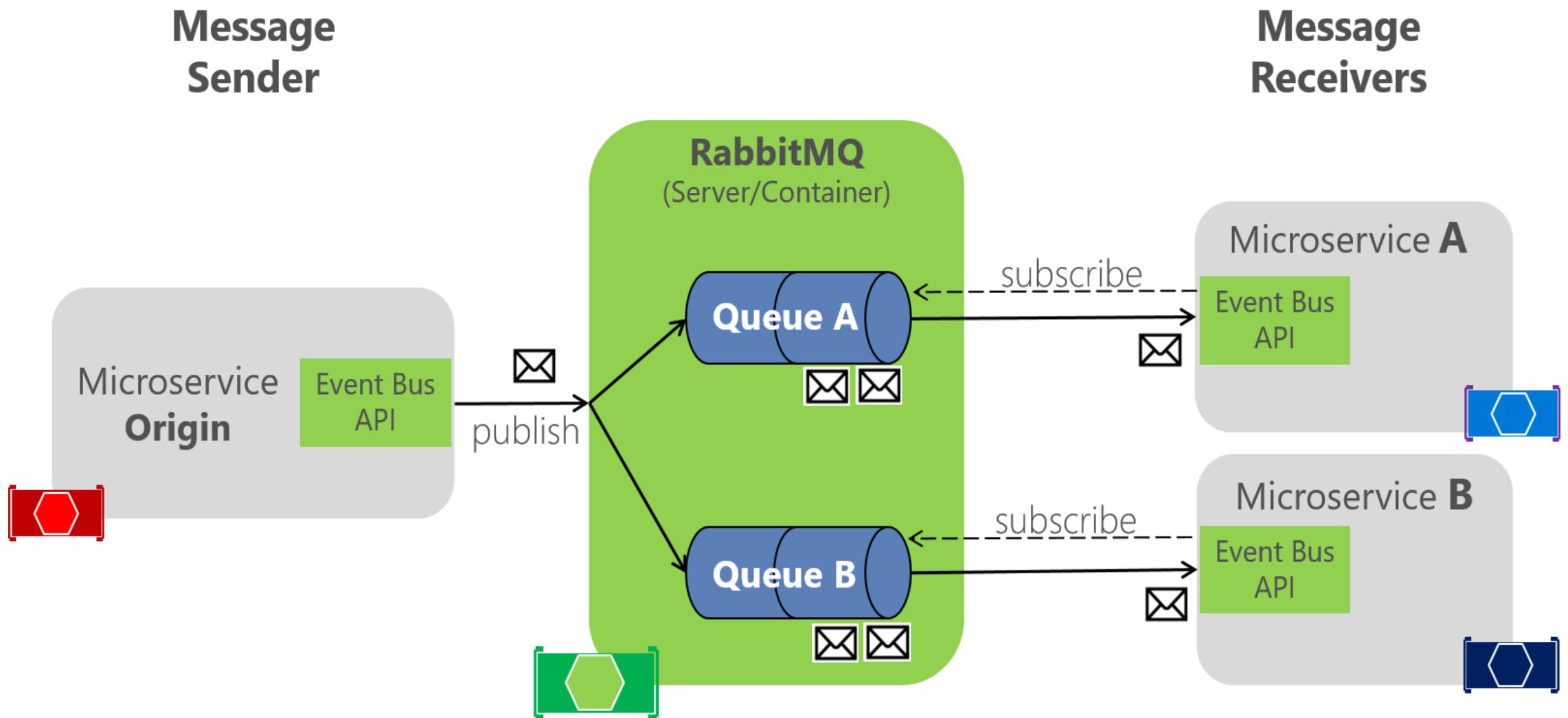
MassTransit

for Checkout Order Between Basket-Ordering Microservices

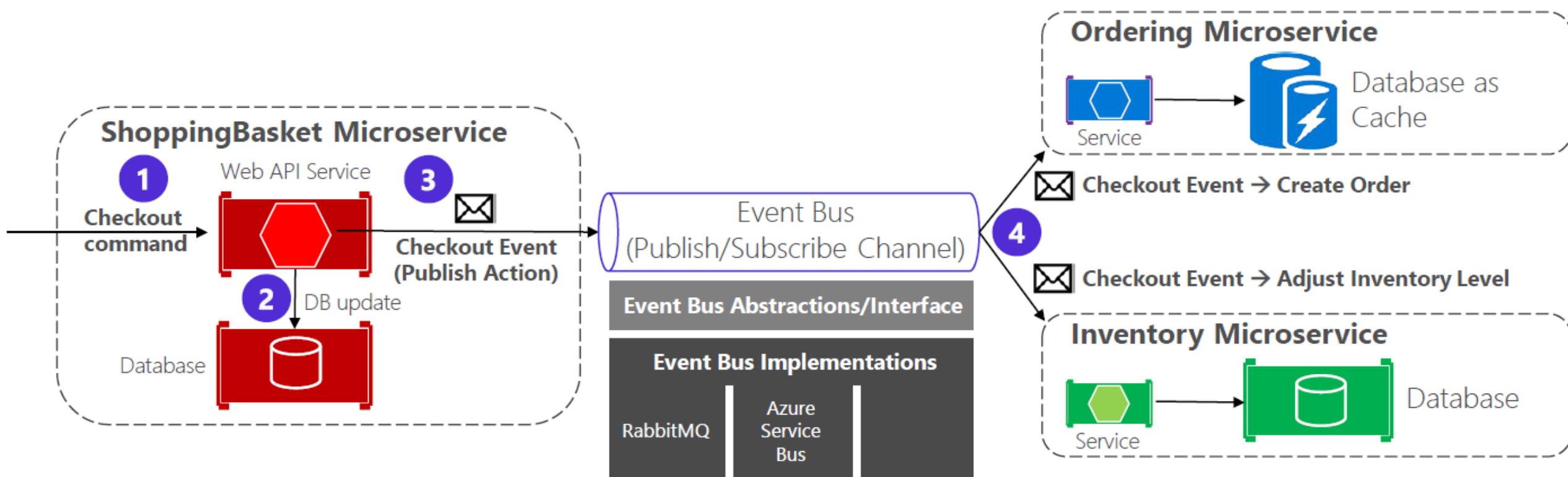
Big Picture



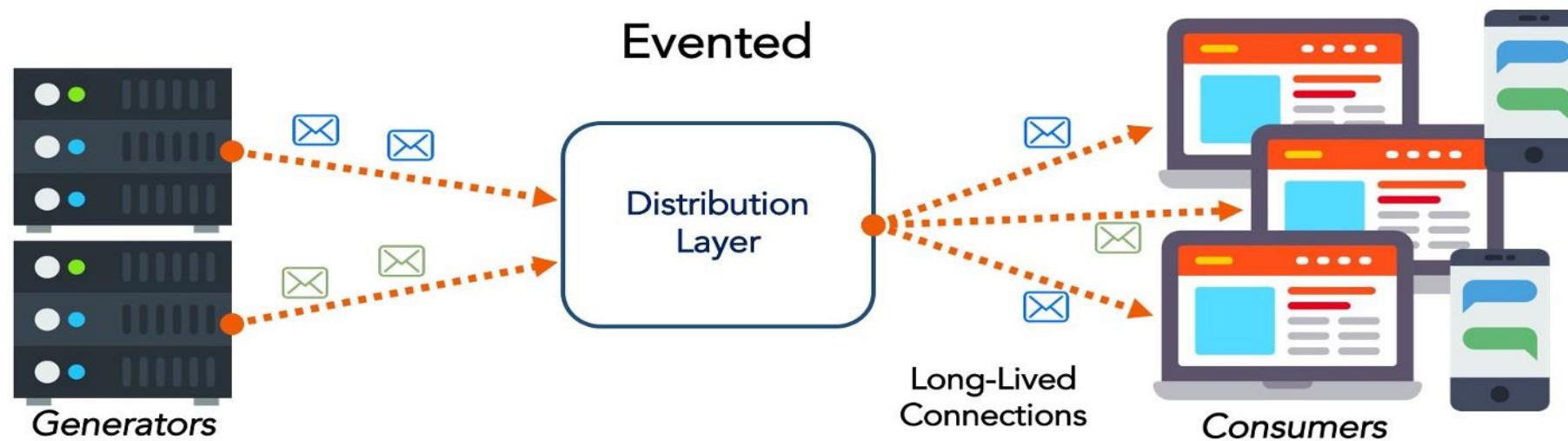
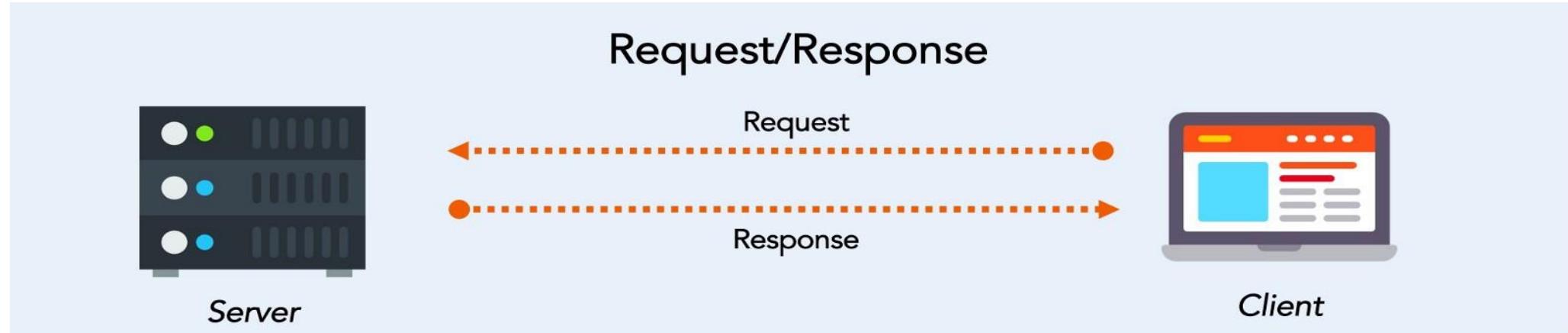
Pub/Sub RabbitMQ Architecture



Publisher/Subscriber of BasketCheckout Event w/ Basket and Ordering Microservices



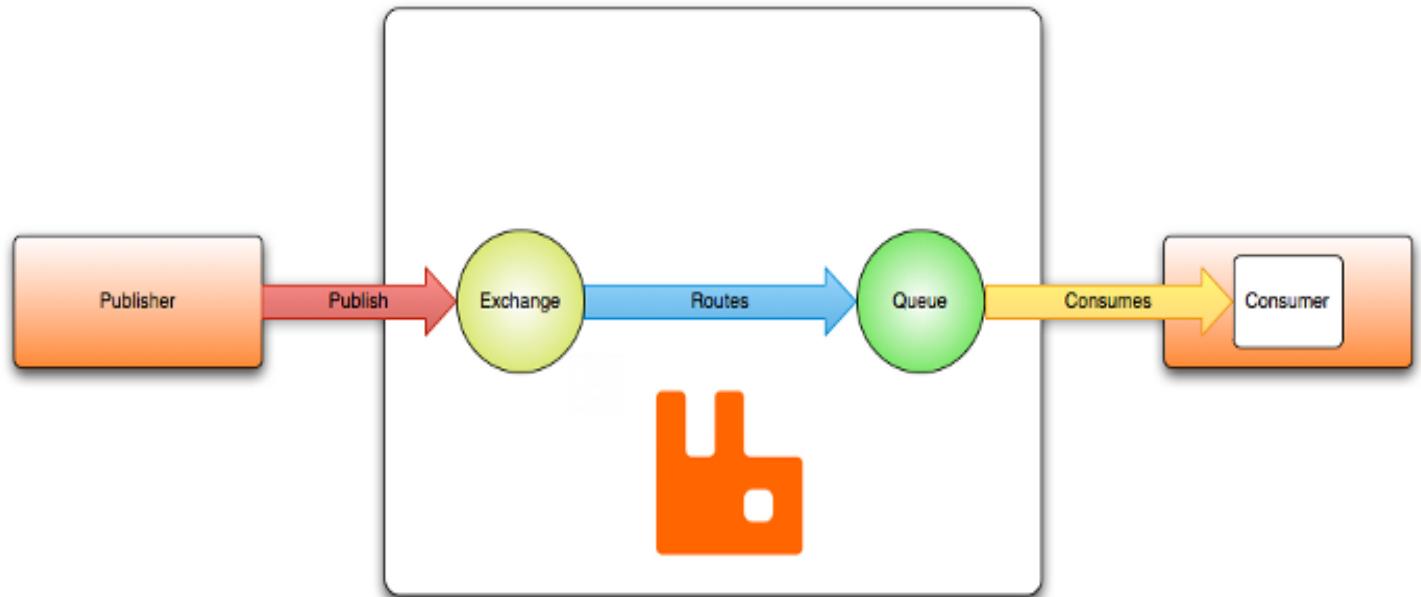
Microservices Communication Types



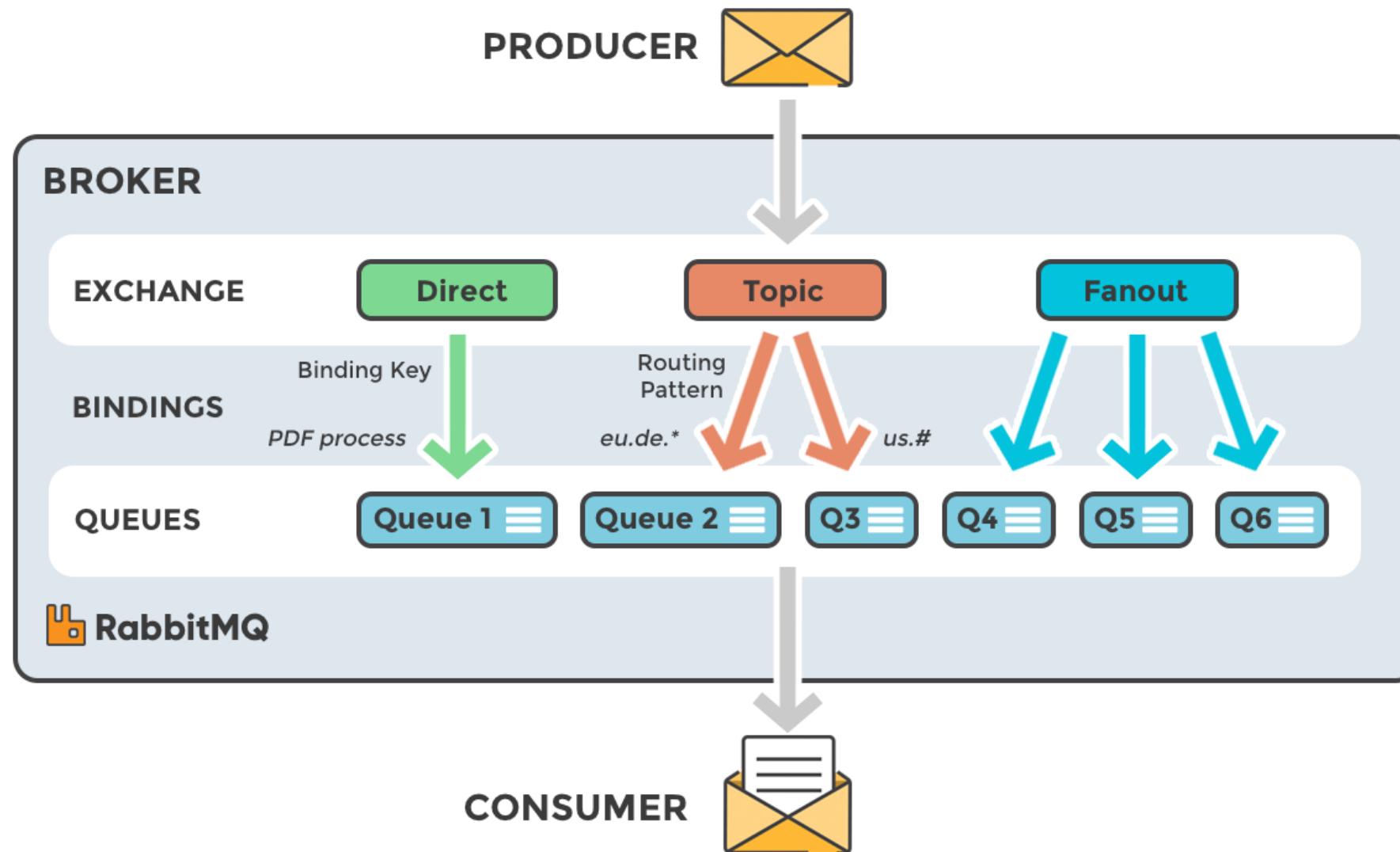
What is RabbitMQ ?

- Message Queue System
- Event-Driven Architecture
- Apache Kafka, Msmq, Microsoft Azure Service Bus, Kestrel, ActiveMQ

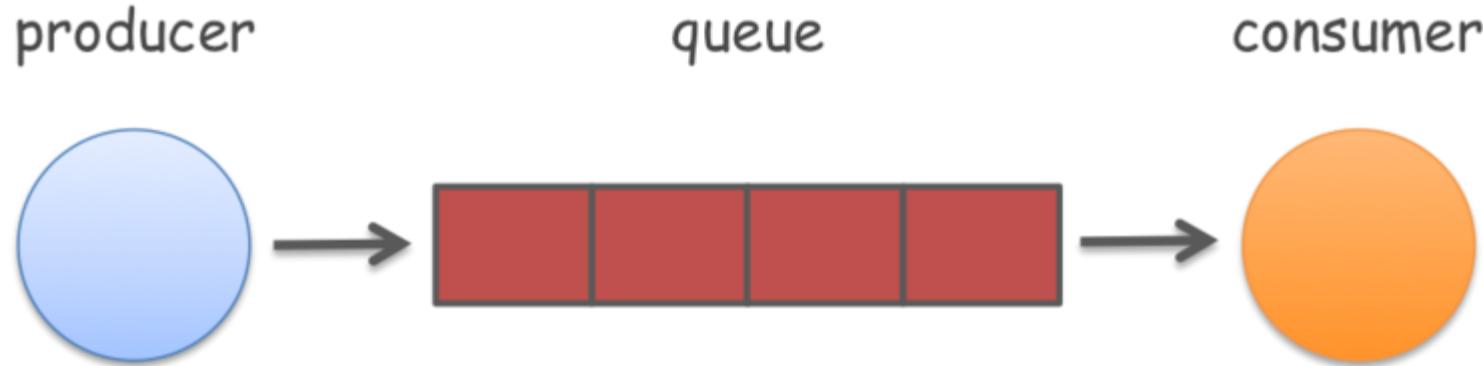
"Hello, world" example routing



Main Components of RabbitMQ

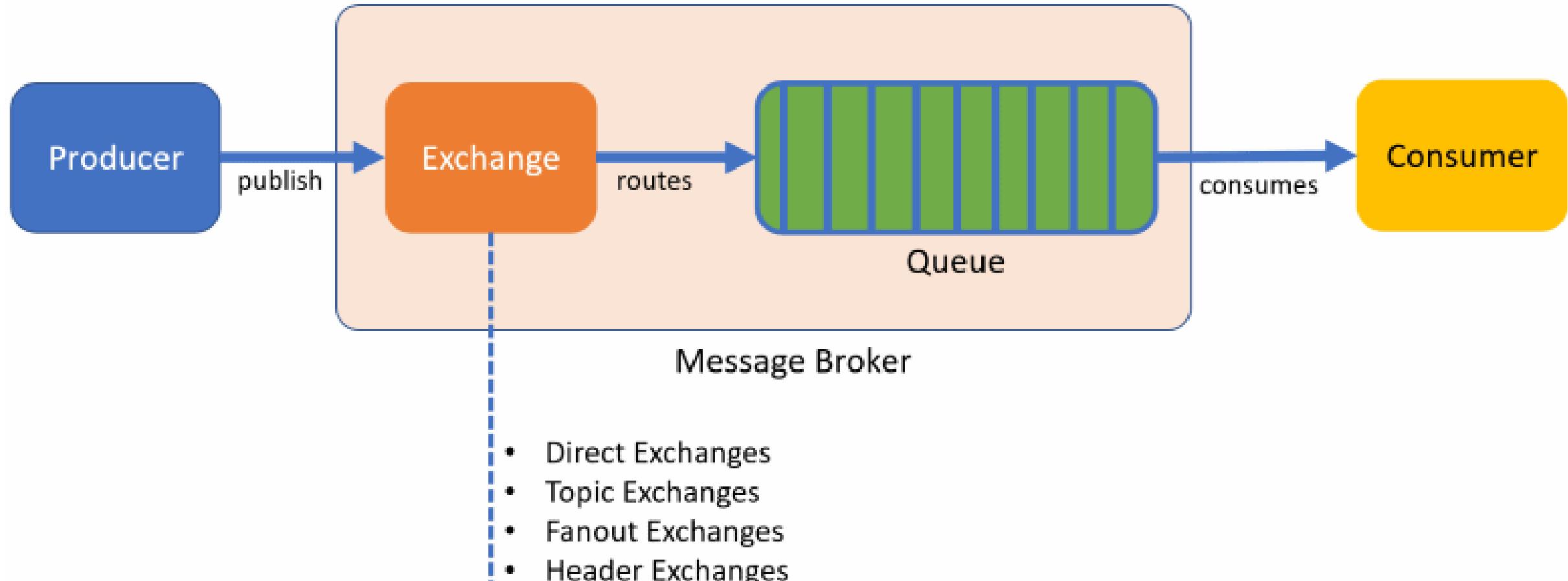


RabbitMQ Queue Properties

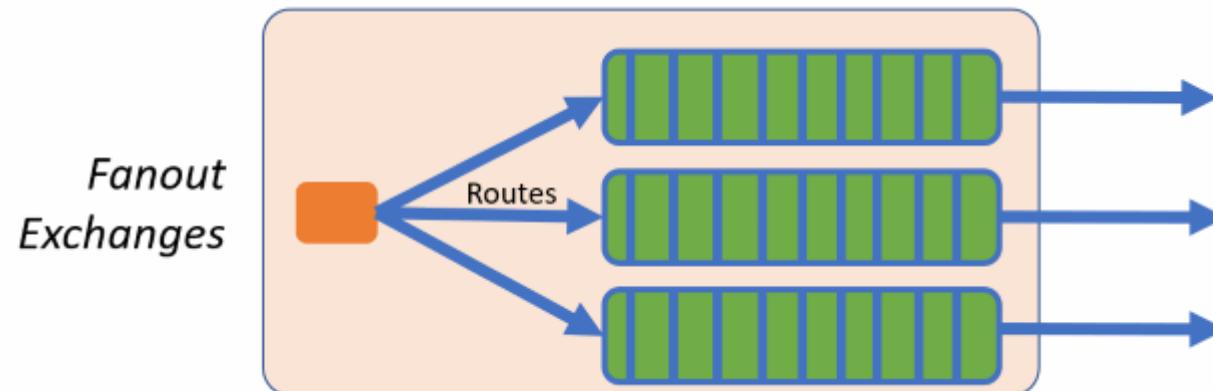
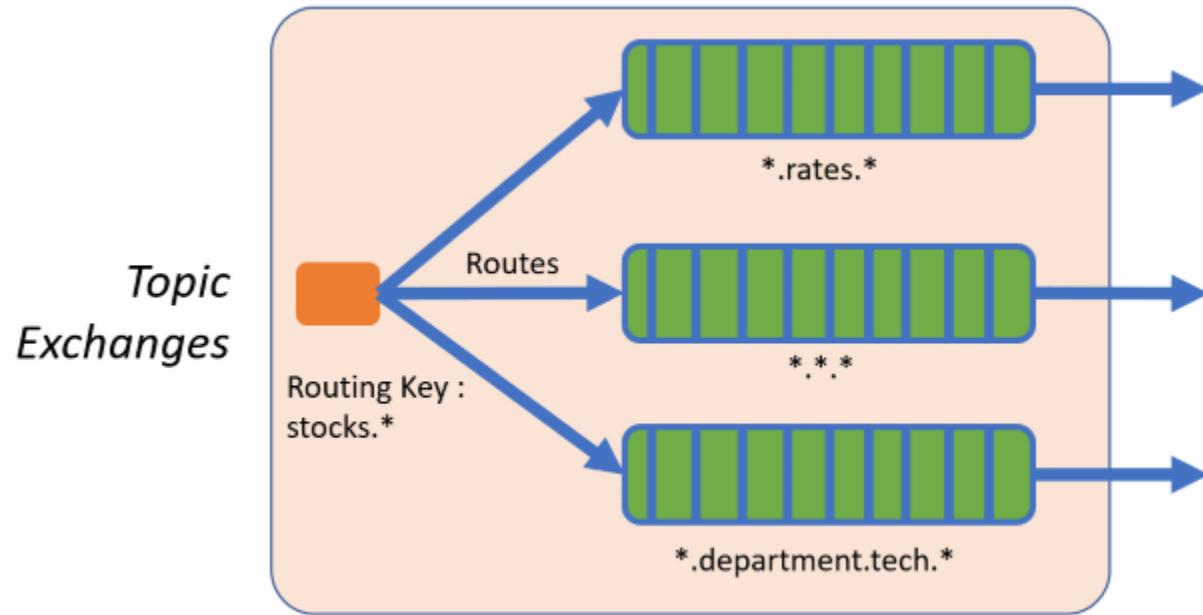


- Name
- Durable
- Exclusive
- AutoDelete

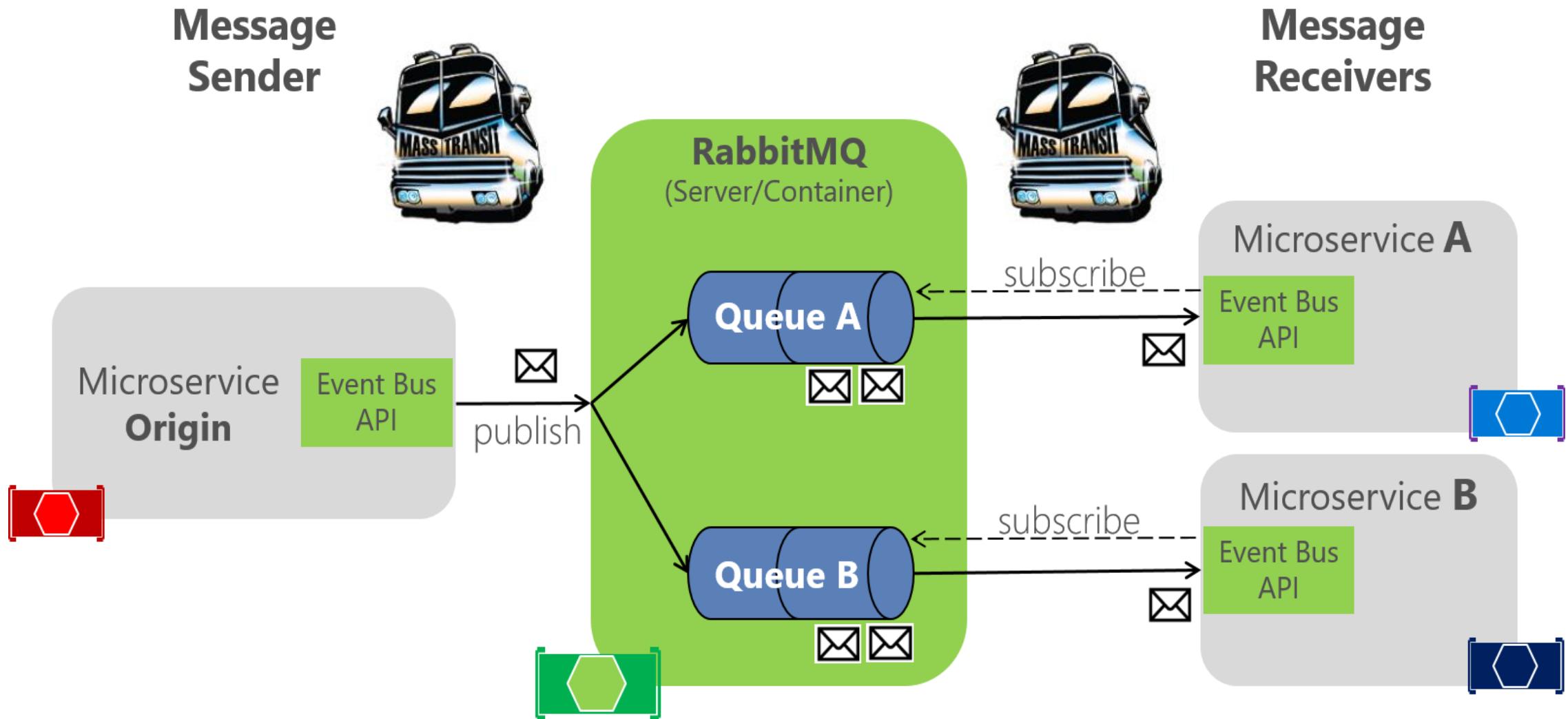
RabbitMQ Exchange Types



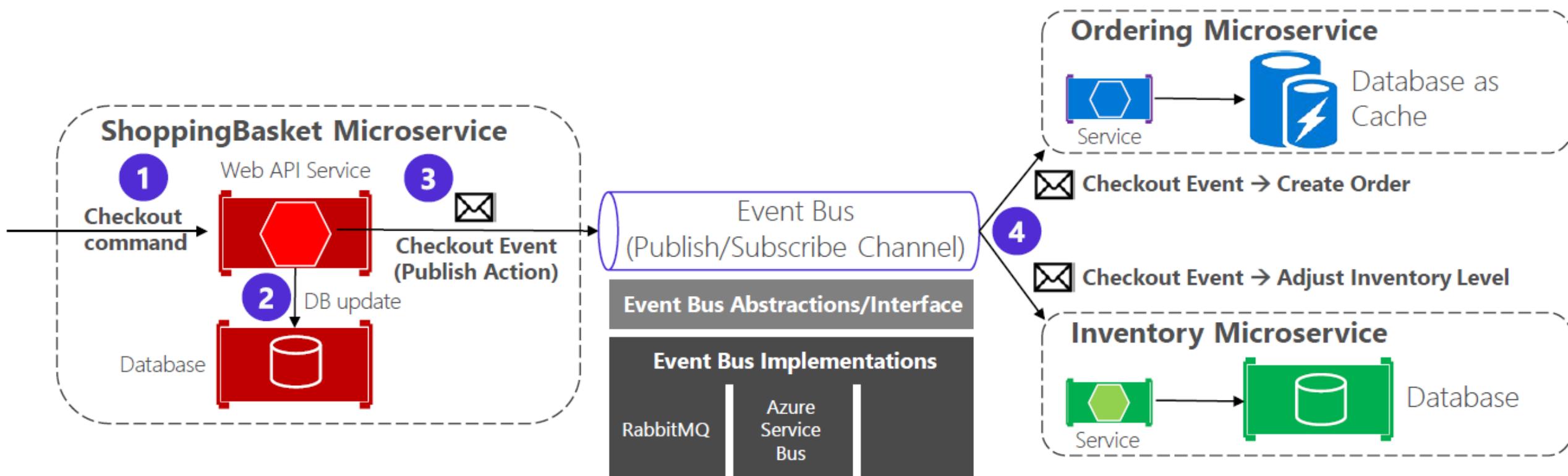
RabbitMQ Topic & Fanout Exchange Types



Analysis & Design RabbitMQ & BuildingBlocks EventBus.Messages



Publisher/Subscriber of BasketCheckout Event



RabbitMQ Nuget Packages

- Install-Package MassTransit
- Install-Package MassTransit.RabbitMQ
- Install-Package MassTransit.AspNetCore
- REST API principles, CRUD operations
- **Add Project Reference - EventBus.Messages**



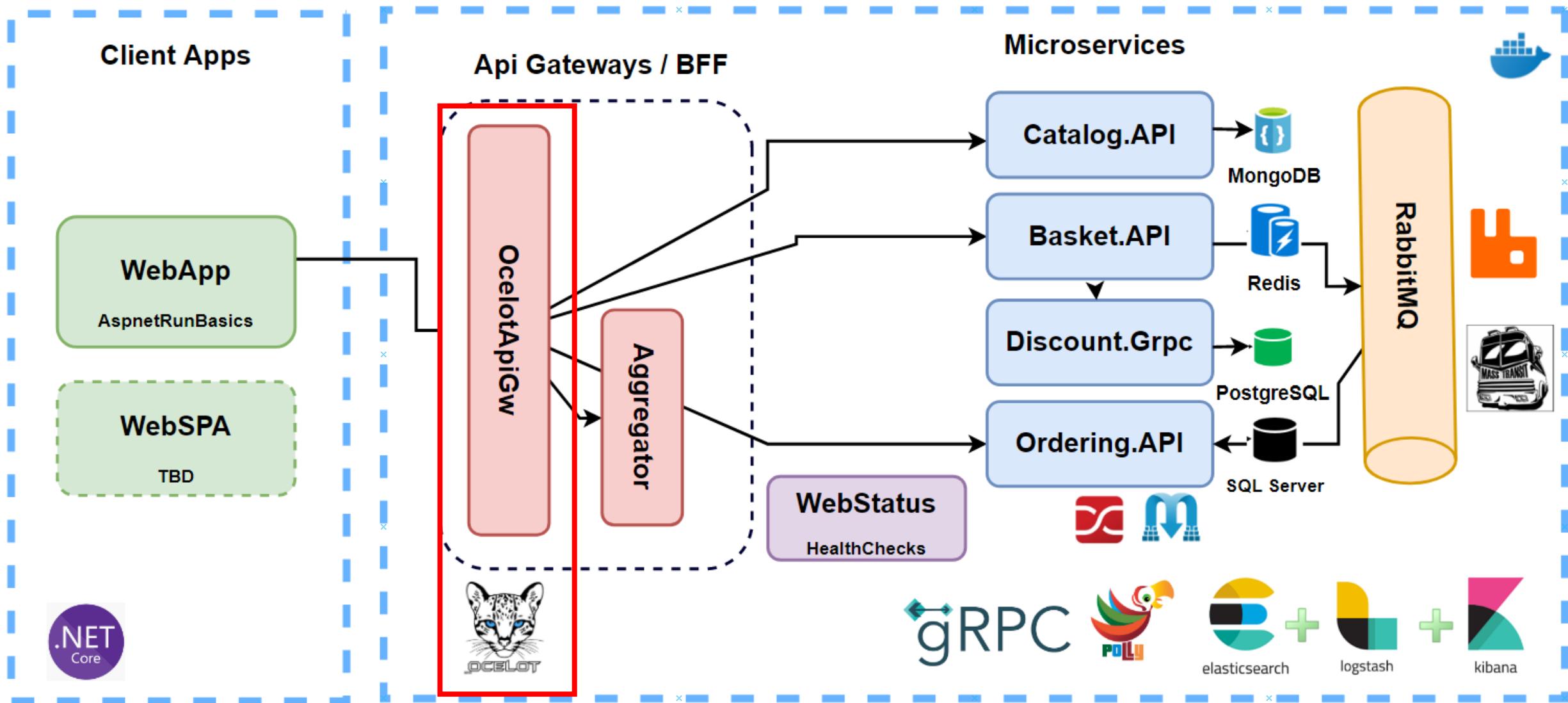
Section 9

Building API Gateways with

Ocelot

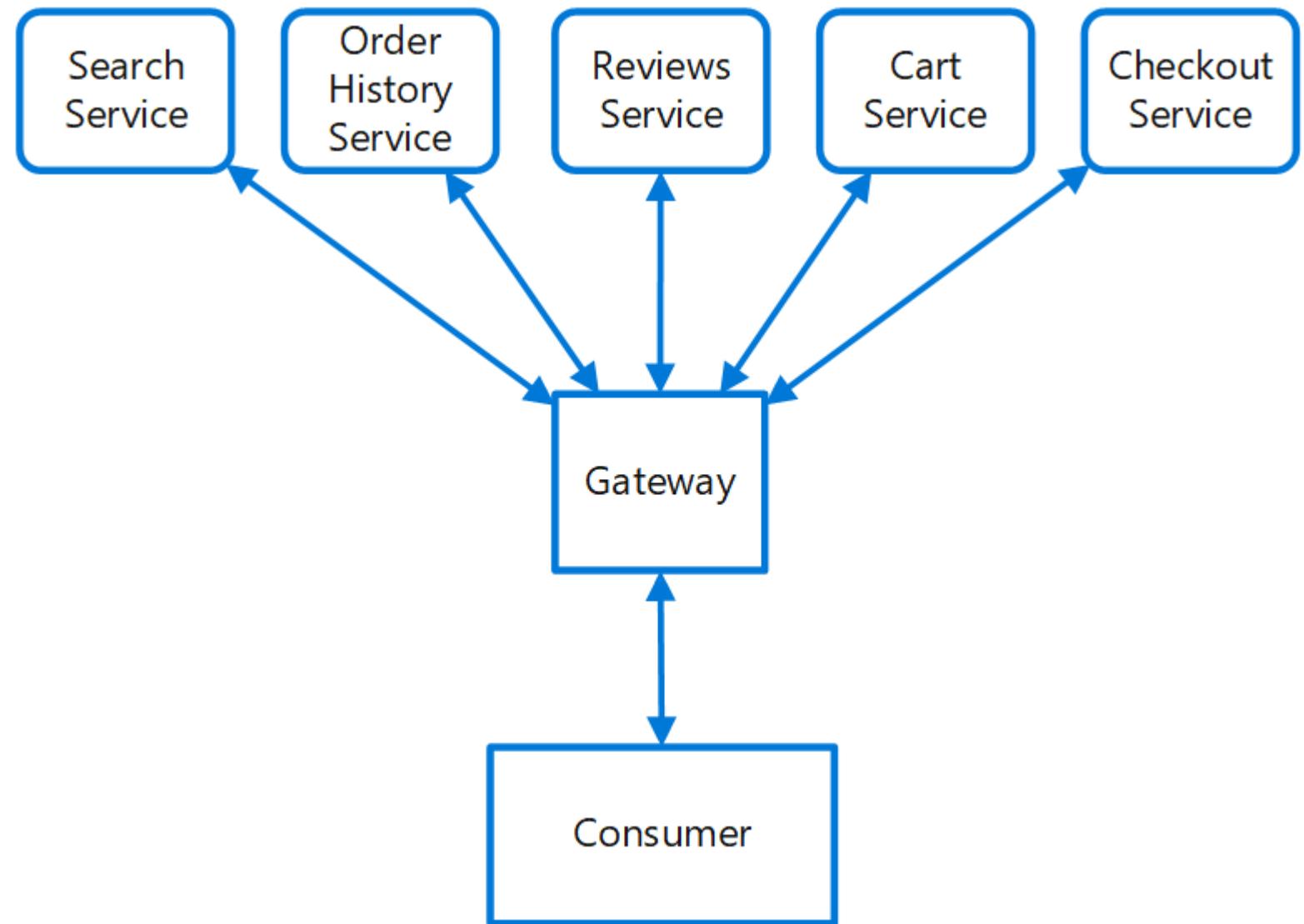
Applying Gateway Routing Pattern

Big Picture



Gateway Routing pattern

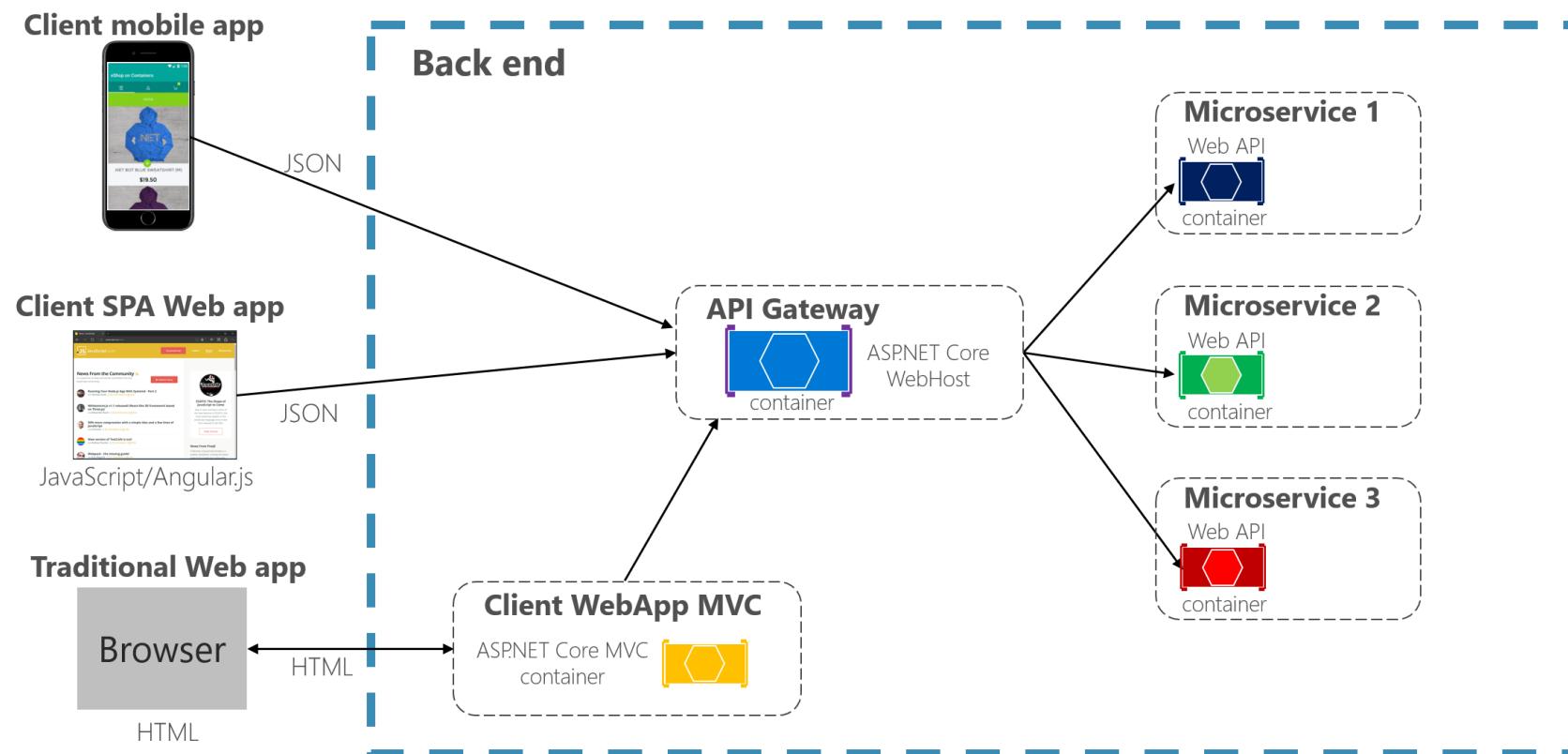
- Routing
- Data Aggregator
- Protocol Abstraction
- Centralized Error Management



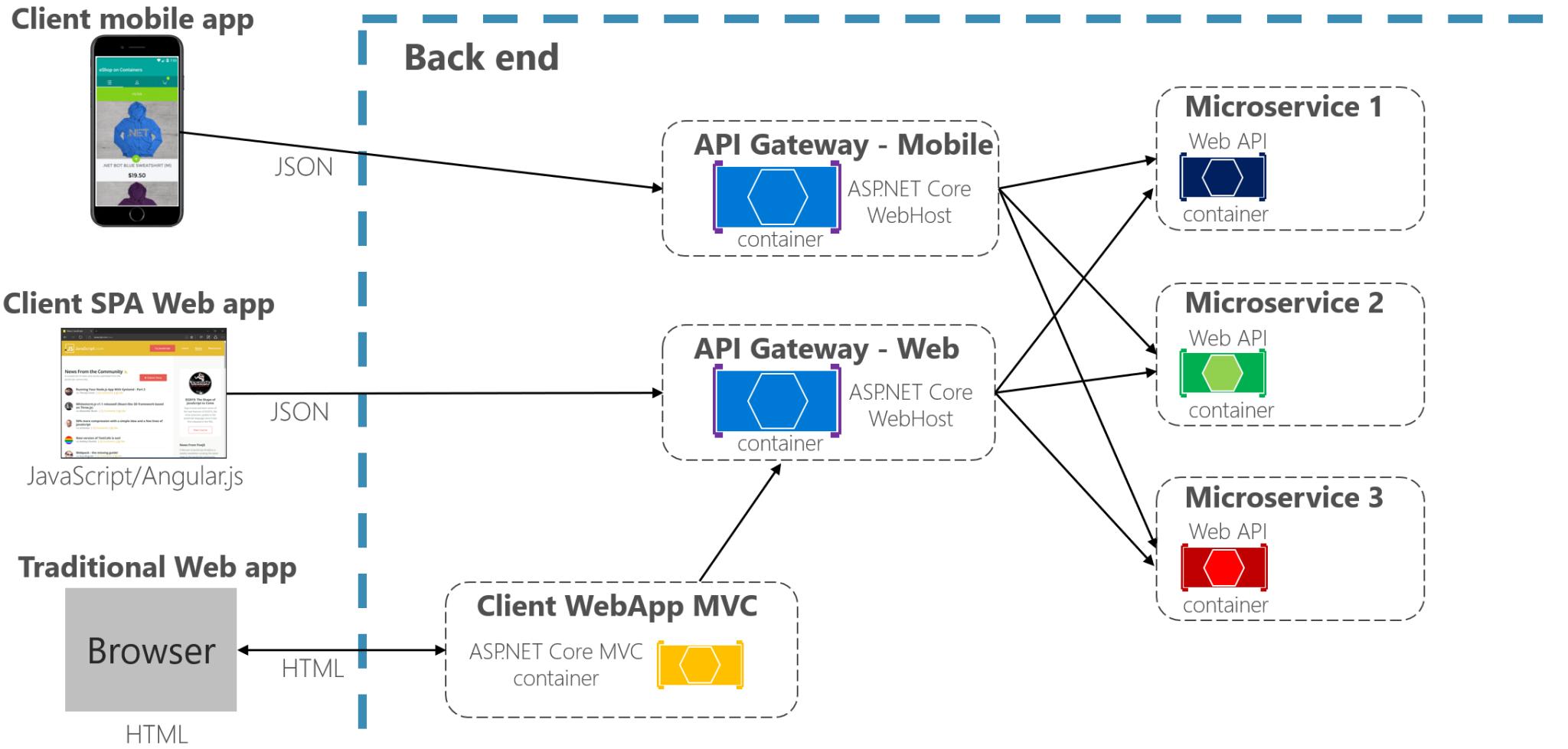
API Gateway Pattern

- Reverse proxy
- Gateway routing
- Requests aggregation
- Cross-cutting concerns or gateway offloading

Using a single custom **API Gateway service**



Using multiple API Gateways / BFF

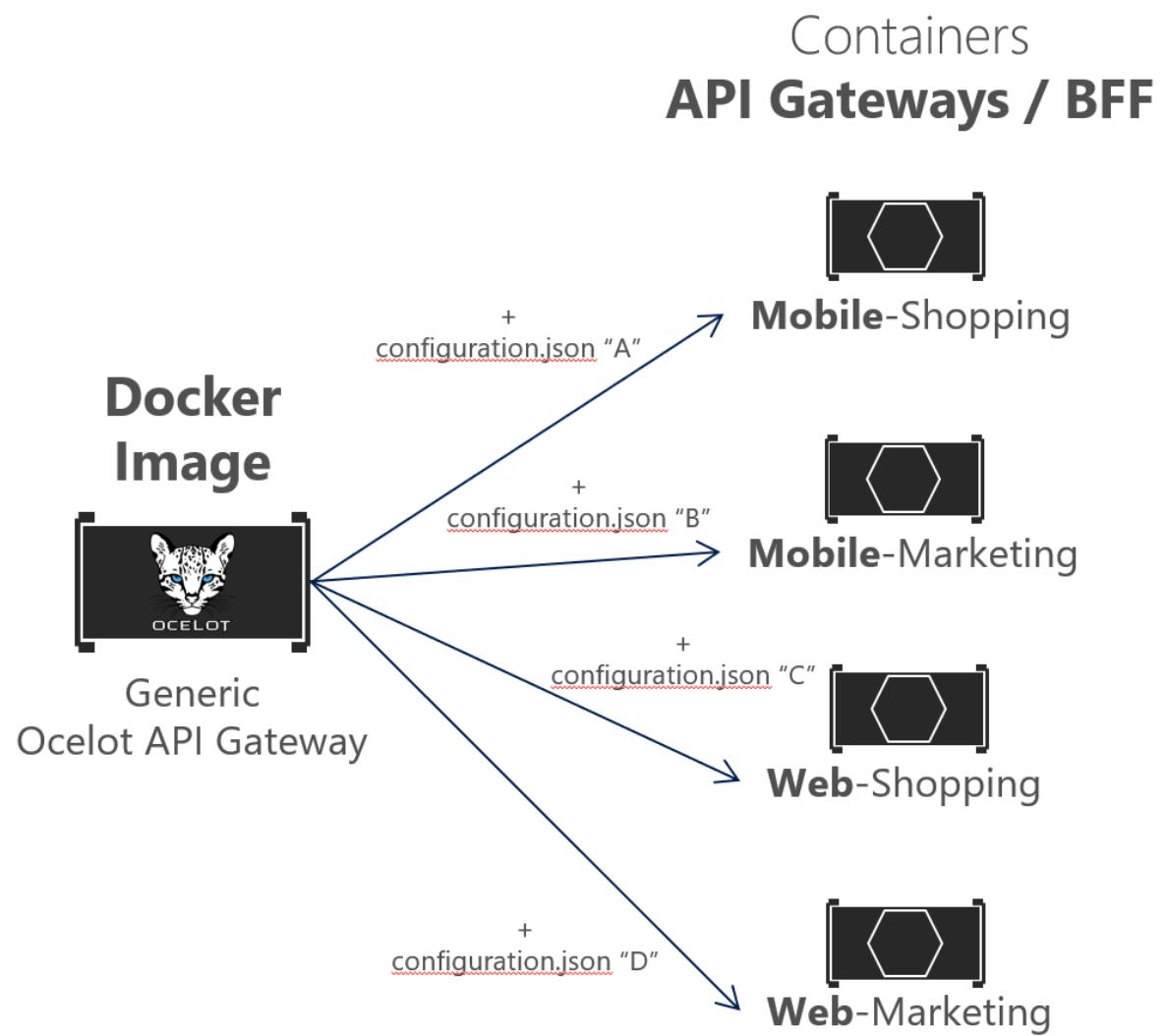


Main features in the API Gateway pattern

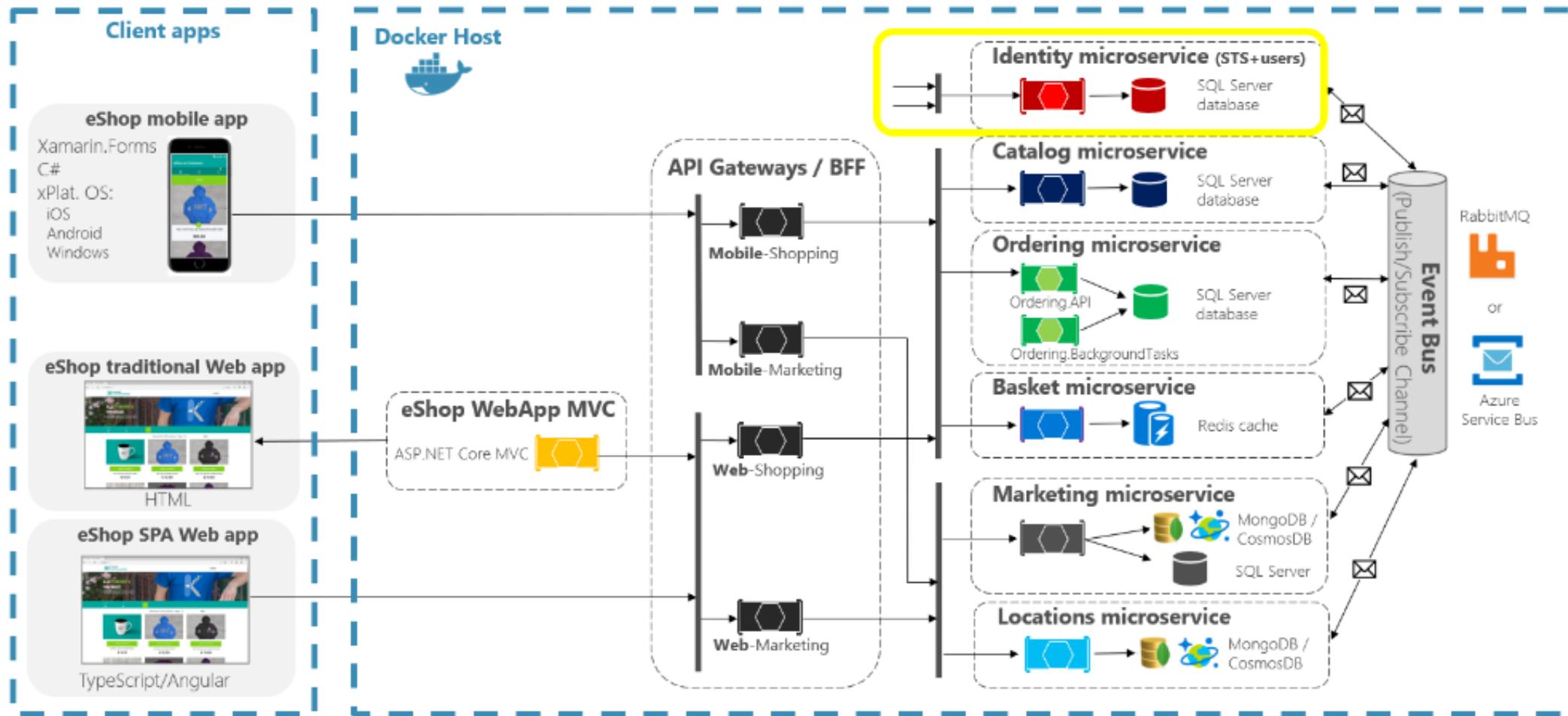
Ocelot Features	
Routing	Authentication
Request Aggregation	Authorization
Service Discovery with Consul & Eureka	Throttling
Load Balancing	Logging, Tracing
Correlation Pass-Through	Headers/Query String Transformation
Quality of Service	Custom Middleware

Ocelot API Gateway

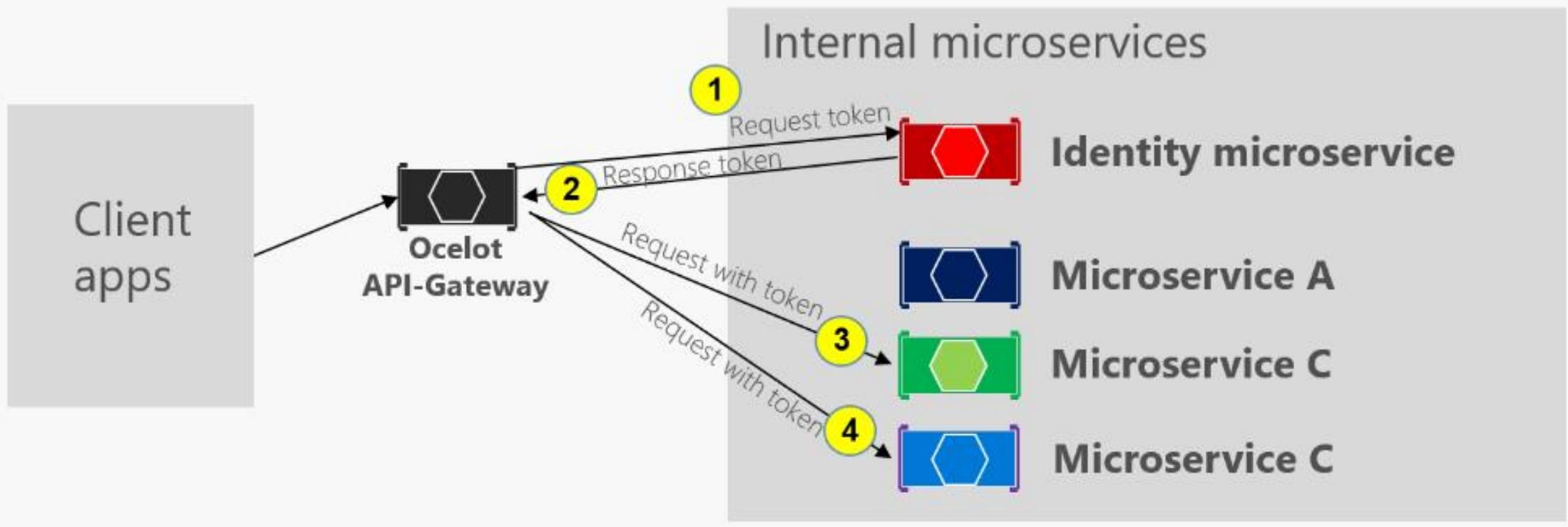
- Lightweight API Gateway
- .NET Core based API Gateway
- Open Source
- Fast & Scalable



Authentication and Authorization in Ocelot



Authentication in Ocelot API Gateway





aspnetrun

The best path to leverage your aspnet skills. Onboarding to .Net Software Architect jobs. Download latest real world asp.net core microservices applications.

Istanbul

<https://aspnetrun.azurewebsites.net>

ezozkme@gmail.com

Repositories 12

Packages

People 1

Teams

Projects 1

Settings

Pinned repositories

Customize pinned repositories

run-aspnetcore

Template



A starter kit for your next ASP.NET Core web application. Boilerplate for ASP.NET Core reference application, demonstrating a layered application architecture with applying Clean Architecture and D...

C# 223 55

run-aspnetcore-microservices

Microservices on .Net platforms which used Asp.Net Web API, Docker, RabbitMQ, MassTransit, Grpc, Ocelot API Gateway, MongoDB, Redis, PostgreSQL, SqlServer, Dapper, Entity Framework Core, CQRS and C...

C# 388 164

run-aspnetcore-realworld

E-Commerce real world example of run-aspnetcore ASP.NET Core web application. Implemented e-commerce domain with clean architecture for ASP.NET Core reference application, demonstrating a layered a...

SCSS 206 75

run-aspnet-identityserver4



Secure microservices with using standalone Identity Server 4 and backing with Ocelot API Gateway. Protect our ASP.NET Web MVC and API applications with using OAuth 2 and OpenID Connect in IdentityS...

C# 35 17

run-aspnet-grpc

Using gRPC in Microservices for Building a high-performance Interservice Communication with .Net 5. See gRPC Microservices and Step by Step Implementation on .NET Course w/ discount->

C# 33 10

run-devops

Deploying .Net Microservices into Kubernetes, and moving deployments to the cloud Azure Kubernetes Services (AKS) with using Azure Container Registry (ACR) and how to Automating Deployments with Az...

C# 4 7

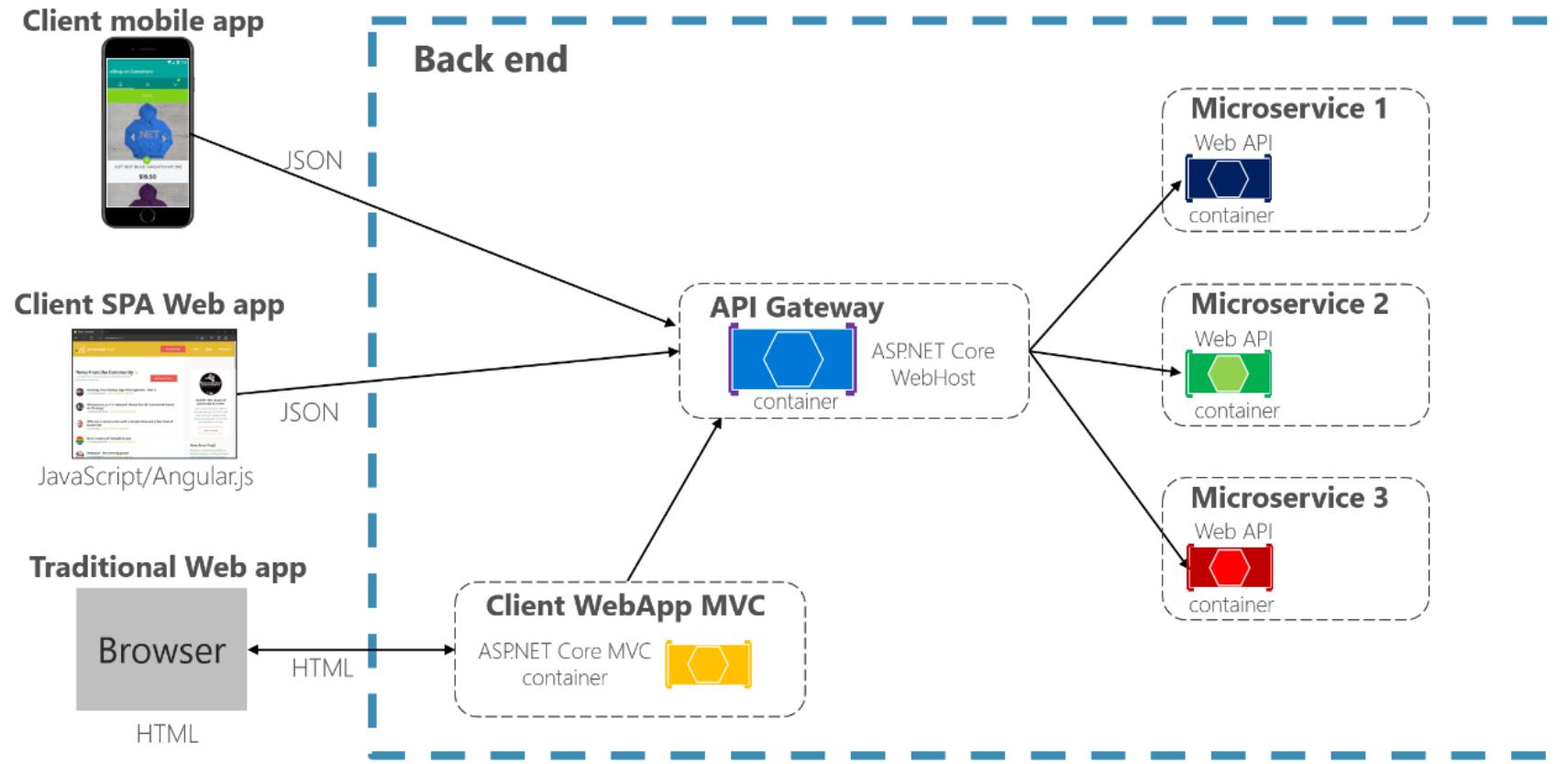
Analysis & Design of API Gateway

- Route Catalog APIs with /Catalog path
- Route Basket APIs with /Basket path
- Route Ordering APIs with /Ordering path

Method	Request URI	Use Case
GET/POST	/Catalog	Route /api/v1/Catalog apis
GET	/Catalog/{id}	Route /api/v1/Catalog apis
GET/POST	/Basket	Basket /api/v1/Basket apis
POST	/Basket/Checkout	Basket /api/v1/Basket apis
GET	/Order	Order /api/v1/Order apis

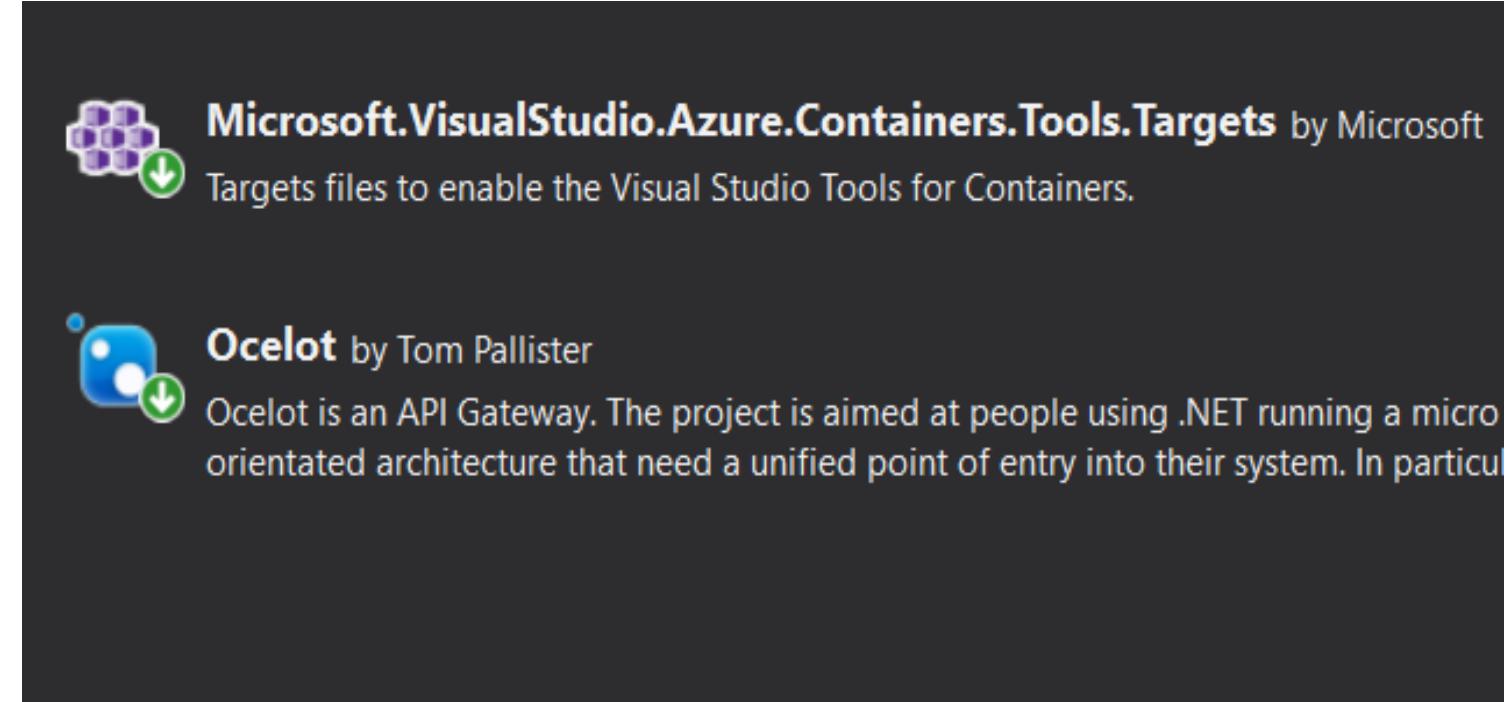
Architecture of API Gateway Microservices

Using a single custom **API Gateway service**



Nuget Packages of API Gateway

- Ocelot Nuget Package



The screenshot shows the NuGet package search results for 'Ocelot'. It displays two packages:

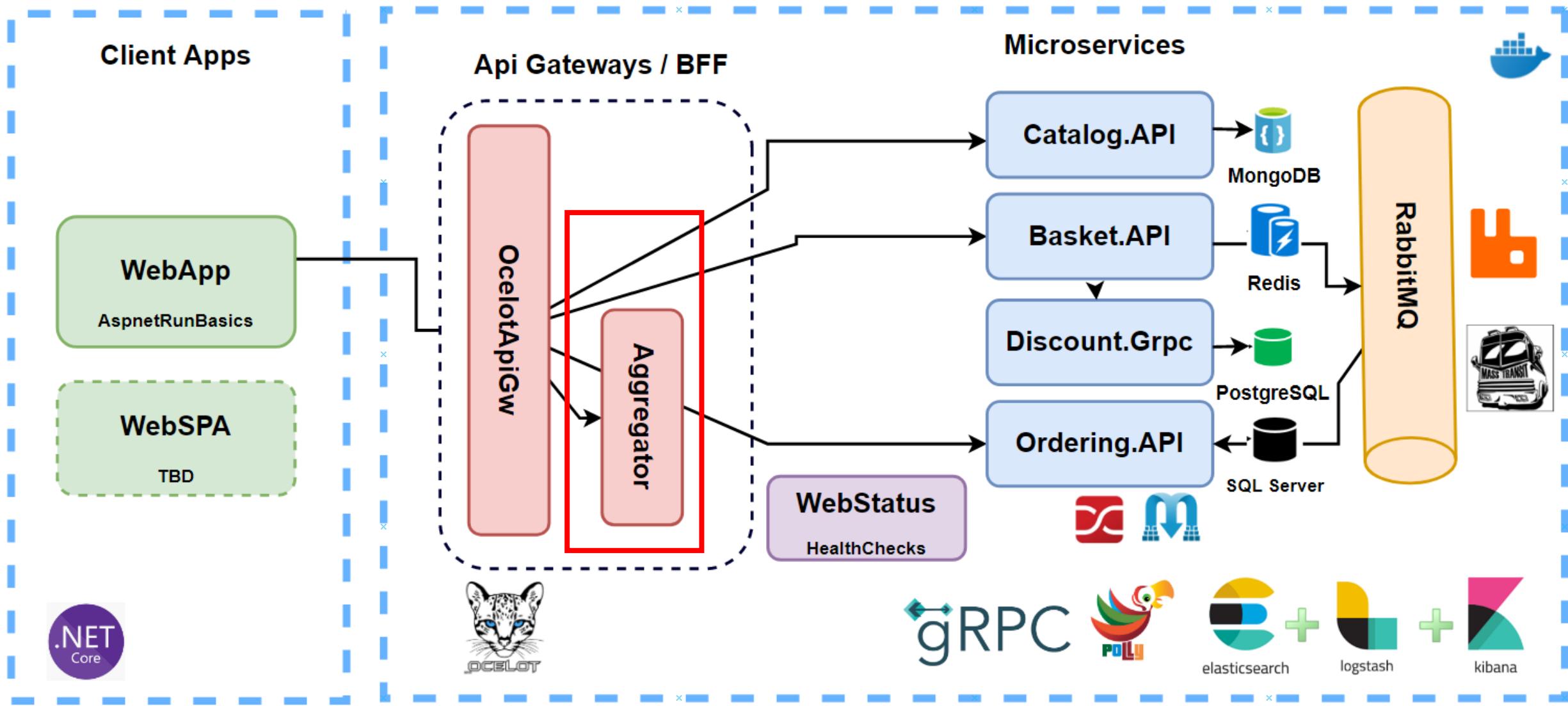
- Microsoft.VisualStudio.Azure.Containers.Tools.Targets** by Microsoft: Targets files to enable the Visual Studio Tools for Containers.
- Ocelot** by Tom Pallister: Ocelot is an API Gateway. The project is aimed at people using .NET running a micro orientated architecture that need a unified point of entry into their system. In particu

Section 10

Api Gateway - Requests Aggregation Pattern

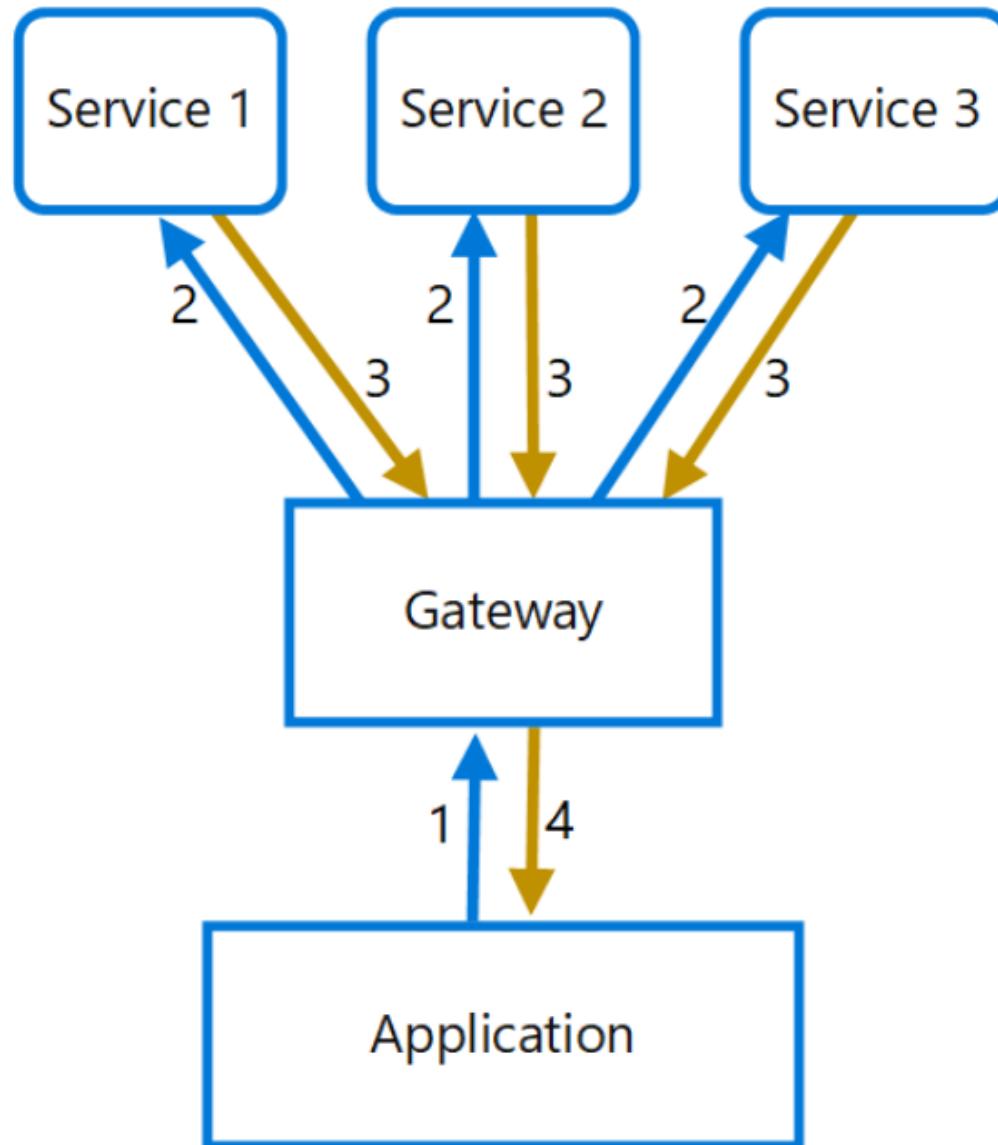
Developing Shopping.Aggregator

Big Picture



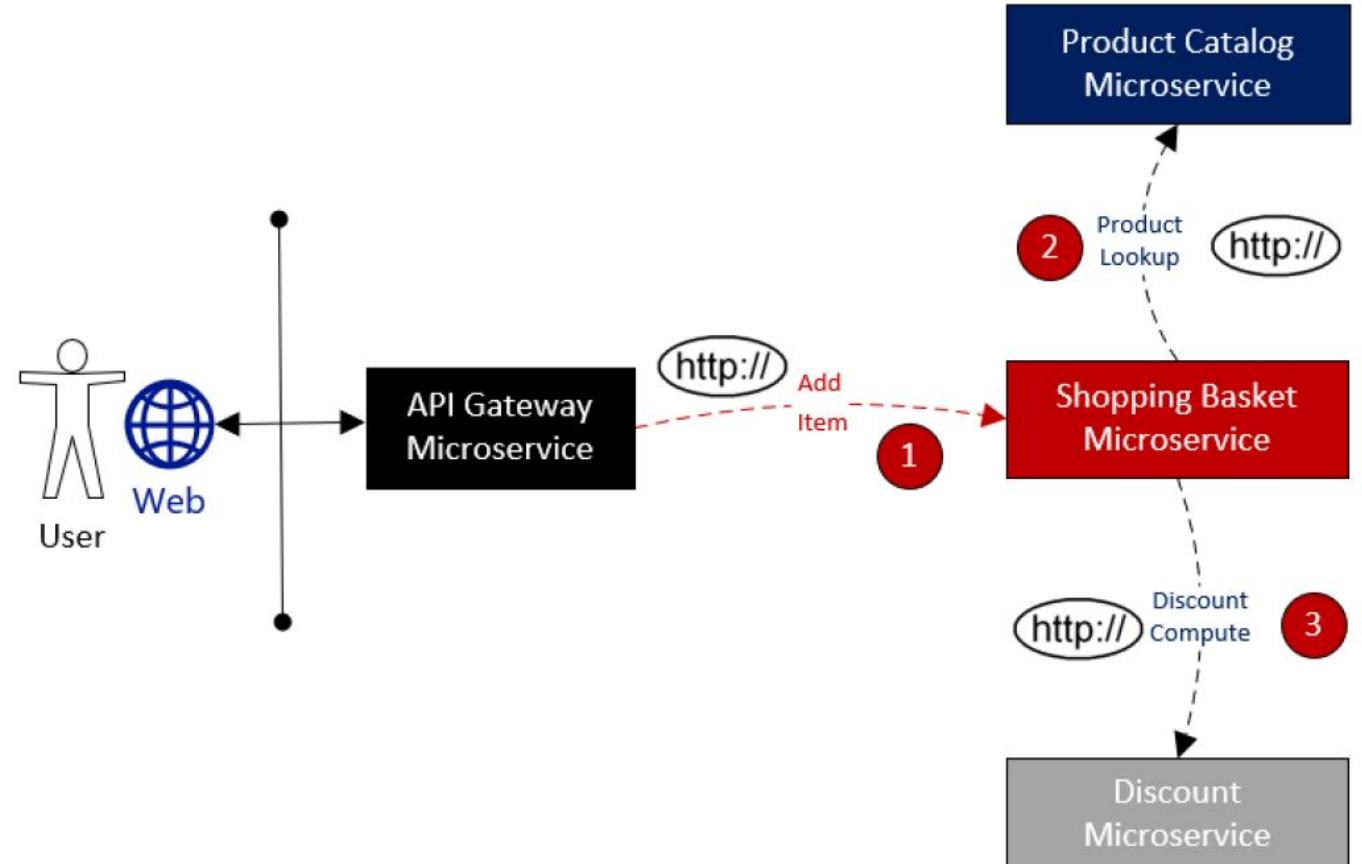
Gateway Aggregation pattern

- Single request
- Multiple calls to different backend systems
- Dispatches requests to the various backend systems
- Reduce chattiness between the client and the services



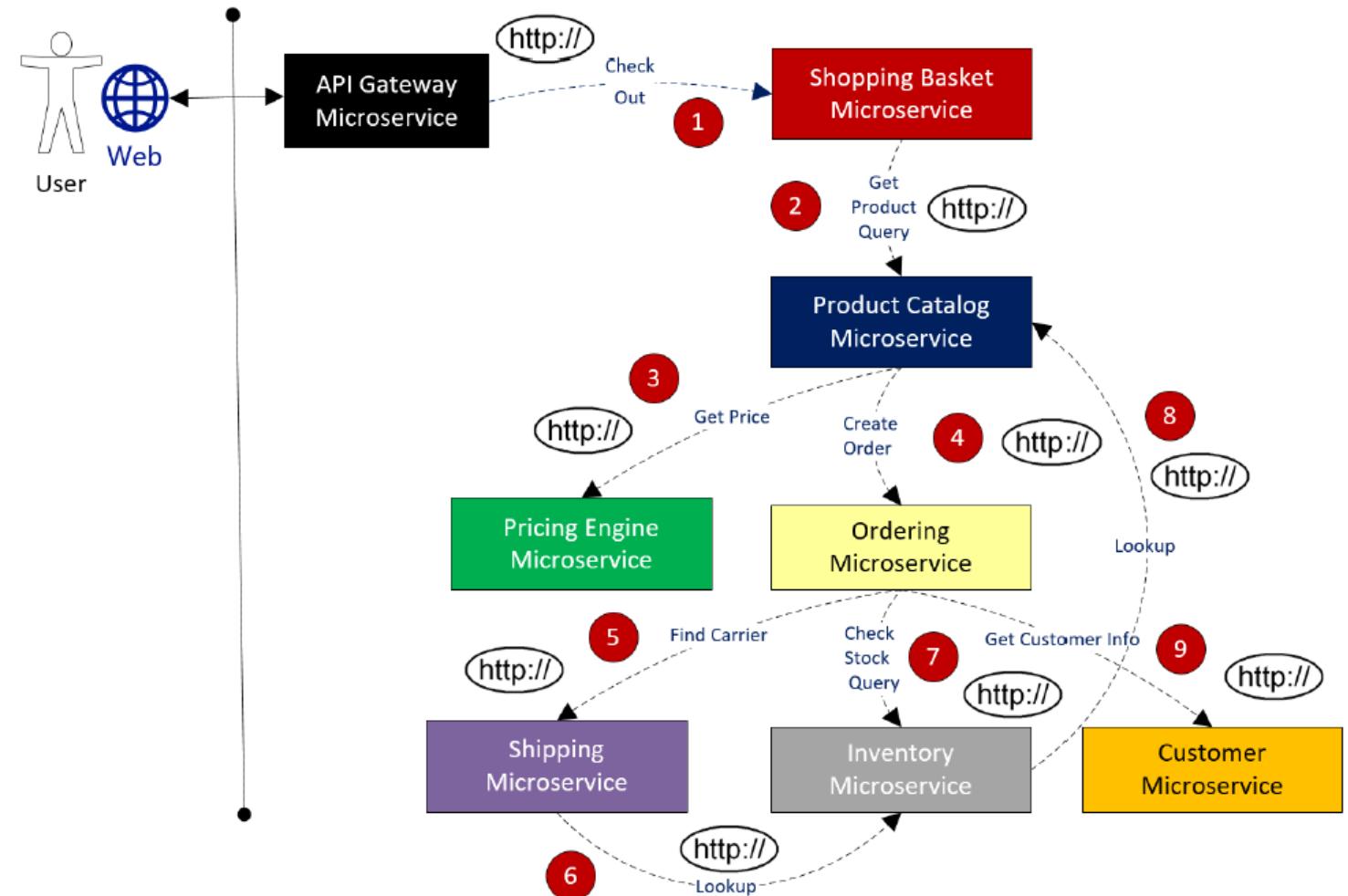
Direct Synchronous Communication

- Impact performance – each call adds latency
- Impact reliability – an unresponsive service can impact entire operation

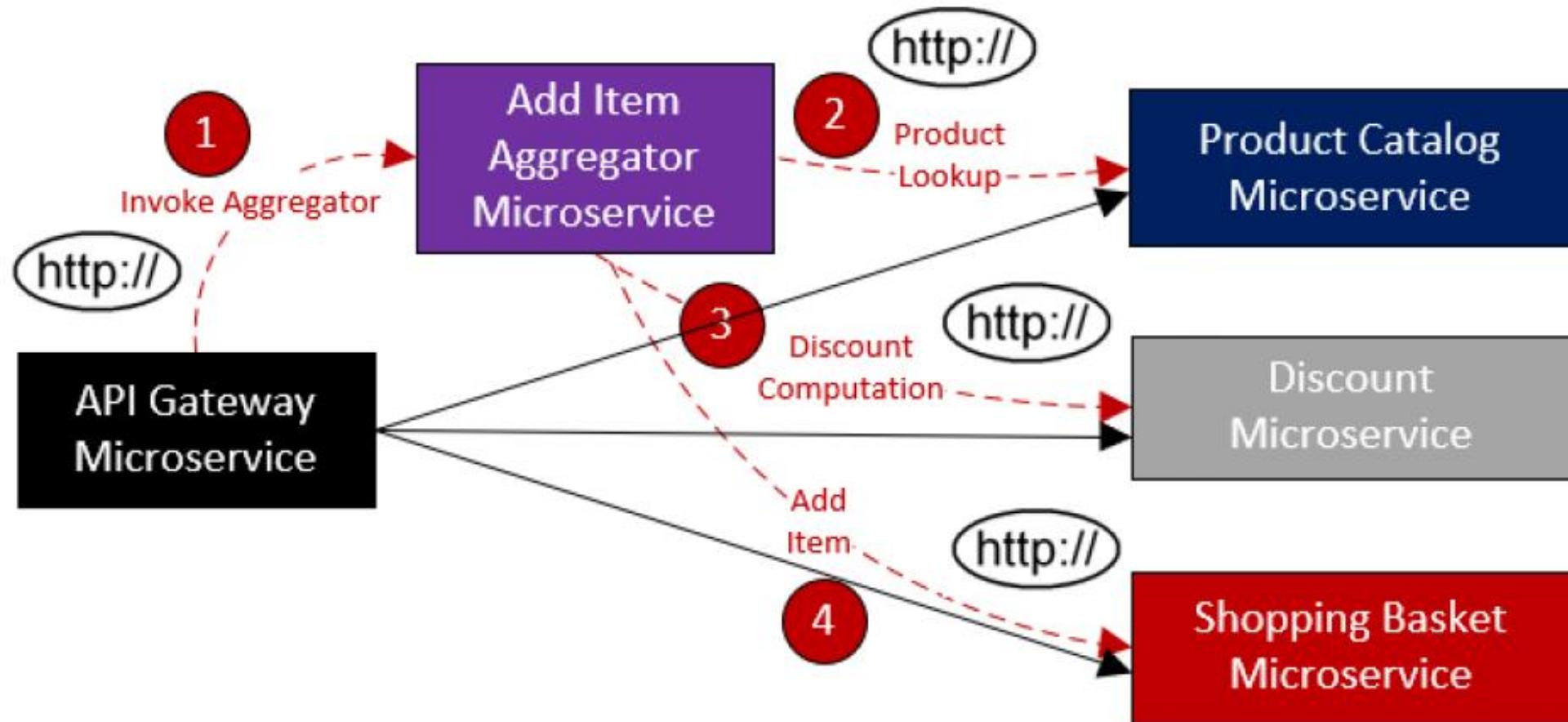


Chaining HTTP Calls

- Deep chaining or nesting of HTTP calls - simple to implement, but an antipattern

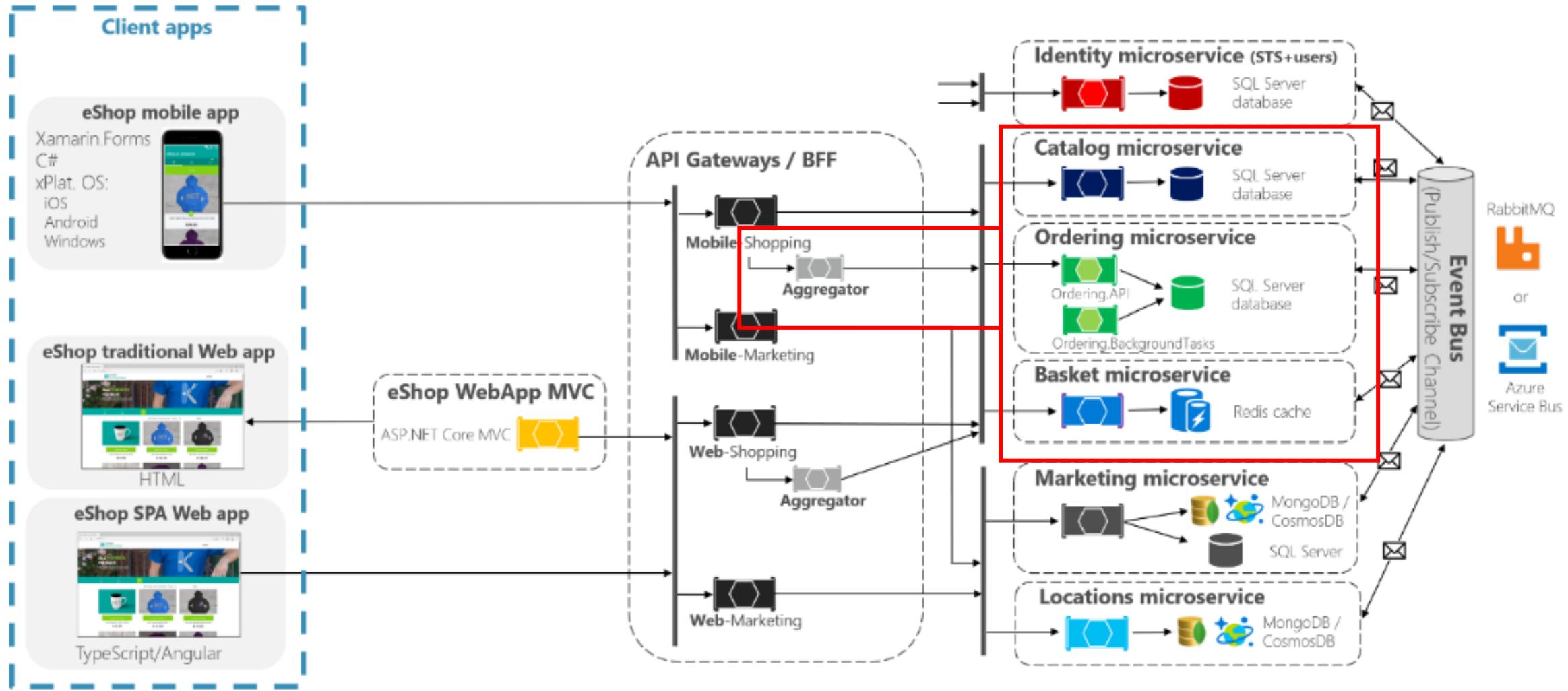


Gateway Aggregation pattern



Architecture of Shopping.Aggregator

(API Gateways / BFF and Aggregator-services details)



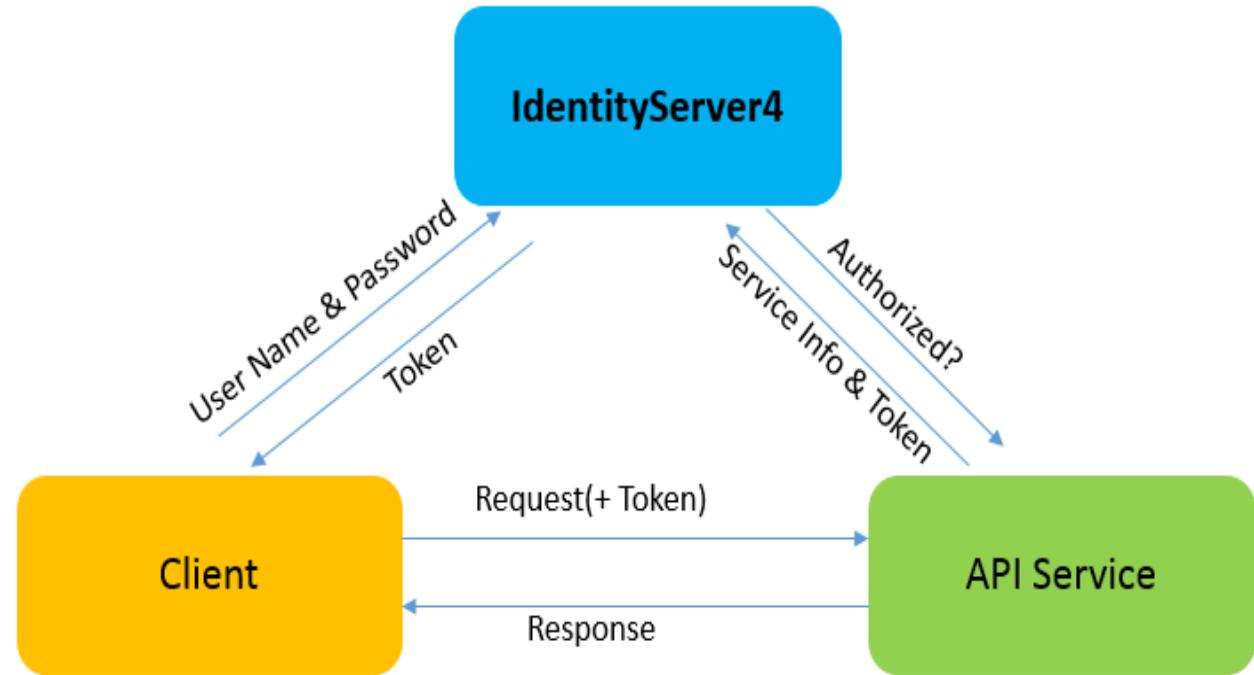
Section 11

Securing Microservices with IdentityServer4 and Ocelot

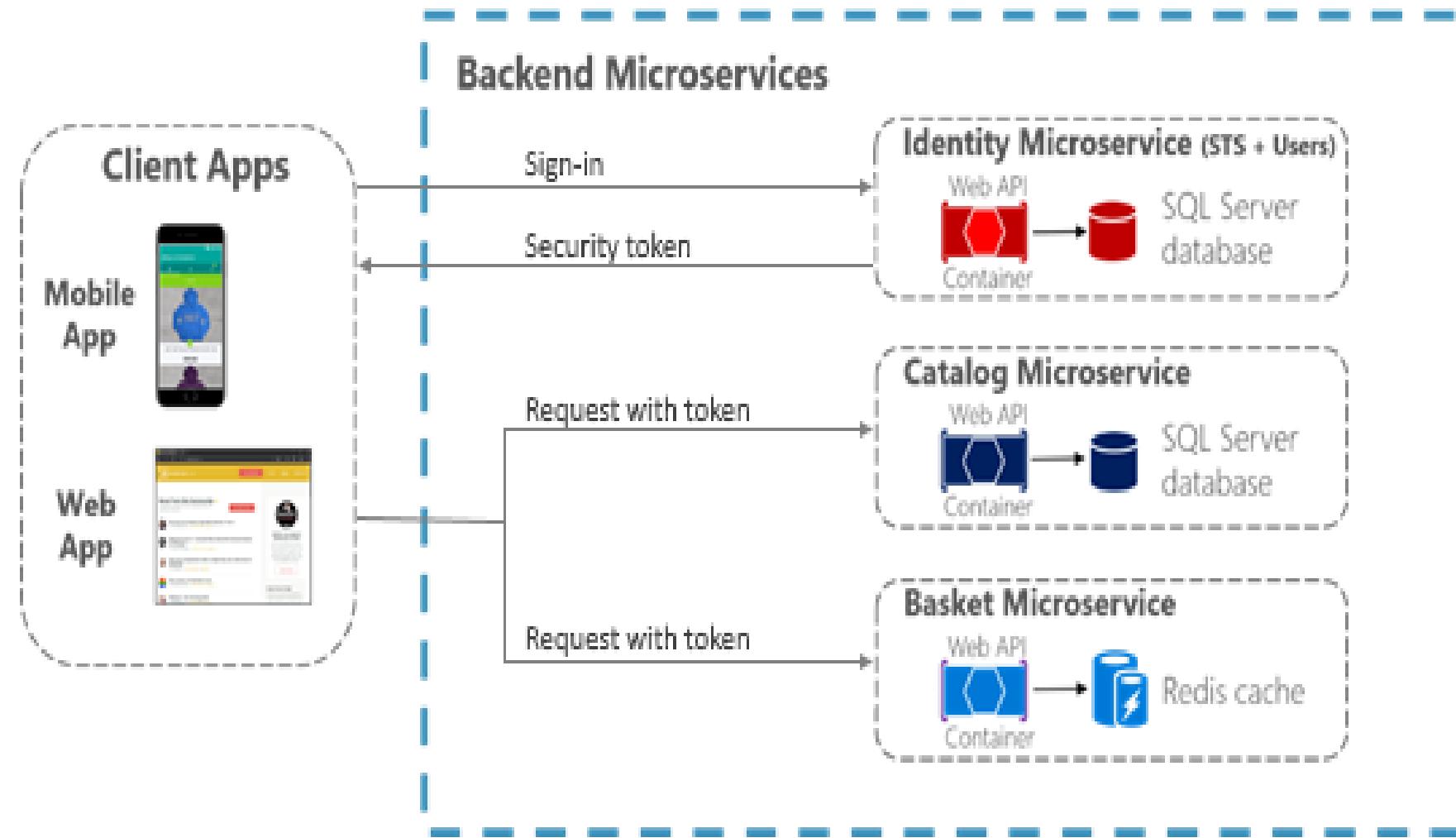
Cross-Cutting Concerns of Security

Microservices Security with IdentityServer4

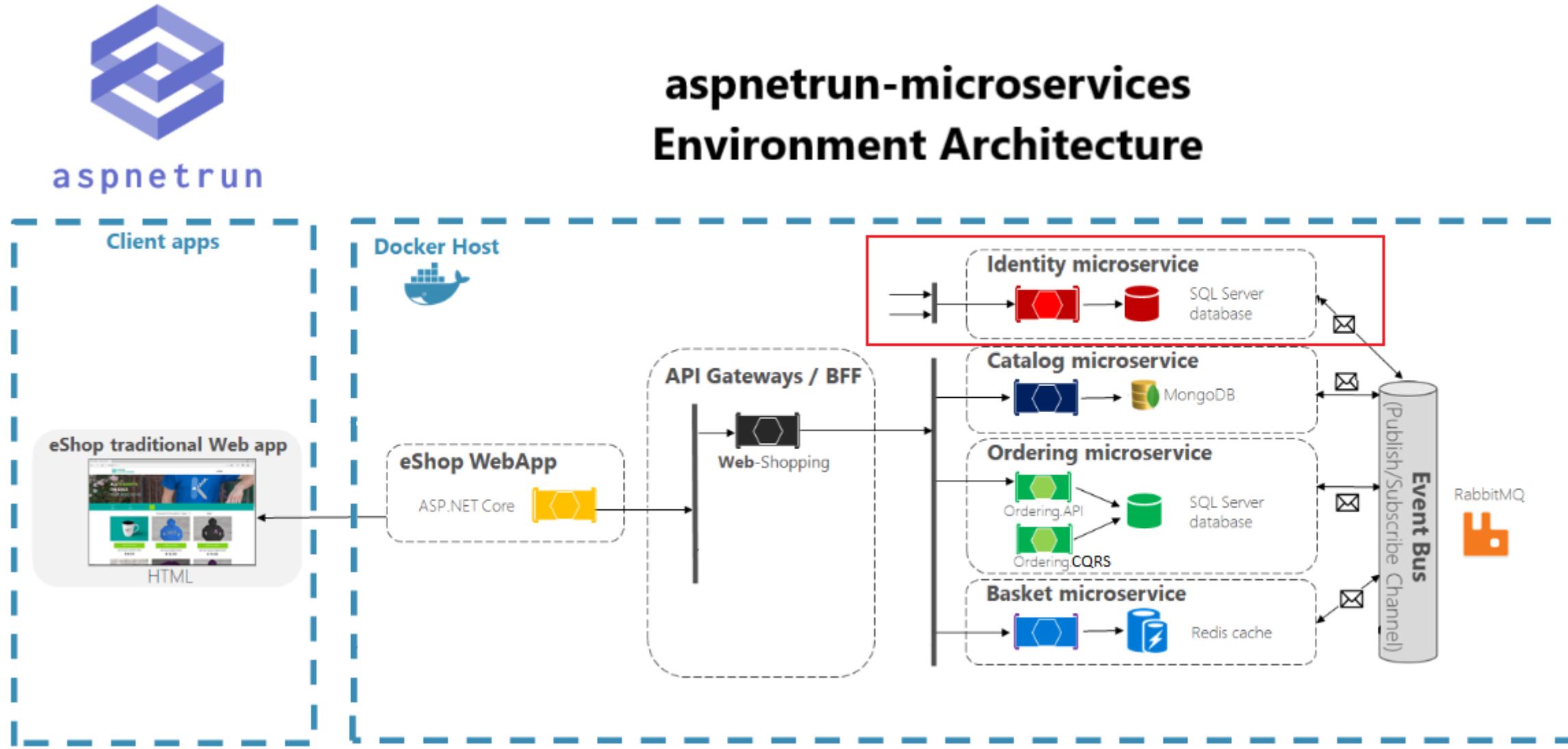
- IdentityServer4 - Authentication as a Service
- Protected API Resources with OAuth 2.0
- OpenID Connect for MVC Interactive Client Apps
- Backing with Ocelot API Gateway



Identity Server 4 in Microservices World



Identity Server in Microservices Architecture





aspnetrun

The best path to leverage your aspnet skills. Onboarding to .Net Software Architect jobs. Download latest real world asp.net core microservices applications.

Istanbul

<https://aspnetrun.azurewebsites.net>

ezozkme@gmail.com

Repositories 12

Packages

People 1

Teams

Projects 1

Settings

Pinned repositories

Customize pinned repositories

run-aspnetcore

Template



A starter kit for your next ASP.NET Core web application. Boilerplate for ASP.NET Core reference application, demonstrating a layered application architecture with applying Clean Architecture and D...

C# 223 55

run-aspnetcore-microservices

Microservices on .Net platforms which used Asp.Net Web API, Docker, RabbitMQ, MassTransit, Grpc, Ocelot API Gateway, MongoDB, Redis, PostgreSQL, SqlServer, Dapper, Entity Framework Core, CQRS and C...

C# 388 164

run-aspnetcore-realworld

E-Commerce real world example of run-aspnetcore ASP.NET Core web application. Implemented e-commerce domain with clean architecture for ASP.NET Core reference application, demonstrating a layered a...

SCSS 206 75

run-aspnet-identityserver4



Secure microservices with using standalone Identity Server 4 and backing with Ocelot API Gateway. Protect our ASP.NET Web MVC and API applications with using OAuth 2 and OpenID Connect in IdentityS...

C# 35 17

run-aspnet-grpc

Using gRPC in Microservices for Building a high-performance Interservice Communication with .Net 5. See gRPC Microservices and Step by Step Implementation on .NET Course w/ discount->

C# 33 10

run-devops

Deploying .Net Microservices into Kubernetes, and moving deployments to the cloud Azure Kubernetes Services (AKS) with using Azure Container Registry (ACR) and how to Automating Deployments with Az...

C# 4 7

Securing Microservices with IdentityServer4 and Ocelot

Secure .Net Microservices with IdentityServer4 OAuth2,OpenID

Securing .Net 5 Microservices with IdentityServer4 using OAuth2, OpenID Connect and Ocelot Api Gateway

- Github Repository -> <https://github.com/aspnetrun/run-aspnet-identityserver4>

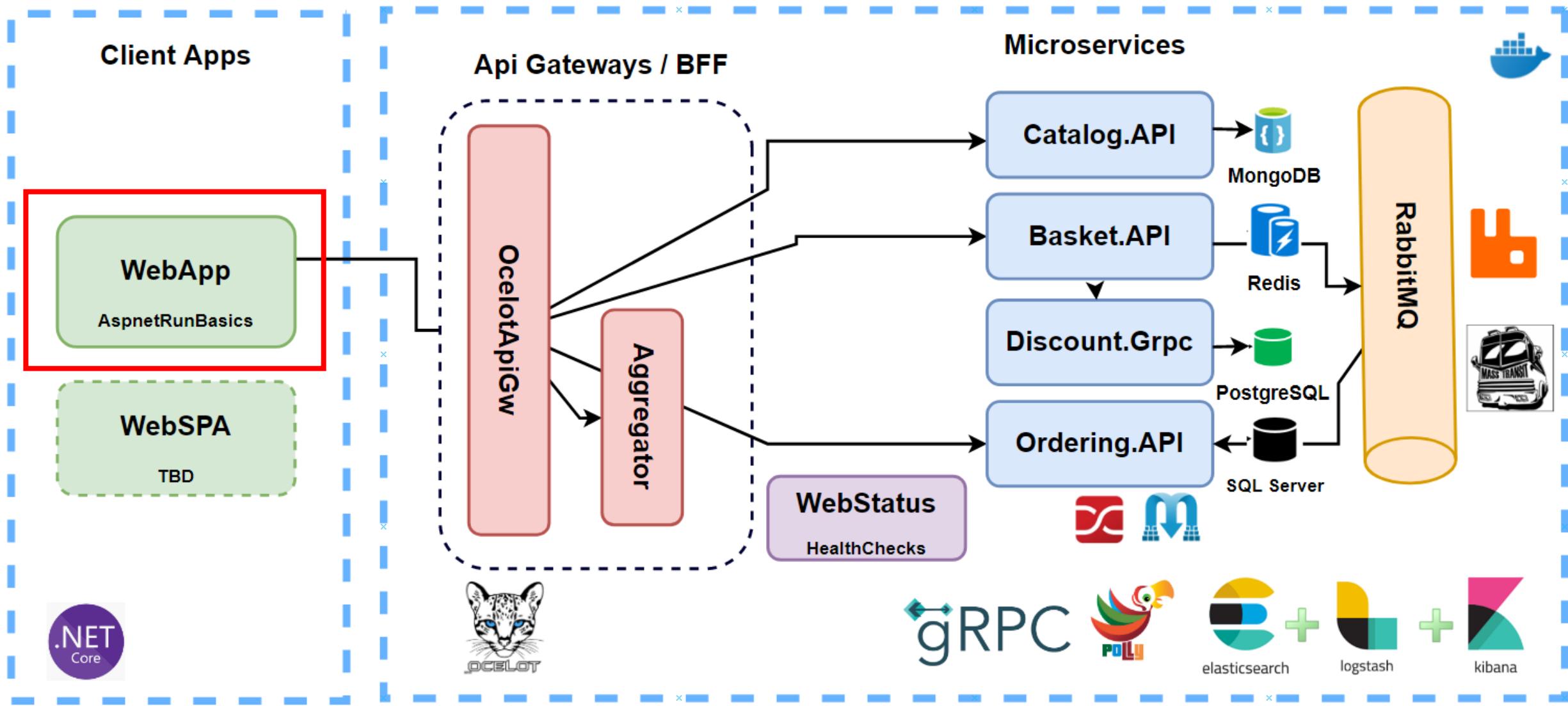
Section 12

Building Shopping Web

Application Microservices

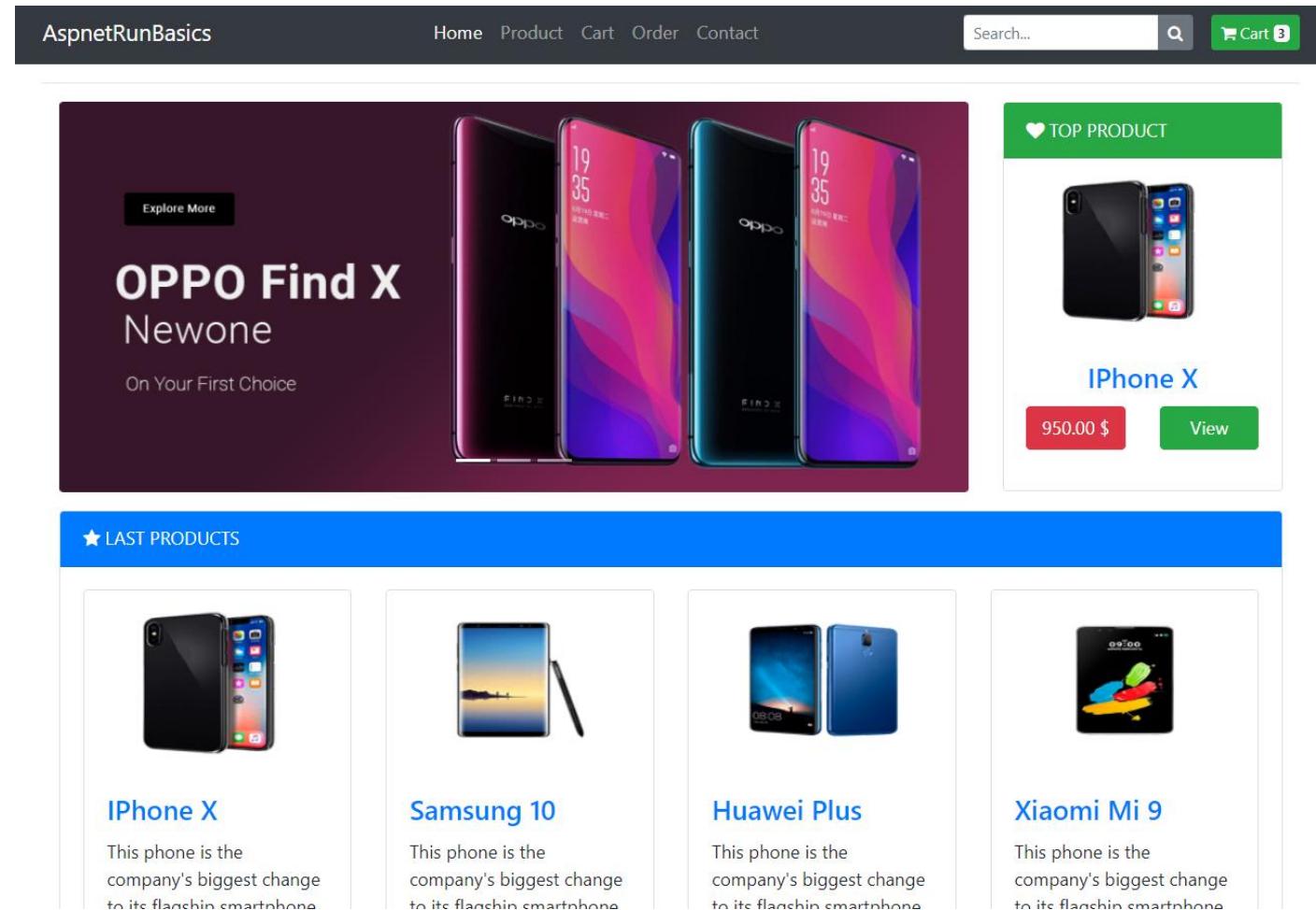
With AspNetBasics repository

Big Picture



Shopping Web Application Microservices

- Default Web Application
- Razor Templates
- HttpClientFactory
- Consume Ocelot API Gateway



Base Repository of Shopping App

- Base Application
- Github Repository of aspnetcore-basics
- Razor Pages
- Bootstrap4

The screenshot shows a GitHub repository page for the project 'run-aspnetcore-basics'. The repository is owned by 'aspnetrun'. The page includes a navigation bar with links for Code, Issues (0), Pull requests (1), Actions, Projects (0), Wiki, Security (0), Insights, and Settings. The main content area displays a brief description: 'Implementation of Real-World example in One Solution - One Project for web application development with Asp.Net Core & E...'. Below the description is a list of tags: aspnet-core, entity-framework-core, razor-pages, e-commerce, shopping-cart, real-world-project, real-world, aspnet-core-identity, aspnet-core-extensions, aspnetcore-authorization, aspnet-core-authentication, aspnet-core-validation, aspnetcore-basic-authentication, business, configuration, validation, bootstrap4, starter-kit, starter-template, bootstrap-theme. A 'Manage topics' section is also present. At the bottom, metrics are shown: 118 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. A 'Branch: master' dropdown and a 'New pull request' button are located at the bottom left, while 'Create new file', 'Upload files', 'Find file', and a 'Clone' button are at the bottom right.

<https://github.com/aspnetrun/run-aspnetcore-basics>

Analysis & Design of Shopping Web App

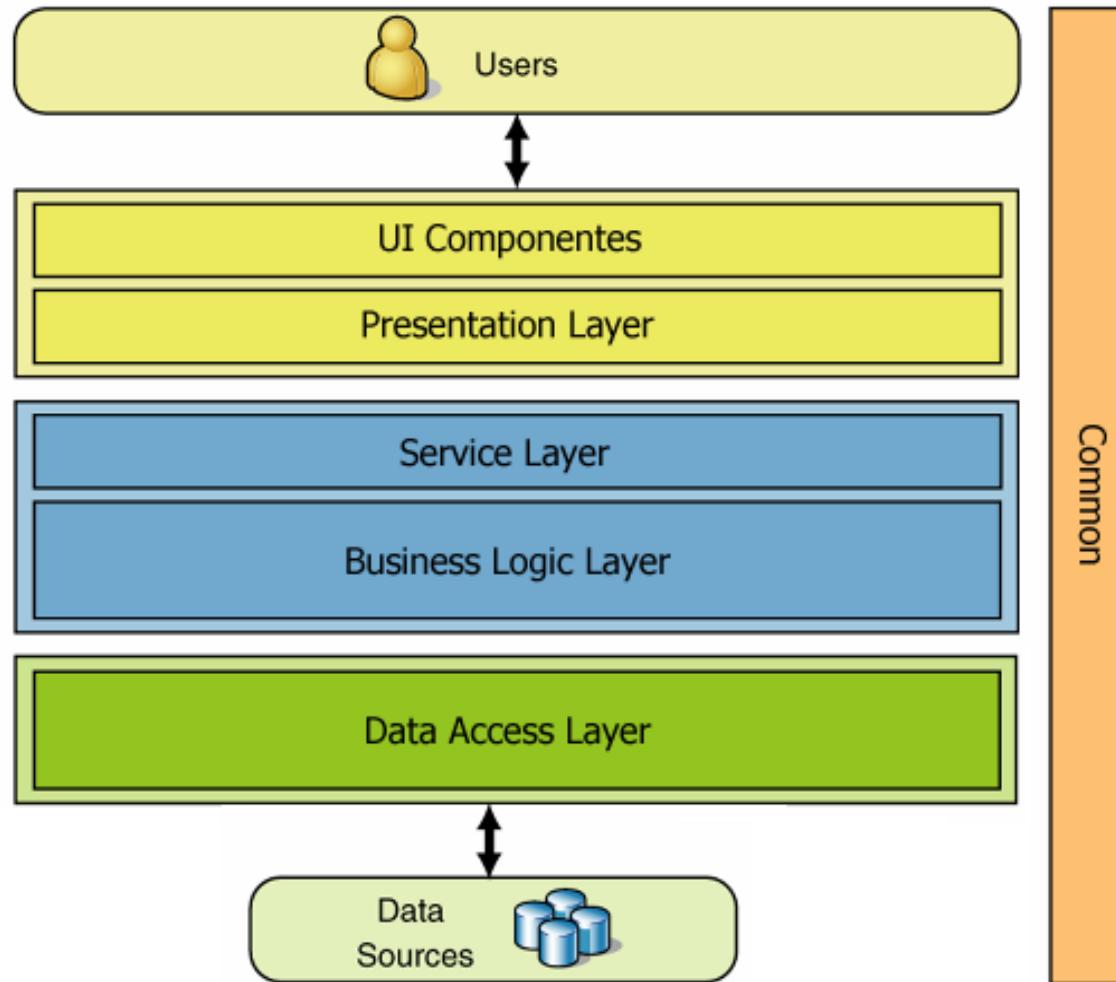
- Listing Products and Categories
- Add Product to Shopping Cart
- Checkout Order

The screenshot shows a shopping web application with the following layout:

- Header:** "AspNetRunBasics" logo, navigation menu with links to Home, Product, Cart, Order, Contact, a search bar with placeholder "Search...", and a cart icon showing 3 items.
- Hero Section:** A large banner for the "OPPO Find X" smartphone. It features four phones in black, white, red, and blue. The text "Explore More", "OPPO Find X", "Newone", and "On Your First Choice".
- Top Right Sidebar:** A green box labeled "TOP PRODUCT" featuring an iPhone X and the text "950.00 \$" and "View".
- Section Header:** "★ LAST PRODUCTS" in a blue bar.
- Product Cards:** Four cards showing recent products:
 - iPhone X:** Image, "iPhone X" name, and a descriptive text: "This phone is the company's biggest change to its flagship smartphone".
 - Samsung 10:** Image, "Samsung 10" name, and a descriptive text: "This phone is the company's biggest change to its flagship smartphone".
 - Huawei Plus:** Image, "Huawei Plus" name, and a descriptive text: "This phone is the company's biggest change to its flagship smartphone".
 - Xiaomi Mi 9:** Image, "Xiaomi Mi 9" name, and a descriptive text: "This phone is the company's biggest change to its flagship smartphone".

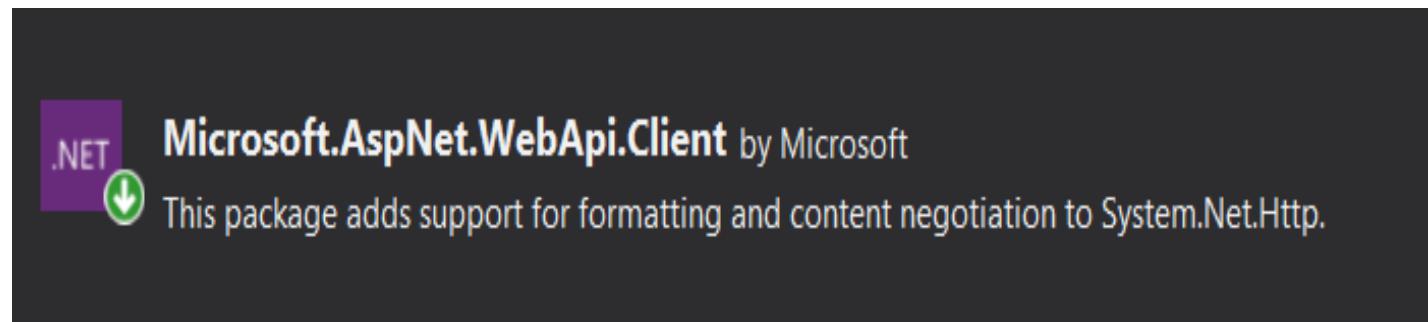
Shopping Microservices Architecture

- Data Access Layer
- Business Logic Layer
- Presentation Layer



Nuget Packages of Shopping Microservices

- WebApi.Client
- HttpClientFactory

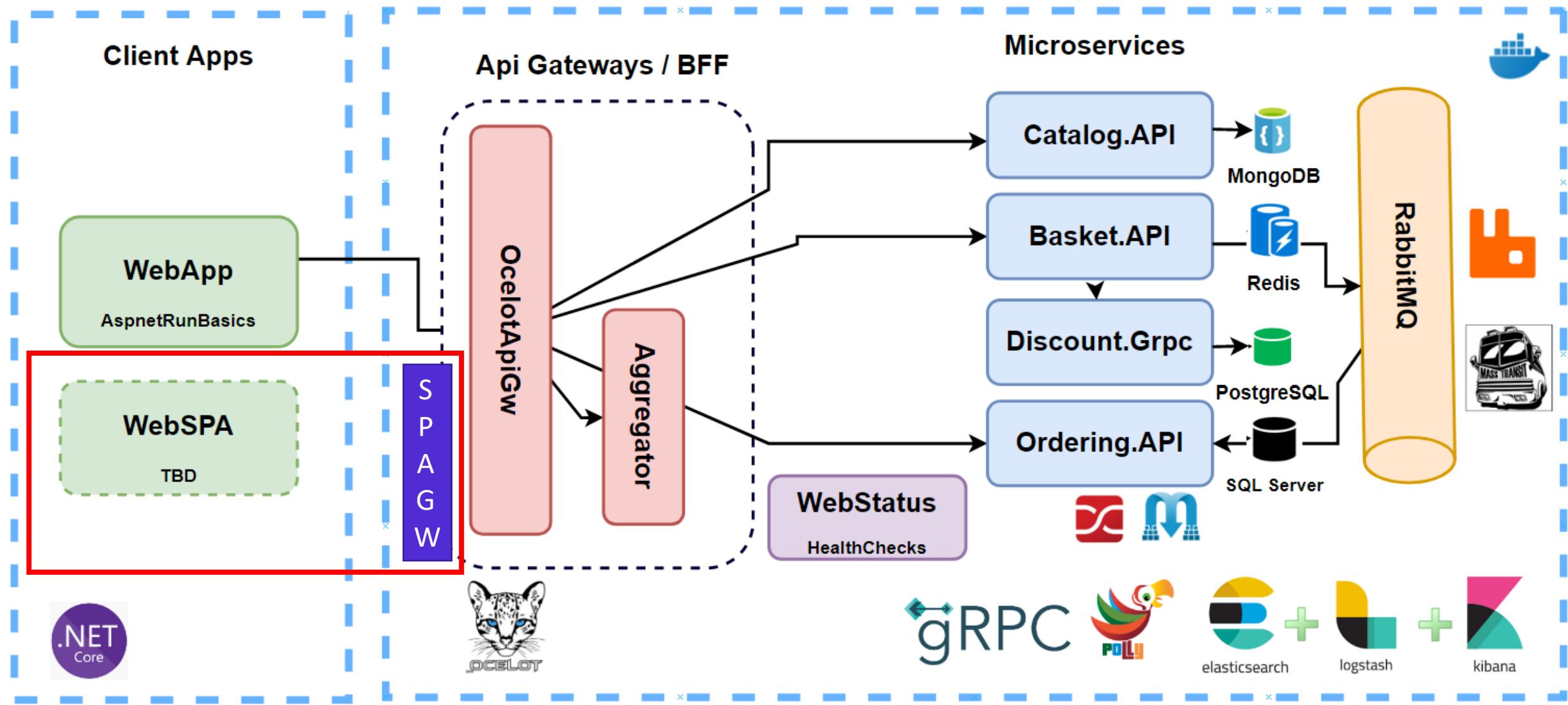


Section 13

Assignment - Developing Blazor Single Page Application with Custom Api Gateway

for performing CRUD Operations over Catalog, Discount, Basket and Ordering microservices

Big Picture



Section 14

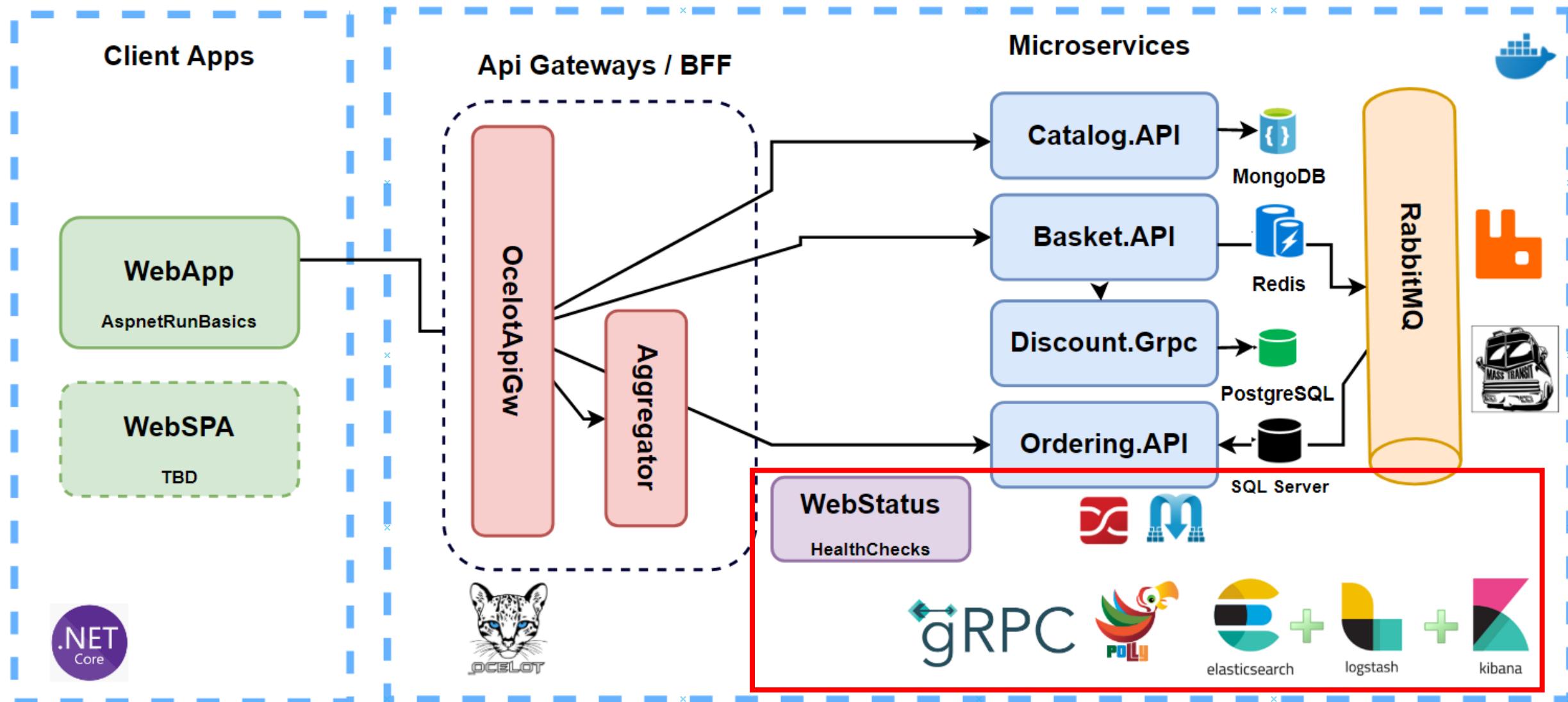
Cross-Cutting Concerns -

Microservices Observability with

Distributed Logging

And Health Monitoring, Resilient and Fault Tolerance with using Polly

Big Picture





aspnetrun

The best path to leverage your aspnet skills. Onboarding to .Net Software Architect jobs. Download latest real world asp.net core microservices applications.

📍 Istanbul

🔗 <https://aspnetrun.azurewebsites.net>

✉ ezozkme@gmail.com

Repositories 13

Packages

People 1

Teams

Projects 1

Settings

Pinned repositories

Customize pinned repositories



learn



run-aspartcore-microservices



The best path to .Net Microservices Udemy Learning Path. .Net world evolving to the microservices and Cloud-native systems to provide rapid change, large scale, and resilience cutting-edge systems....



2



1



C#



420



175



run-aspartcore

Template



A starter kit for your next ASP.NET Core web application. Boilerplate for ASP.NET Core reference application, demonstrating a layered application architecture with applying Clean Architecture and D...



C#



228



57



run-aspartidentityserver4



run-aspart-grpc



Secure microservices with using standalone Identity Server 4 and backing with Ocelot API Gateway. Protect our ASP.NET Web MVC and API applications with using OAuth 2 and OpenID Connect in IdentityS...



C#



39



18



C#



34



10



run-devops



Deploying .Net Microservices into Kubernetes, and moving deployments to the cloud Azure Kubernetes Services (AKS) with using Azure Container Registry (ACR) and how to Automating Deployments with Az...



C#



4



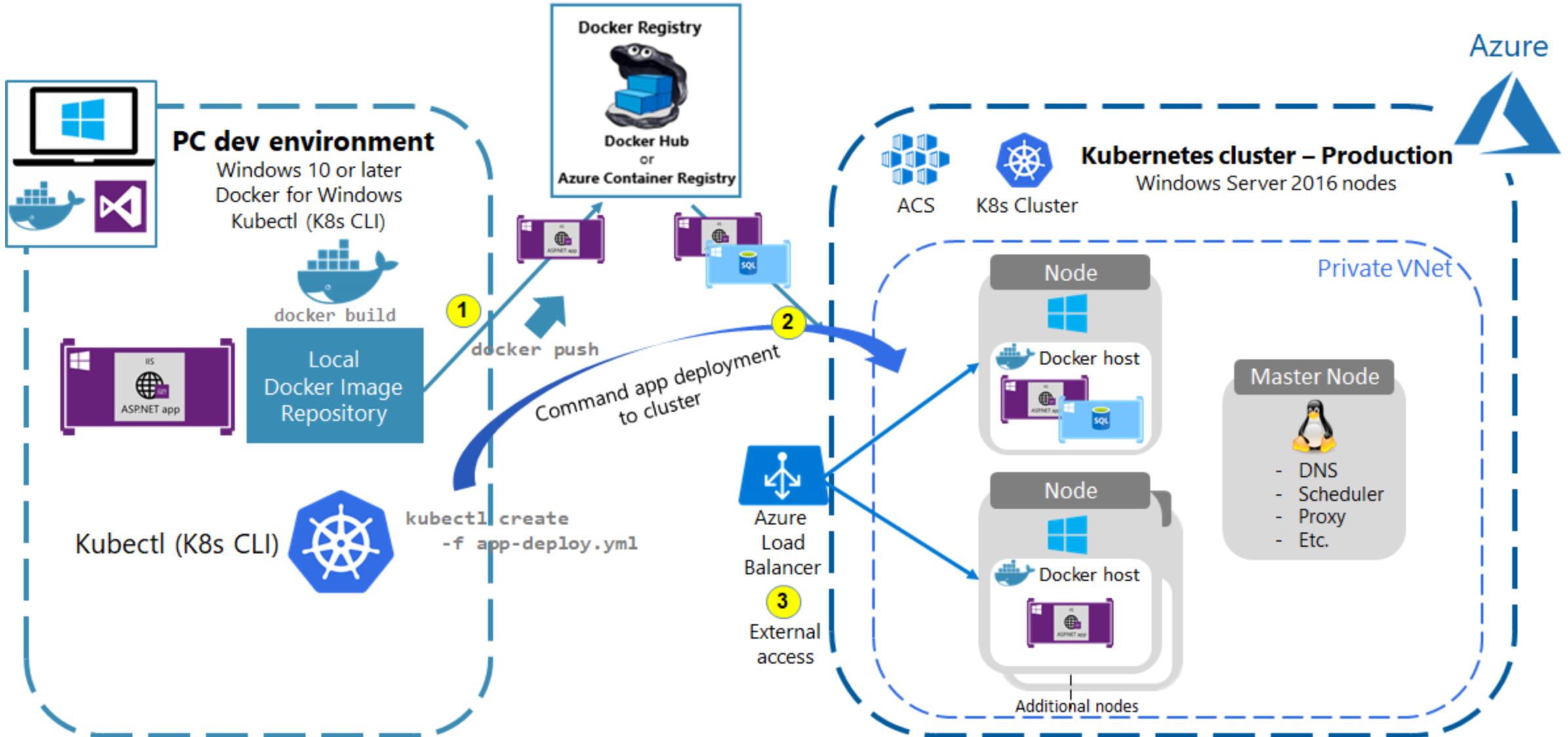
8

Section 15

Deploying Microservices to Kubernetes, Automating with Azure DevOps into AKS

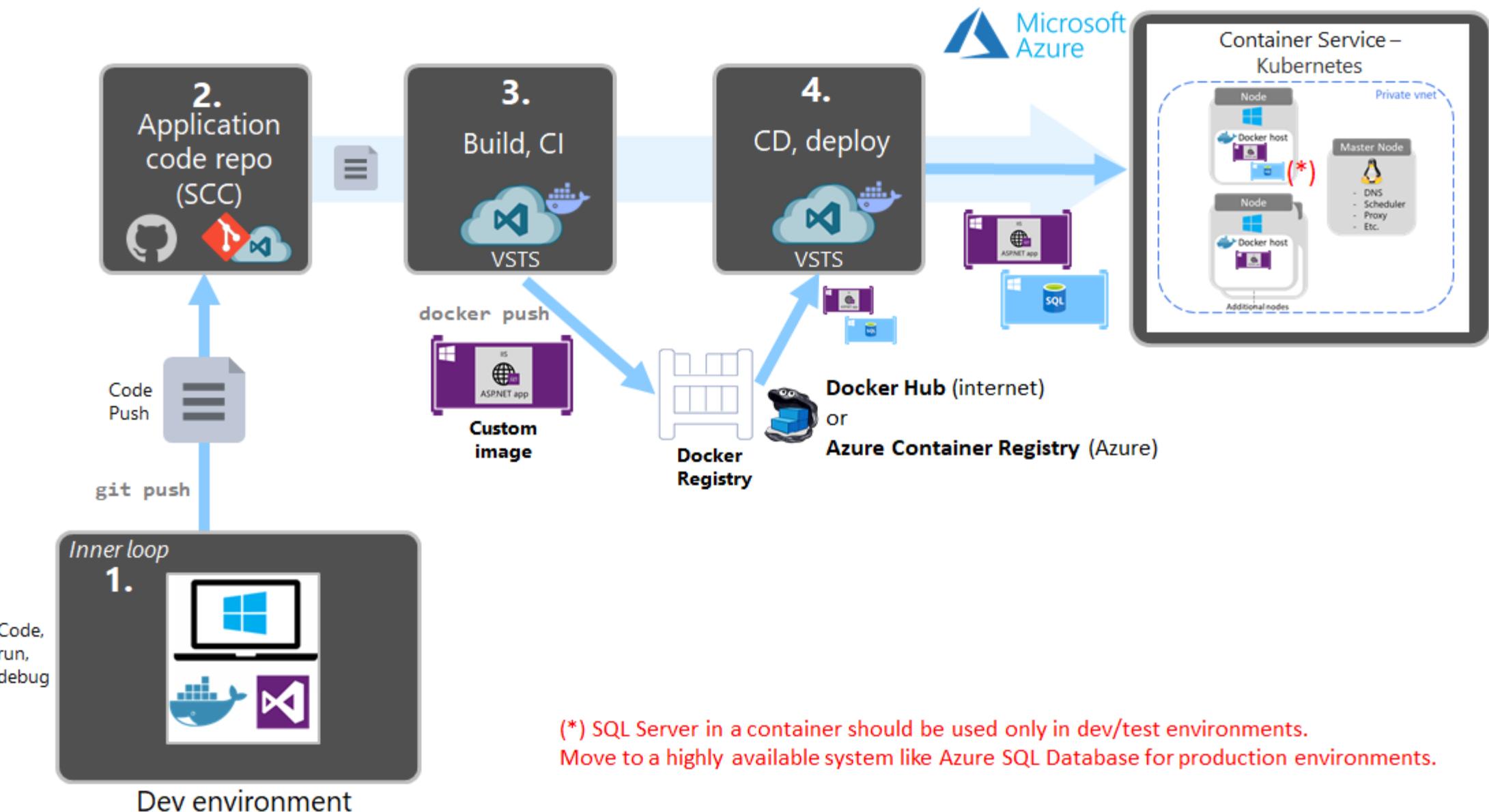
Using Azure Pipelines with Azure Kubernetes Services (AKS)

Scenario: Direct deployment to a Kubernetes cluster in Azure Container Service



(*) SQL Server in a container should be used only in dev/test environments.
Move to a highly available system like Azure SQL Database for production environments.

Scenario: Deploy to Kubernetes through CI/CD pipelines





aspnetrun

The best path to leverage your aspnet skills. Onboarding to .Net Software Architect jobs. Download latest real world asp.net core microservices applications.

Istanbul

<https://aspnetrun.azurewebsites.net>

ezozkme@gmail.com

Repositories 12

Packages

People 1

Teams

Projects 1

Settings

Pinned repositories

Customize pinned repositories

run-aspnetcore

Template



A starter kit for your next ASP.NET Core web application. Boilerplate for ASP.NET Core reference application, demonstrating a layered application architecture with applying Clean Architecture and D...



223



55

run-aspnetcore-microservices

Microservices on .Net platforms which used Asp.Net Web API, Docker, RabbitMQ, MassTransit, Grpc, Ocelot API Gateway, MongoDB, Redis, PostgreSQL, SqlServer, Dapper, Entity Framework Core, CQRS and C...



388



164

run-aspnetcore-realworld

E-Commerce real world example of run-aspnetcore ASP.NET Core web application. Implemented e-commerce domain with clean architecture for ASP.NET Core reference application, demonstrating a layered a...



206



75

run-aspnet-identityserver4



Secure microservices with using standalone Identity Server 4 and backing with Ocelot API Gateway. Protect our ASP.NET Web MVC and API applications with using OAuth 2 and OpenID Connect in IdentityS...



35



17

run-aspnet-grpc



Using gRPC in Microservices for Building a high-performance Interservice Communication with .Net 5. See gRPC Microservices and Step by Step Implementation on .NET Course w/ discount->



33



10

run-devops



Deploying .Net Microservices into Kubernetes, and moving deployments to the cloud Azure Kubernetes Services (AKS) with using Azure Container Registry (ACR) and how to Automating Deployments with Az...



4



7

Deploying Microservices to Kubernetes, Automating with Azure DevOps into AKS

Deploying .Net Microservices with K8s, AKS and Azure DevOps

Deploying .Net Microservices to Kubernetes, move cloud Azure Kubernetes Services(AKS), Automating with Azure DevOps

Hot & New

4.5 ★★★★★ (18 ratings) 258 students

- Github Repository -> <https://github.com/aspnetrun/run-devops>

Thanks!

Follow me on github

<https://github.com/mehmetozkaya>

<https://github.com/aspnetrun>

Follow me on twitter

<https://twitter.com/ezozkme>

Follow me on medium

<https://mehmetozkaya.medium.com/>

Send mail me anything you can ask

ezozkme@gmail.com

Check my other courses on udemy:

<https://www.udemy.com/user/ff0e5c8c-dd71-443e-be0a-e73ba821f7d7/>

