Particular Software

NServiceBus and Service Fabric

daniel.marbach@particular.net
@danielmarbach

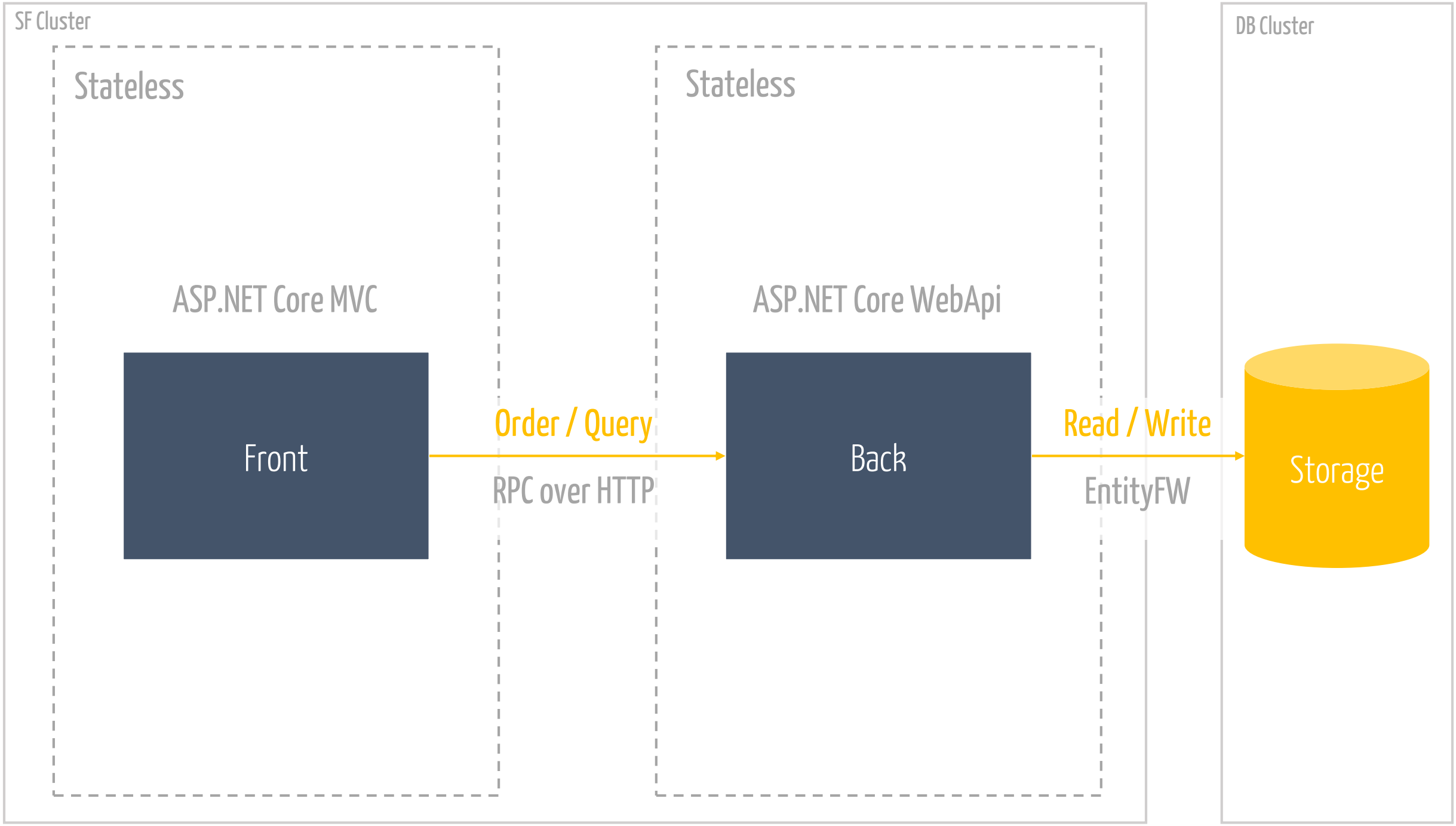
bob.langley@particular.net
@boblangley

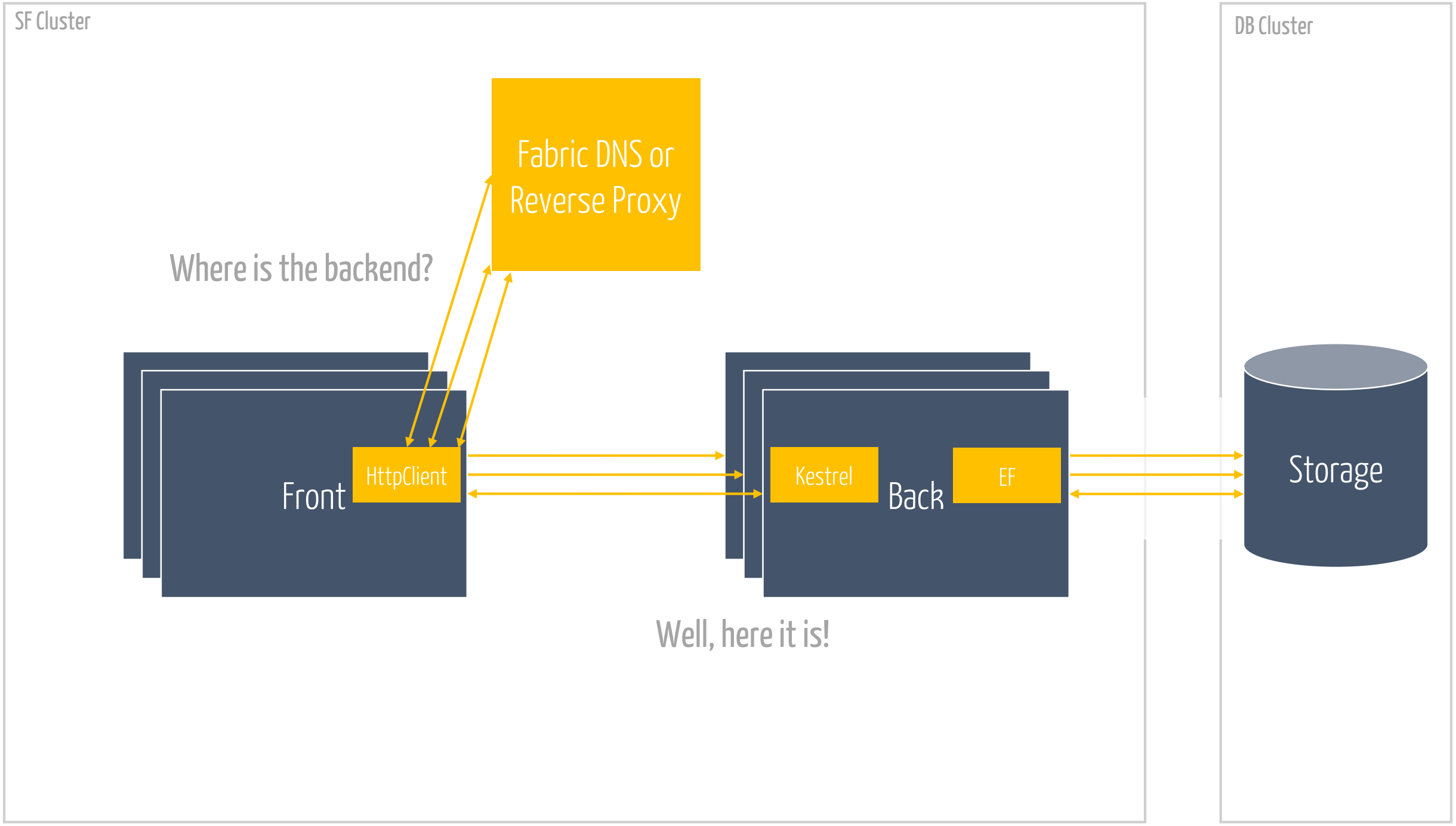
A Tale of chocolate

A few hours

later...

Microservices 1.0


SF Cluster
Stateless
Stateless
DB Cluster
ASP.NET Core MVC
ASP.NET Core WebApi
Front
Order / Query
RPC over HTTP
Back
Read / Write
EntityFW
Storage

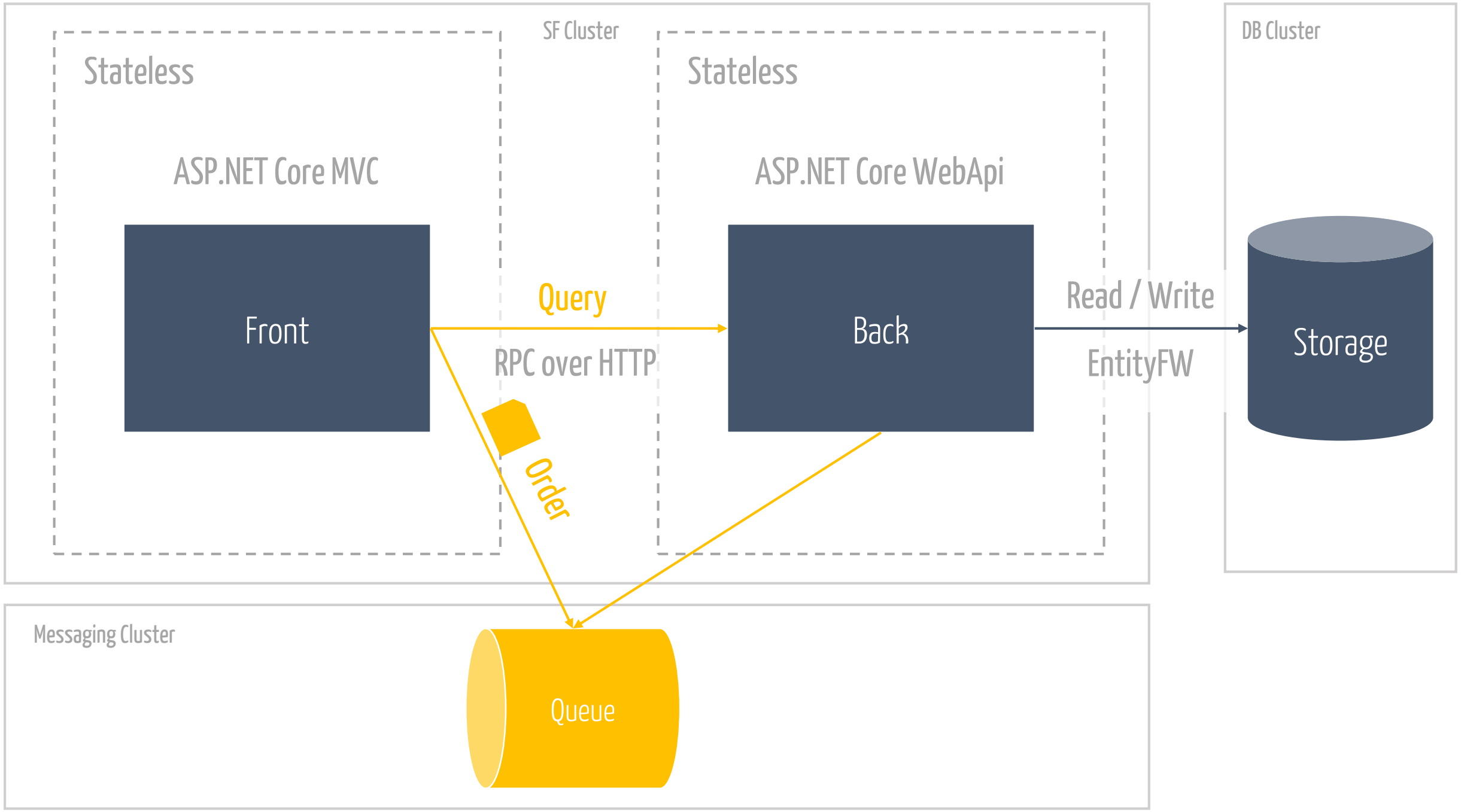

SF Cluster
DB Cluster
Fabric DNS or
Reverse Proxy
Where is the backend?
Front
HttpClient
Kestrel
Back
EF
Storage
Well, here it is!

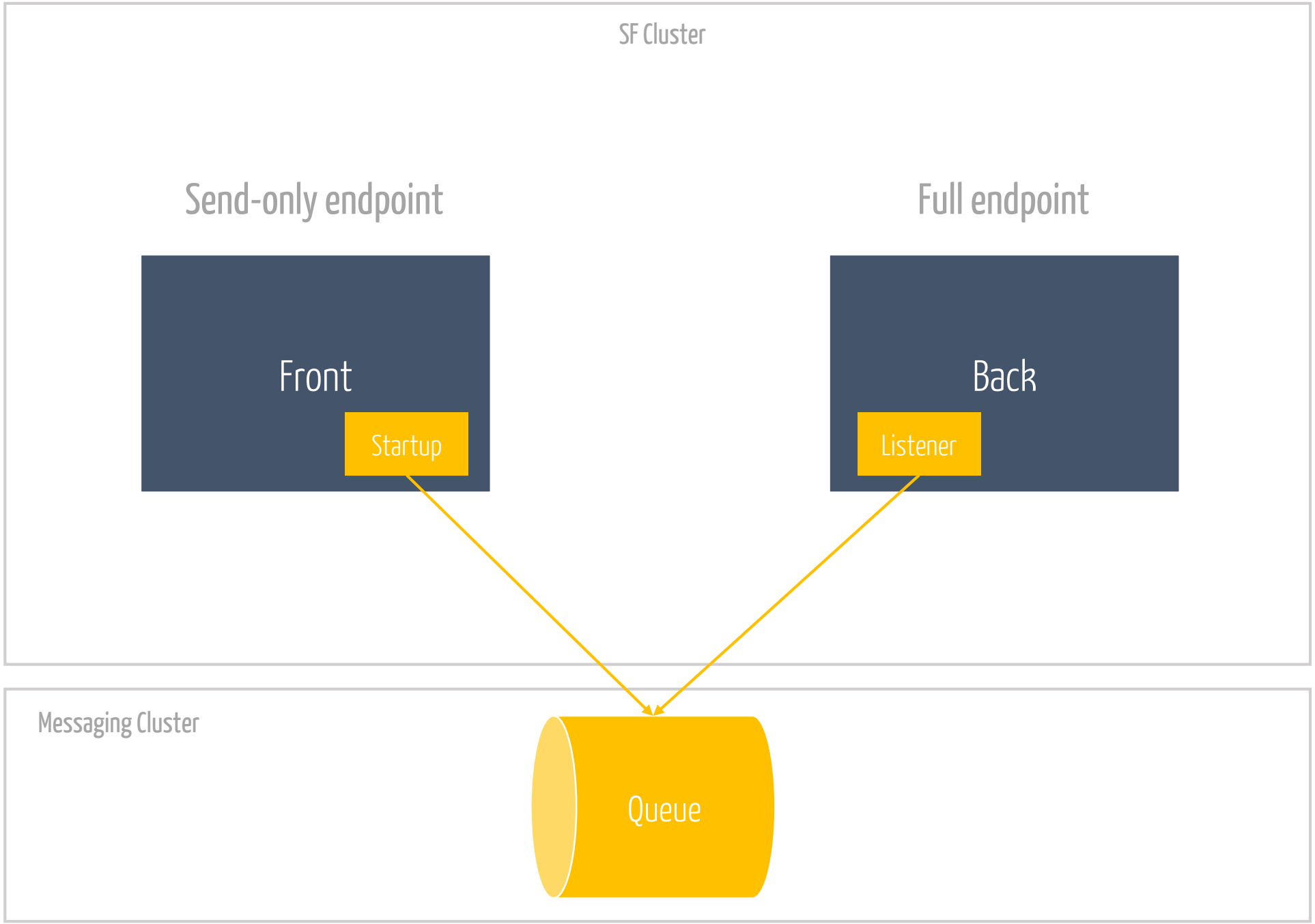
Demo

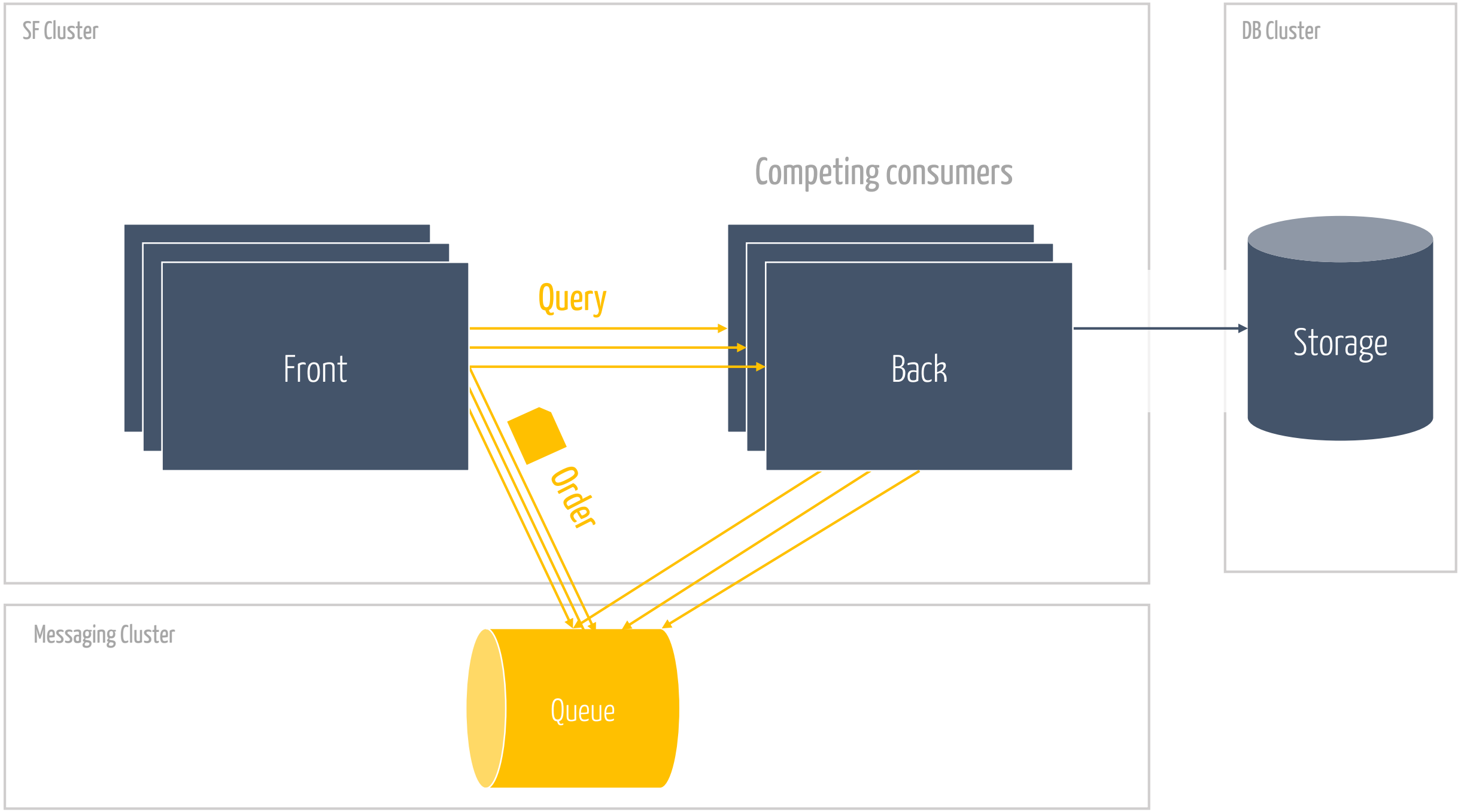
The RPC callstack of

doom

Microservices 2.0


SF Cluster
Stateless
ASP.NET Core MVC
Front
Stateless
ASP.NET Core WebApi
Back
Query
RPC over HTTP
Order
Read / Write
EntityFW
DB Cluster
Storage
Messaging Cluster
Queue


SF Cluster
Send-only endpoint
Full endpoint
Front
Startup
Back
Listener
Messaging Cluster
Queue


SF Cluster
DB Cluster
Competing consumers
Front
Query
Back
Storage
Order
Messaging Cluster
Queue

Demo

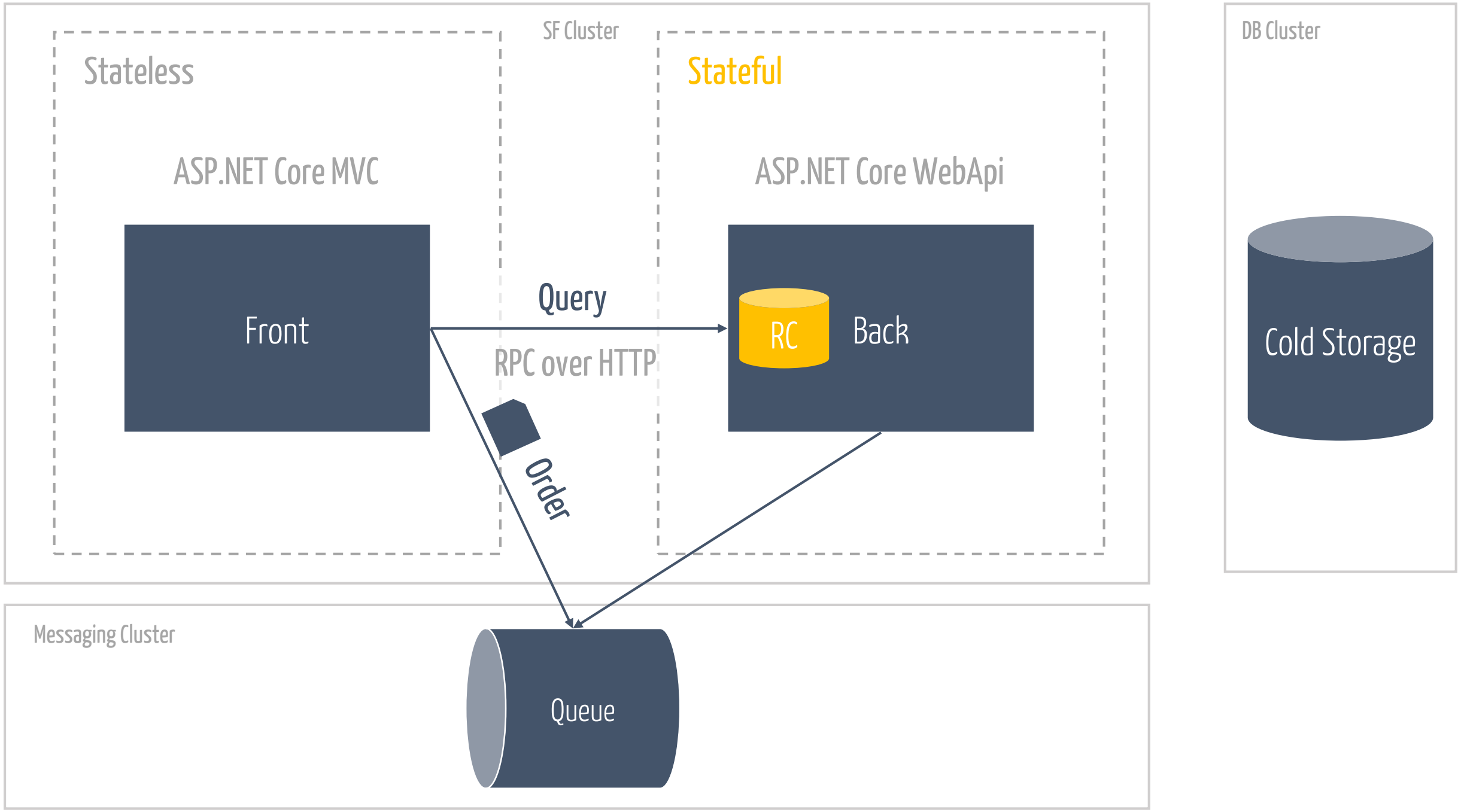
almost

a happy life

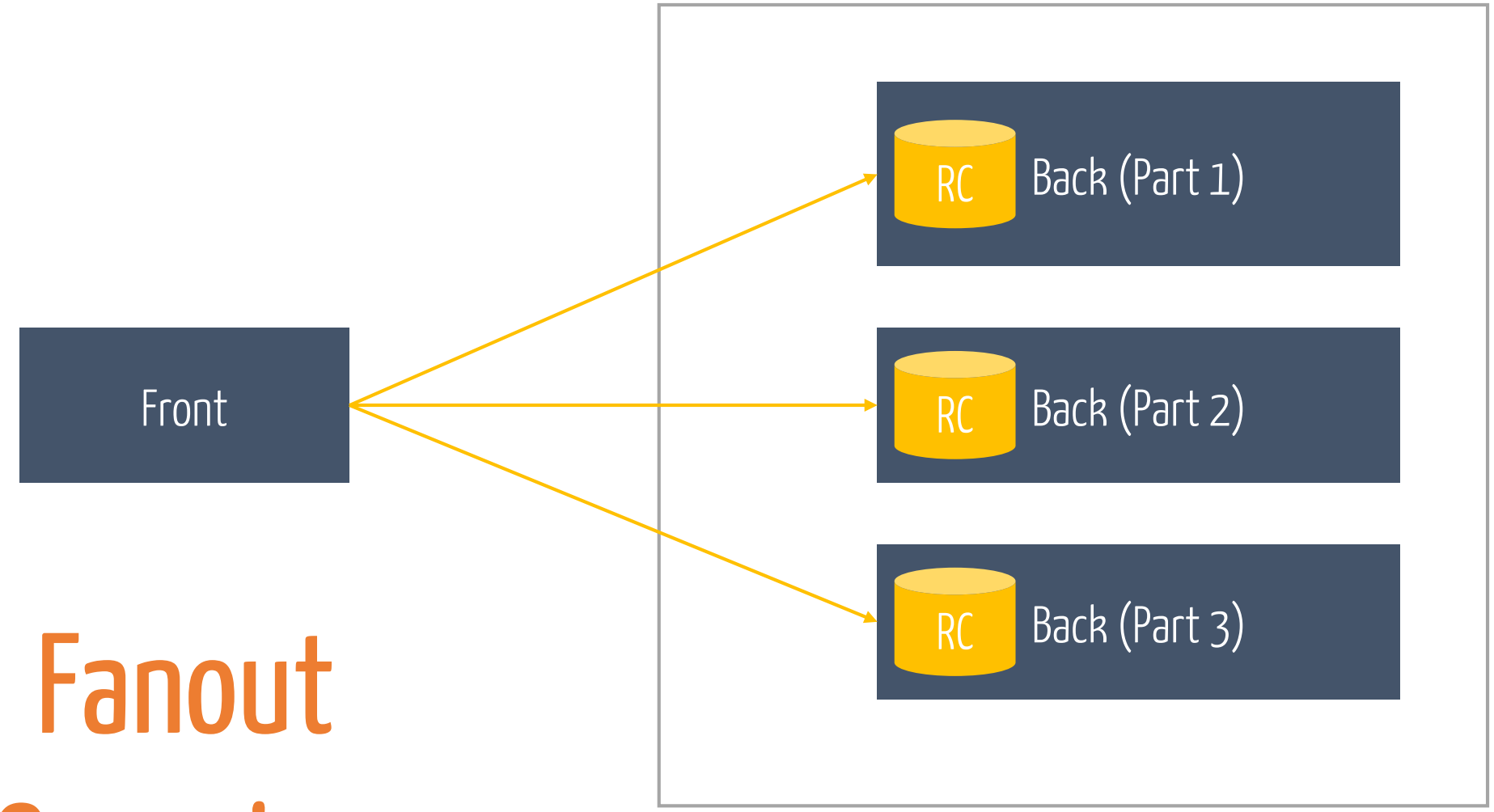
almost

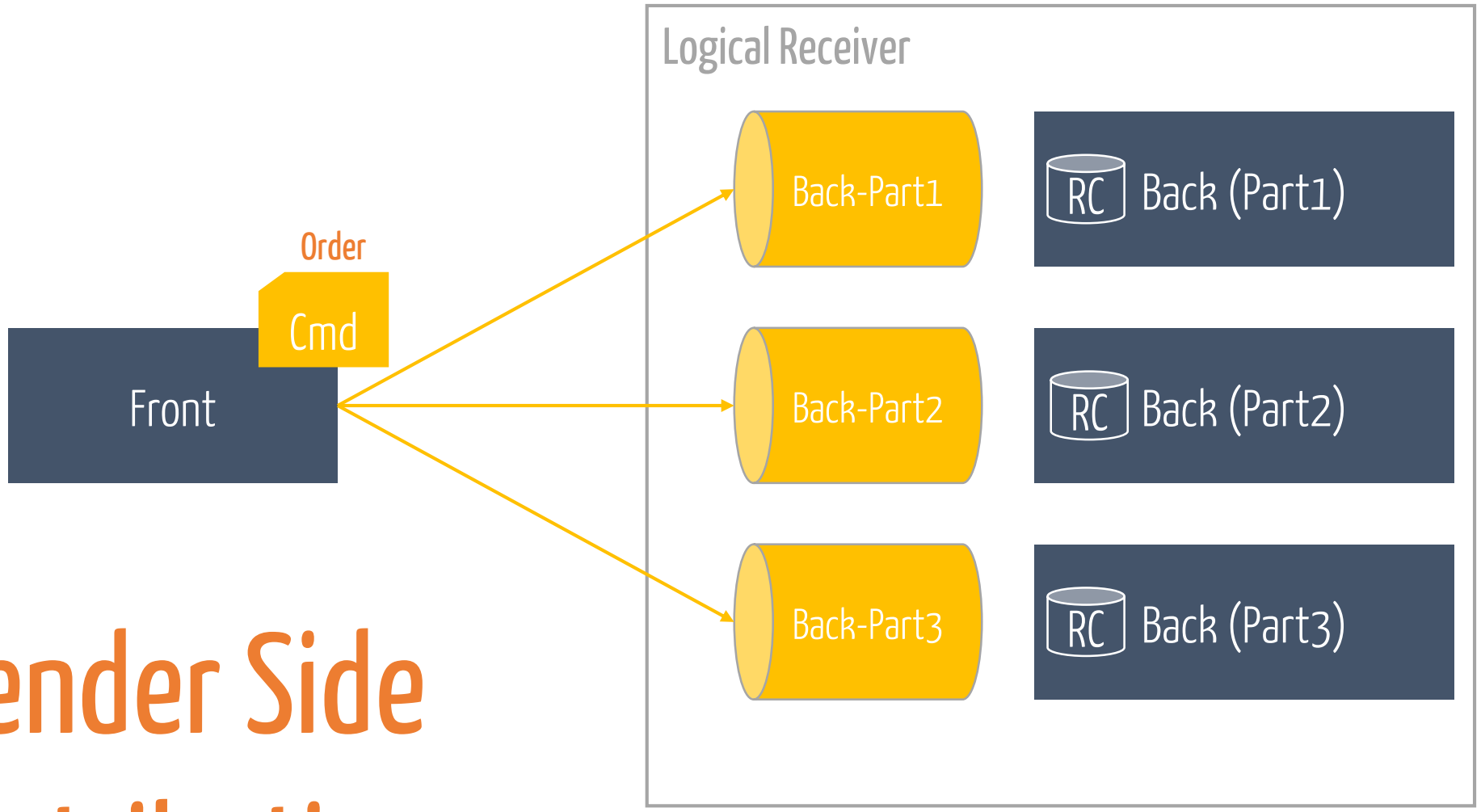
a happy life

Microservices 3.0


SF Cluster
Stateless
ASP.NET Core MVC
Front
Stateful
ASP.NET Core WebApi
RC
Back
Query
RPC over HTTP
Order
DB Cluster
Cold Storage
Messaging Cluster
Queue

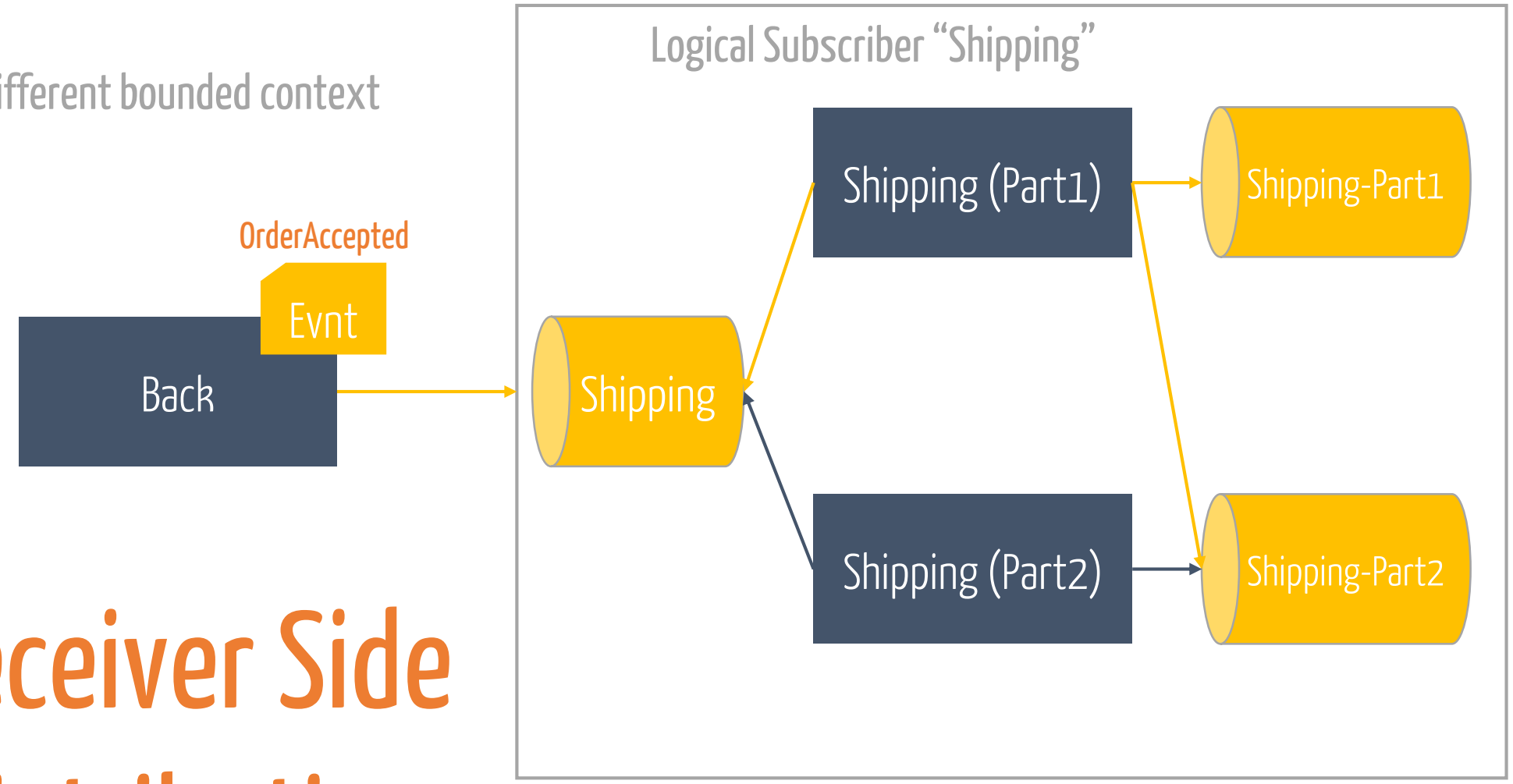
Fanout Querying





Sender Side Distribution

Receiver Side Distribution



Demo

Slides, Links...

github.com/danielmarbach/service-fabric-webinar

docs.particular.net/
tutorials/quickstart

NServiceBus Quick Start

In this tutorial, we'll see why software systems built on asynchronous messaging using NServiceBus are superior to traditional synchronous HTTP-based web services. We'll also show how NServiceBus guarantees reliability and extensibility that can't be achieved with REST.

This tutorial skips over some concepts and implementation details in order to get up and running quickly. If you'd prefer to go more in-depth, check out our Introduction to NServiceBus tutorial. It will teach you the NServiceBus API and important concepts you need to learn to build successful message-based software systems.

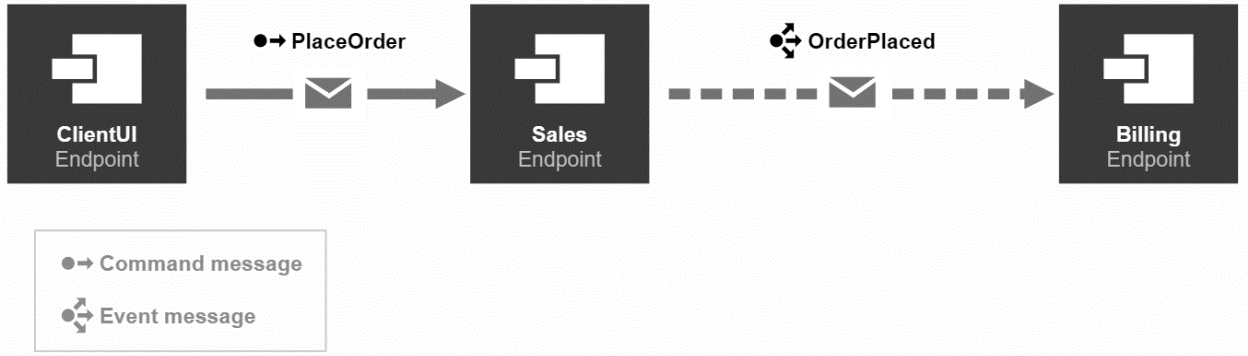
To get started, download the solution, extract the archive, and then open the **RetailDemo.sln** file with Visual Studio 2017.

Download the solution now

Project structure

The solution contains four projects. The **ClientUI**, **Sales**, and **Billing** projects are endpoints that communicate with each other using NServiceBus messages. The **ClientUI** endpoint mimics a web application and is an entry point in our system. The **Sales** and **Billing** endpoints contain business logic related to processing and fulfilling orders. Each endpoint references the **Messages** assembly, which contains the definitions of messages as POCO class files.

As shown in the diagram, the **ClientUI** endpoint sends a **PlaceOrder** command to the **Sales** endpoint. As a result, the **Sales** endpoint will publish an **OrderPlaced** event using the publish/subscribe pattern, which will be received by the **Billing** endpoint.



The solution mimics a real-life retail system, where the command to place an order is sent as a result of a customer interaction, and the actual processing occurs in the background. Publishing an event allows us to isolate the code to bill the credit card from the code to place the order, reducing coupling and making the system easier to maintain over the long term. Later in this tutorial, we'll see how to add a second subscriber in the **Shipping** endpoint which would begin the process of shipping the order.

NServiceBus

Getting Started
Service Platform
Quick Start Tutorial
Messaging Basics Tutorial
Getting Started
Sending a command
Multiple endpoints
Publishing events
Retrying errors
Message Replay Tutorial
Step by Step Sample
Configuration choices Sample
Concepts
Platform Installer
License
Extensions
Contributing
Architectural Principles
Bus vs. Broker

Upgrade Guides
Messaging
Hosting
Handlers and Sagas
Testing
Recoverability
Pipeline
Serialization
Containers
Logging
Security
Operations

docs.particular.net/

samples/azure/azure-service-fabric-routing/

Home / Samples / Azure Samples

Edit | Feedback

Service Fabric Partition Aware Routing

Download | Nugets Used | Edit Code

Component: NServiceBus | Nuget: NServiceBus (Version: 6.x)

This sample currently makes use of a pre-release version of NServiceBus.Persistence.ServiceFabric.

The sample demonstrates how the NServiceBus API can be used to implement partition aware routing for services hosted inside a Service Fabric cluster. It takes advantage of routing system extensibility points and custom pipeline behaviors to support various types of NServiceBus communication patterns. It is assumed that the NServiceBus users are able to define mapping between message type and service partition for each message. It is also assumed that `send local`, `timeout` and `reply` messages are partition affine i.e. should be processed in the context of originating partition. The sample consists of services hosted inside and outside the Service Fabric and enables proper communication between the two.

Scenario

The scenario used in this sample covers a voting system. In this voting system the cast votes are counted by candidate. The endpoint responsible for counting candidate votes is subscribed to an event published when votes are cast.

Next to this the system also counts the total number of votes cast in each zip code. In order to achieve this the candidate voting endpoint issues a `request` to the zip code counting endpoint to track the zip code. The zip code counting endpoint will `reply` back with the intermediary results.

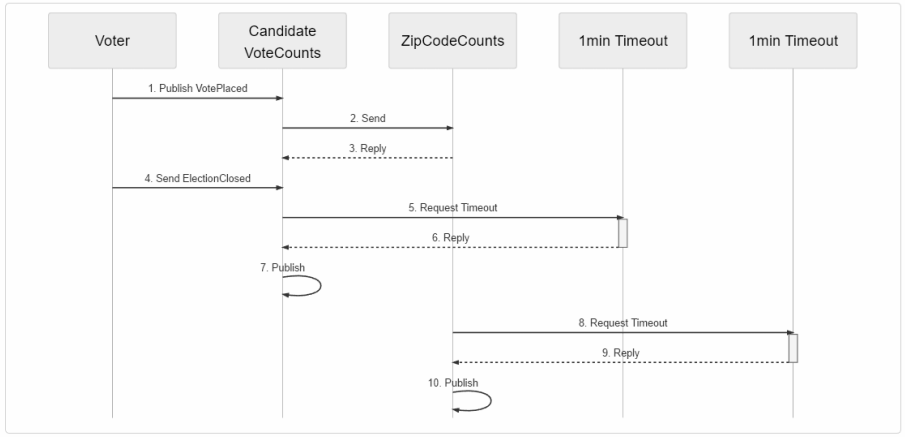
When the election is closed, the candidate vote counting endpoint will `publish` the results per candidate and report them using Service Fabric diagnostics infrastructure (ETW Event Viewer windows).

After the counting time expires, using a timeout the zip code counting endpoint `sends a local command` to report the statistics per zip code.

The sample assumes that:

- There are only 2 candidates in the election, called "John" and "Abby",
- Zip codes are integers in the range of 0 to 99000.

This simplifies partition id value calculation. In a real world scenario a hash function could be used to perform mapping from arbitrary input types.



NServiceBus
Transports
Persistence
ServiceInsight
ServicePulse
ServiceControl
Samples
General
Azure
Azure Service Bus Transport
Long running operations with Azure Service Bus Transport
Service Fabric Partition Aware Routing
Azure Blob Storage DataBus
Custom ASB Namespace Partitioning
Custom Sanitization with Azure Service Bus Transport
Native Integration with Azure Service Bus Transport
Azure Service Bus Performance Tuning
Polymorphic events with Azure Service Bus Transport
Shared Hosting in Azure Cloud Services
Azure Storage Persistence
Azure Storage Queues Transport
Container
Custom Checks
Encryption

Particular Software

Q&A

daniel.marbach@particular.net
@danielmarbach

bob.langley@particular.net
@boblangley

Particular
Software

Thanks