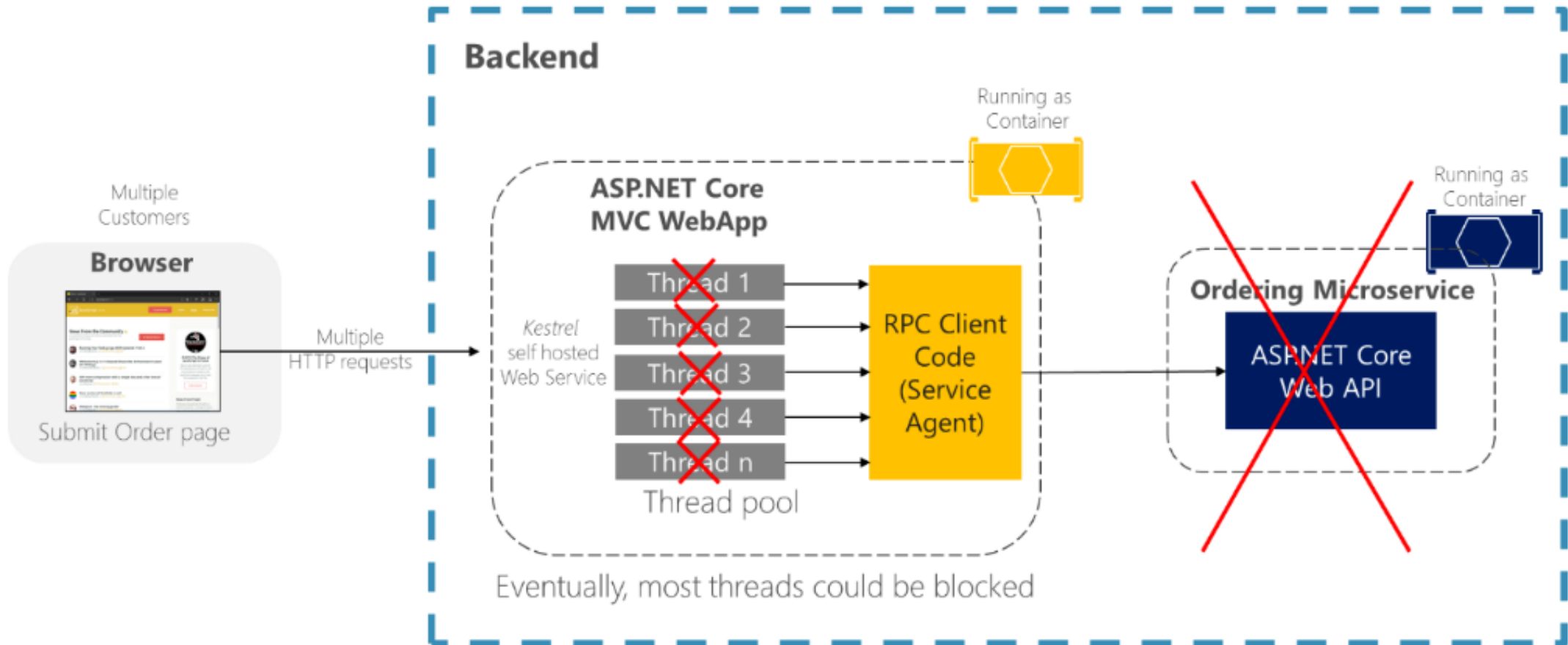




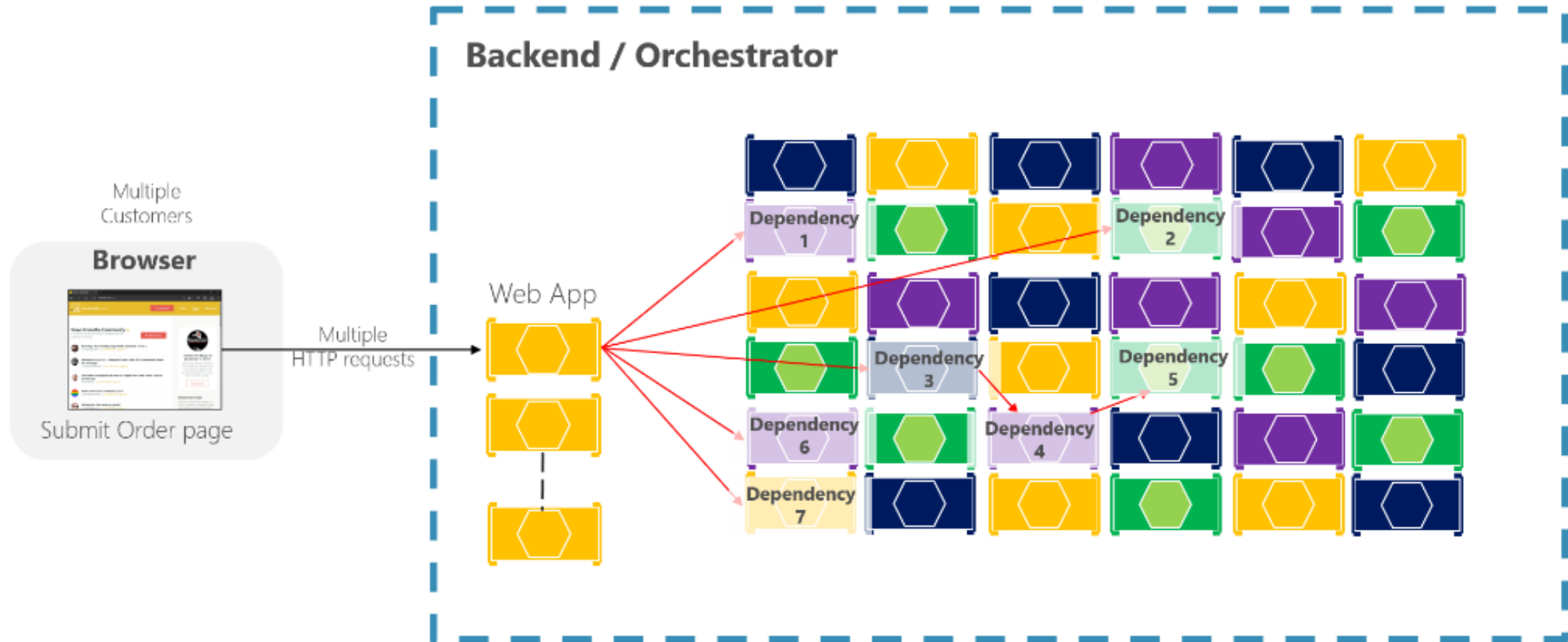
# Microservices = Distributed Systems

## Partial failures



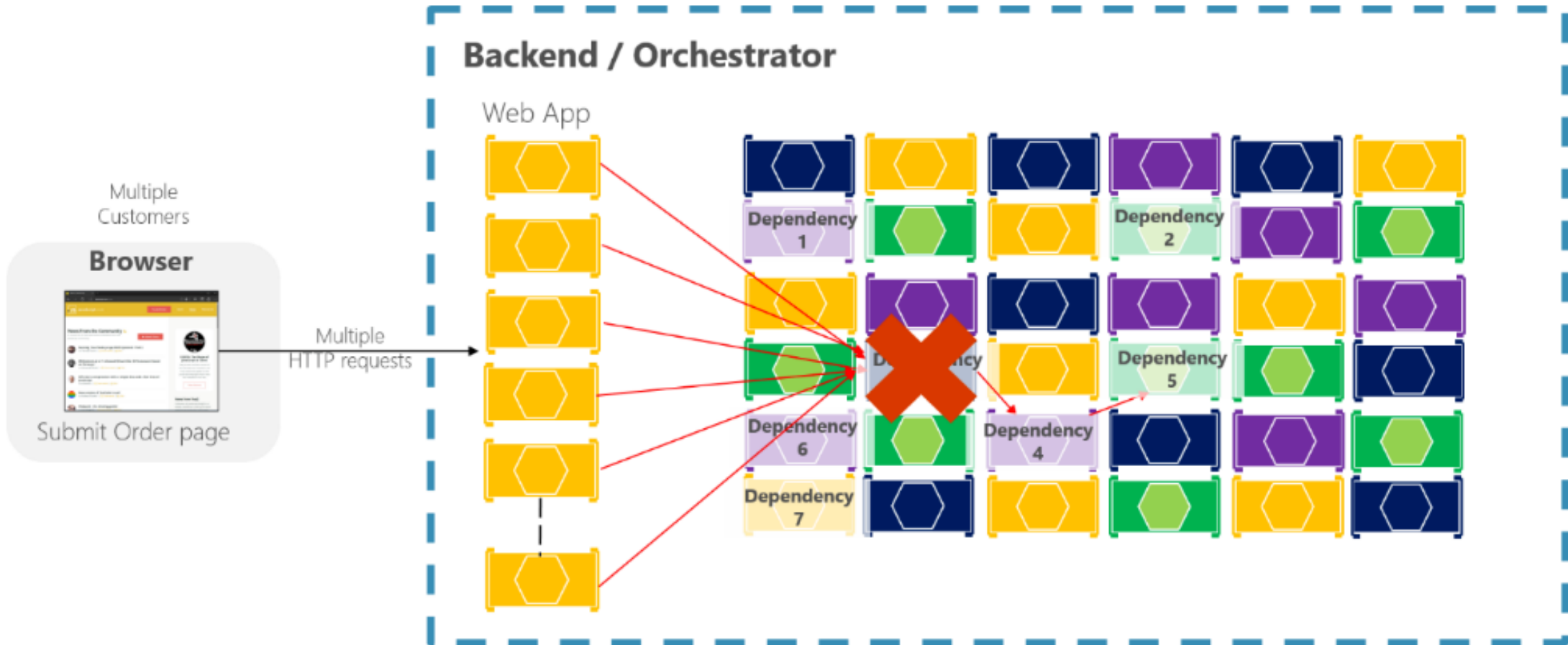
# Distributed Dependencies

## Multiple distributed dependencies

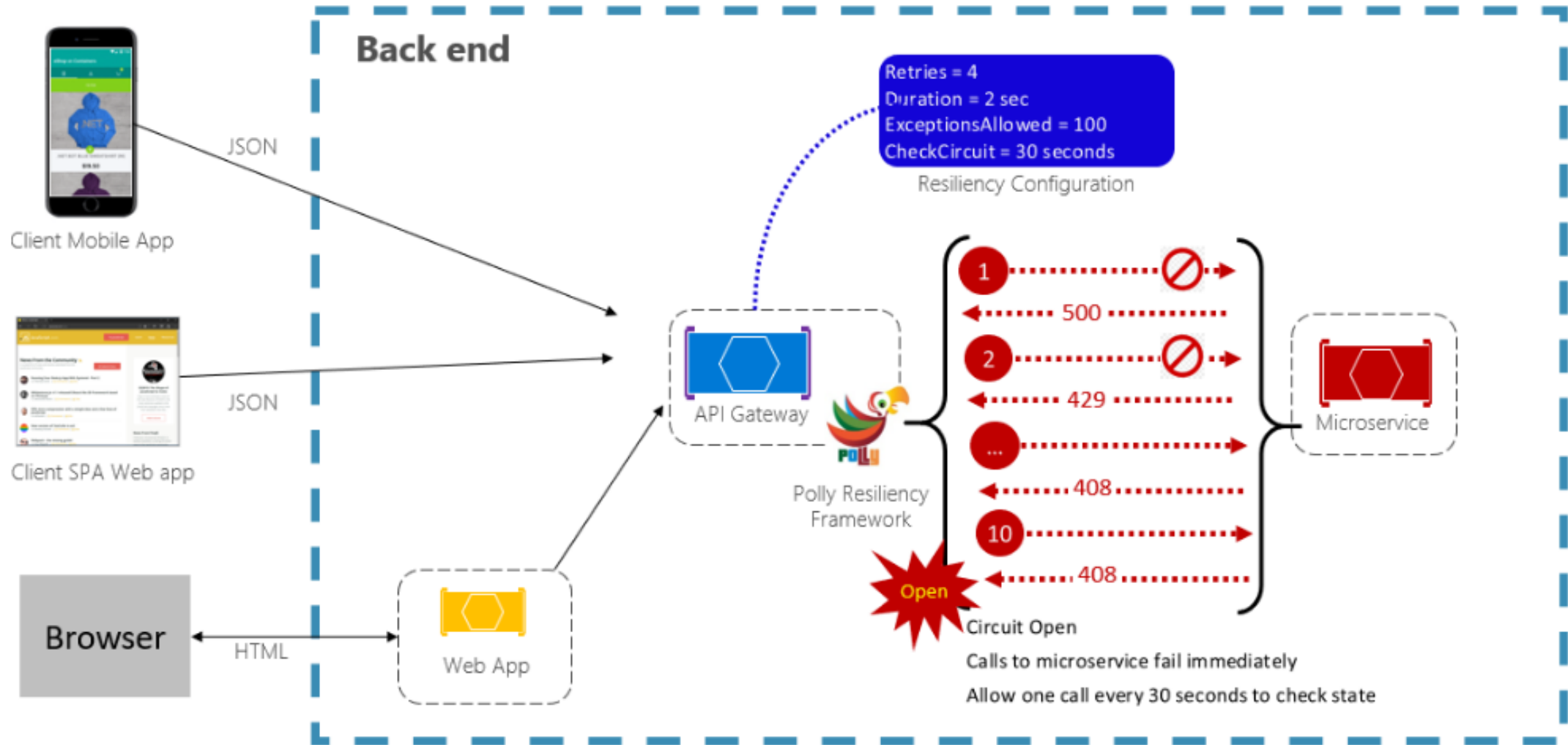


# Dealing with Failures

## Partial Failure Amplified in Microservices

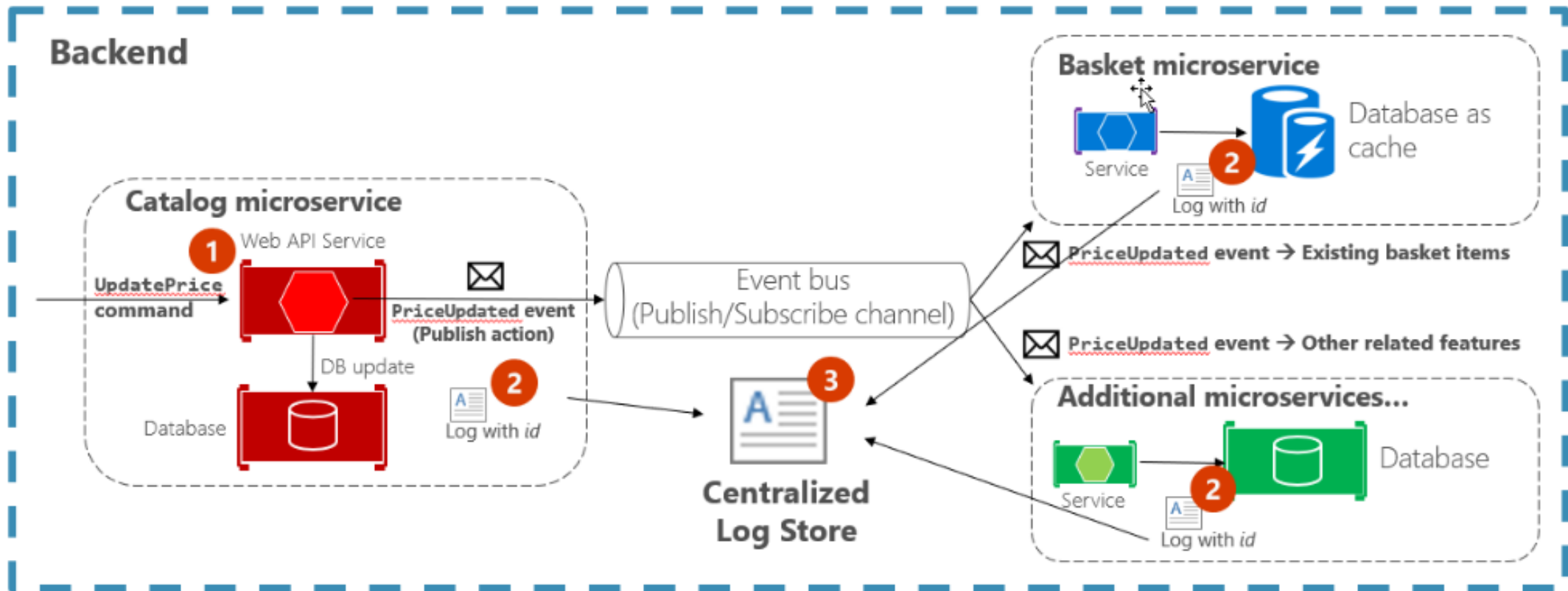


# Microservices Resilience



# Microservices Observability

## Implementing centralized logging



# Microservices Monitoring

The screenshot shows a web browser window titled 'Health Checks UI' with the URL 'localhost:5555/healthchecks-ui#/healthchecks'. The main content area is titled 'Health Checks status' and includes a 'Refresh every 5 seconds' control with a 'Change' button. Below this is a table of health checks:

+	NAME	HEALTH	ON STATE FROM	LAST EXECUTION
-	Todo API	✓ Healthy	Healthy 2 minutes ago	5/3/2020, 6:56:29 PM

NAME	HEALTH	DESCRIPTION	DURATION	DETAILS
todo-db-check	✓ Healthy		00:00:00.0041068	🕒
todo-custom-check	✓ Healthy		00:00:00.0000116	🕒

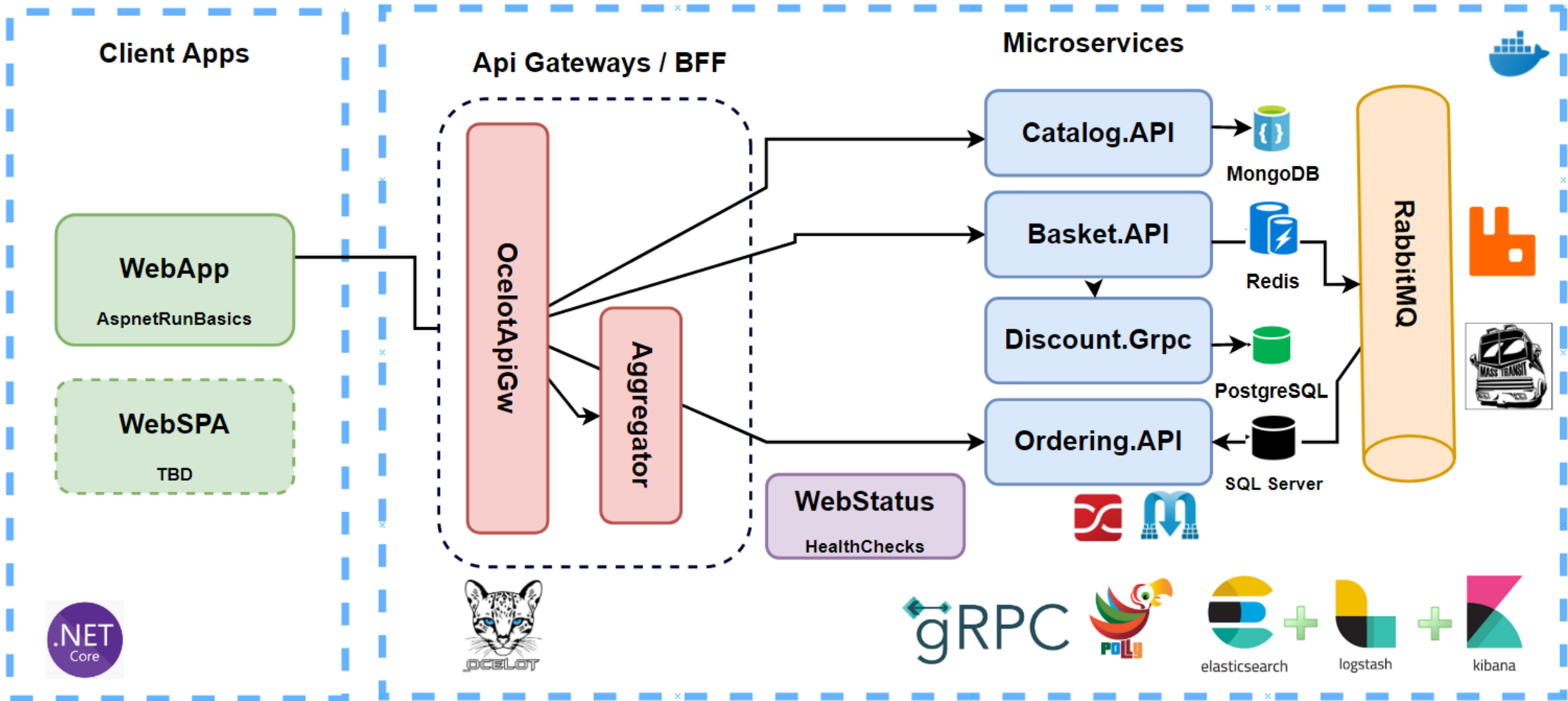
The sidebar on the left contains a hamburger menu icon, a profile picture of a blue horse, and two menu items: 'Health Checks' (with a stethoscope icon) and 'Webhooks' (with a gear icon).

# Microservices Cross-Cutting Concerns

- Microservices Observability with Distributed Logging
- Microservices Resilience and Fault Tolerance with applying Retry and Circuit-Breaker patterns using Polly
- Microservices Monitoring with Health Checks using WatchDog
- Microservices Tracing with OpenTelemetry using Zipkin



# Big Picture



# 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

The screenshot shows the GitHub repository page for 'aspnetrun / run-aspnetcore-microservices'. The repository is sponsored and has 5 users who use it, 6 unwatchers, 92 stars, and 11 forks. The repository description is: 'Building Microservices on .Net platforms which used Asp.Net Web API, Docker, RabbitMQ, Ocelot API Gateway, MongoDB, Redis, SqlServer, Entity Framework Core, CQRS and Clean Architecture implementation. Download Microservices Architecture and Step by Step Implementation on .NET Book -> <https://aspnetrun.azurewebsites.net/M...>'. The repository has 87 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. The license is MIT. The repository is tagged with various topics such as aspnet-core, docker, ocelot-gateway, micorservices, aspnetcore-microservices, rabbitmq, mongodb, redis, sql-server, cqrs-pattern, clean-architecture, event-sourcing, eventbus, event-driven, microservices-architecture, aspnet-web-api, swagger, rest-api, api-gateway, and mediator-pattern.

**[Final Application - https://github.com/aspnetrun/run-aspnetcore-microservices](https://github.com/aspnetrun/run-aspnetcore-microservices)**

**[Base Application - https://github.com/mehmetozkaya/AspnetMicroservices](https://github.com/mehmetozkaya/AspnetMicroservices)**

# Tools that we will use



.NET 5.x or above SDK



Visual Studio 2019 v16.x or above



Docker Desktop



Postman



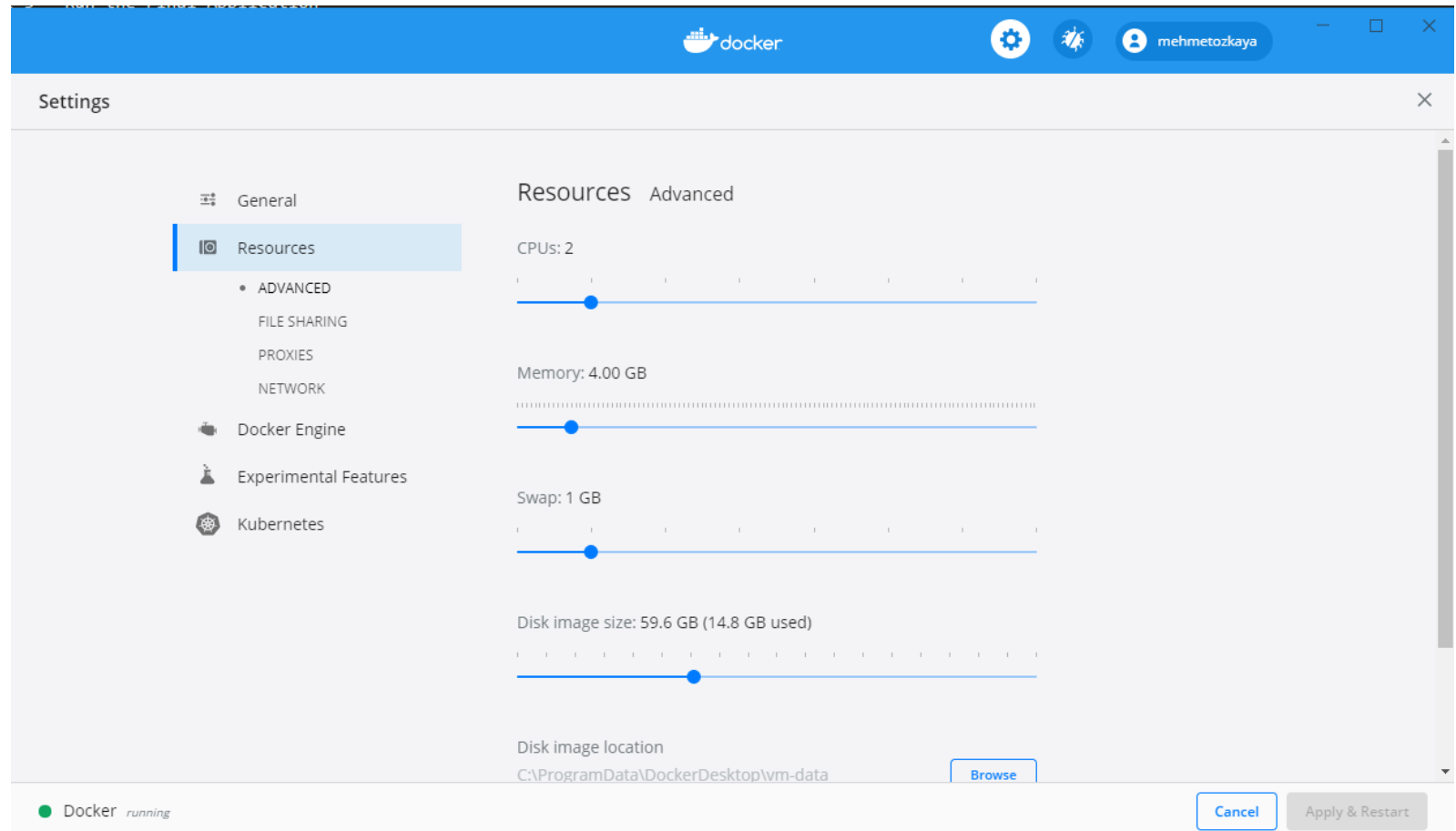
Windows 10

Pro 64Bit

Microsoft

# Configure Docker Desktop

- Once Docker for Windows is installed, go to the **Settings** > **Advanced** option
- Memory – 4 GB
- CPU - 2



# Run Base Application

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

The screenshot shows a web application interface for 'AspnetRunBasics'. The top navigation bar includes links for Home, Product, Cart, Order, and Contact, along with a search bar and a cart icon showing 3 items. The main content area features a large banner for 'OPPO Find X Newone' with an 'Explore More' button and the tagline 'On Your First Choice'. To the right, a 'TOP PRODUCT' section highlights the 'iPhone X' for 950.00 \$, with a 'View' button. Below this, a 'LAST PRODUCTS' section displays four product cards: iPhone X, Samsung 10, Huawei Plus, and Xiaomi Mi 9. Each card includes a product image, the product name, and a placeholder description: 'This phone is the company's biggest change to its flagship smartphone'.

[Base Application - https://github.com/mehmetozkaya/AspnetMicroservices](https://github.com/mehmetozkaya/AspnetMicroservices)

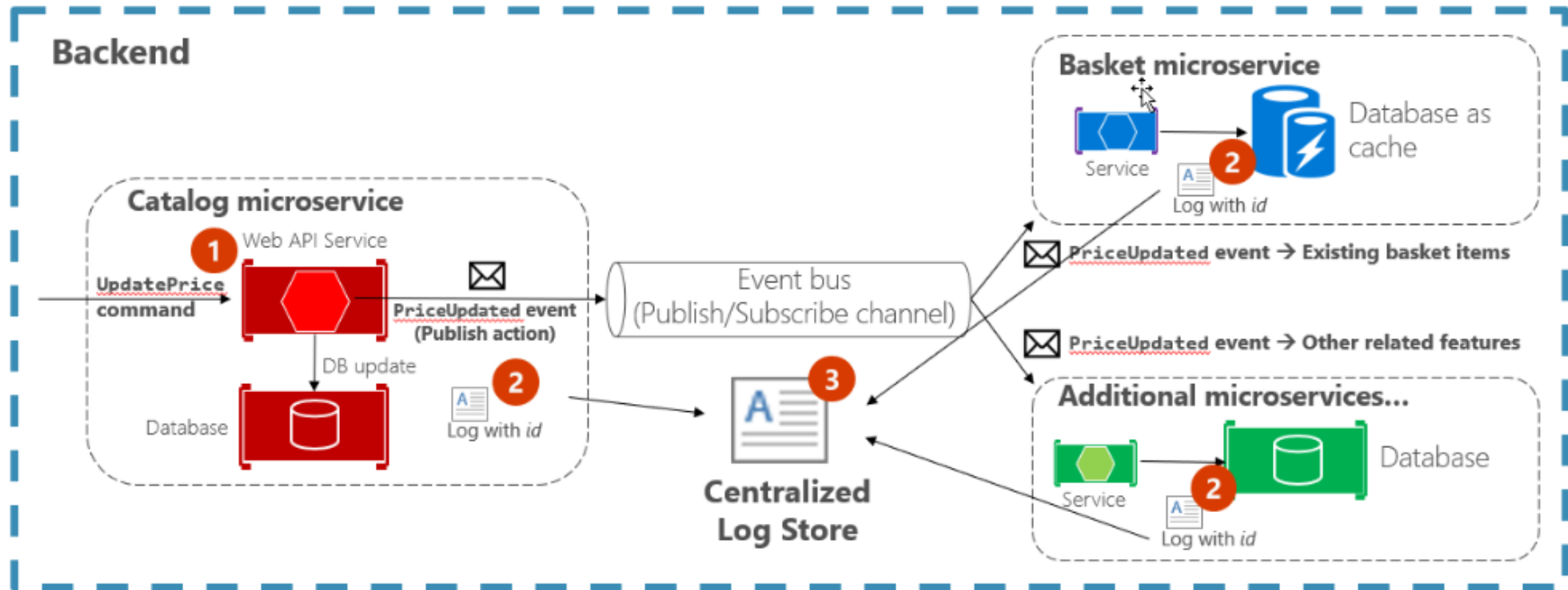
# Section 2

# Microservices Observability with Distributed Logging

Applying Elastic Stack which includes elasticsearh + logstach + kibana  
and SeriLog nuget package for .net microservices

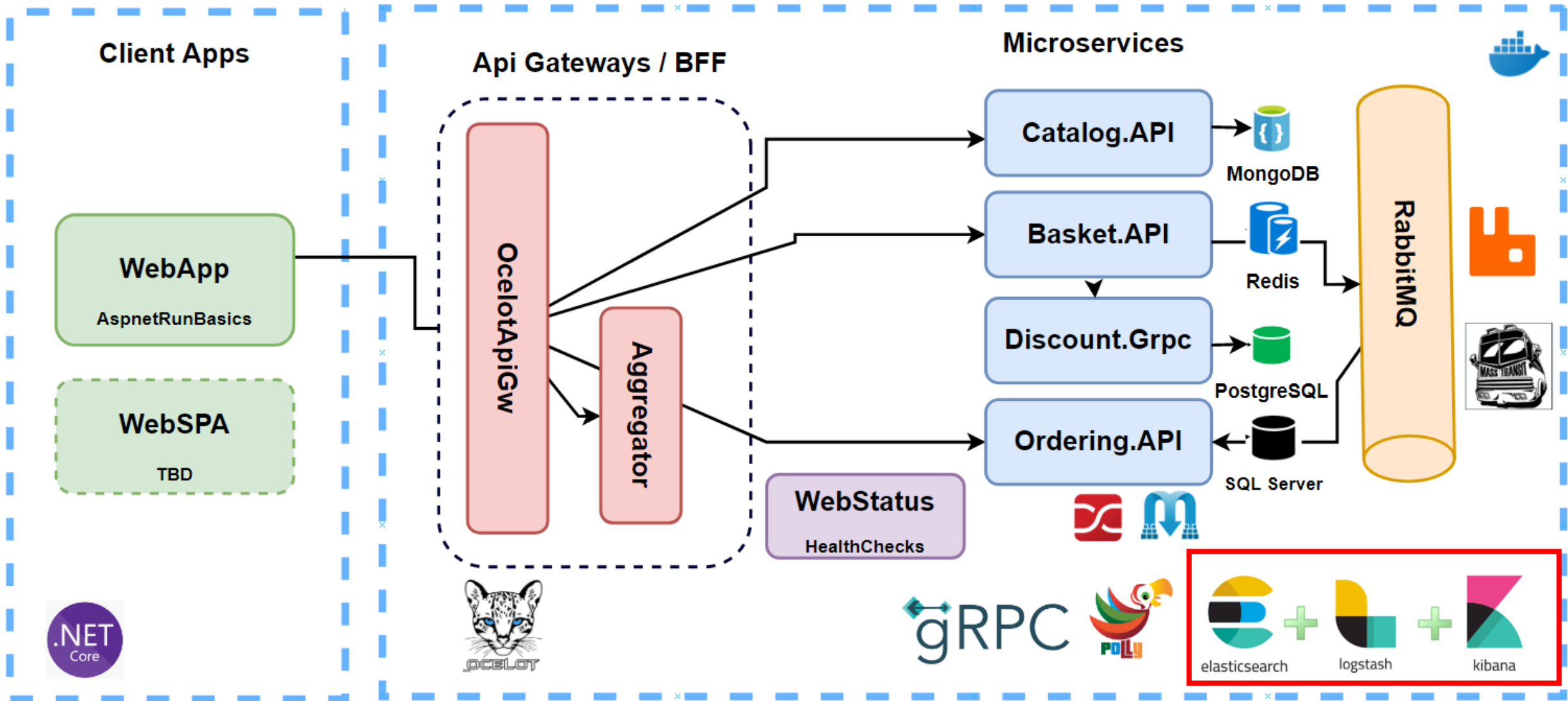
# Microservices Observability

## Implementing centralized logging



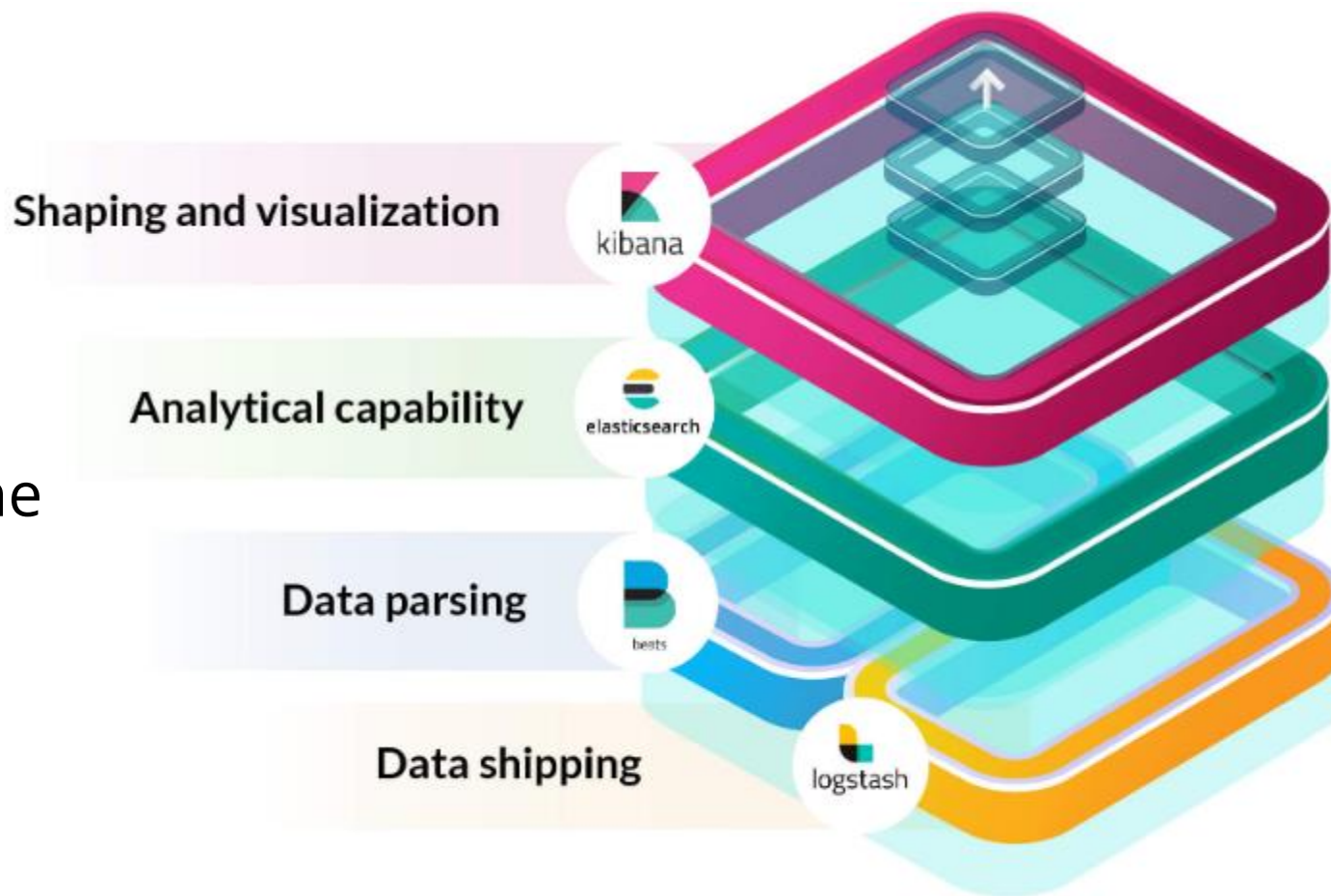


# Big Picture



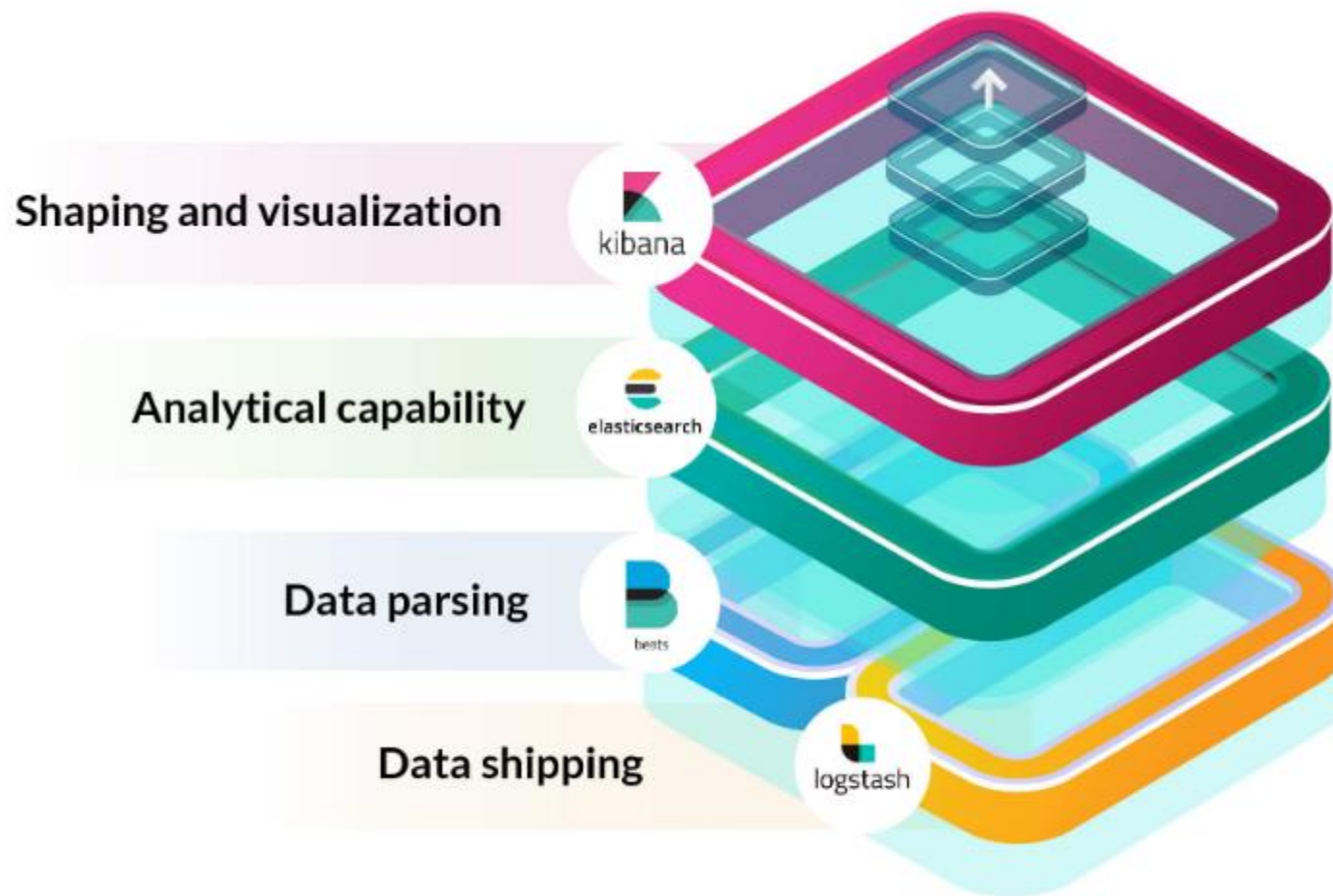
# What is Elastic Search ?

- Distributed, open-source search and analytics engine
- Built on Apache Lucene
- Full-text search search engine
- Indexing logs and analytical data



# Why is Elasticsearch so popular?

- Free and open source
- RESTful API
- Easy To Query
- Very fast
- Scalable
- Easy to Install



# What is Kibana ?



through multiple nodes by separating them into a single flow

# What is Serilog ?

- ASP.NET Logging Library
- Sink to ElasticSearch
- Structured Logs



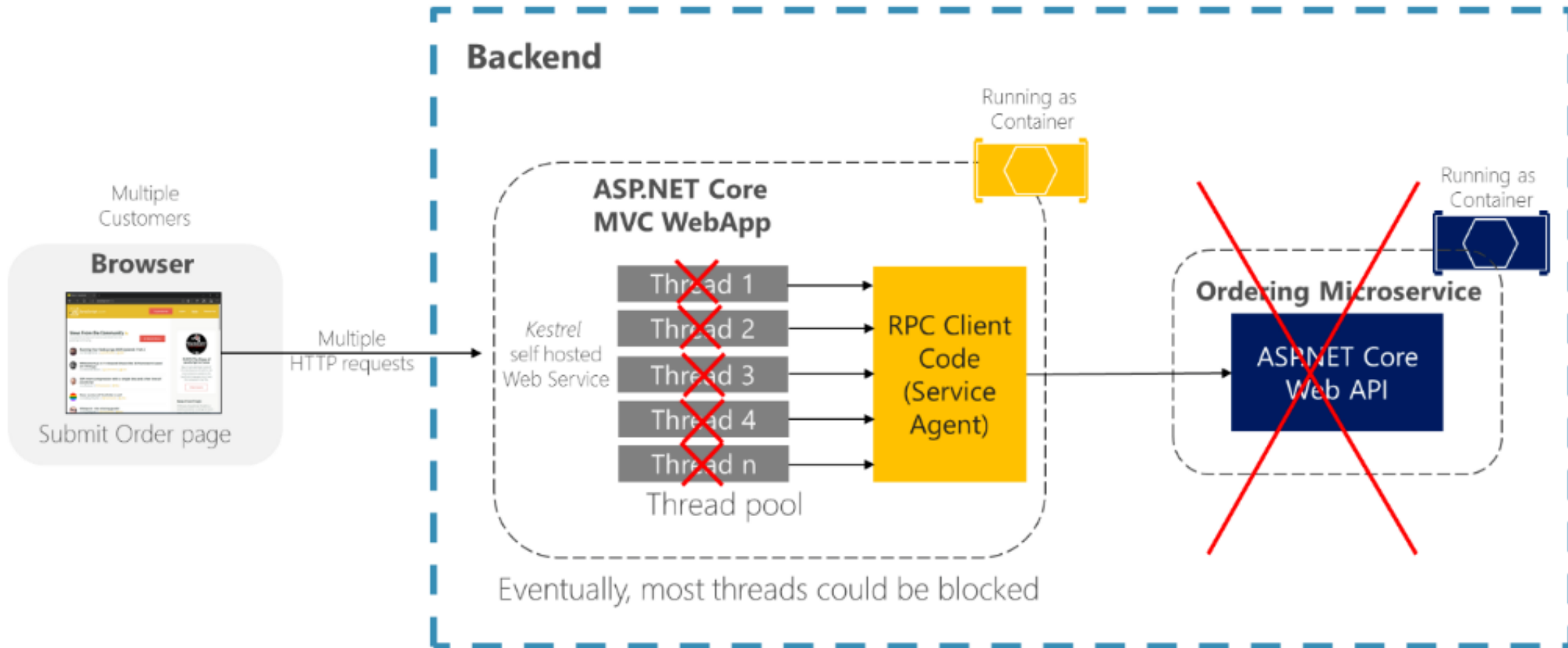
# Section 3

# Microservices Resilience and Fault Tolerance

Applying Retry and Circuit-Breaker patterns using Polly

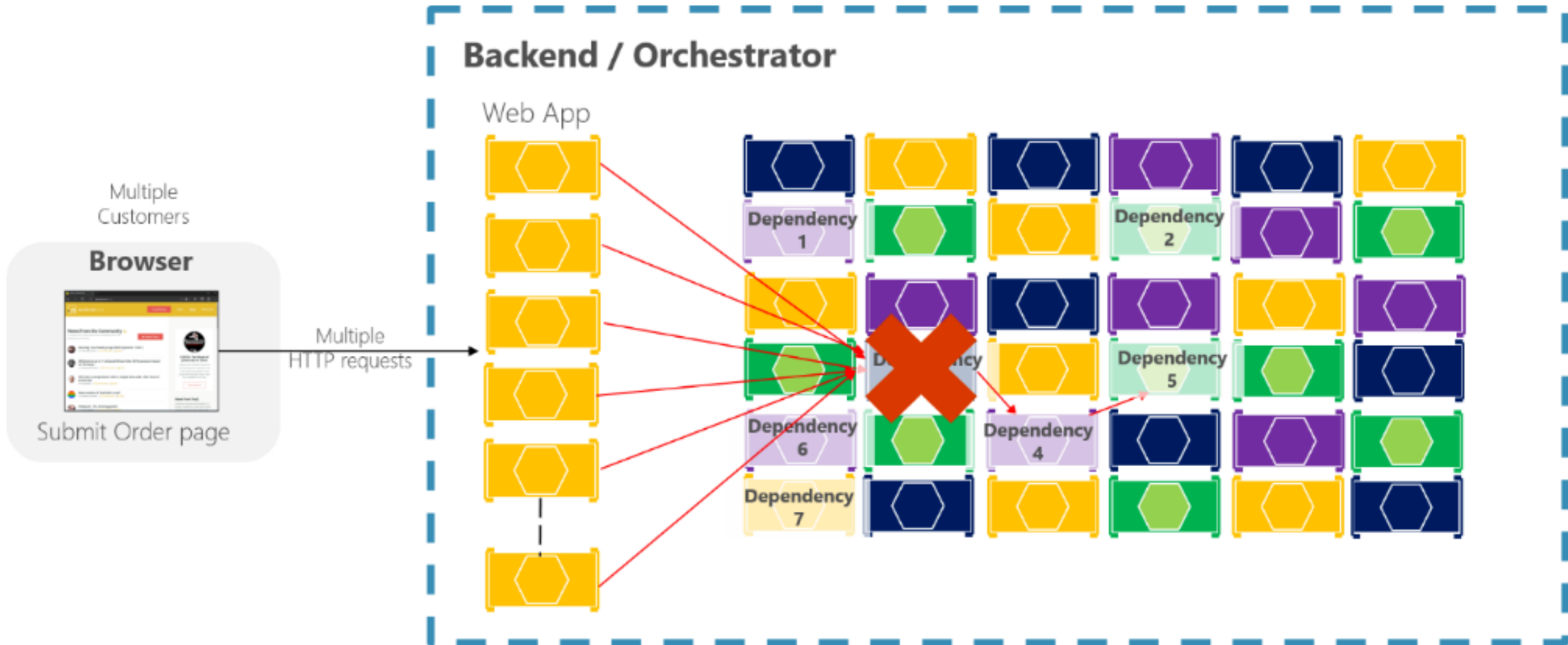
# Microservices = Distributed Systems

## Partial failures



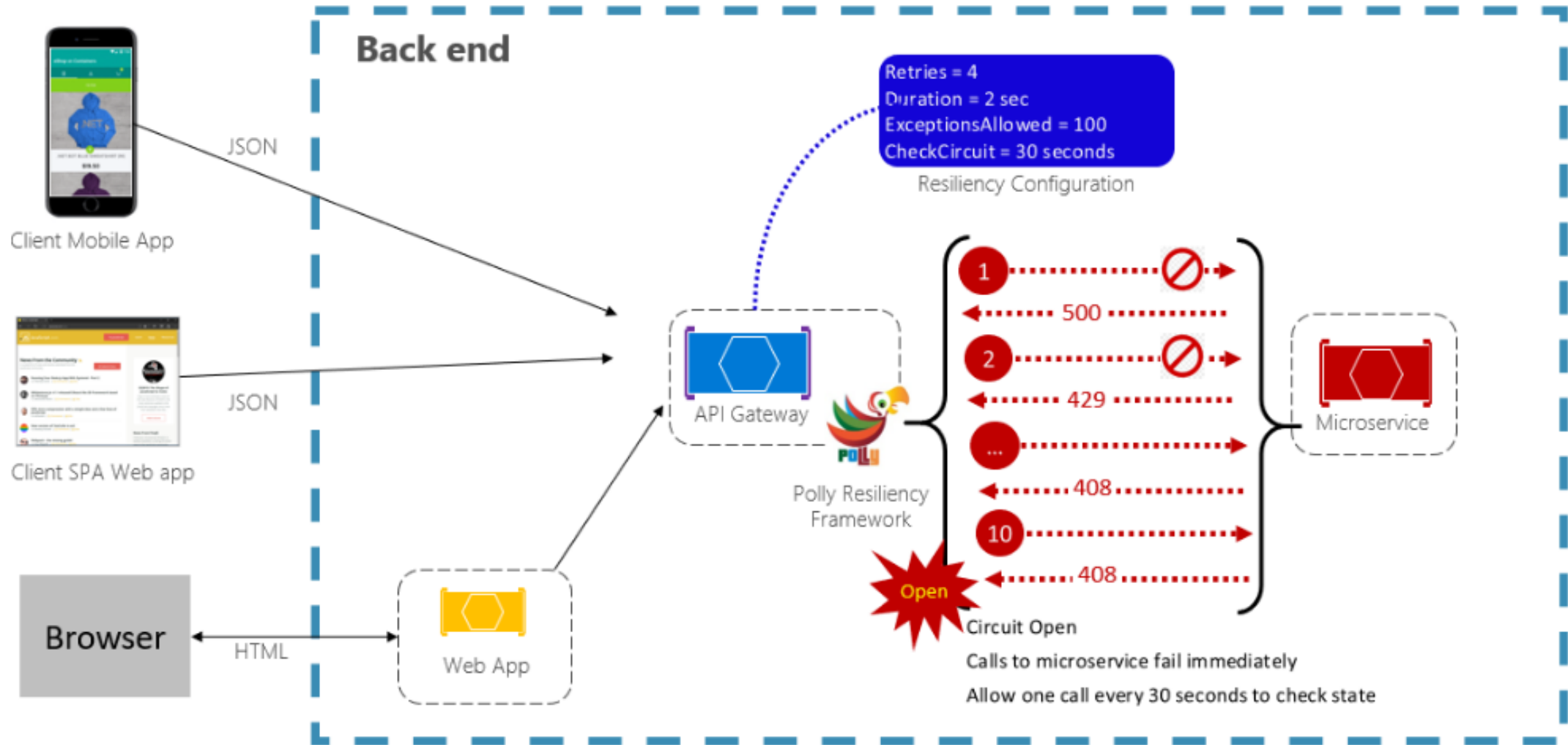
# Dealing with Failures

## Partial Failure Amplified in Microservices

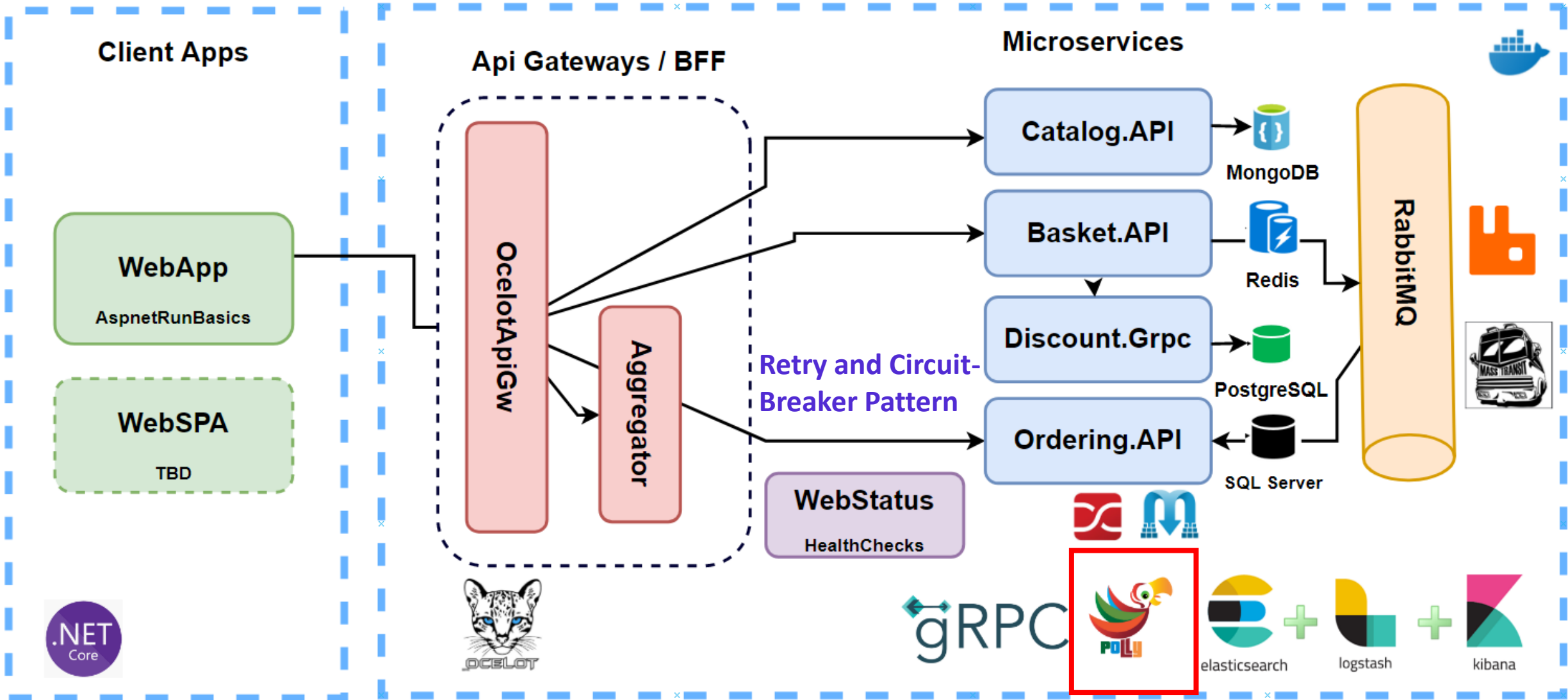




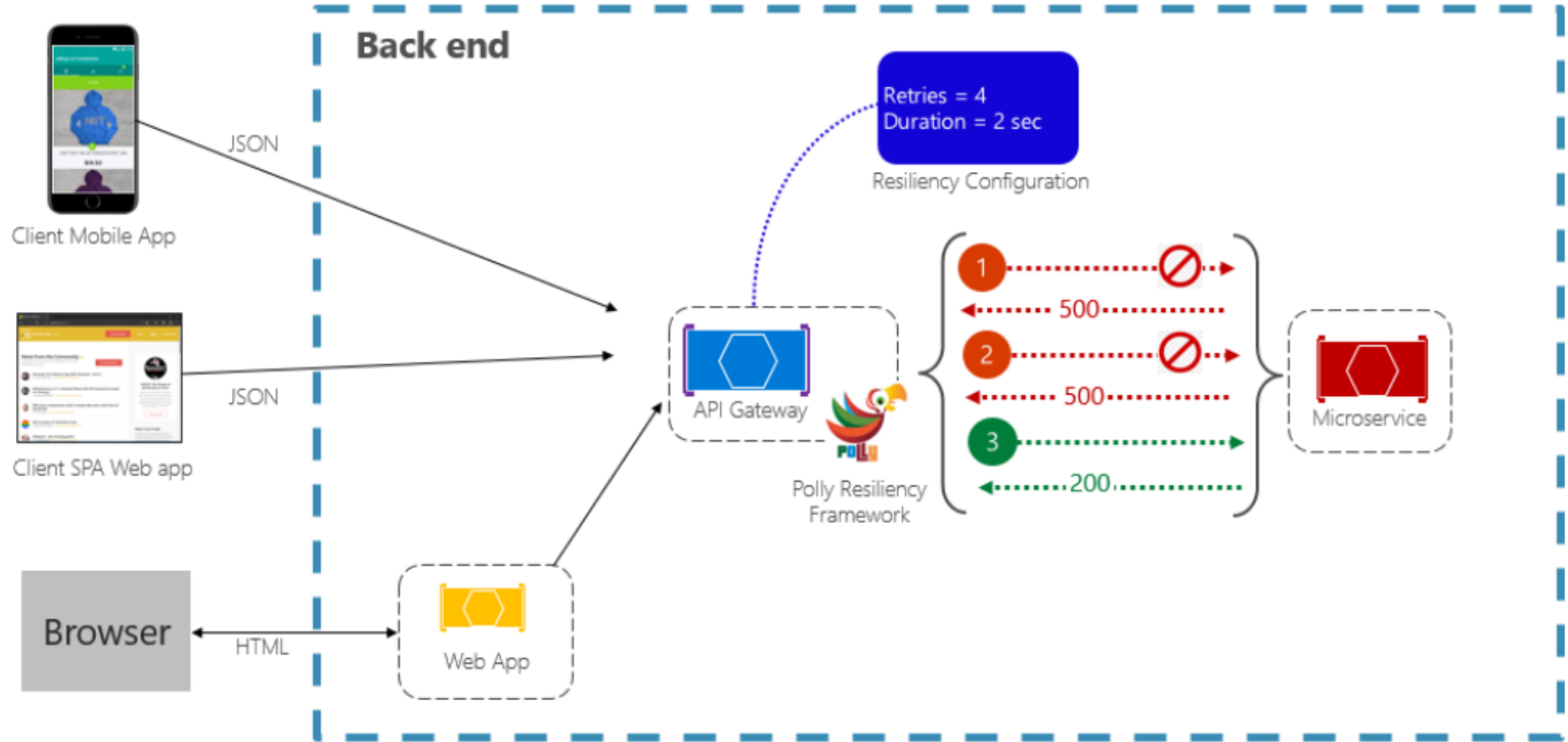
# Microservices Resilience



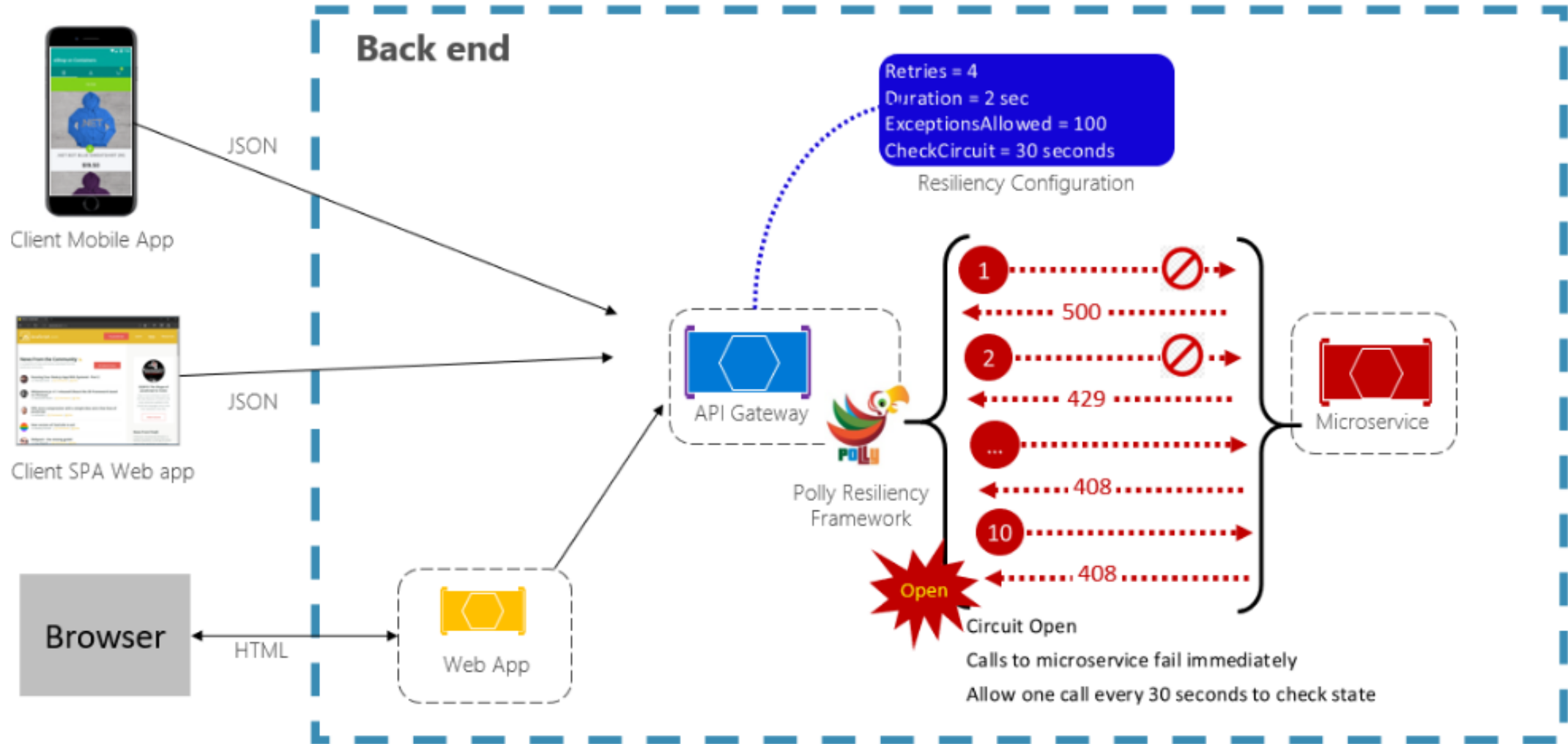
# Big Picture



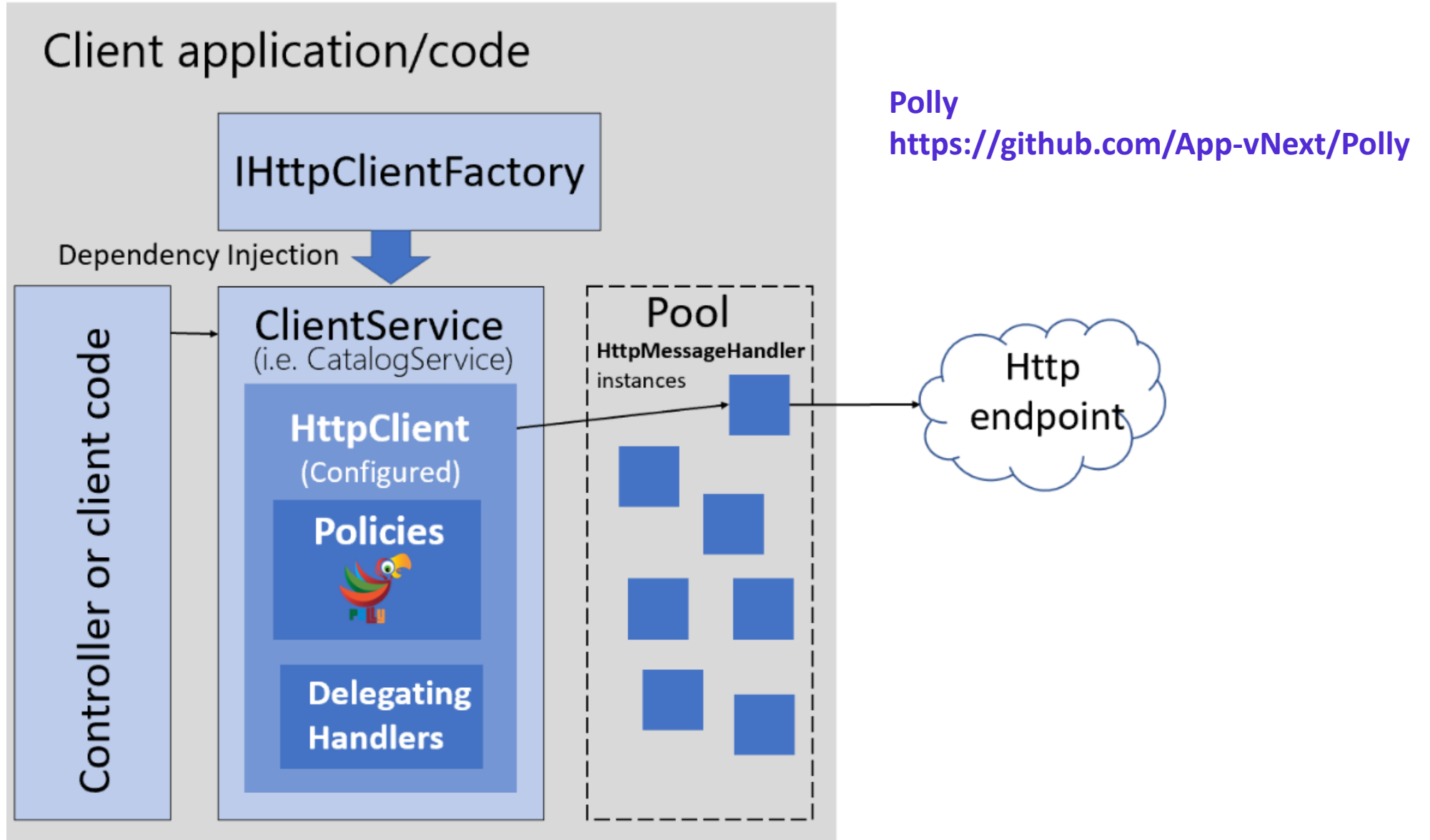
# Microservices Resilience - Retry



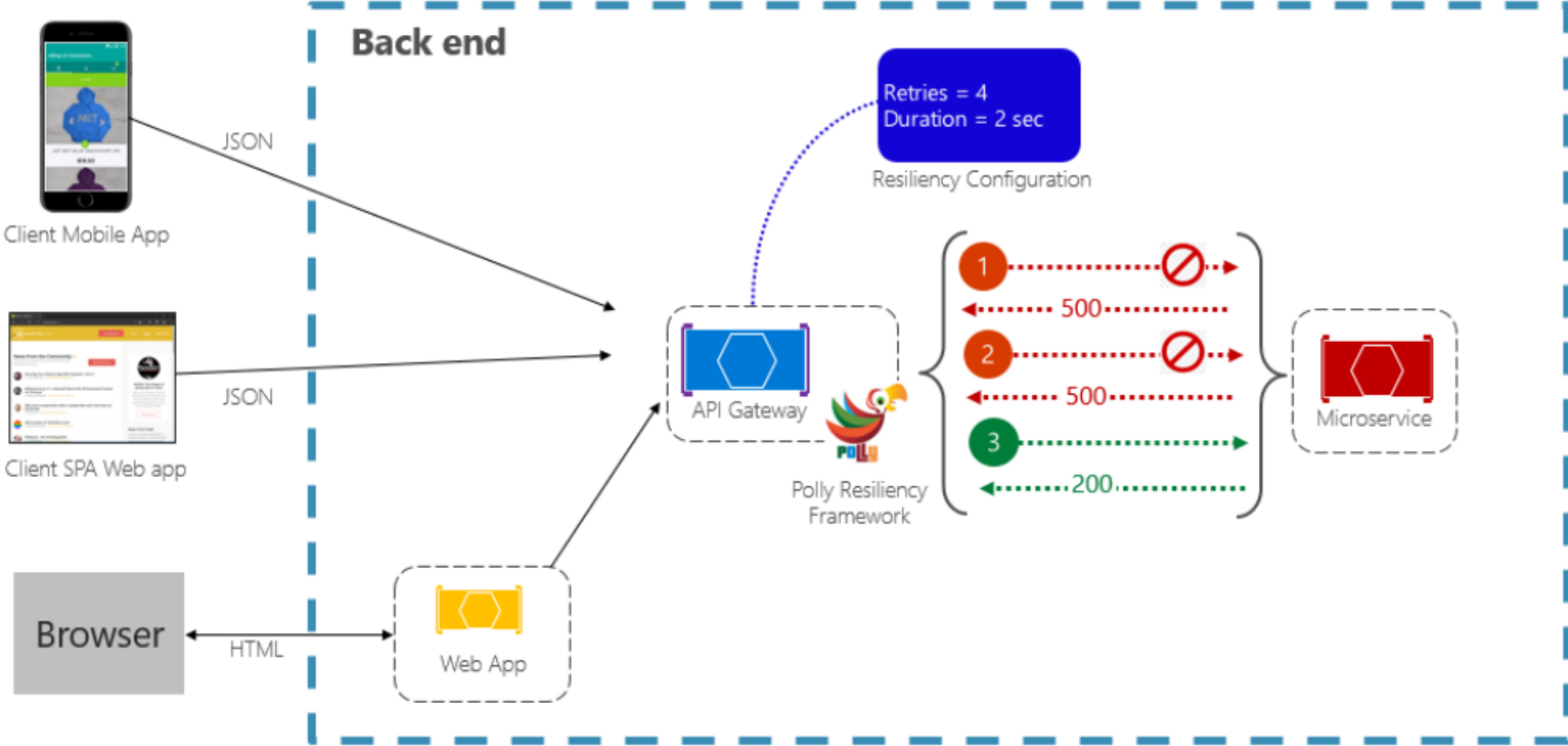
# Microservices Resilience – Circuit Breaker



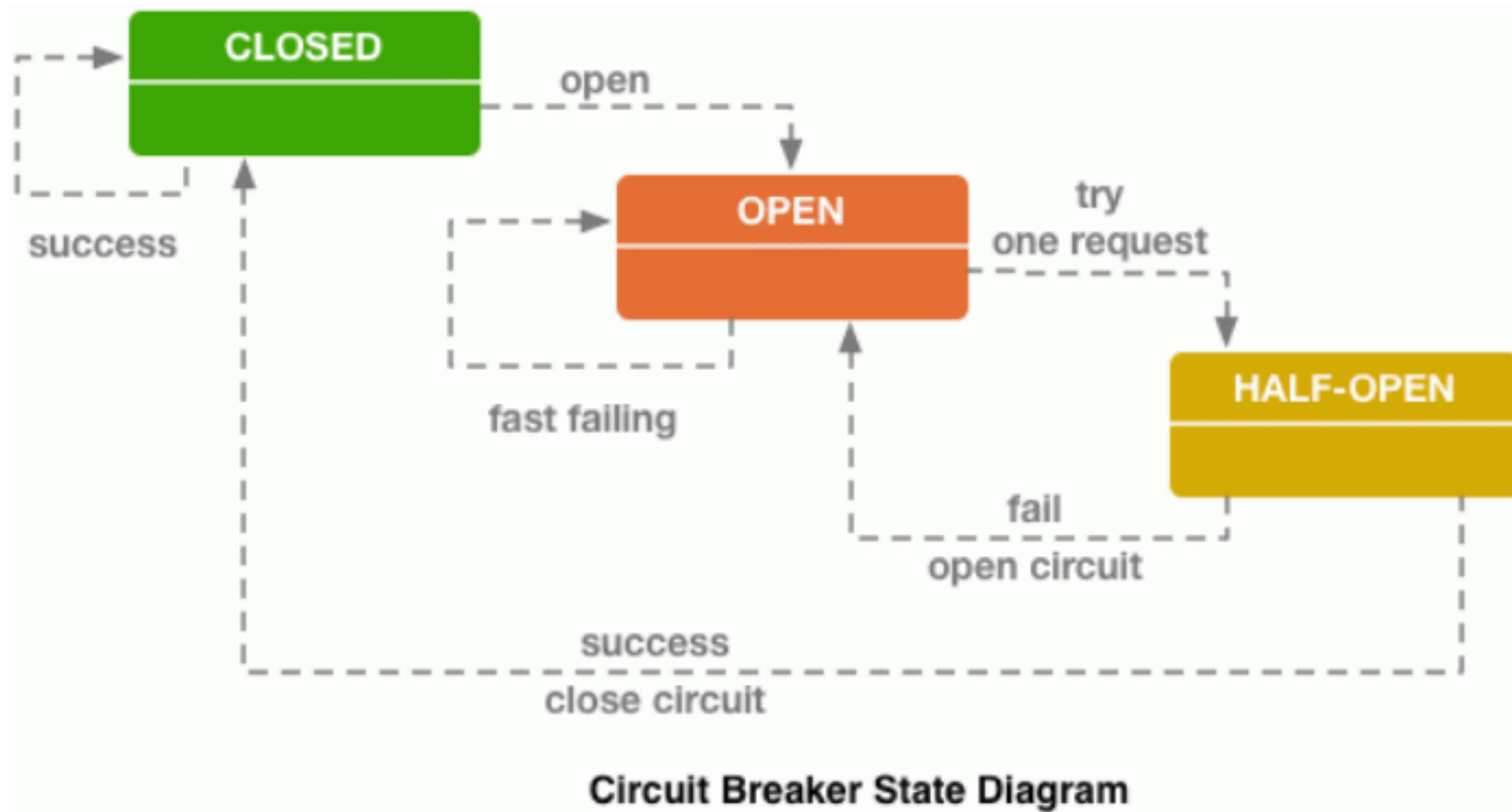
# Polly Policies



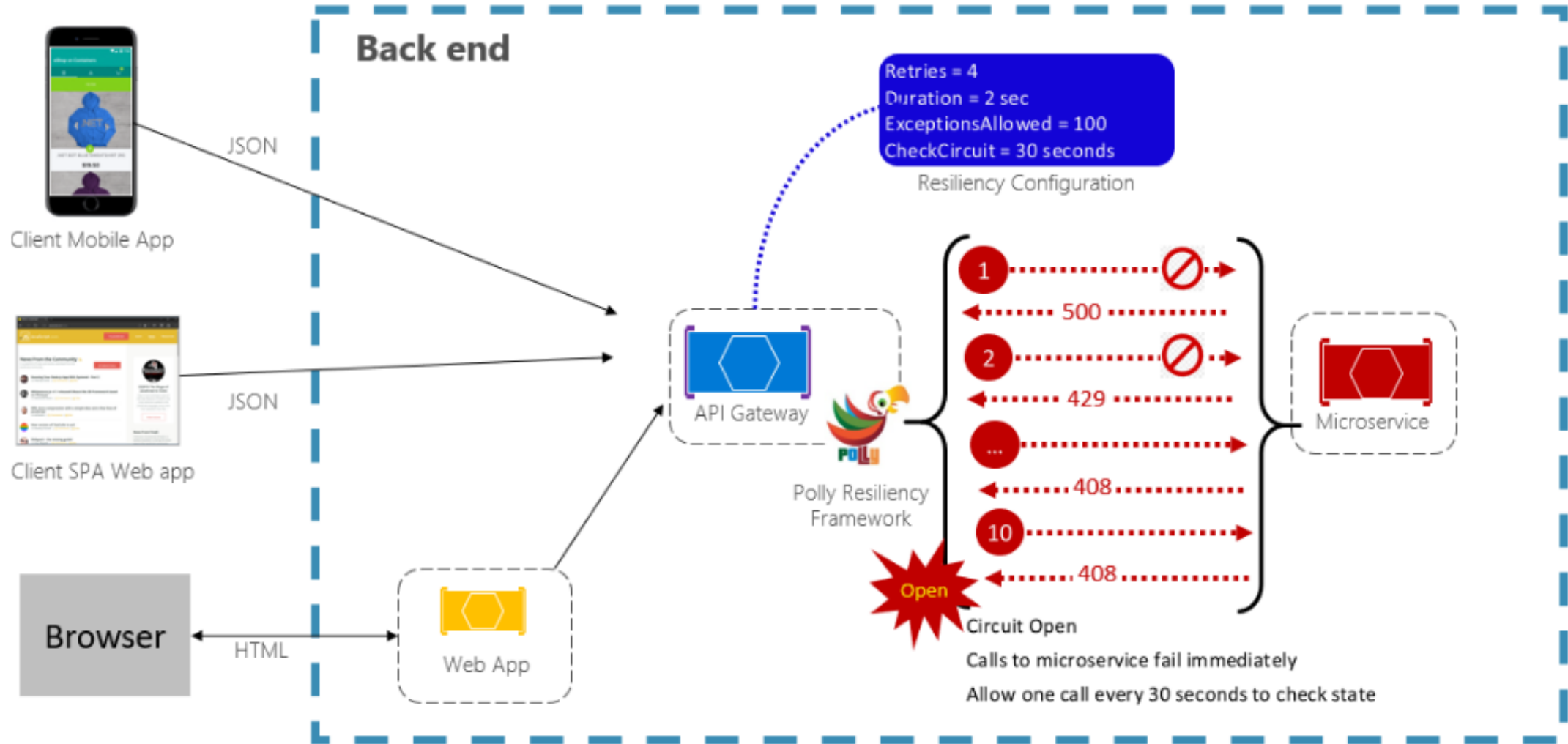
# Retry Pattern



# Circuit Breaker Pattern

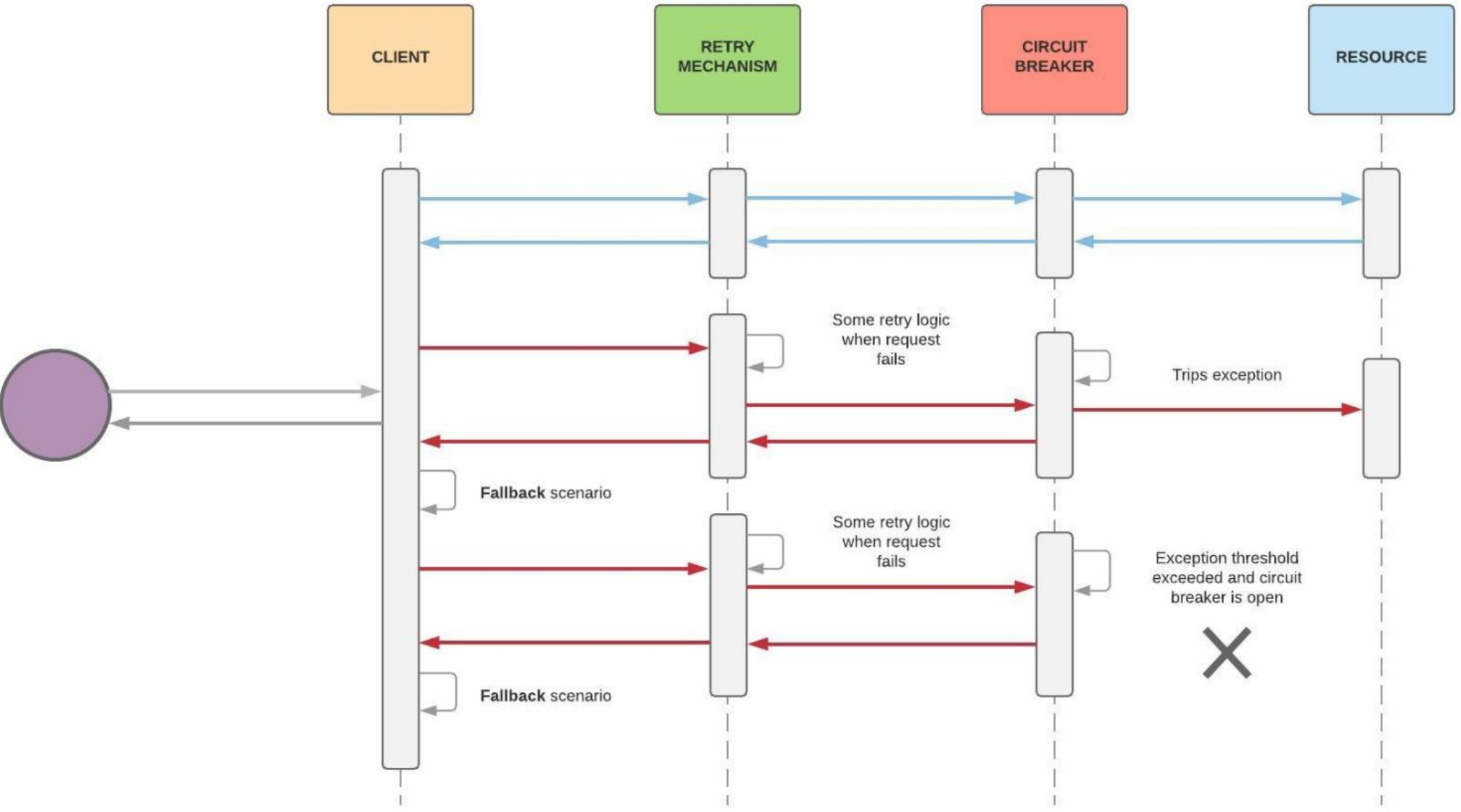


# Circuit Breaker Pattern

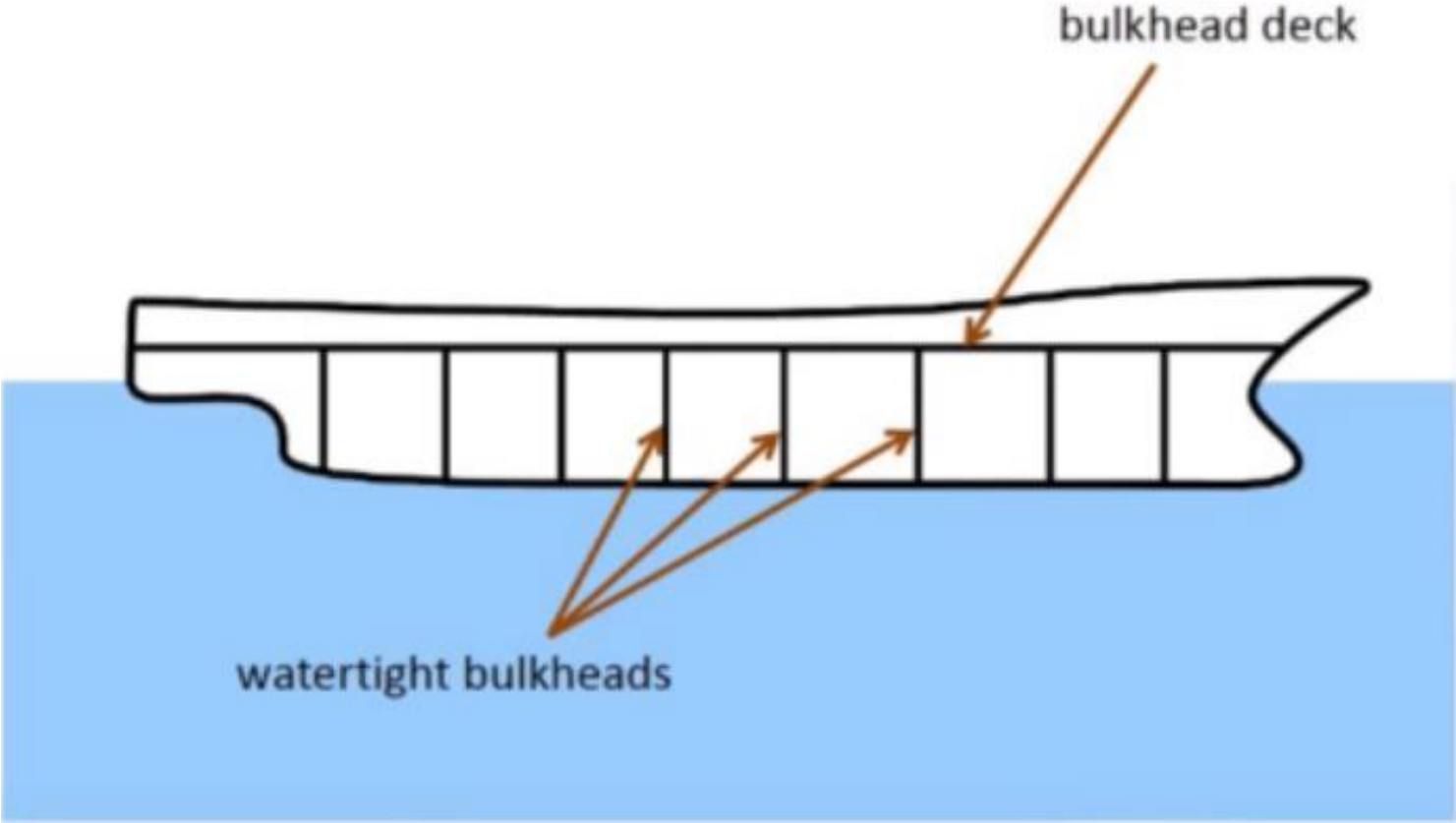




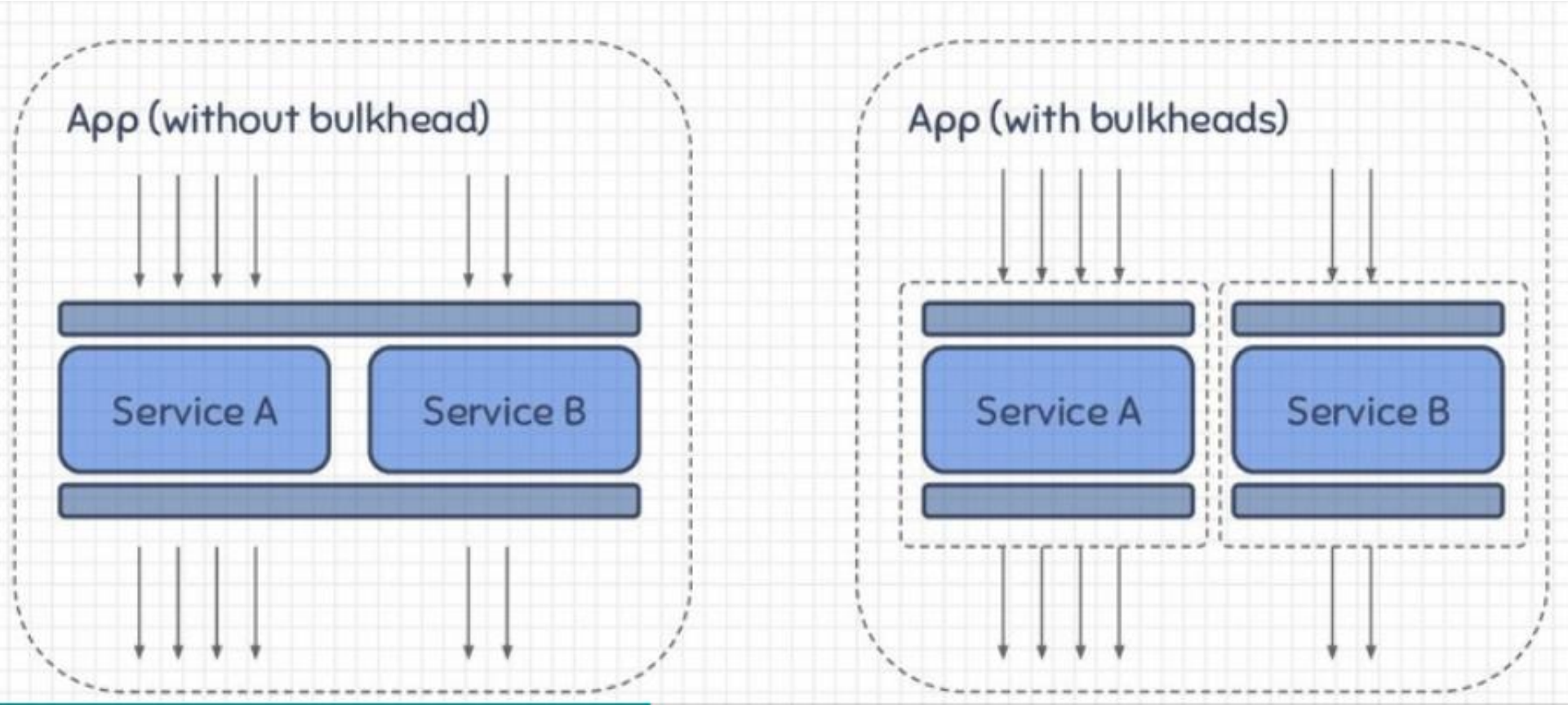
# Retry + Circuit Breaker Pattern



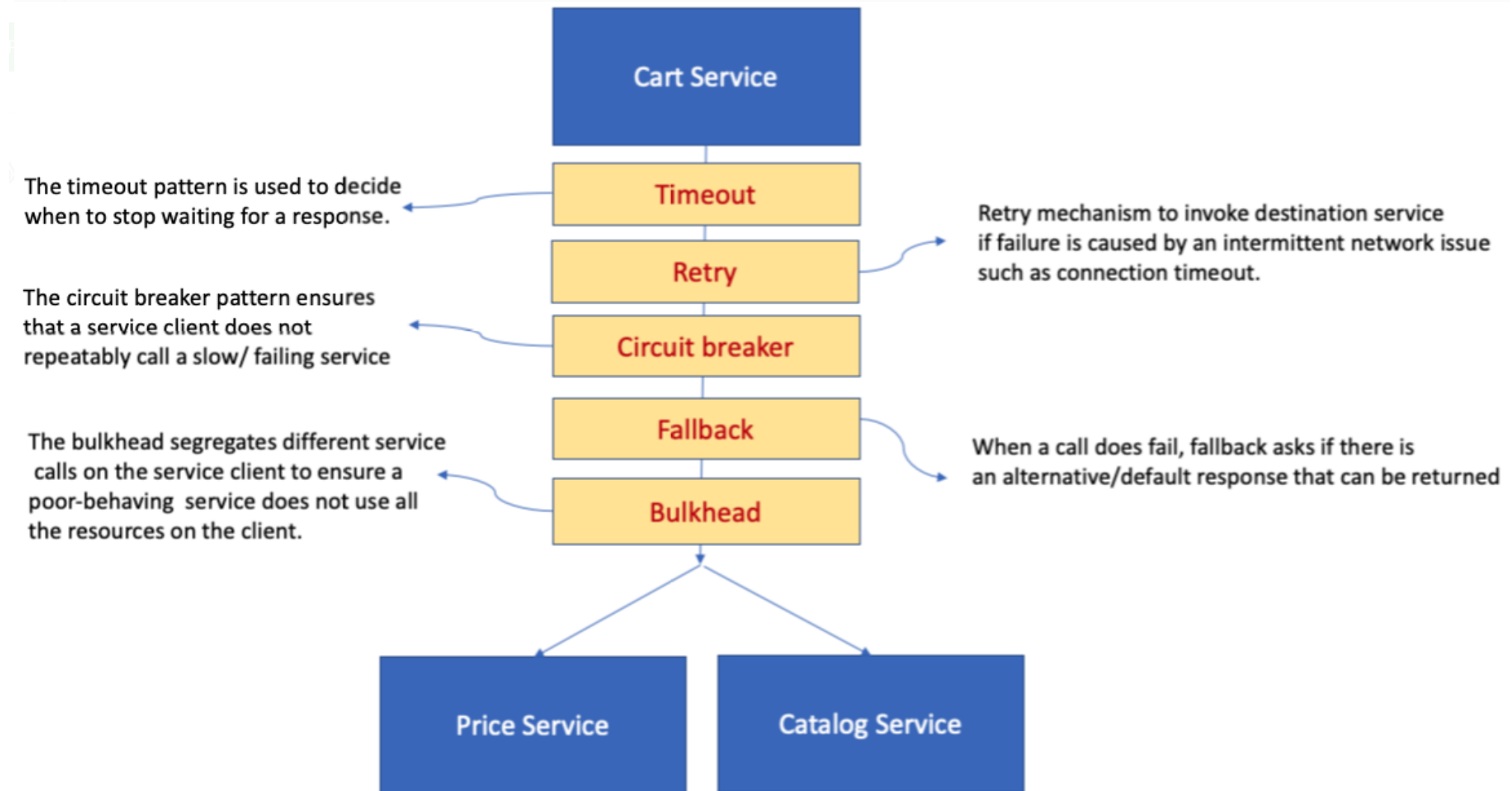
# Bulkhead Pattern



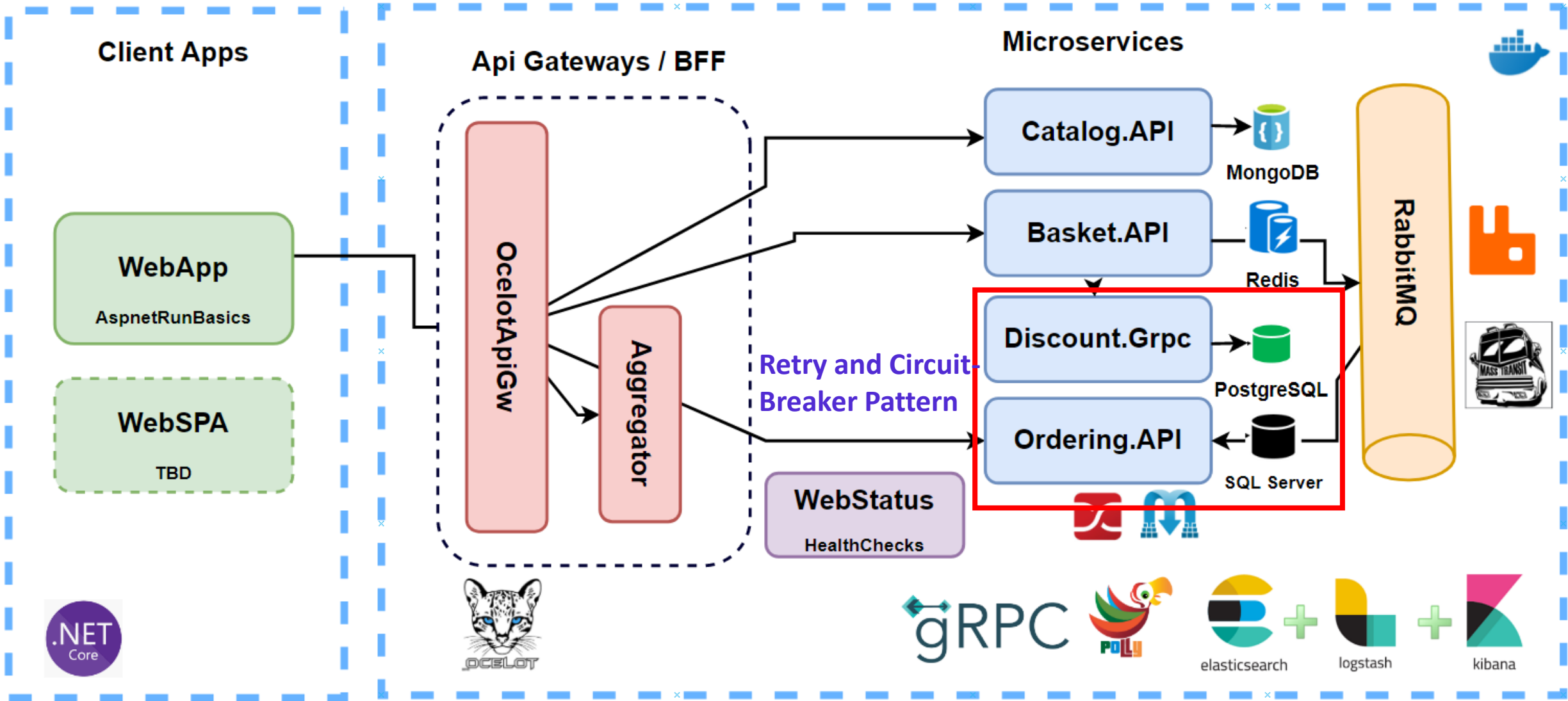
# Bulkhead Pattern



# Order of Resilience Patterns



# Database Migration Retries



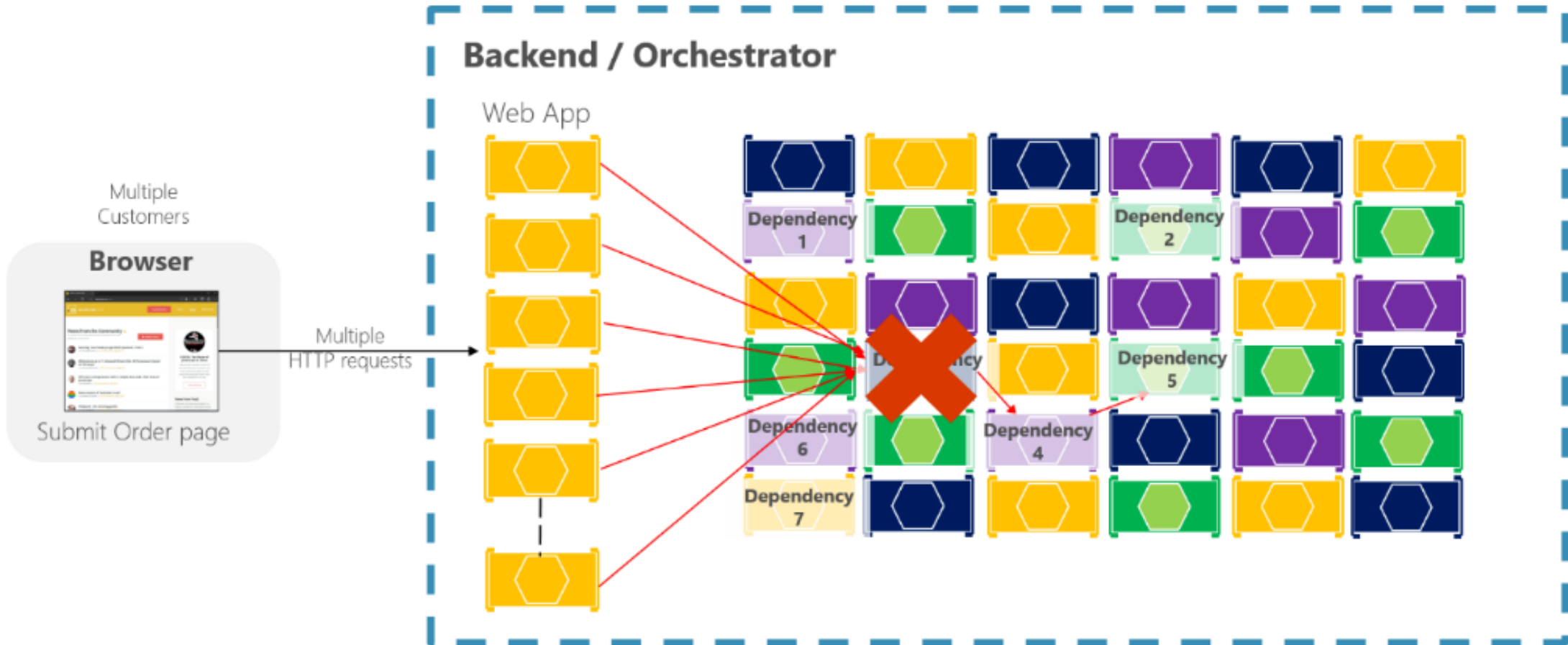
# Section 4

## Microservices Monitoring with Health Checks using WatchDog

Use the HealthChecks feature in your back-end ASP.NET microservices

# Microservice Health Monitoring

## Partial Failure Amplified in Microservices



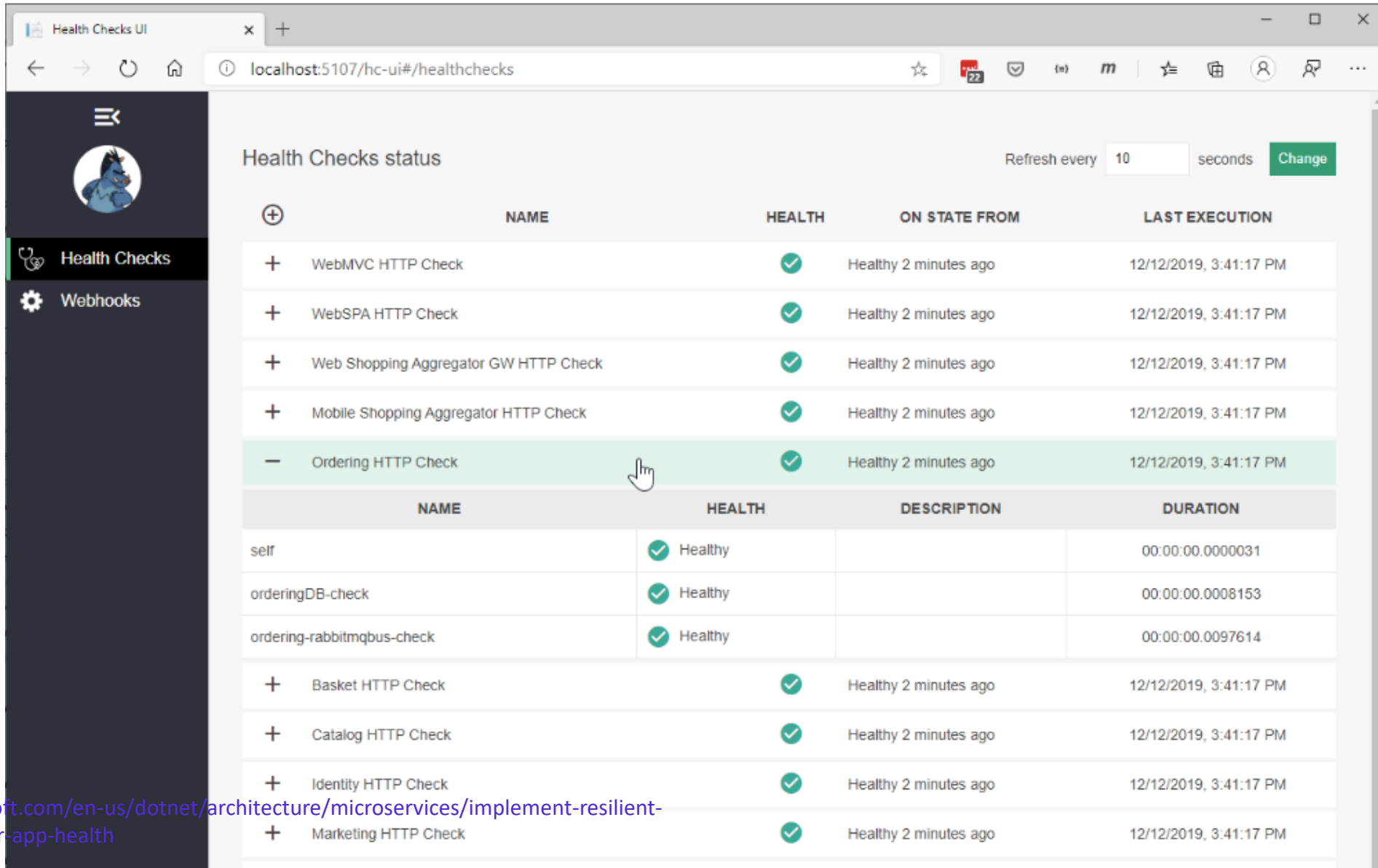
# DEVELOPERS WITHOUT MONITORING







# WatchDog Health Dashboard



The screenshot shows a web browser window titled "Health Checks UI" with the address bar displaying "localhost:5107/hc-ui#/healthchecks". The dashboard features a dark sidebar on the left with a menu icon, a donkey logo, and two menu items: "Health Checks" (selected) and "Webhooks". The main content area is titled "Health Checks status" and includes a "Refresh every" dropdown set to "10" seconds and a "Change" button. Below this is a table of health checks with columns for NAME, HEALTH, ON STATE FROM, and LAST EXECUTION. The "Ordering HTTP Check" row is highlighted in green and has a hand cursor over it. Below the main table is a detailed view for the selected check, showing columns for NAME, HEALTH, DESCRIPTION, and DURATION.

+	NAME	HEALTH	ON STATE FROM	LAST EXECUTION
+	WebMVC HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	WebSPA HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Web Shopping Aggregator GW HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Mobile Shopping Aggregator HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
-	Ordering HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM

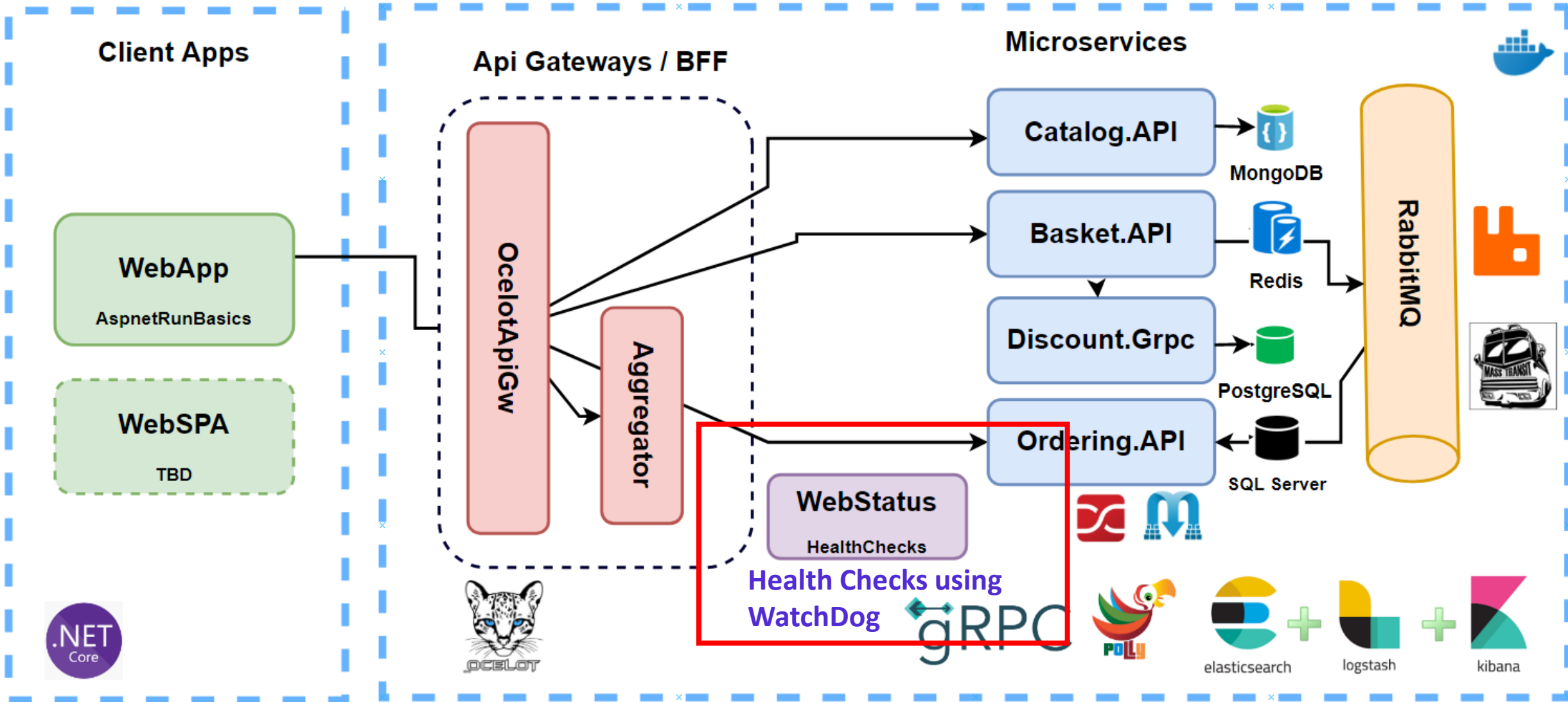
  

NAME	HEALTH	DESCRIPTION	DURATION
self	✓ Healthy		00:00:00.0000031
orderingDB-check	✓ Healthy		00:00:00.0008153
ordering-rabbitmqbus-check	✓ Healthy		00:00:00.0097614

+	Basket HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Catalog HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Identity HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Marketing HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM

# Microservices Monitoring with Health Checks



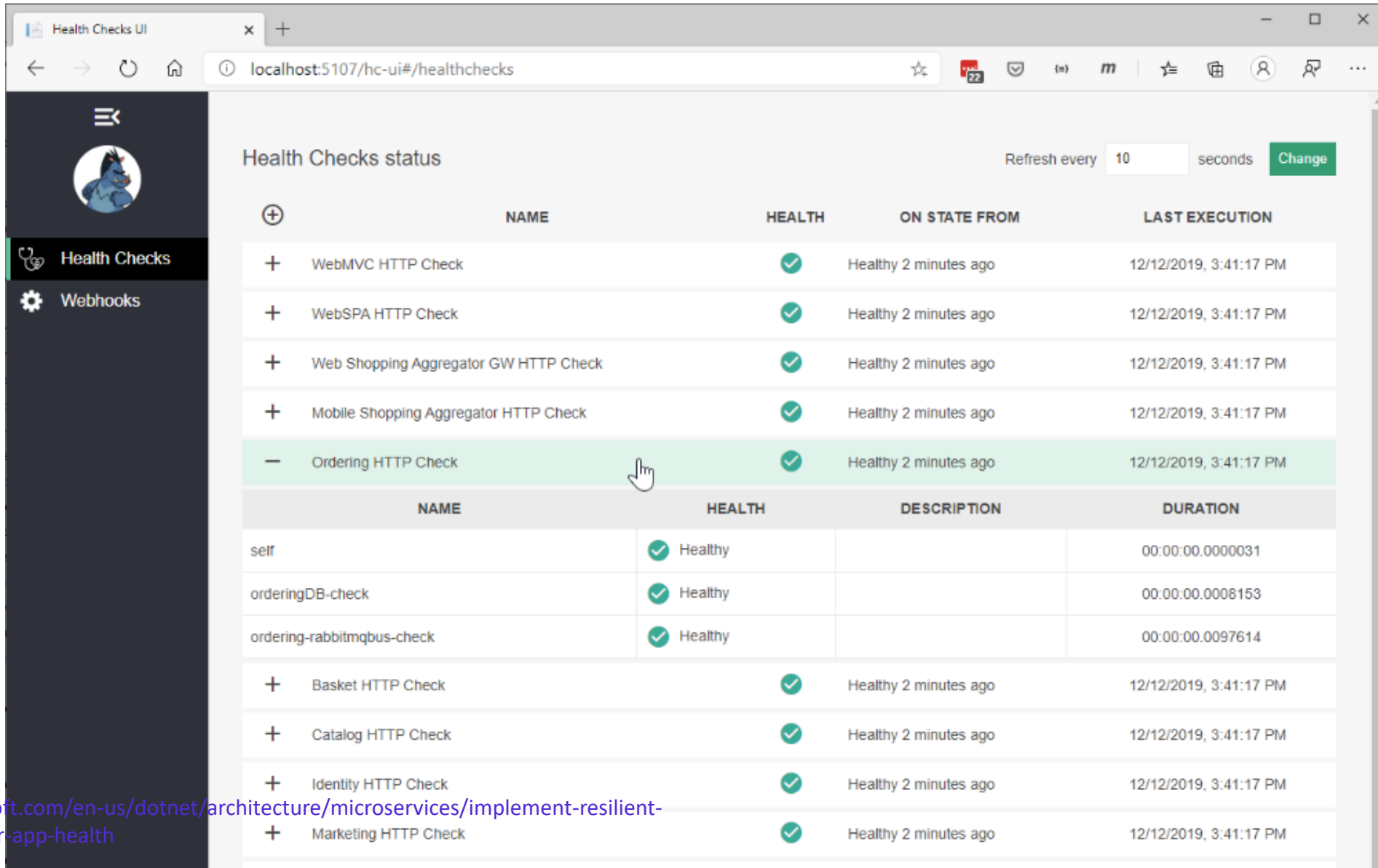
# Asp.Net Health Checks

- Health monitoring is critical
- Revenue loss, customer confidence, and negative effects
- Notified to customers about outages beforehand
- Status page to view results





# WatchDog Health Dashboard



The screenshot shows a web browser window titled "Health Checks UI" with the URL "localhost:5107/hc-ui#/healthchecks". The dashboard displays a "Health Checks status" section with a refresh interval of 10 seconds. A table lists various health checks, all of which are currently "Healthy". The "Ordering HTTP Check" is highlighted in green. Below the main table, a detailed view for the "Ordering HTTP Check" is shown, including its name, health status, description, and duration.

+	NAME	HEALTH	ON STATE FROM	LAST EXECUTION
+	WebMVC HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	WebSPA HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Web Shopping Aggregator GW HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Mobile Shopping Aggregator HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
-	Ordering HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM

NAME	HEALTH	DESCRIPTION	DURATION
self	✓ Healthy		00:00:00.0000031
orderingDB-check	✓ Healthy		00:00:00.0008153
ordering-rabbitmqbus-check	✓ Healthy		00:00:00.0097614

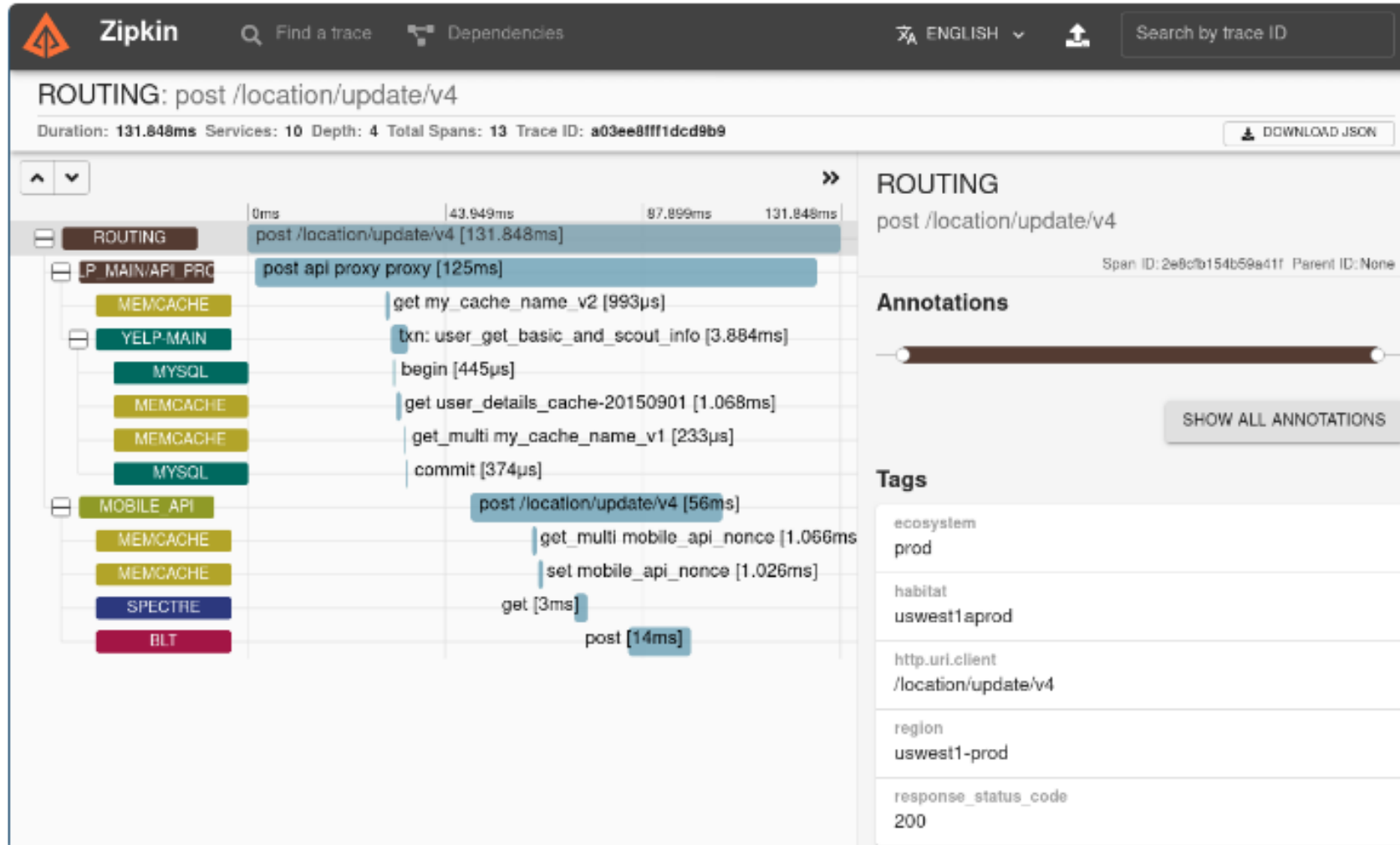
+	Basket HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Catalog HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Identity HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM
+	Marketing HTTP Check	✓	Healthy 2 minutes ago	12/12/2019, 3:41:17 PM

# Section 5

# Microservices Tracing with OpenTelemetry using Zipkin

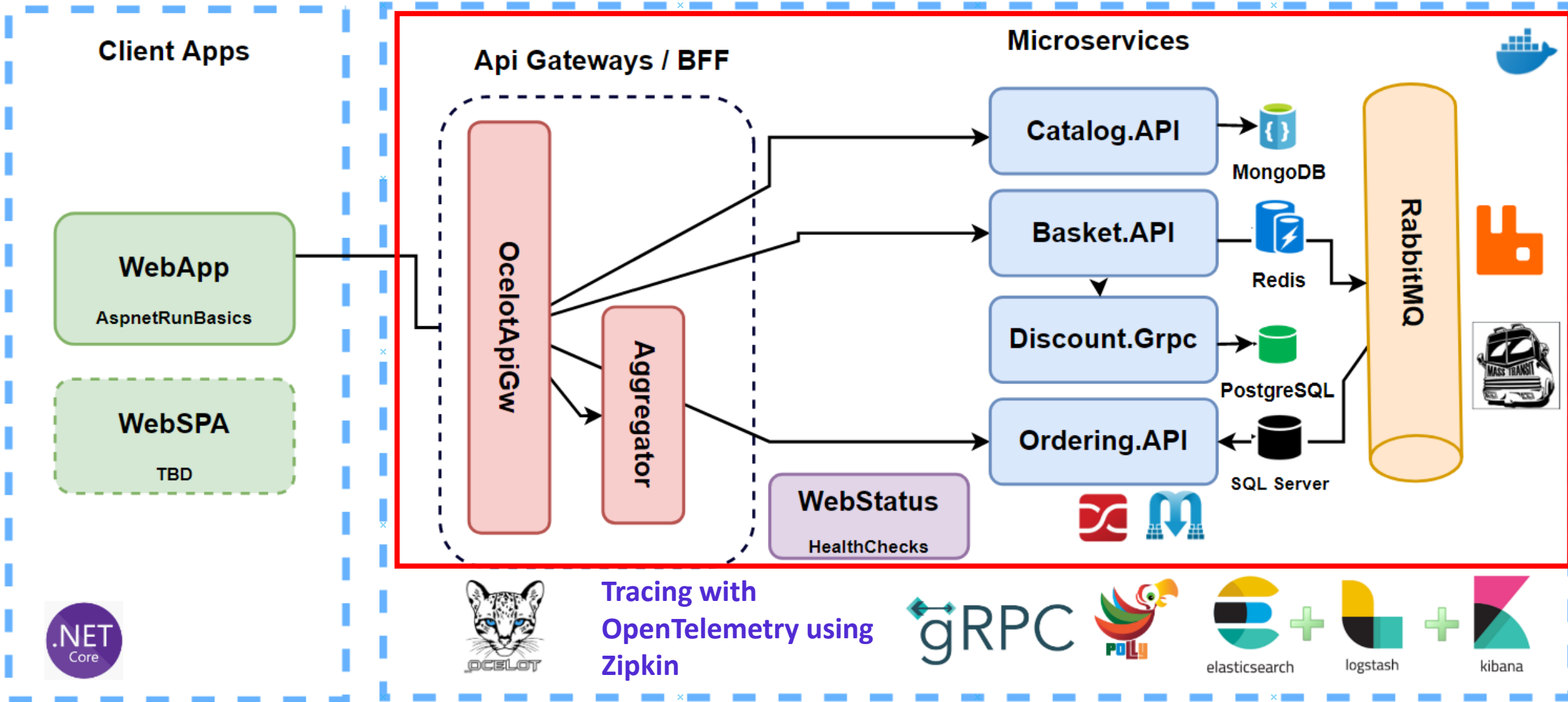
Use Activity and distributed tracing with OpenTelemetry

# Tracing with OpenTelemetry using Zipkin

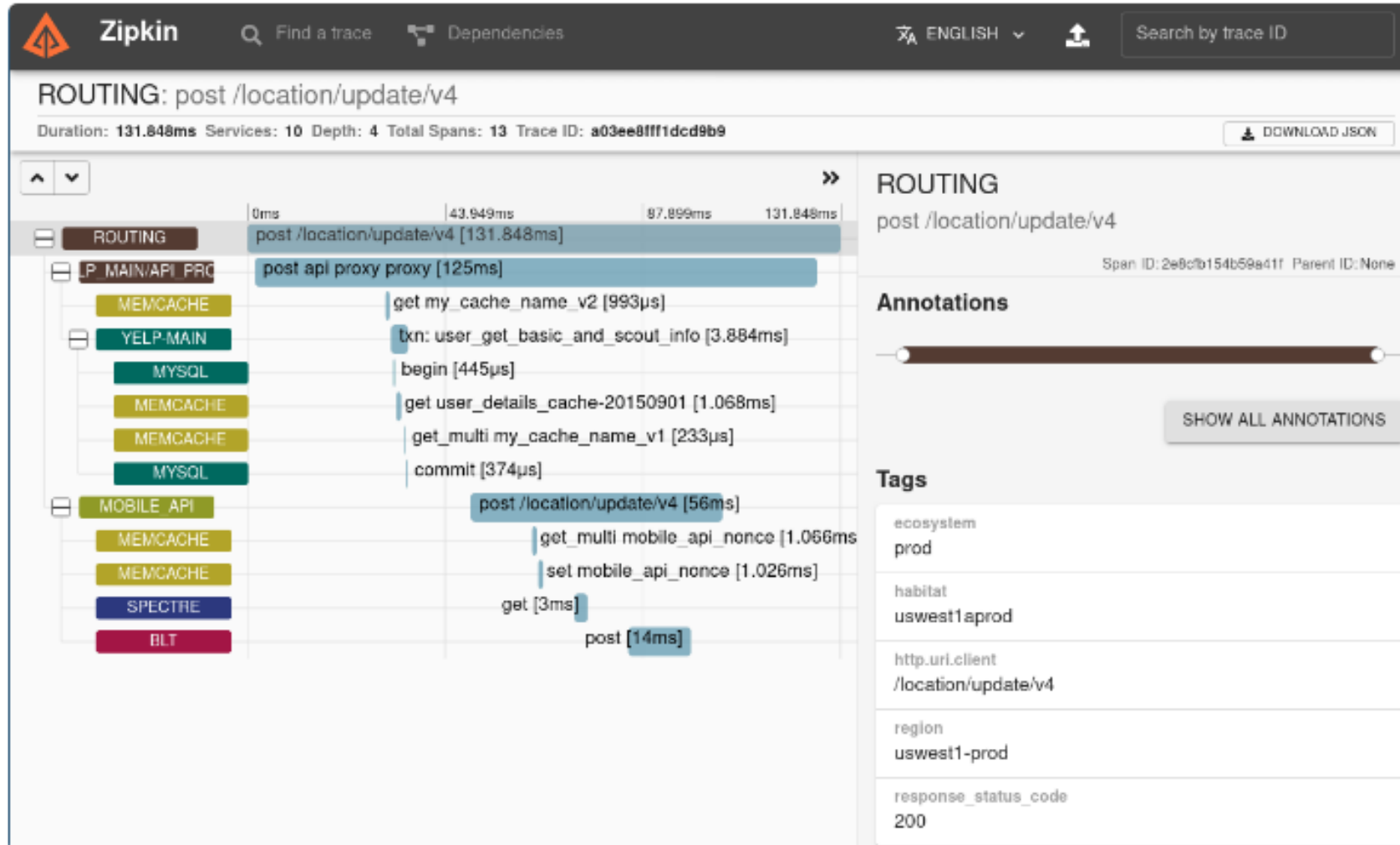




# Database Migration Retries



# Tracing with OpenTelemetry using Zipkin



# OpenTelemetry

- Observability framework
- Collection of tools
- Export telemetry data



ZIPKIN





Collect & Ship Logs

Parse Logs

Store & Search Logs

Visualize Logs



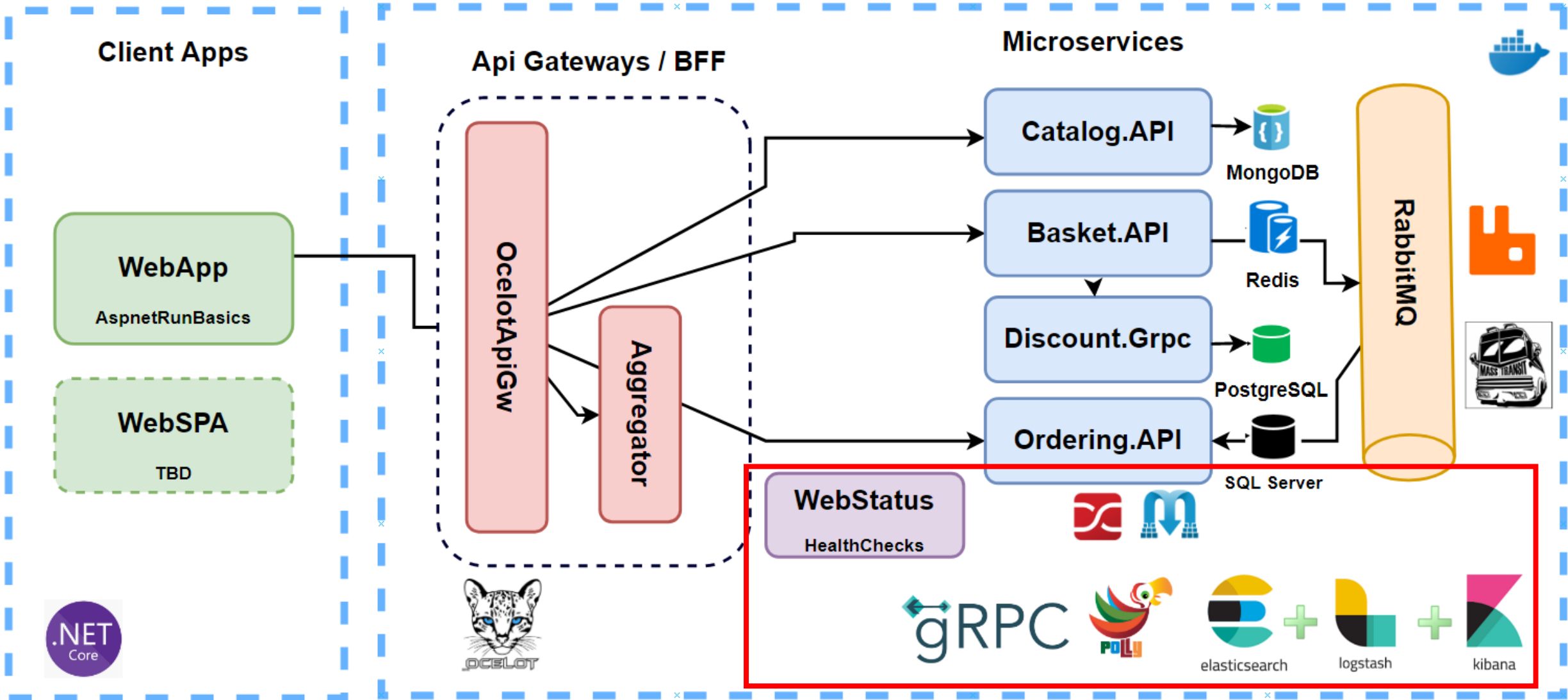
elasticsearch kibana logstash

# 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](mailto:ezozkme@gmail.com)

Repositories 13 Packages People 1 Teams Projects 1 Settings

## Pinned repositories

Customize pinned repositories

learn

The best path to .Net Microservices Udemey 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

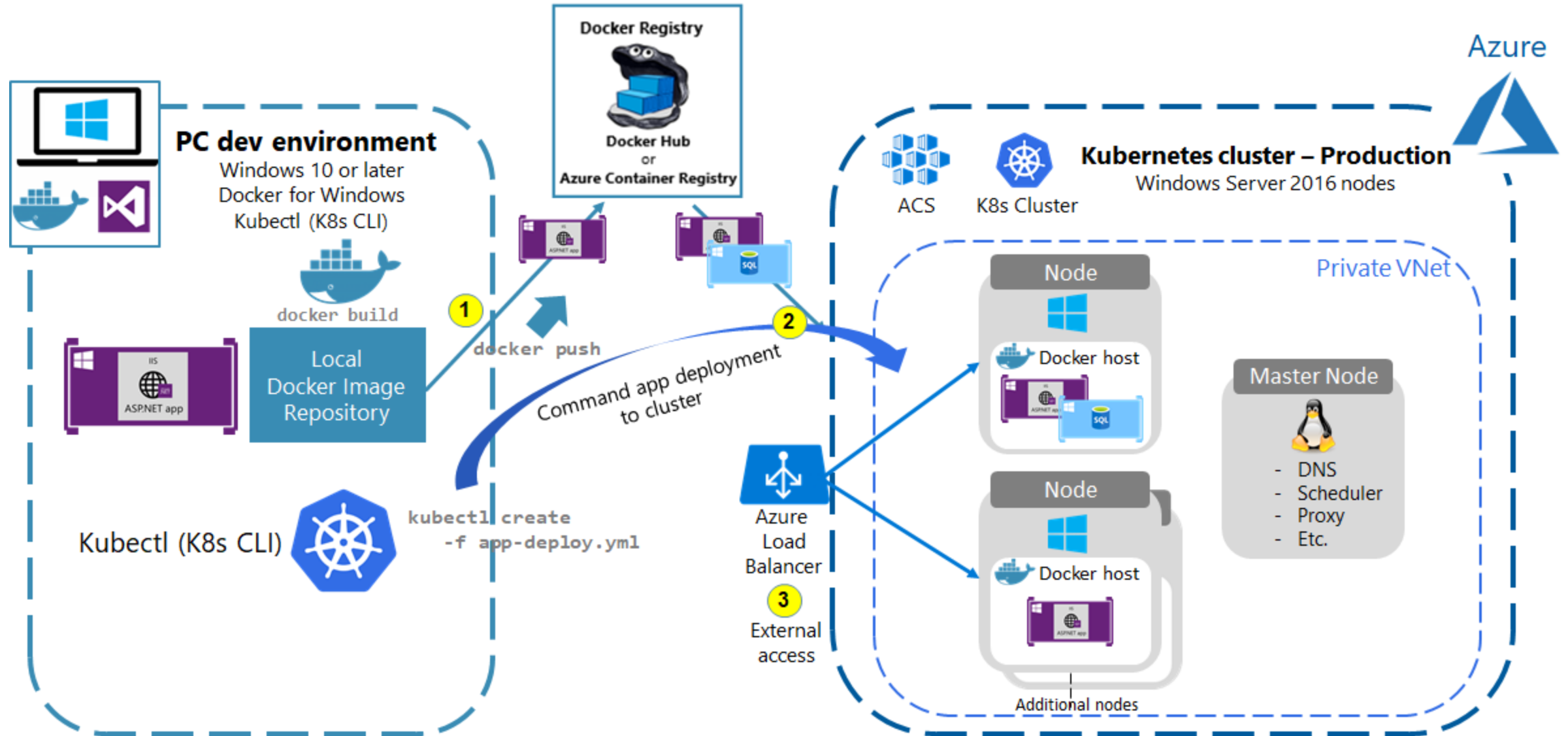


# Section 15

# Deploying Microservices to Kubernetes, Automating with Azure DevOps into AKS

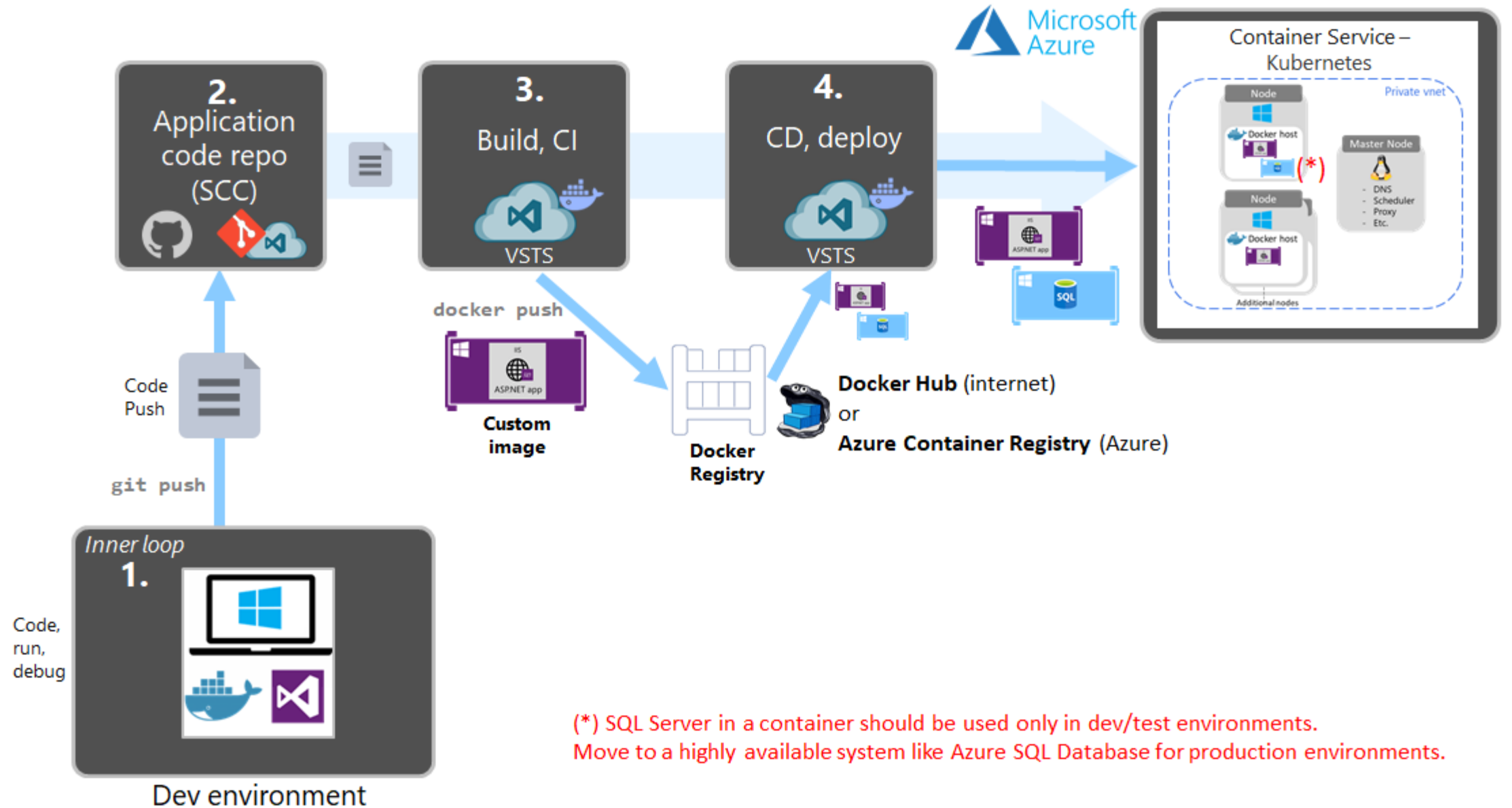
Using Azure Pipelines with Azure Kubernetes Services (AKS)

# 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](mailto: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

# 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](mailto:ezozkme@gmail.com)

Check my other courses on udemy:

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

