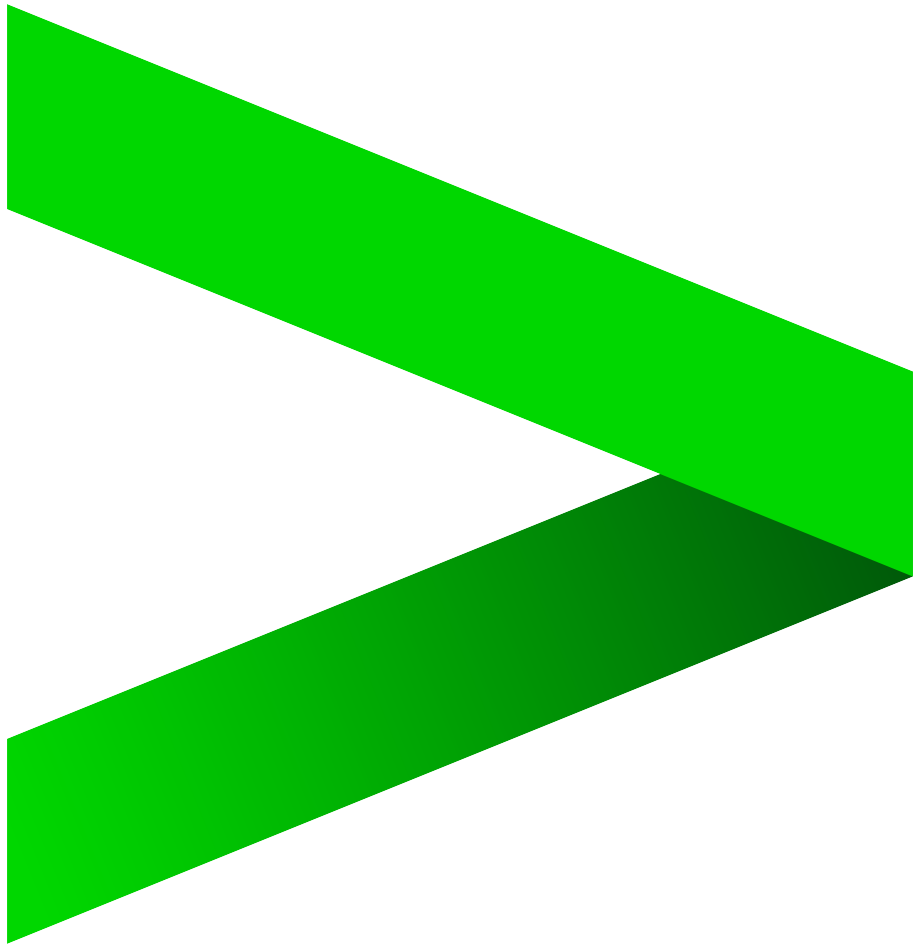
THE EVENT-DRIVEN UI

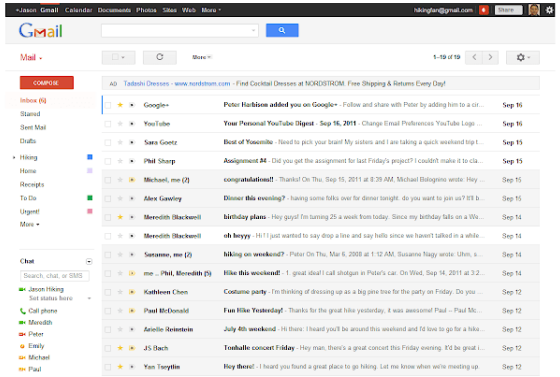
BRIDGING THE GAP

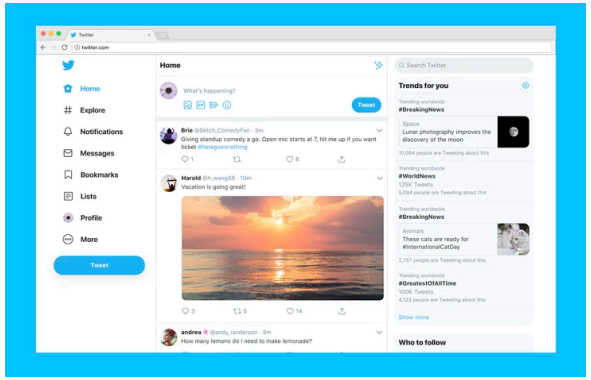


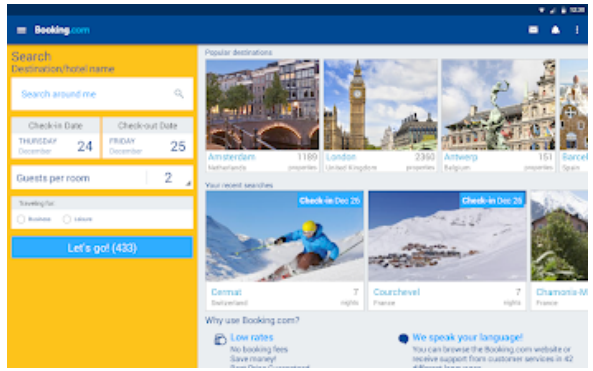
KEVIN BADER

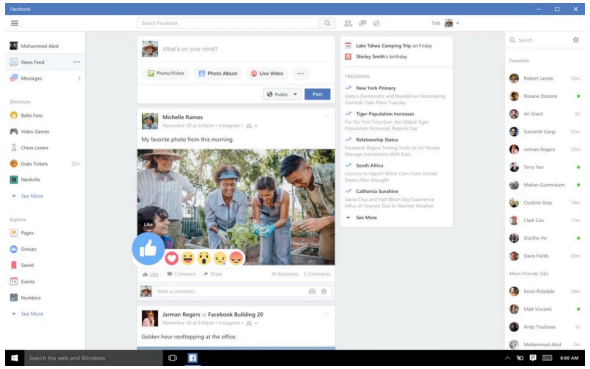
accenture technology

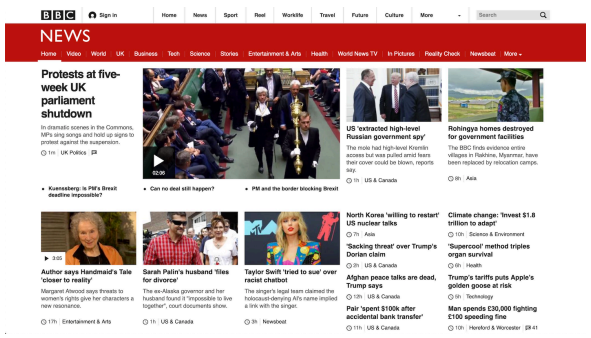
LIVE UPDATES!

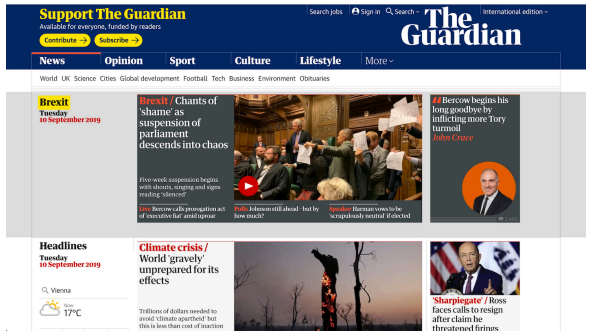










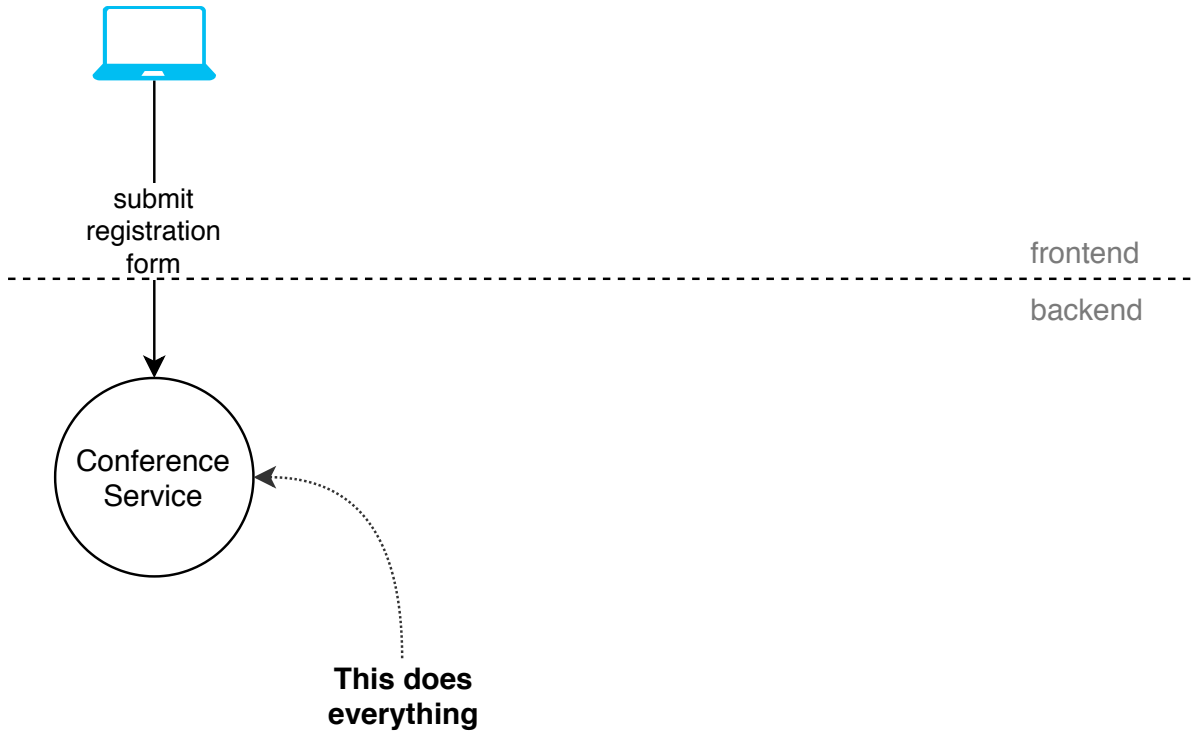


LIVE UPDATES

Example: Conference Registration

- **Allow people to register to the conference**
- **When someone has registered:**
  - send an email to the organizing team
  - send an email to the new participant
- **Assign a mentor to the participant**
  - It might take a while to find a good match..

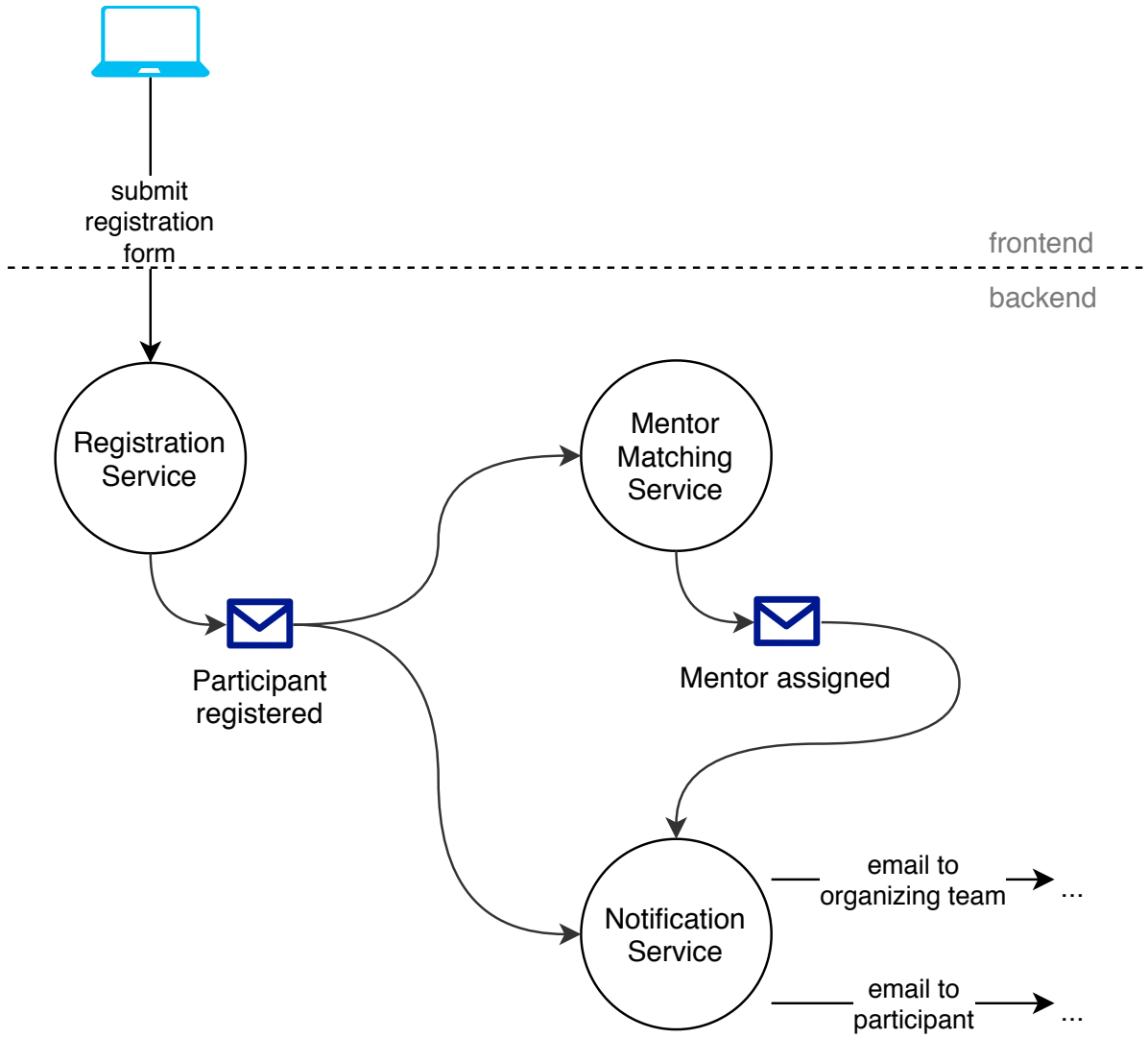
LIVE UPDATES FOR CONFERENCE REGISTRATION



One backend, one frontend:

one source of data and events.

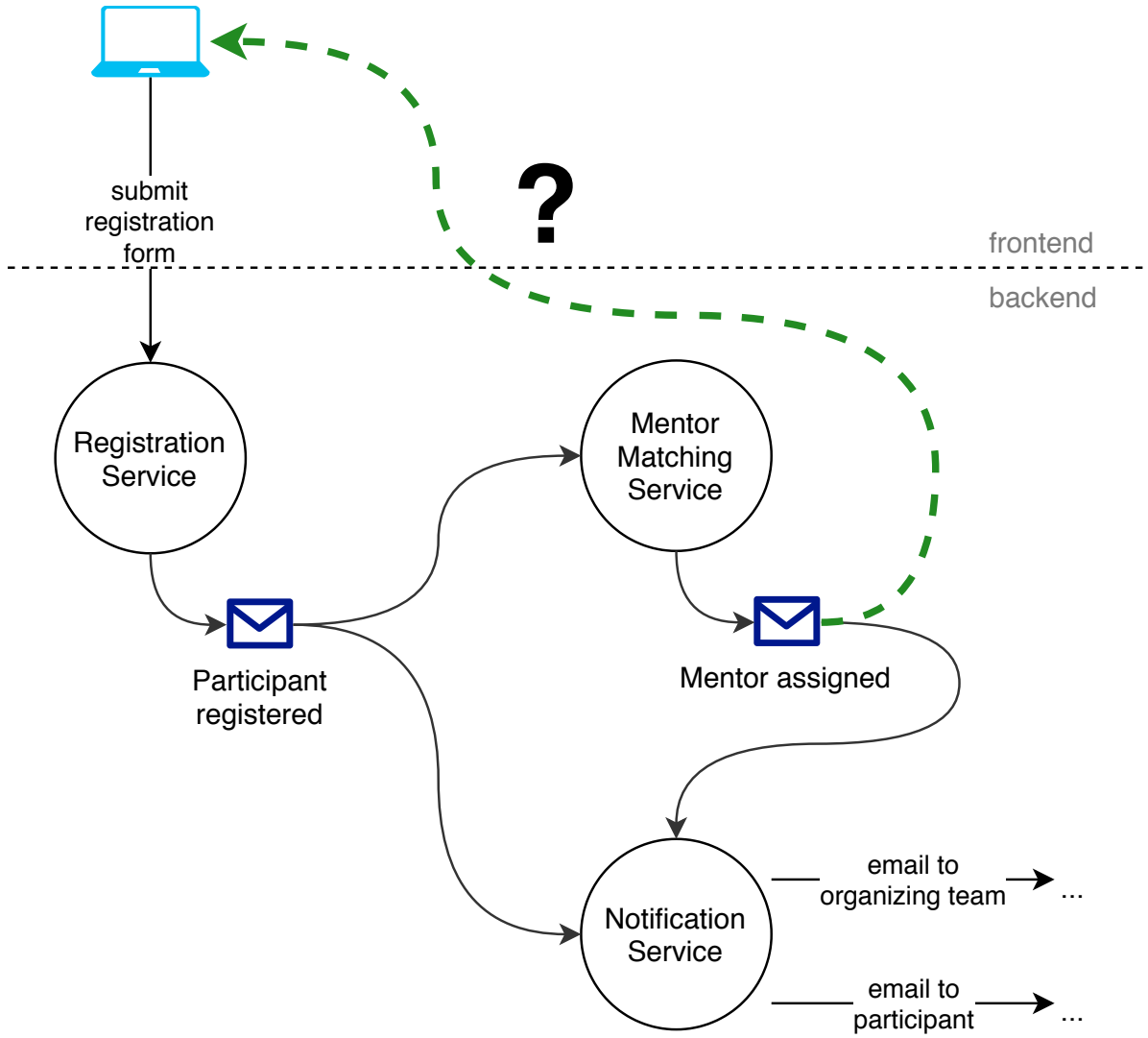
LIVE UPDATES FOR CONFERENCE REGISTRATION



Event-driven microservices.

Still one frontend.

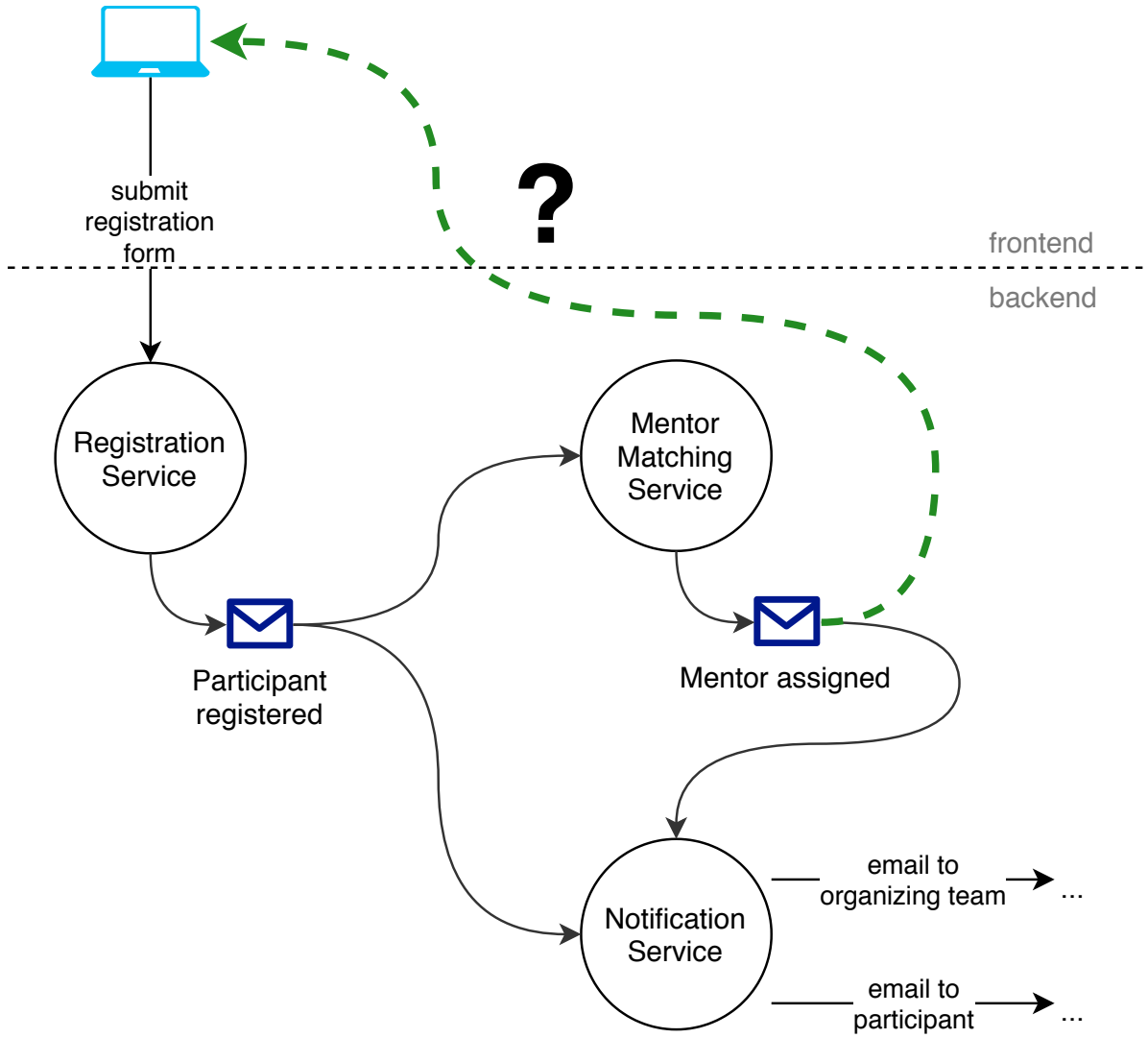
LIVE UPDATES FOR CONFERENCE REGISTRATION



Event-driven microservices.

Still one frontend.

LIVE UPDATES FOR CONFERENCE REGISTRATION



Event-driven microservices.

Still one frontend.

Microservice:
How do I know if someone is currently online?

Frontend:
How to subscribe to events?

AN EVENT-DRIVEN WORLD

Microservices emit events.

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

Frontends not aware how events are stored.

Only open standards.

Enable the frontend to participate in choreography:

- Frontends subscribe to events
- Microservice emit events and don’t target specific frontends
- Enhances decoupling!

AN EVENT-DRIVEN WORLD

Microservices emit events.

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

Frontends not aware how events are stored.

Only open standards.

Frontends communicates the user's actions.

Microservices can react:

- start a (long-running) process
- analyze a user's behavior
- notify the support staff
- ...

AN EVENT-DRIVEN WORLD

Microservices emit events.

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

Frontends not aware how events are stored.

Only open standards.

- Decouples teams
- Topic partitioning should not be part of the interface

AN EVENT-DRIVEN WORLD

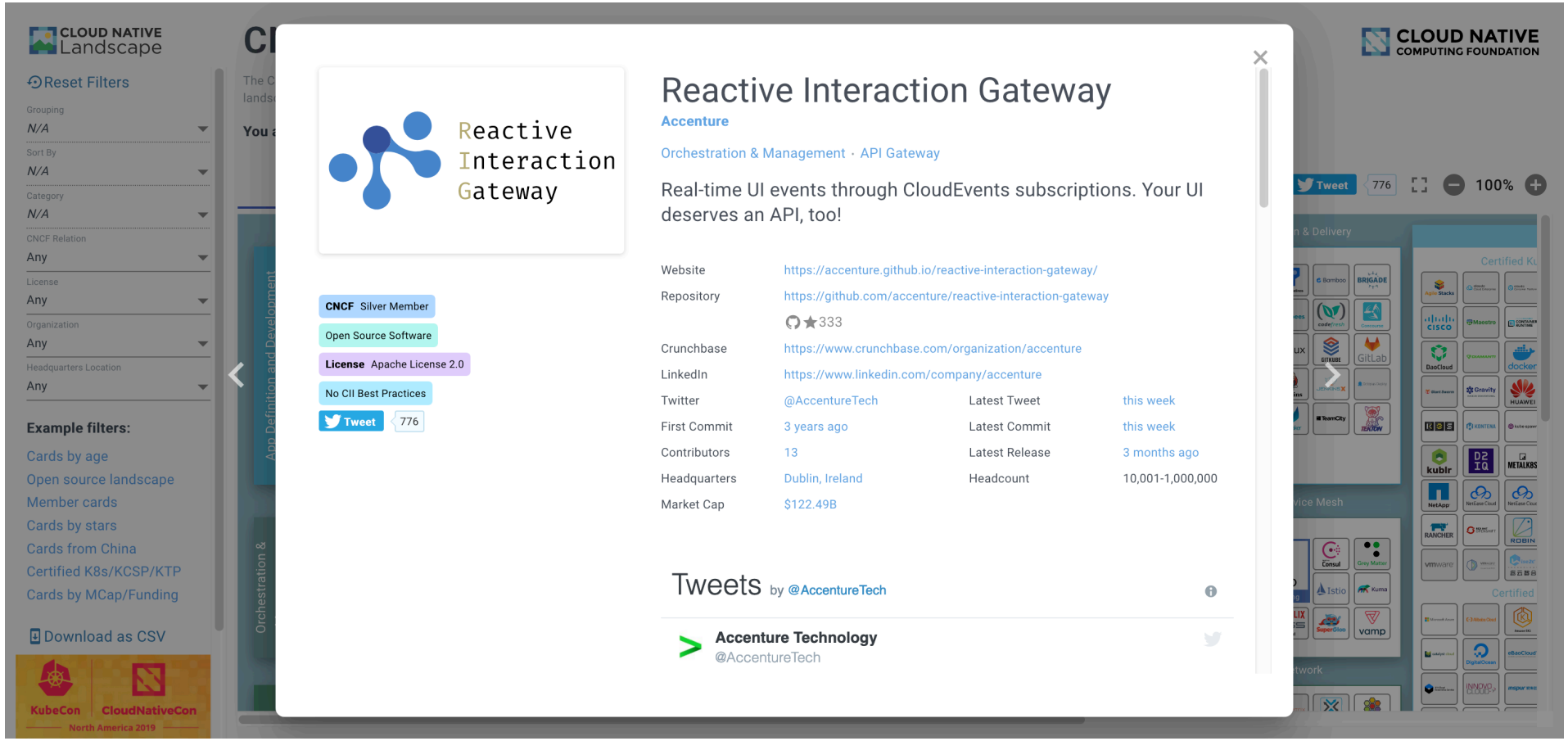
Microservices emit events.

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

Frontends not aware how events are stored.

Only open standards.

REACTIVE INTERACTION GATEWAY (RIG)

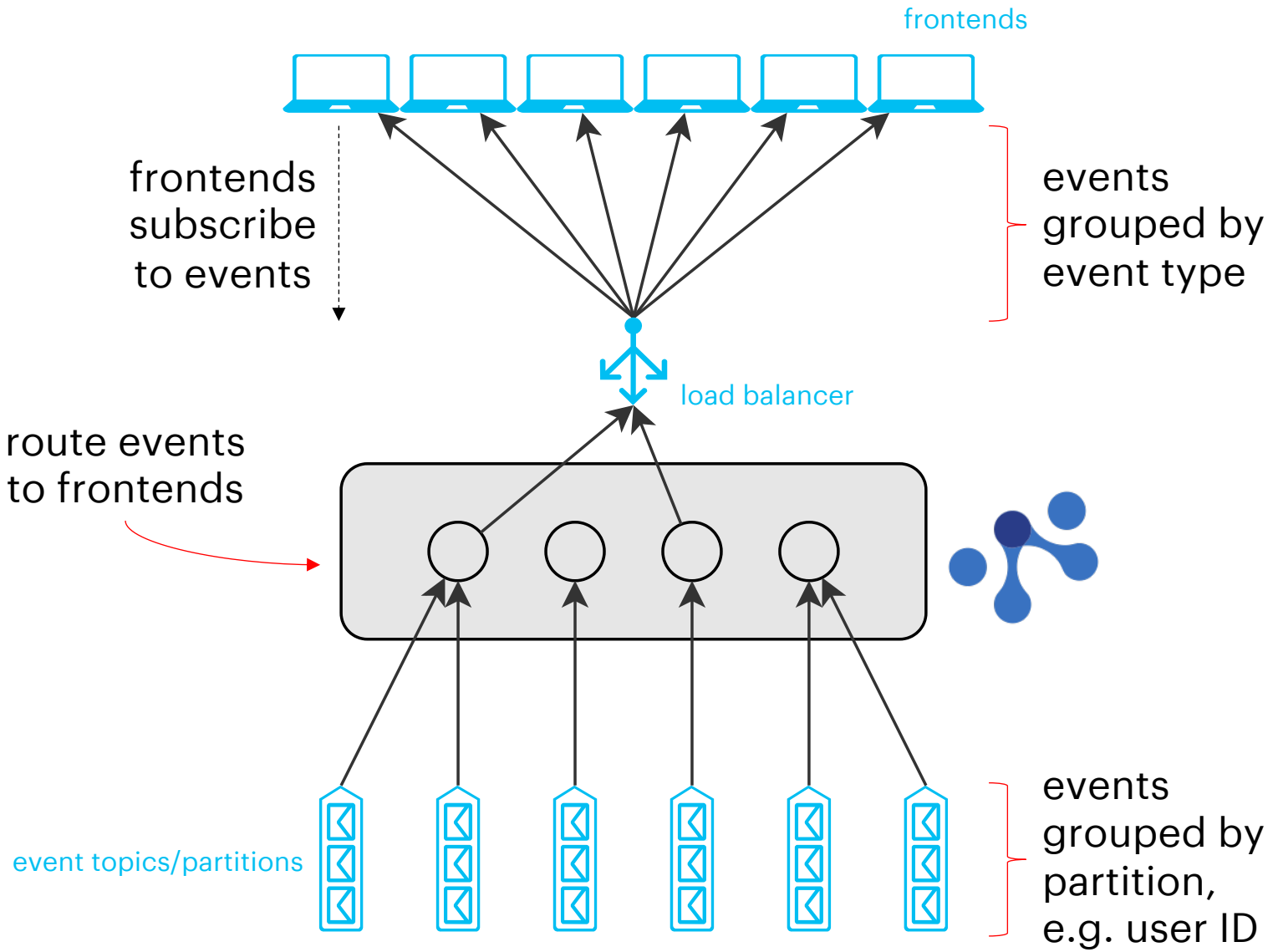


CHALLENGES

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

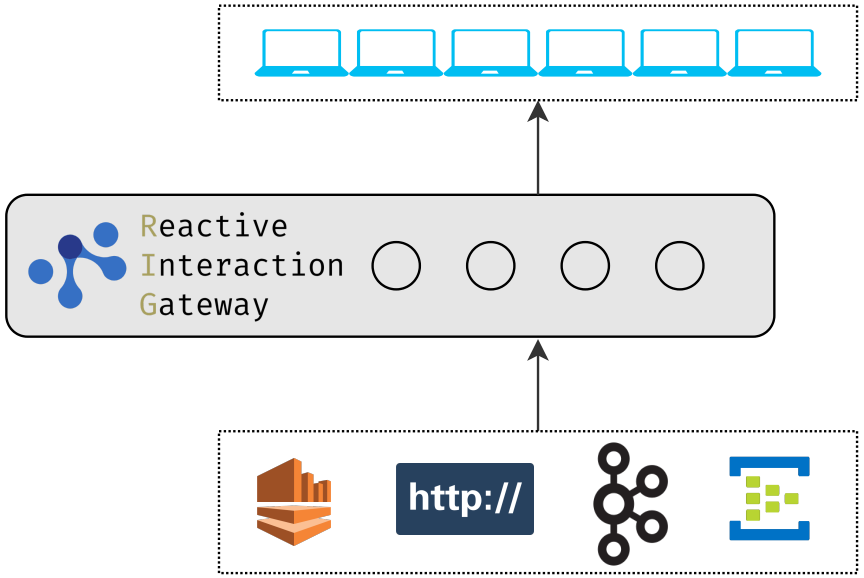
1. SCALING OUT

- **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



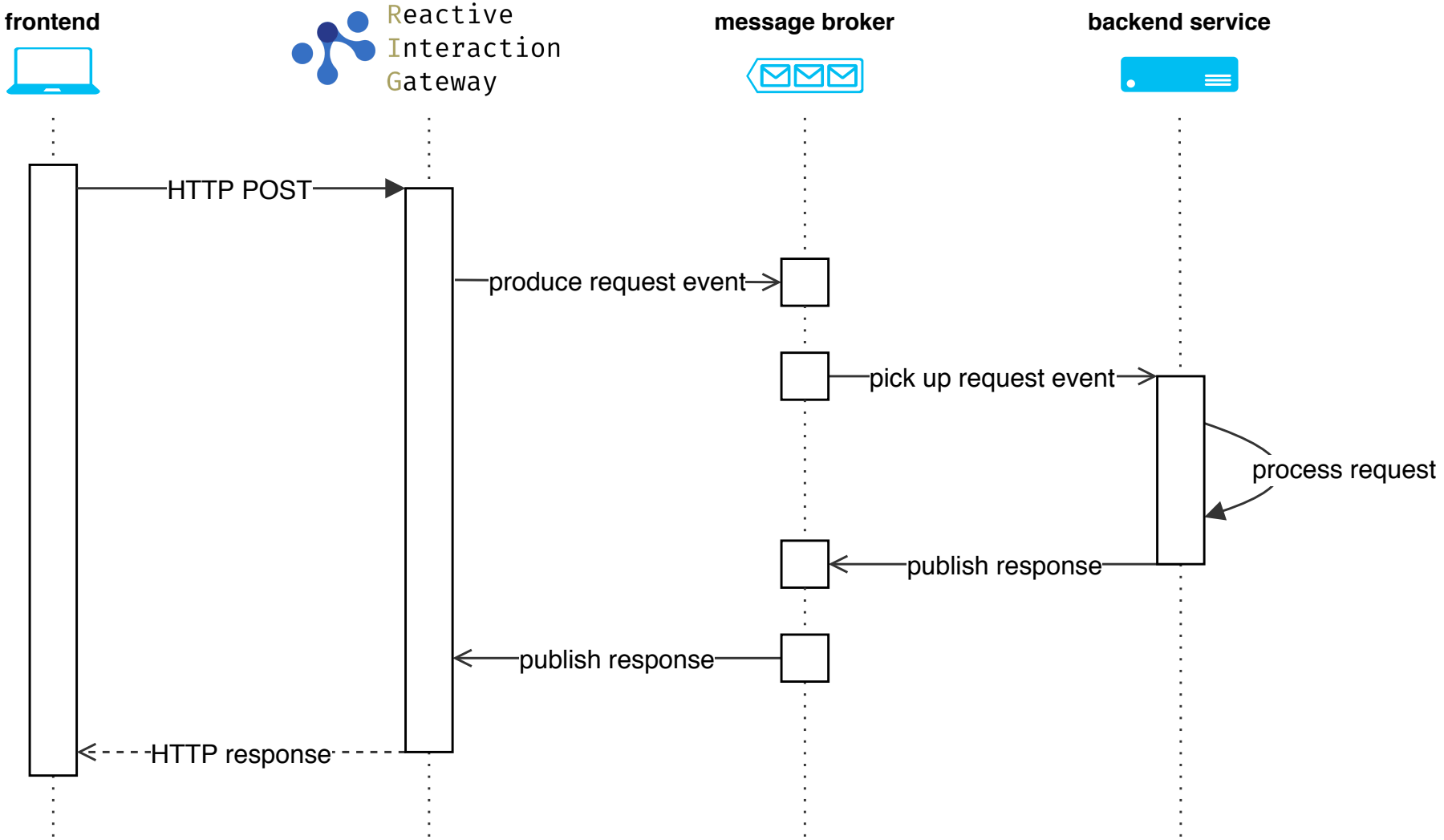
3. SYNC REQUEST, ASYNC PROCESSING

- **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 3rd party clients often expect immediate response**
  - Requires “conversion” of asynchronously processed result into synchronous request-response

3. SYNC REQUEST, ASYNC PROCESSING

the frontend sees a **traditional API**

while processing is actually **asynchronous**



4. AUTHORIZATION

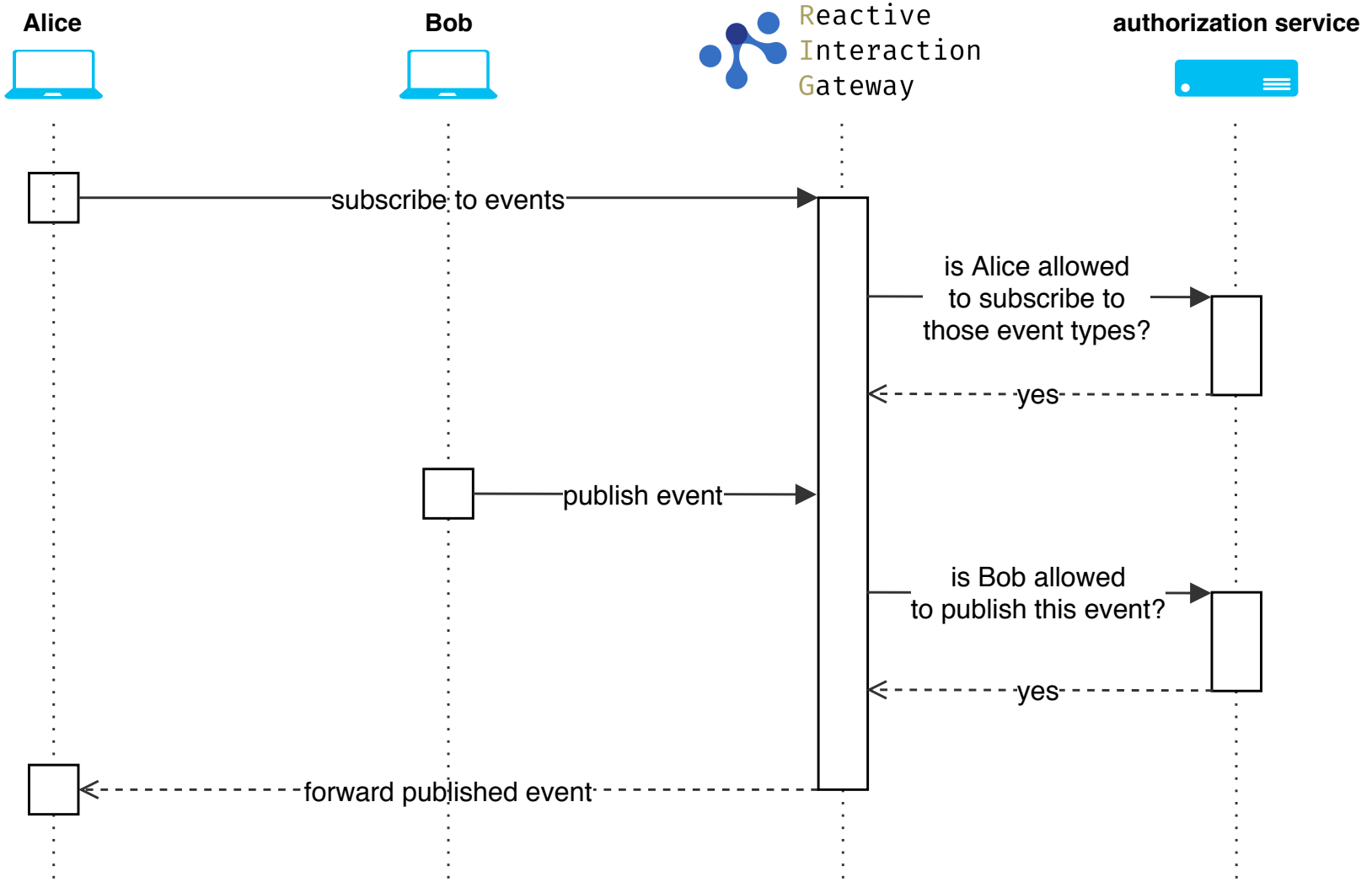
- **Is ESSENTIAL: any event may be subscribed to**
- **As little business logic at possible**
- **As pluggable as possible**

4. AUTHORIZATION

subscribing to events
publishing events

authorized by

JWT validation or
calling a service



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 ✓

REACTIVE INTERACTION GATEWAY

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

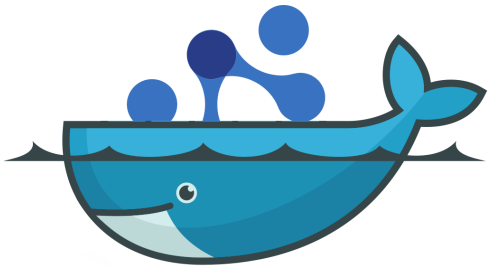
REACTIVE INTERACTION GATEWAY

- **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

Thanks to:

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

GitHub: kevinbader
Twitter: @KevnBadr