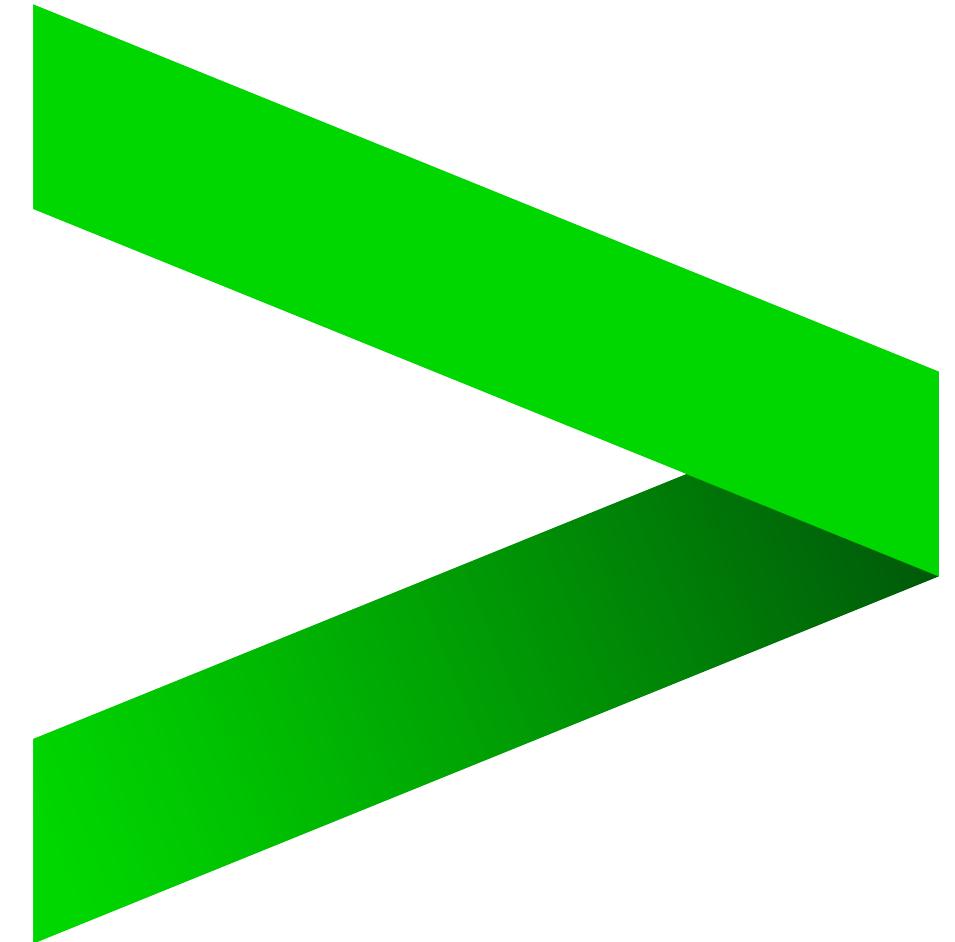


# **THE EVENT-DRIVEN UI BRIDGING THE GAP**

**KEVIN BADER**

accenture technology

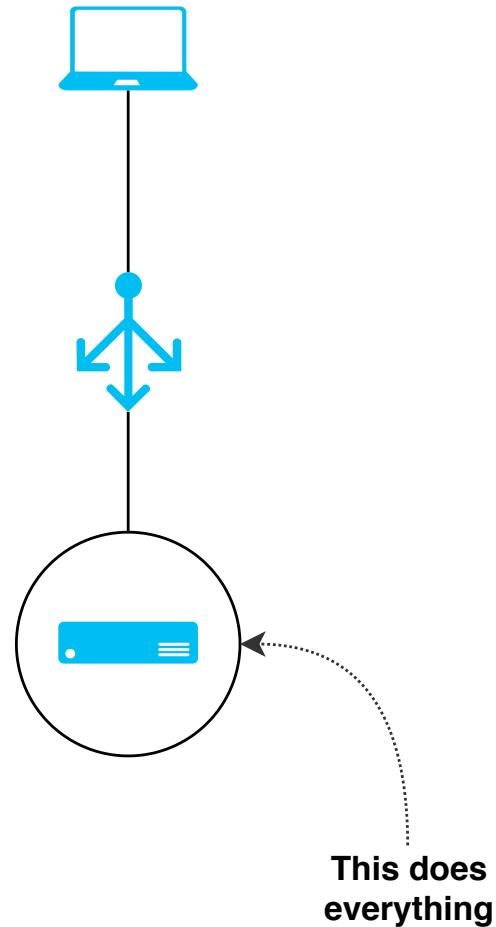


# **AGENDA**

- **Situation**
- **Vision**
- **Challenges**
- **Solution**
- **Conclusion**

# SITUATION

**One backend, one frontend:  
one source of data and events.**

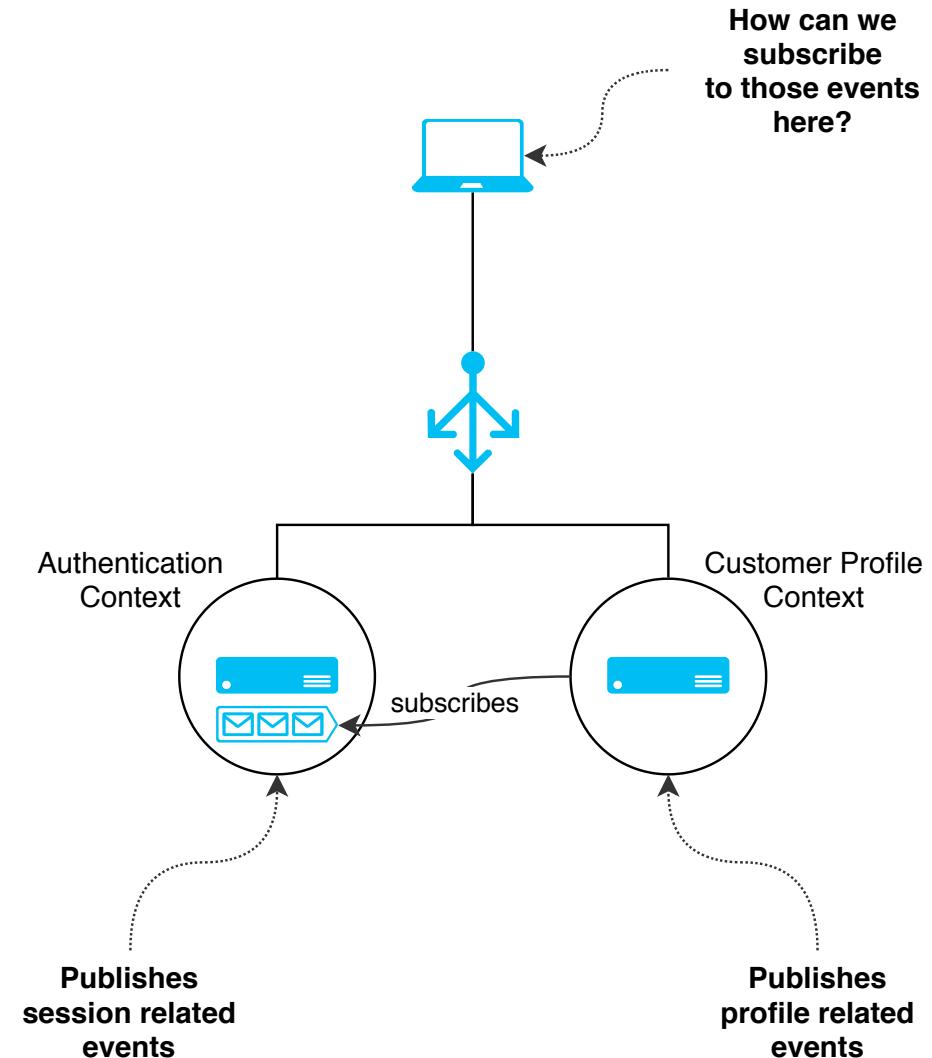


# SITUATION

**Microservices group functionality by domain/context.**

**Events decouple microservices.**

**Still only one frontend.**



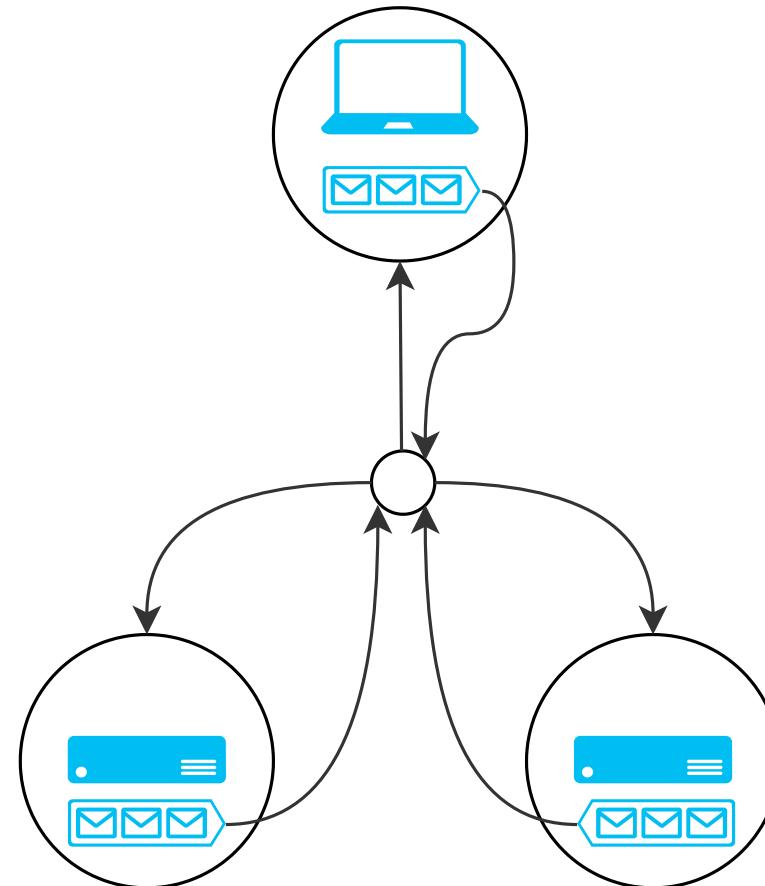
# VISION: AN EVENT-DRIVEN WORLD

**Microservices emit events.**

**Frontends consume and produce events the same way other microservices do.**

**Only open standards.**

**Frontends not aware how events are stored.**



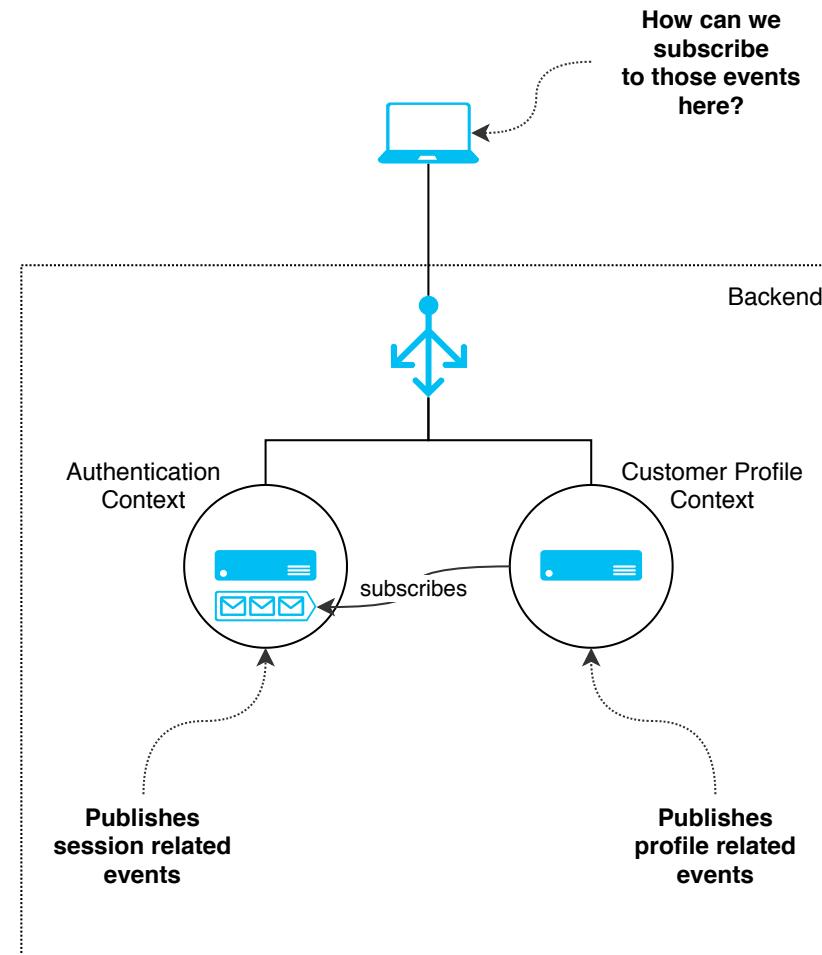
# VISION: GOALS

## Microservices should

- not handle long-lived connections
- not publish “special” events for frontend consumption

## Frontends should

- be agnostic of event partitioning on the backend
- not rely on proprietary formats
- be able to publish events
- be able to control what events they are subscribed to

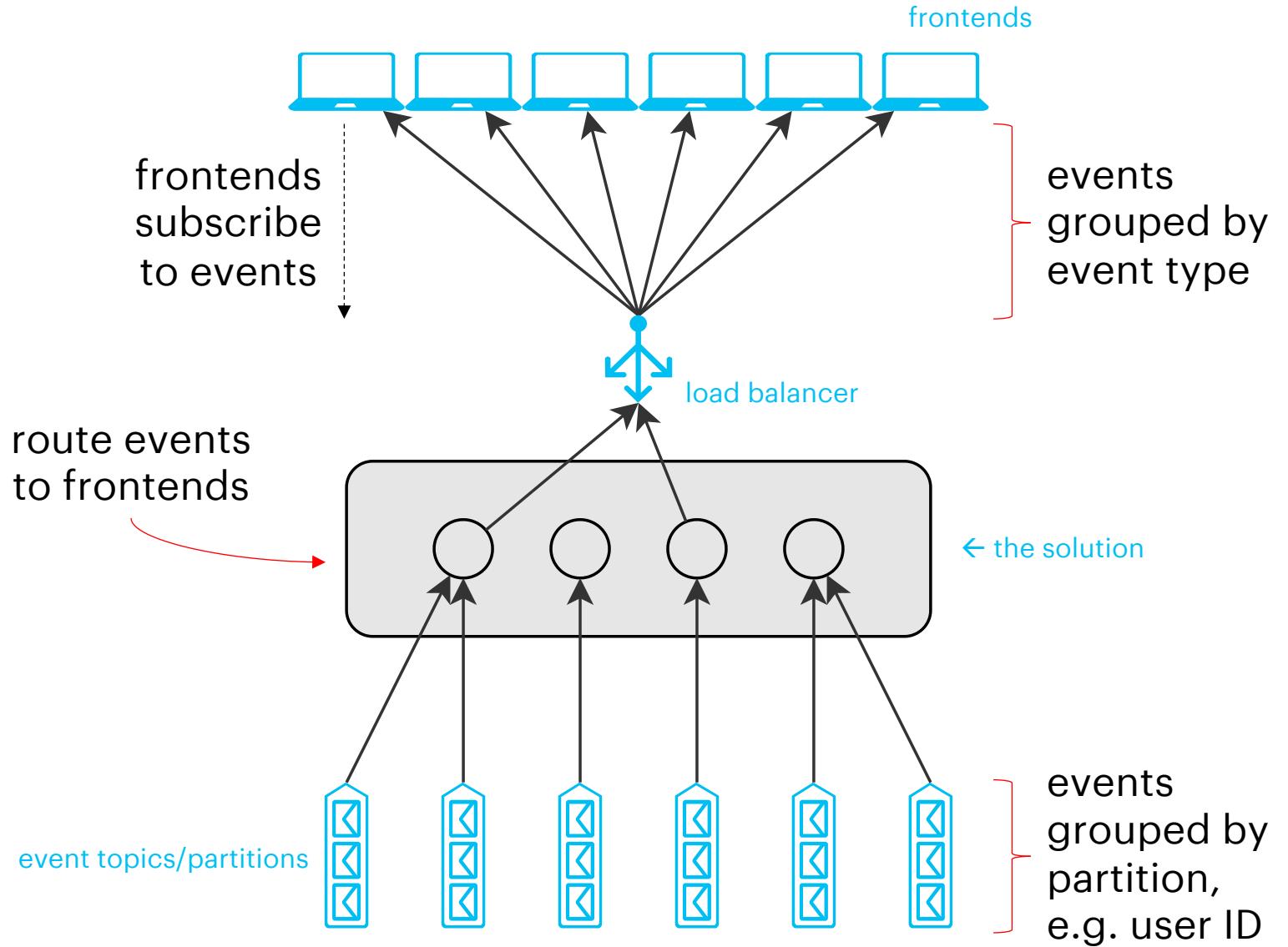


# CHALLENGES

- 1. Scaling out: number of users, rate of events**
- 2. Event sources**
- 3. Synchronous request, asynchronous processing**
- 4. Authorization**

# 1. SCALING OUT

- **Many users, few online**
- **Events from all microservices**



## 2. EVENT SOURCES

- **Kafka as the de-facto standard for implementing event-driven architecture:**
  - Confluent Kafka platform
  - Confluent Cloud on GCP
  - Azure Event Hubs has Kafka-compatible API
  - Amazon Managed Streaming for Kafka (MSK)
- **Publish via HTTP**
  - Easier to setup and use during dev and test
  - Used when decrypting data on-the-fly

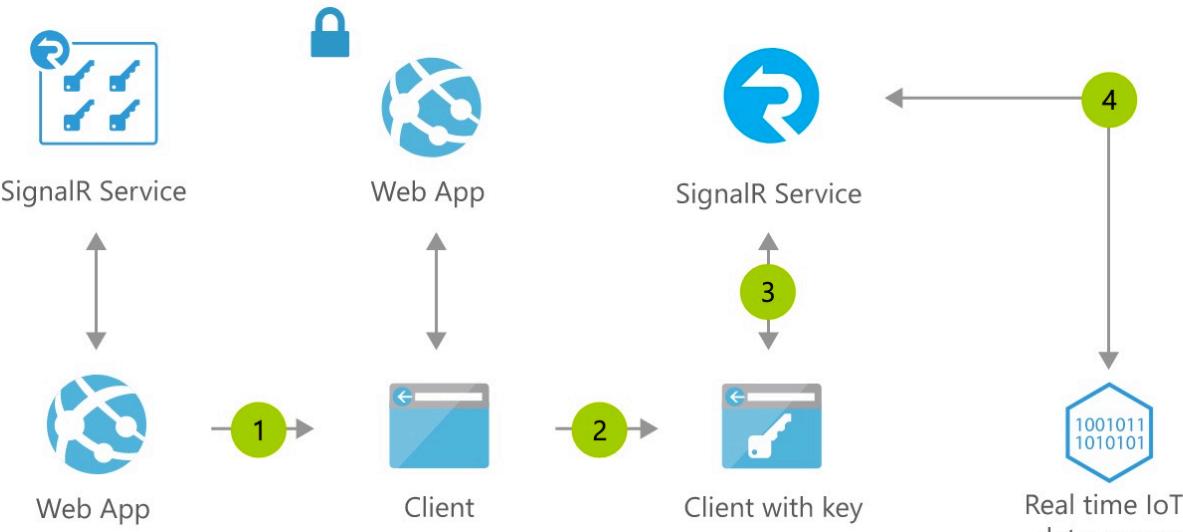
# 3. SYNC REQUEST, ASYNC PROCESSING

- **Asynchronous, event-driven processing is the new default**
  - Decoupling: easy to add/remove microservices
  - Deployment: easy to deal with upgrades/rollbacks/downtime
- **But: frontend and 3<sup>rd</sup> party clients often expect immediate response**
  - Requires “conversion” of asynchronously processed result into synchronous request-response

# 4. AUTHORIZATION

- **Is ESSENTIAL: any event may be subscribed to**
- **As little business logic at possible**
- **As pluggable as possible**

# CANDIDATE: ASP.NET SignalR



OVERVIEW

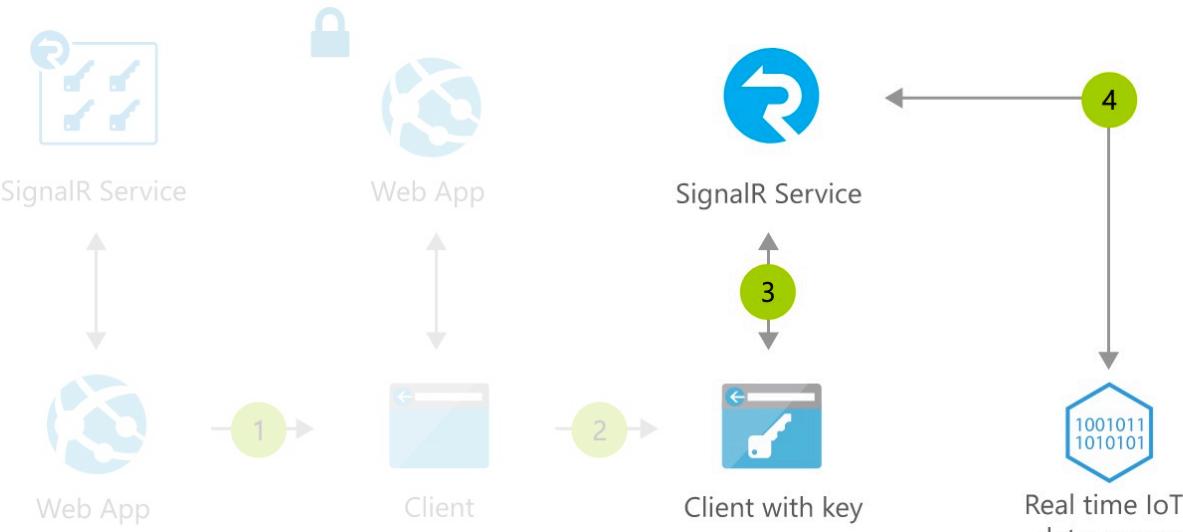
FLOW

- 1 Web app connects to SignalR Service and receives token
- 2 User connects to web app and gets SignalR endpoint and token
- 3 User connects to SignalR Service
- 4 Data from real-time source sent to SignalR Service and user

[Learn more >](#)

<https://azure.microsoft.com/en-us/services/signalr-service/>

# CANDIDATE: ASP.NET SignalR



OVERVIEW

FLOW

- 1 Web app connects to SignalR Service and receives token
- 2 User connects to web app and gets SignalR endpoint and token
- 3 User connects to SignalR Service
- 4 Data from real-time source sent to SignalR Service and user

[Learn more >](#)

<https://azure.microsoft.com/en-us/services/signalr-service/>

# CANDIDATE: ASP.NET SignalR

## Microservices should

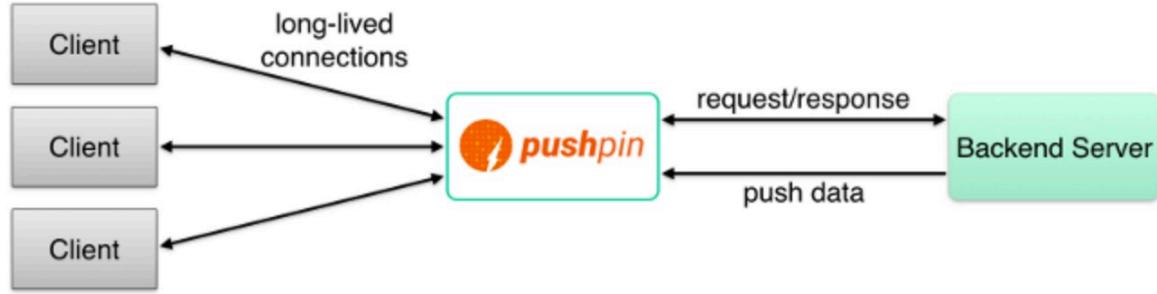
- not handle long-lived connections ✓
- not publish “special” events for frontend consumption X

## Frontends should

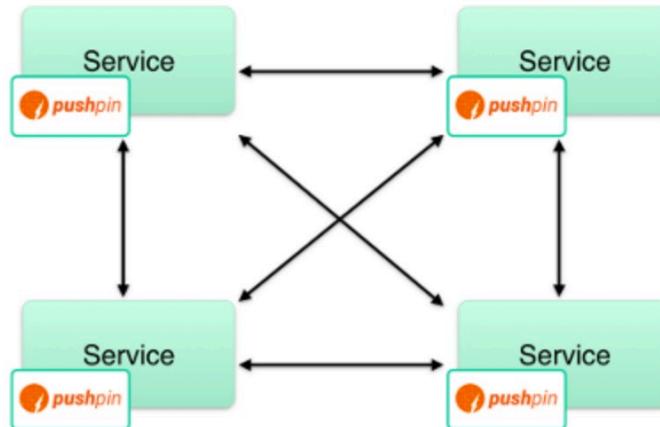
- be agnostic of event partitioning on the backend ✓
- not rely on proprietary formats ✓
- be able to publish events X
- be able to control what events they are subscribed to X



# CANDIDATE: Pushpin



*Pushpin has no built-in support for connecting to specific queues/brokers. Instead, you can write a small worker program that runs alongside Pushpin, to receive from the queue and send to Pushpin. Often you'll need to transform the data as well, and you can write any data transformation code in the same worker program.*



<https://pushpin.org/docs/about/>

# CANDIDATE: Pushpin

## Microservices should

- not handle long-lived connections ✓
- not publish “special” events for frontend consumption ✗

## Frontends should

- be agnostic of event partitioning on the backend ✓
- not rely on proprietary formats ✓
- be able to publish events ✗
- be able to control what events they are subscribed to ✗



# SOLUTION: Reactive Interaction Gateway

The screenshot shows the Cloud Native Landscape interface. On the left, there's a sidebar with filters for Grouping (N/A), Sort By (N/A), Category (N/A), CNCF Relation (Any), License (Any), Organization (Any), Headquarters Location (Any), and Example filters (Cards by age, Open source landscape, Cards in categories, Cards by stars, Cards from China). The main area displays the CNCF Cloud Events Landscape, which includes cards for various projects like KV, Reactor, and Reactive Interaction Gateway. The Reactive Interaction Gateway card is highlighted and shown in a larger modal window.

**Reactive Interaction Gateway**

**Accenture**

Orchestration & Management · API Gateway

Real-time UI events through CloudEvents subscriptions. Your UI deserves an API, too!

Website	<a href="https://accenture.github.io/reactive-interaction-gateway/">https://accenture.github.io/reactive-interaction-gateway/</a>
Repository	<a href="https://github.com/accenture/reactive-interaction-gateway">https://github.com/accenture/reactive-interaction-gateway</a>
Crunchbase	<a href="https://www.crunchbase.com/organization/accenture">https://www.crunchbase.com/organization/accenture</a>
LinkedIn	<a href="https://www.linkedin.com/company/accenture">https://www.linkedin.com/company/accenture</a>
Twitter	@AccentureTech
First Commit	2 years ago
Contributors	15
Headquarters	Dublin, Ireland
Market Cap	\$114B
Latest Tweet	this week
Latest Commit	this week
Latest Release	2 months ago
Headcount	10,001-1,000,000

# SOLUTION: Reactive Interaction Gateway

## Microservices should

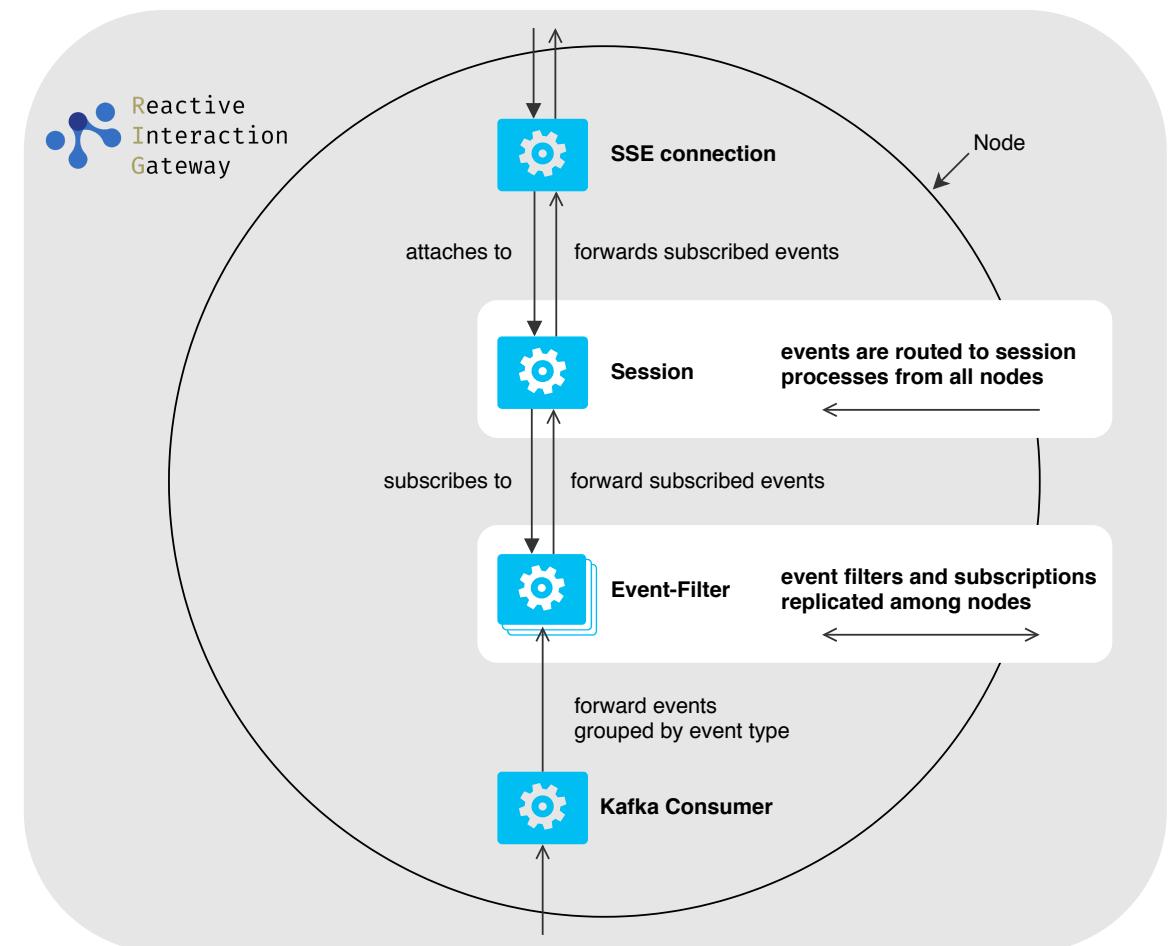
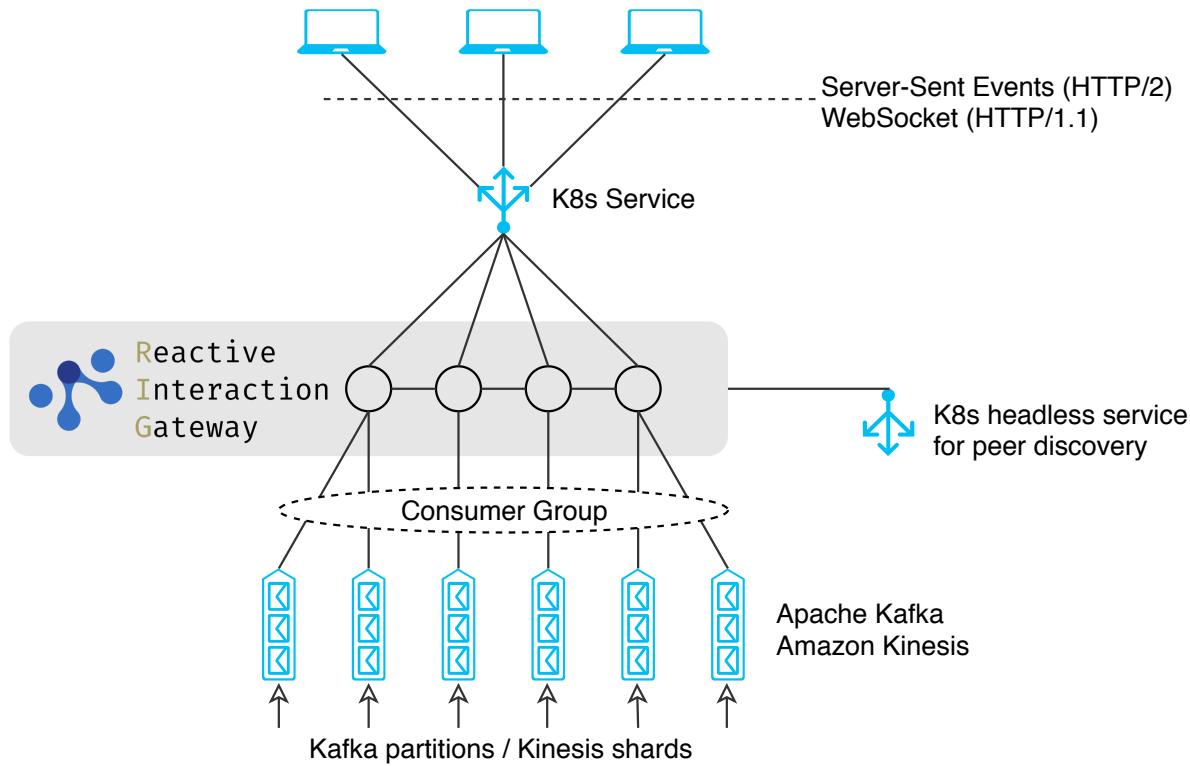
- not handle long-lived connections ✓
- not publish “special” events for frontend consumption ✓

## Frontends should

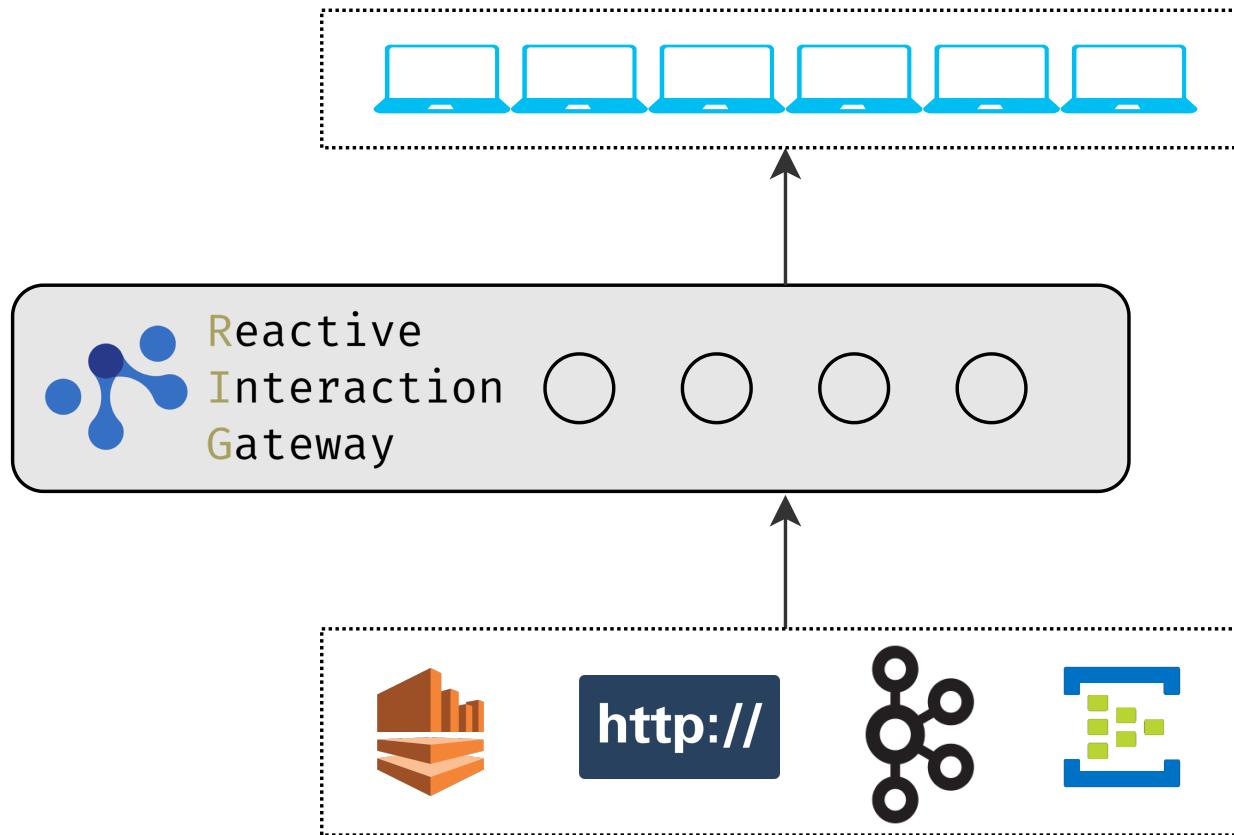
- be agnostic of event partitioning on the backend ✓
- not rely on proprietary formats ✓
- be able to publish events ✓
- be able to control what events they are subscribed to ✓



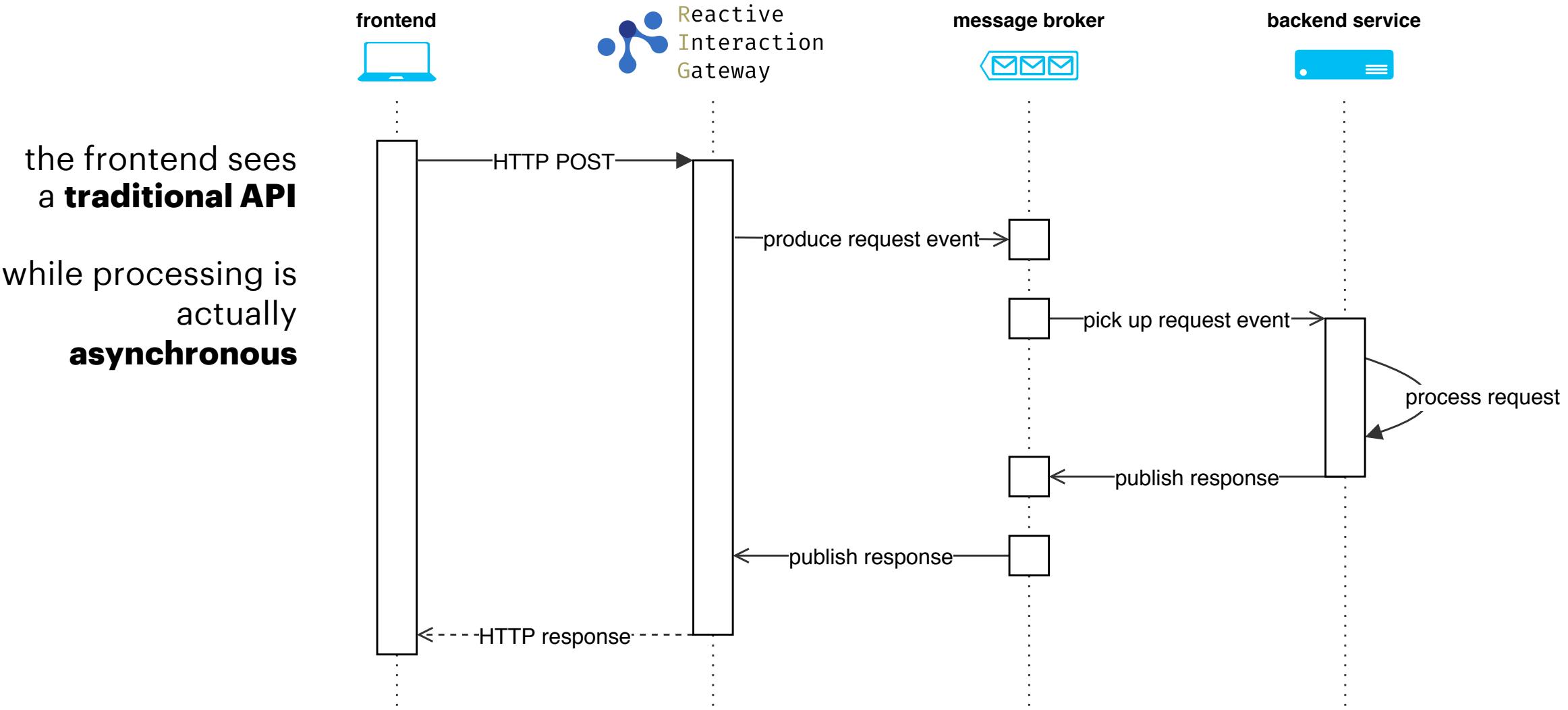
# 1. SCALING OUT



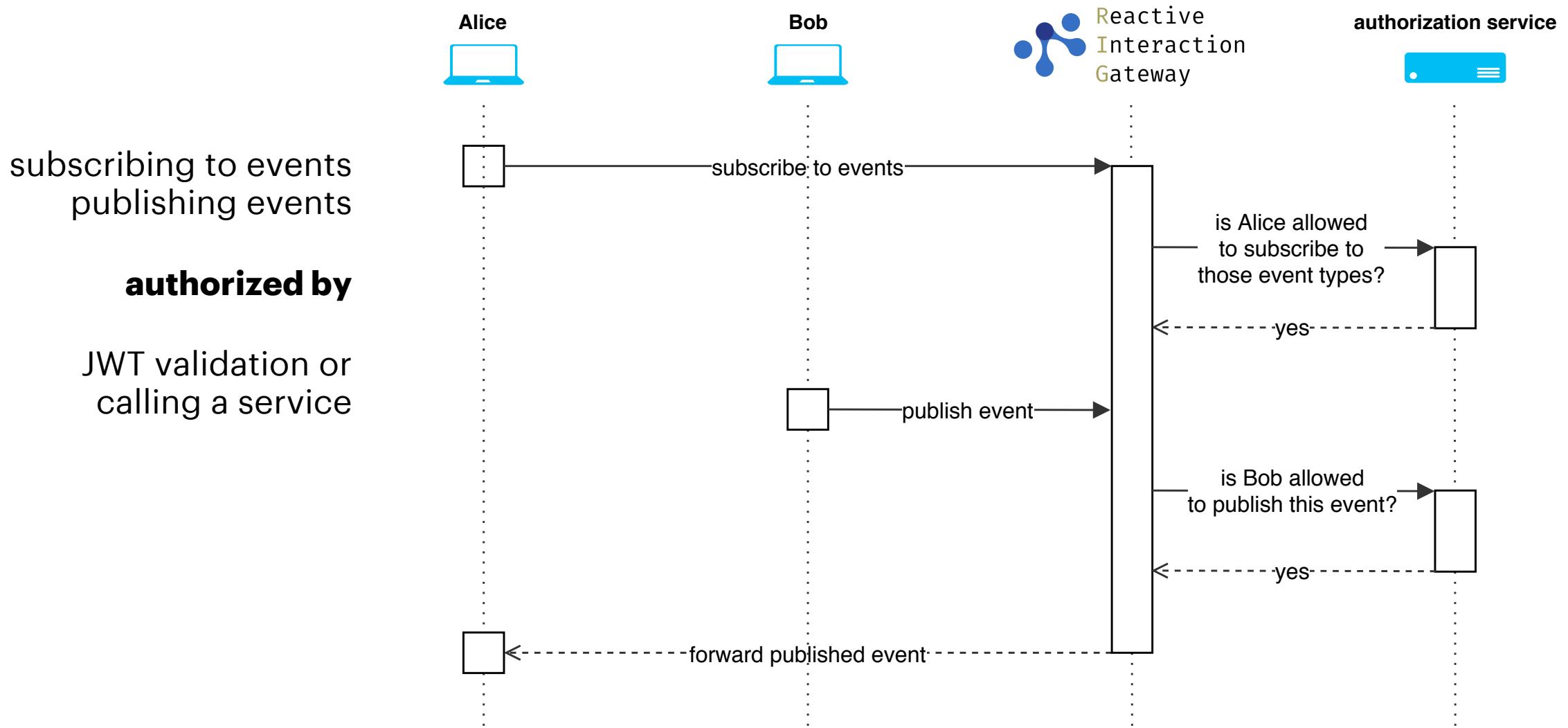
## 2. EVENT SOURCES



# 3. SYNC REQUEST, ASYNC PROCESSING



# 4. AUTHORIZATION



# Reactive Interaction Gateway

- **Free Software, Apache 2.0 License, developed on GitHub**
- **Open standards:**
  - **CloudEvents (CNCF Sandbox project)**
  - **HTTP/1.1 and HTTP/2**
  - **Server-Sent Events (SSE)**
  - **WebSocket**
  - **Kafka**

# Reactive Interaction Gateway

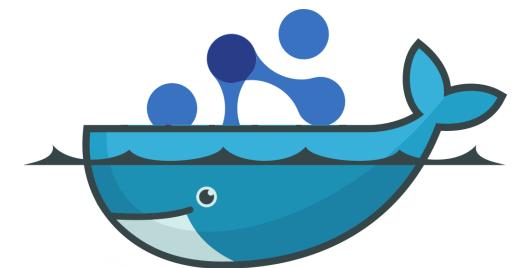
- **No external dependencies**
- **Configuration using environment variables**

- **Available on Docker Hub**

```
$ docker pull accenture/reactive-interaction-gateway
```

- **Scales like a stateless service**

```
$ kubectl scale deployment rig --replicas=10
```



# CONCLUSION

- **Real-time UI for great user experience**
- **Extending event-driven architecture to the frontend decouples frontend and backend**
- **The Reactive Interaction Gateway enables this in a scalable way, using open standards**

**Check out the Reactive Interaction Gateway  
and let us know what you think!**

[github.com/Accenture/reactive-interaction-gateway](https://github.com/Accenture/reactive-interaction-gateway)



Thanks to:

- Dominik Wagenknecht <- inventor
- Mario Macai <- long-term core team member
- Accenture's Software Innovation team

GitHub: kevinbader  
Twitter: @KevnBadr

# APPLICATION-LEVEL CONCERNS

- **Duplicate events**
- **Lost events**
- **Out-of-order events**