Connecting Microservices Synchronously and Asynchronously

Roland Guijt
MICROSOFT MVP, INDEPENDENT CONSULTANT, AUTHOR AND SPEAKER

@rolandguijt rolandguijt.com

Module Overview

Working with multiple microservices

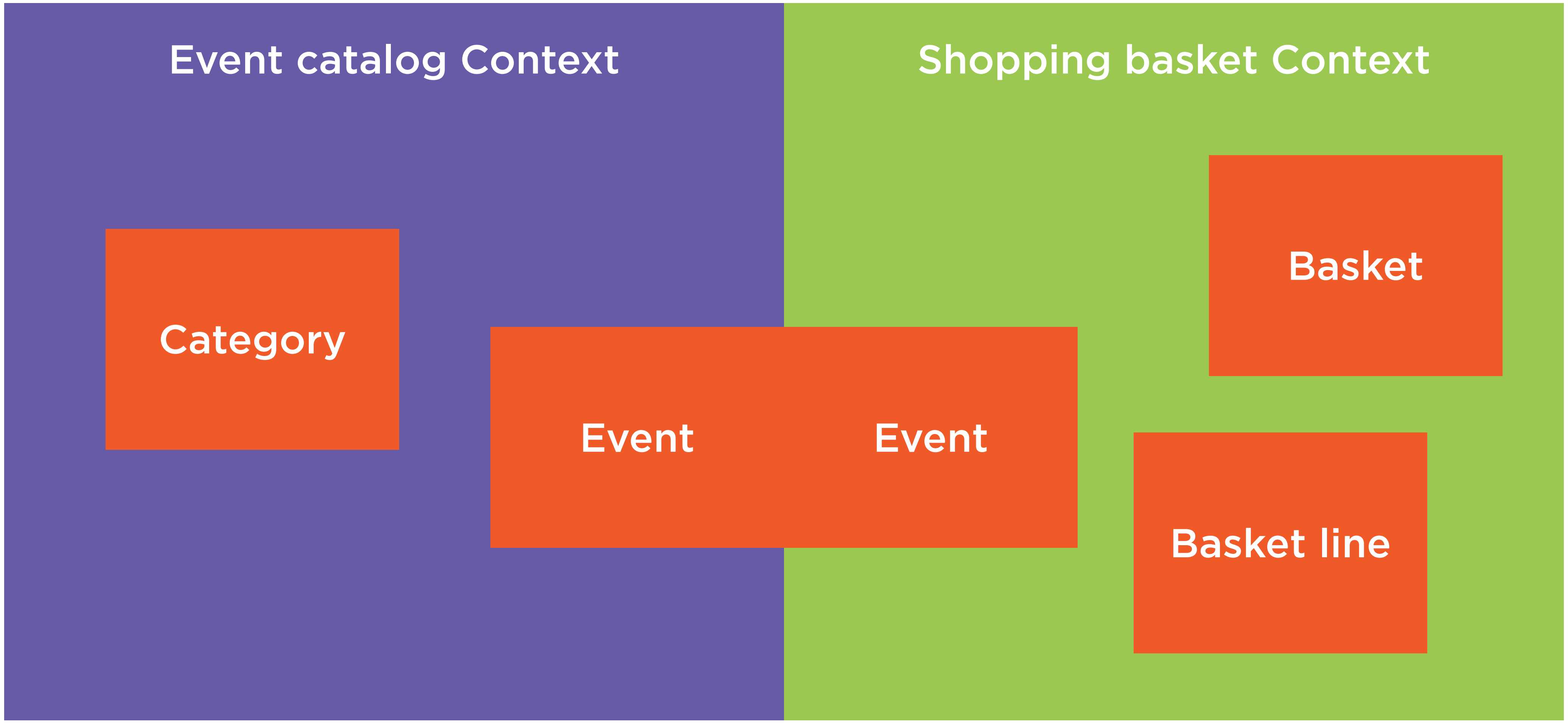
Service-to-service communication

Synchronous and asynchronous communication

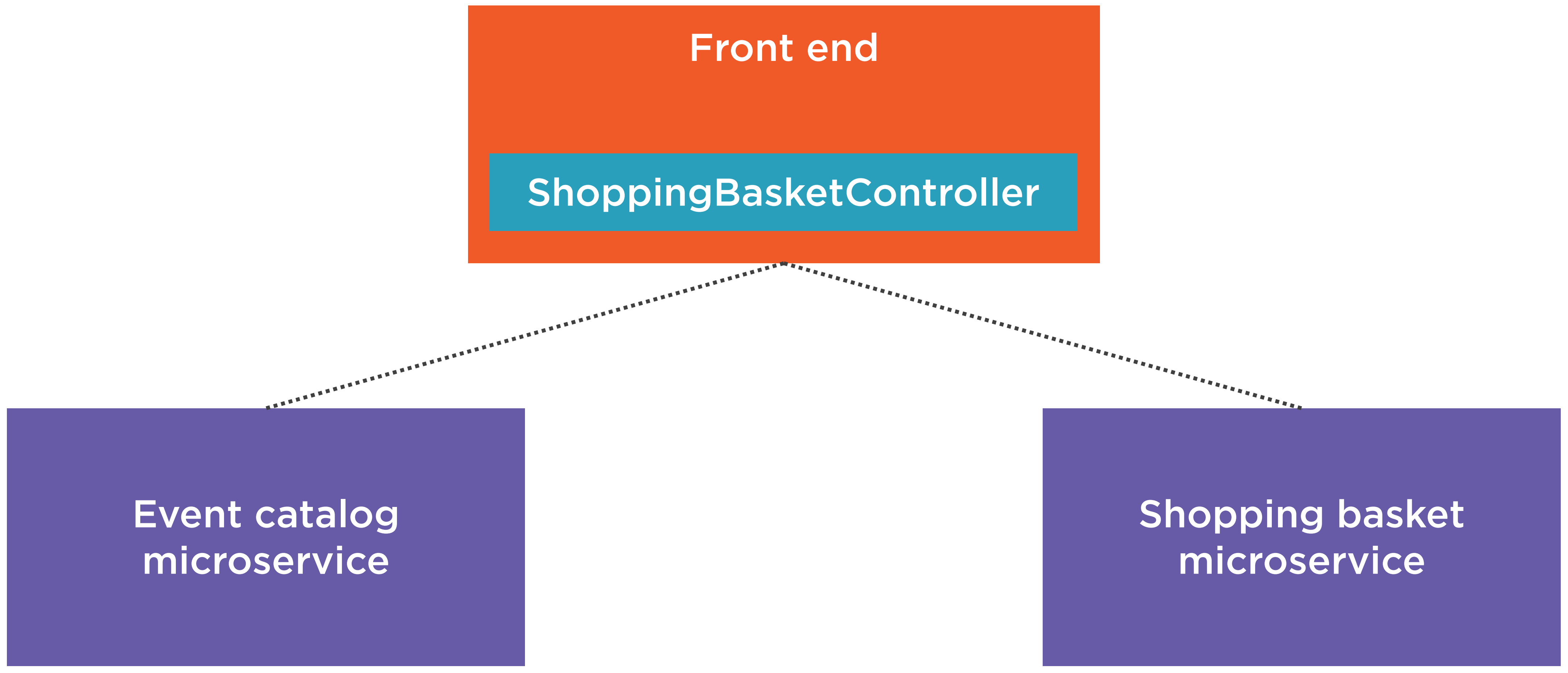
Implementing asynchronous communication

GloboTicket's Domain

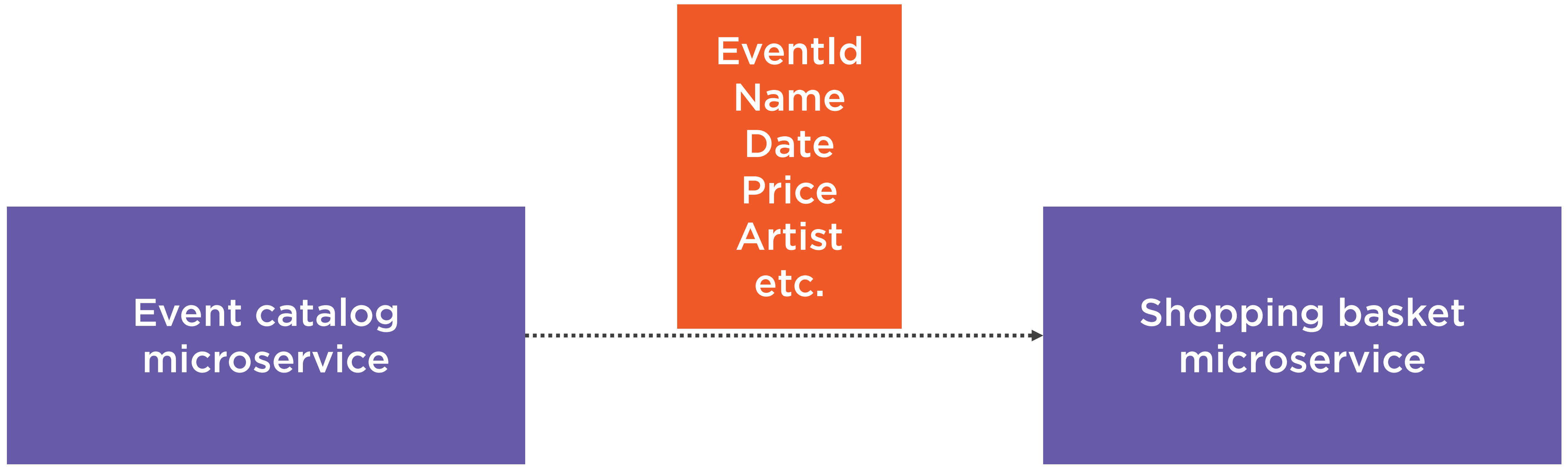
(Excerpt)



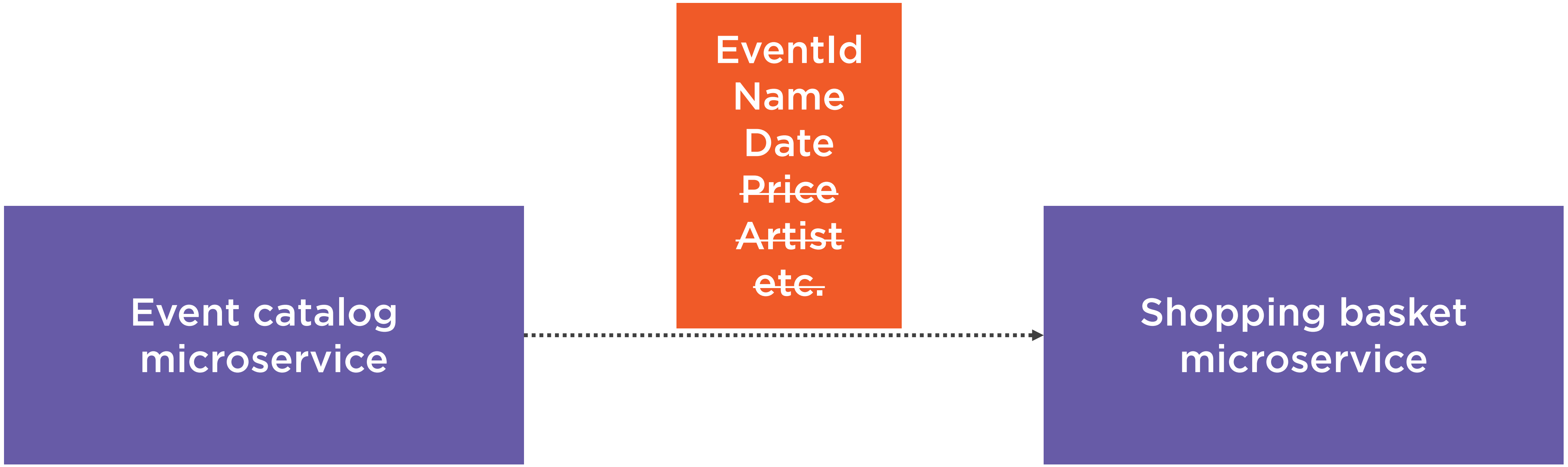
Alternative (Not Recommended)



Serialization and Types



Serialization and Types



That's a Waste of Bandwidth!

Create endpoints for each consuming microservice and front-end?

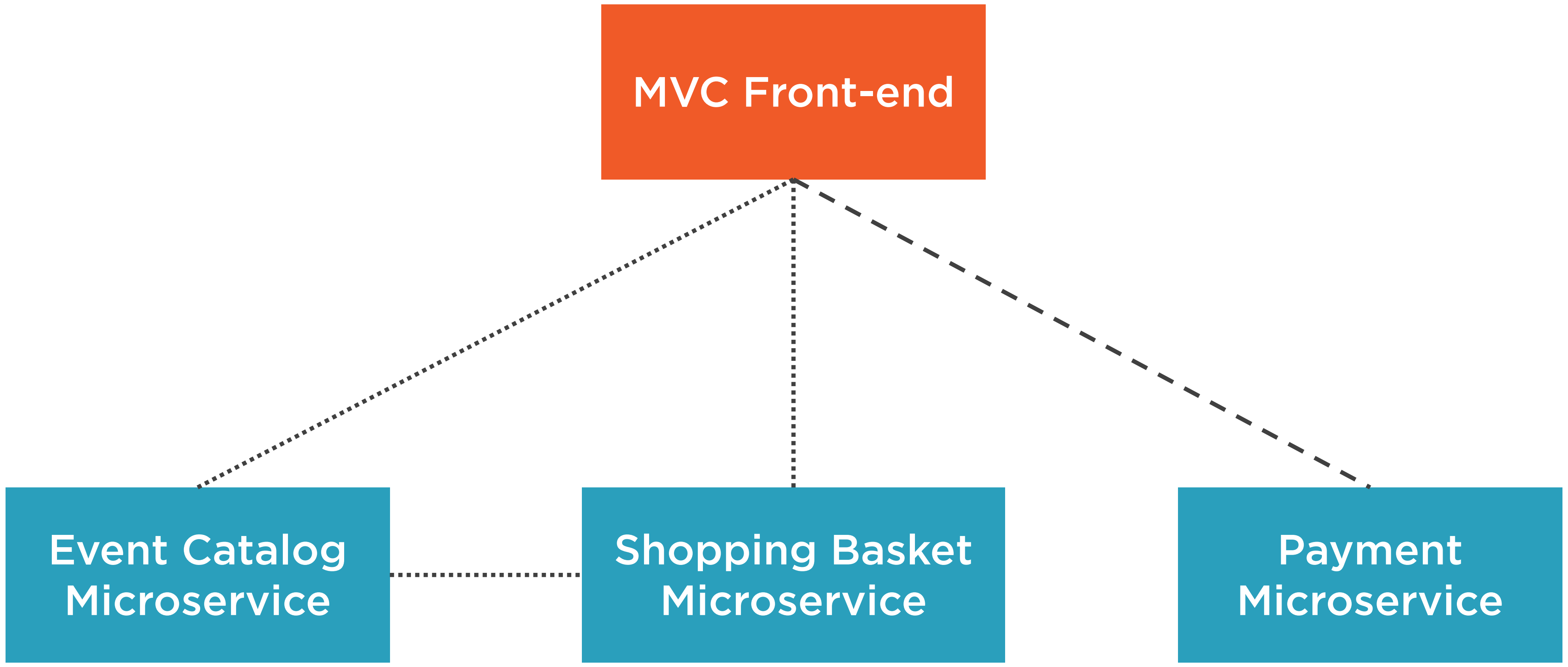
But think of maintainability

And the independence of a microservice

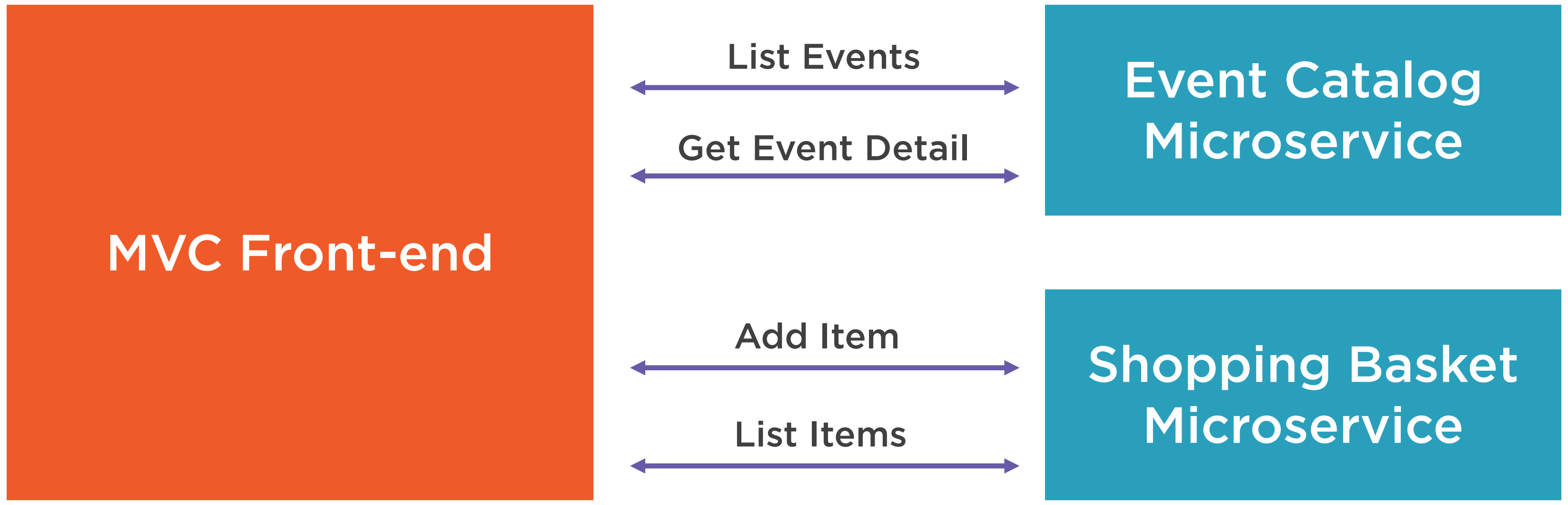
If bandwidth is a problem, consider switching to GraphQL

Building GraphQL APIs with ASP.NET Core

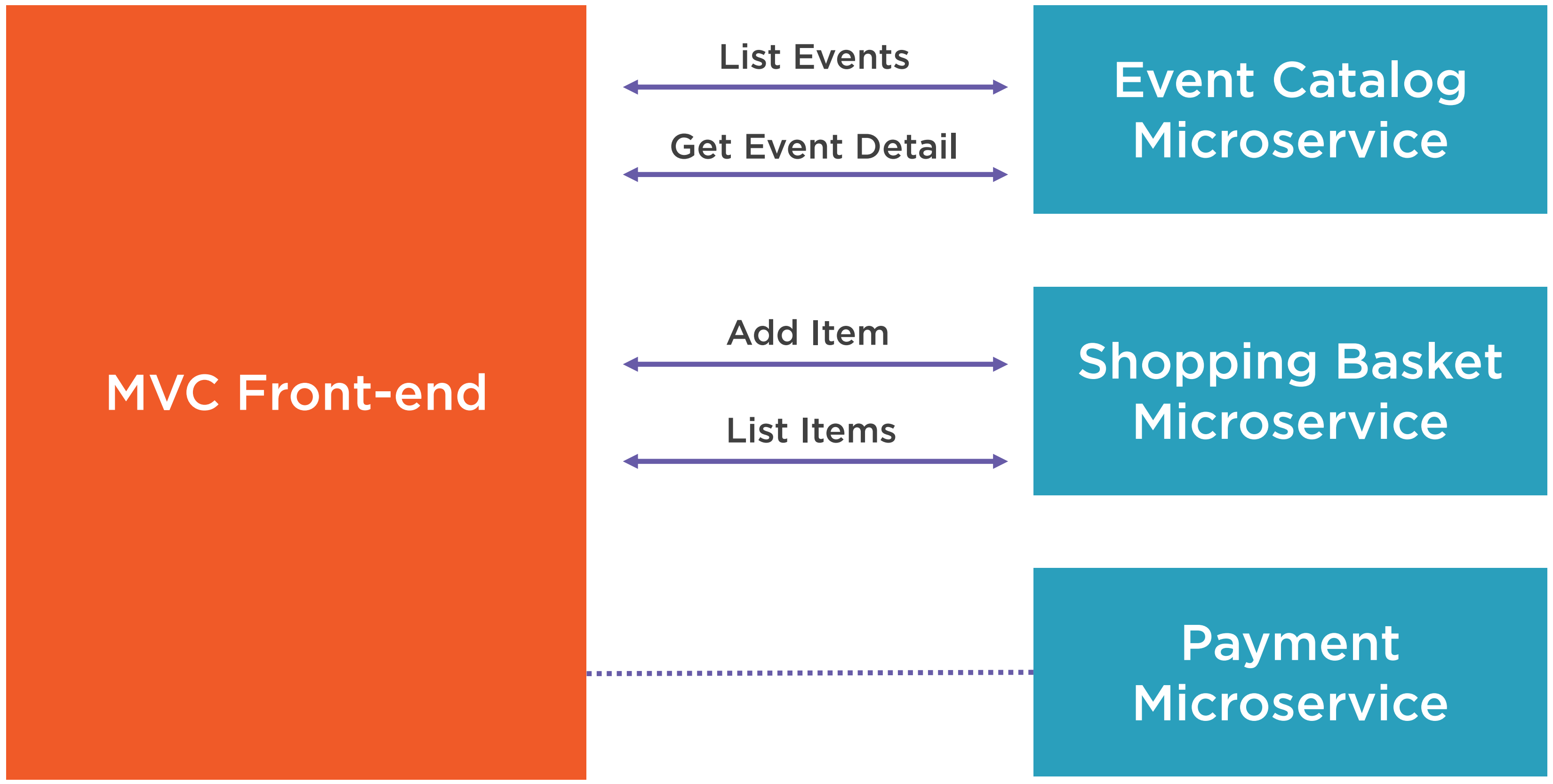
Schematic Architecture So Far



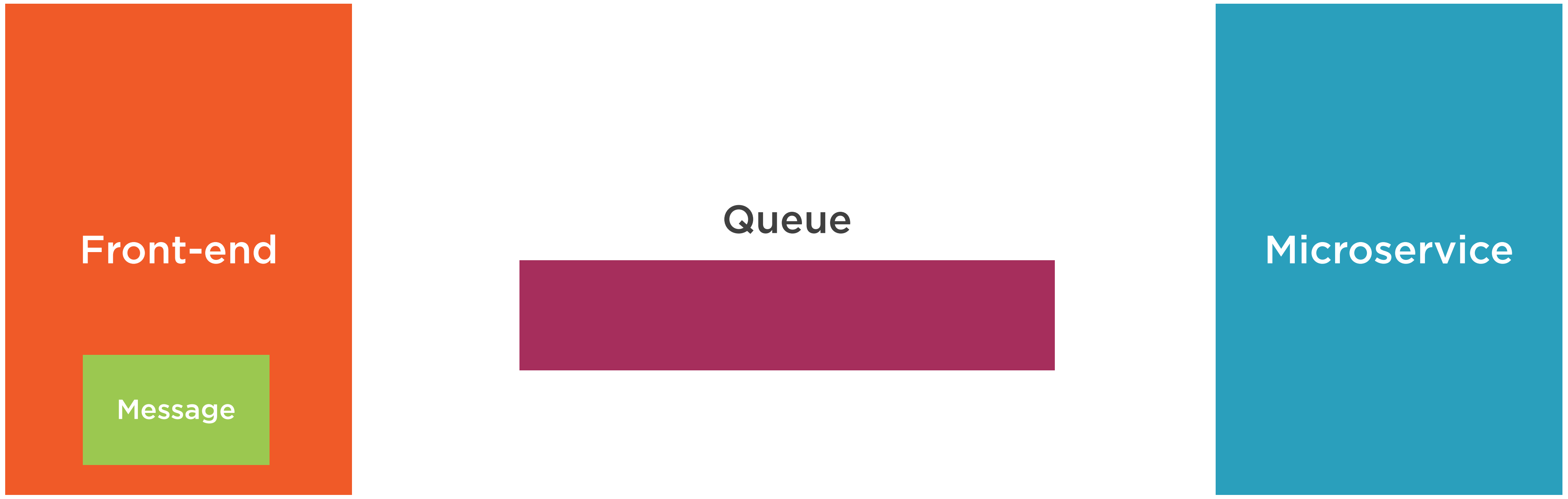
Application Flow



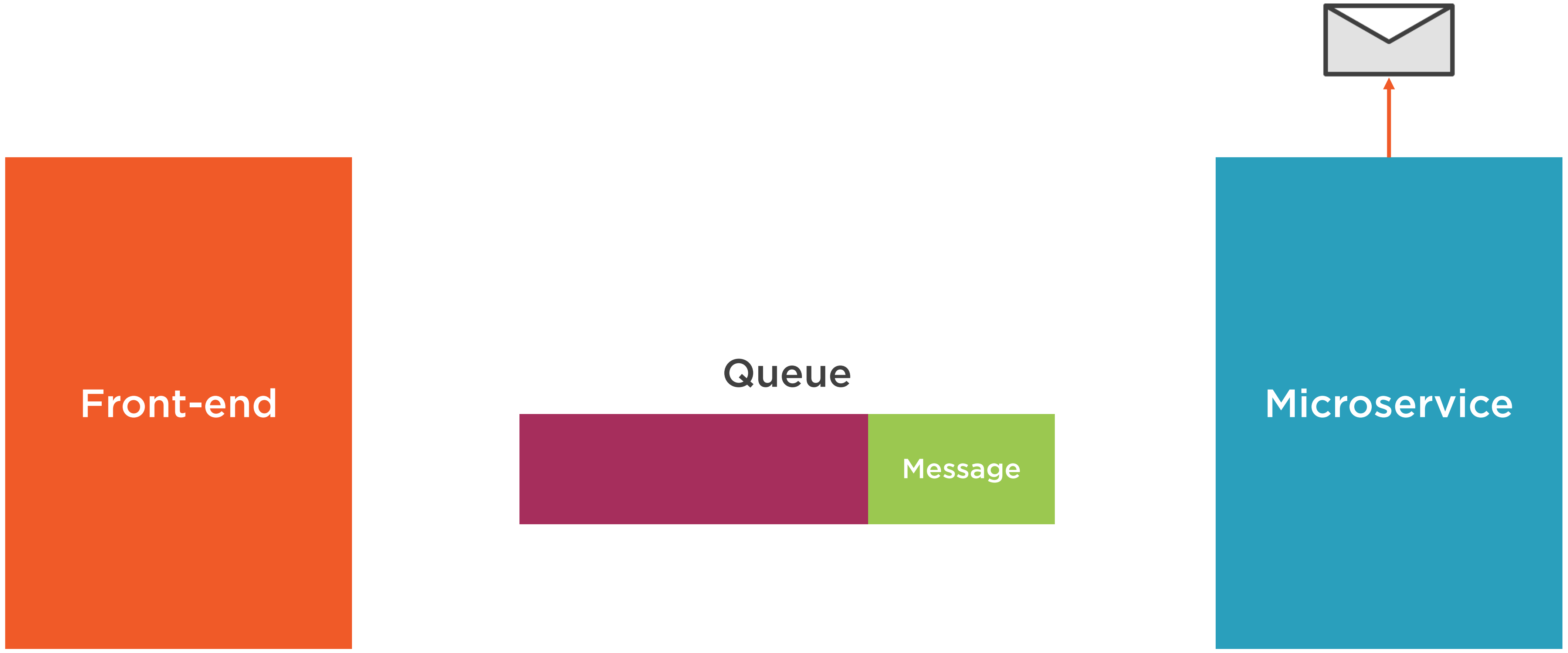
Application Flow



Asynchronous Communication



Asynchronous Communication



Asynchronous Communication

Involves storage between sender and receiver

Work can be done in parallel

Mitigates temporal coupling

Reliable

Message is the contract

Use when no immediate response is required

Worker Services

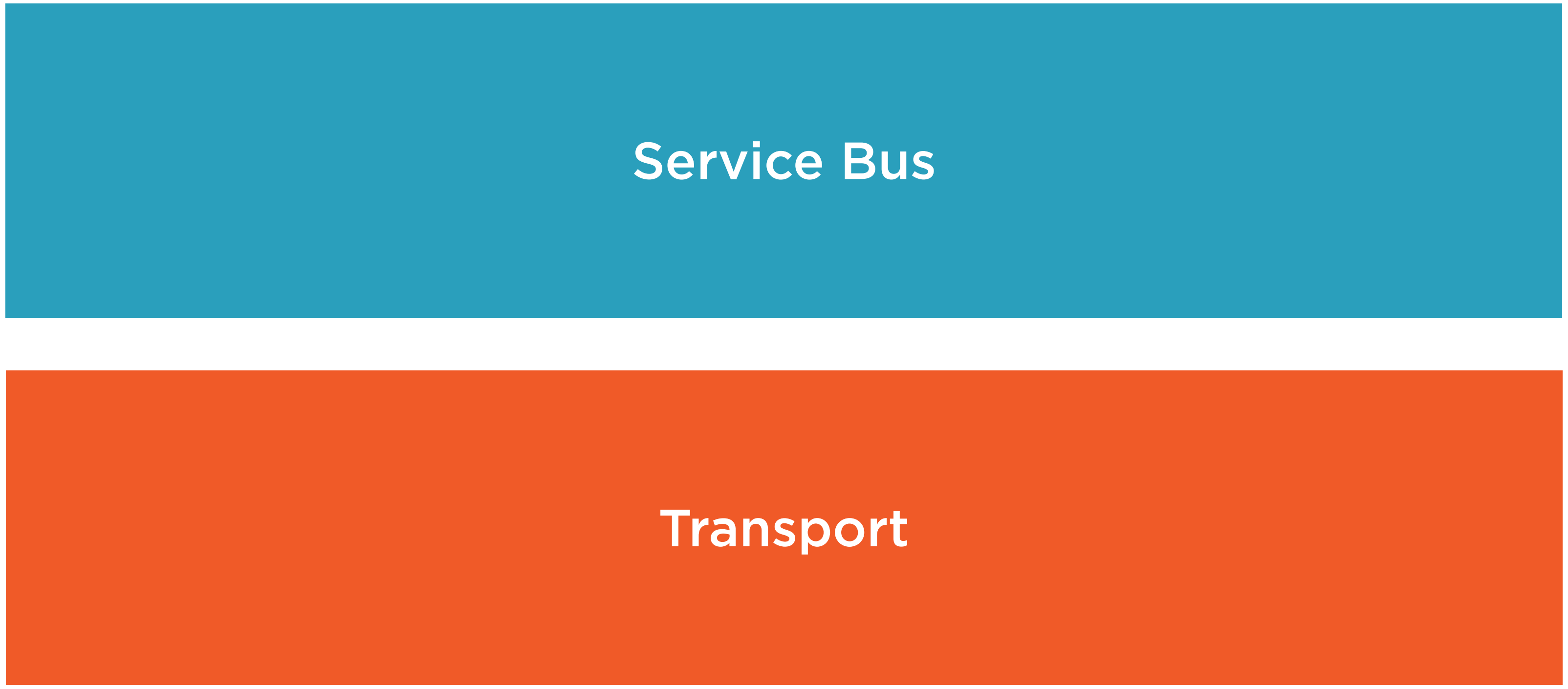
Enable dependency injection

For cleaner and testable code

Sets up familiar configuration and logging types

Building ASP.NET Core Hosted Services and .NET Core Worker Services

Service Bus and Transport



Service Bus and Transport

Service Bus = Rebus

Transport = Azure Storage Queues

Summary

Synchronous communication: Request-response

Asynchronous communication: Messages

Use synchronous communication when immediate response is needed

Use asynchronous communication in all other cases