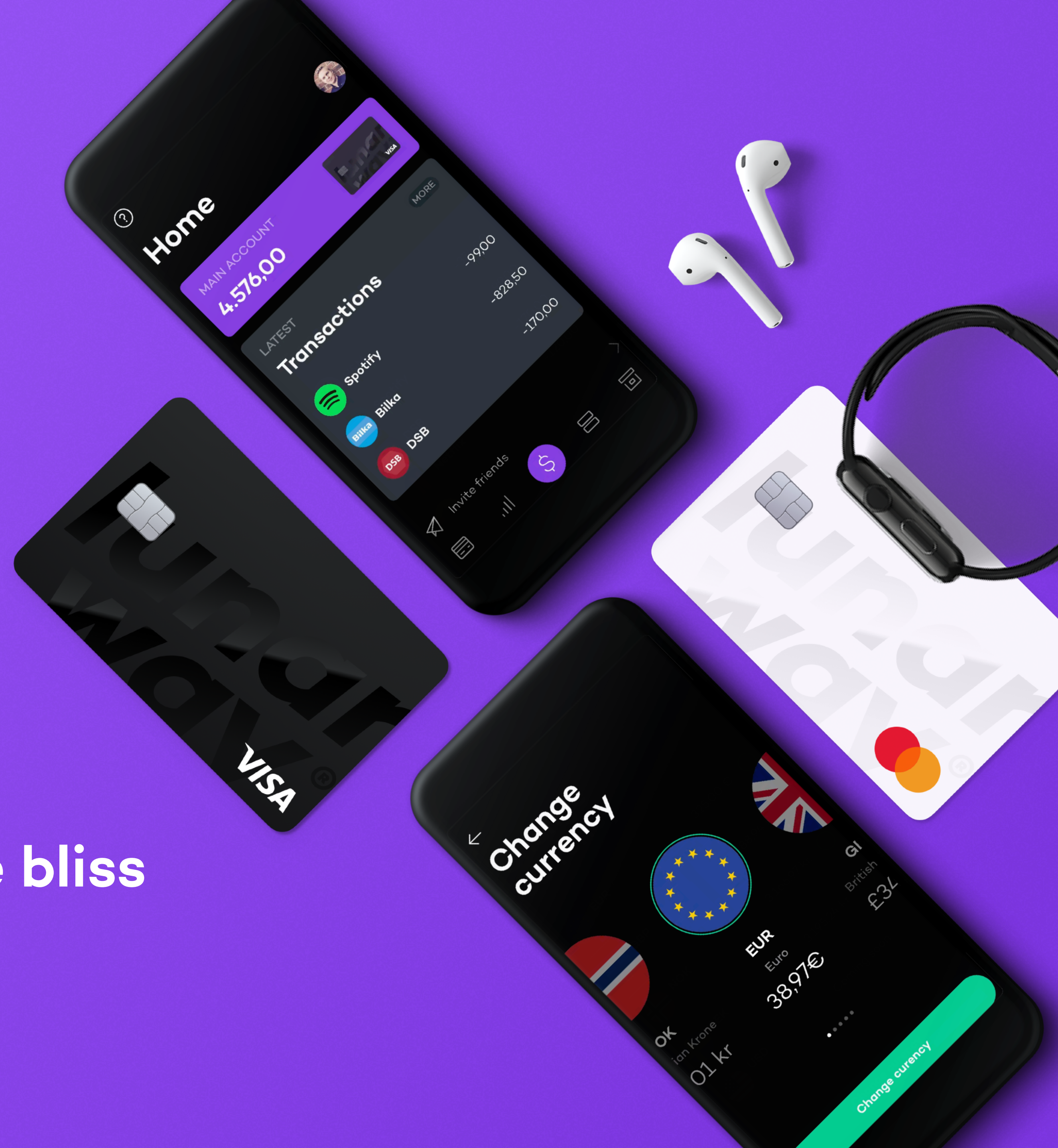


***lunar
way***[®]

To RPC or not to RPC:
Communication strategies
in a micro service world

OR

The Lunar Way to Microservice bliss



About me

- Started programming on the C64 and the Amiga back in 80s
- M.Sc. in Maths
- Professional software developer since 2004
- Started my career in Java
- Left Java in favour of Scala and FP
- Now primarily working in Go and Typescript

Although I miss Scala's type system, the simplicity of Go is often a bliss



Thomas Bøgh Fangel

Web Architect @ Lunar Way

 [@tbfangel](https://twitter.com/tbfangel)

AGENDA

- **Setting the Scene**
- **Microservices and Inter Service Communication**
- **Our Road to Microservices**
- **Use Cases of Async Message Passing**

SIMPLE BANKING
 Visa-kort, Konto og App

7-Eleven 40,00
 H&M -9,99
 Martin W. -130,00
 Spotify -18,70
 Zalando -8,50
 Uber
 McDonald's
 ASOS

lunar way

MAKE MONEY MATTER.

lunar way



DIRTY MONEY

lunar way



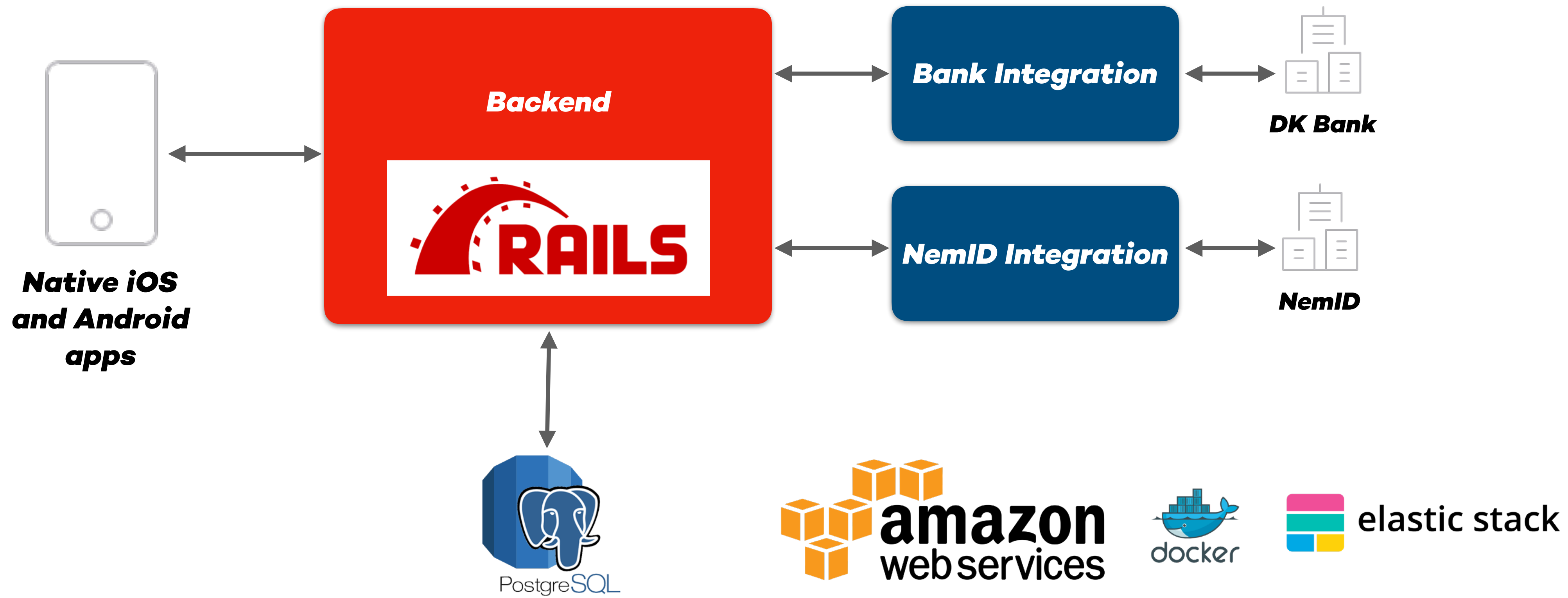
thelunarway

lunar way

lunar way

KAPSEJLADS

The Lunar Way platform of Summer 2016



Platform assessment

The Good

- Well structured REST API for the app

The Bad

- Tightly coupled data model on the backend
- App and backend tightly coupled
- One data entity all over the place
- Big Bang deployments
- Hard to do fast experiments
- Hard to scale

Change required!

Goals

- Scalability
- Resilience
- Autonomy
- Decoupling
- Fast experiments
- Small, independent and fast deployments

A top-down photograph of a group of people sitting on a green lawn. Their hands and feet are arranged in a circle, touching each other. Many of the people have their toenails painted green. Some are wearing colorful beaded bracelets and wristbands. The scene is brightly lit, suggesting an outdoor setting during the day.

Microservices and inter service communication

So, what is a microservice?

Microservices are **small, autonomous** services that **work together**.

– **Sam Newman**

...a single application as a suite of small services, each running in its **own process** and **communicating** with **lightweight mechanisms**...

...services are built around **business capabilities** and **independently deployable** by **fully automated deployment machinery**

– **Martin Fowler**

Why microservices – or why not?

Benefits

- Modularity
- Coherence and low coupling
- Fault tolerance
- Fast development
- Autonomy
- Independent deployment

Challenges

- Sharing data across services
- Debugging and tracing
- Orchestration
- Deployment
- Increased overall complexity
- Insight across service boundaries

Inter service communication

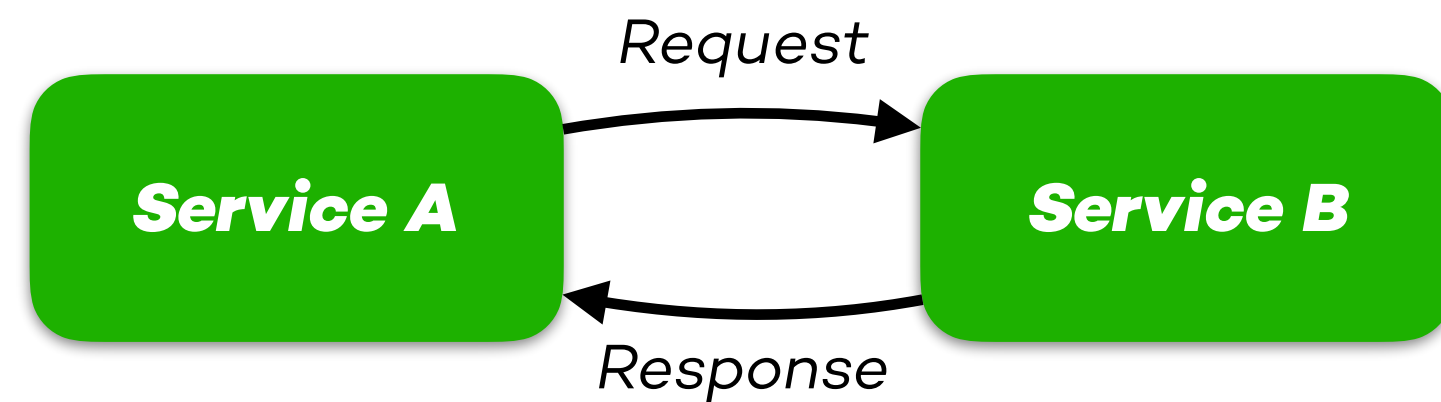
Synchronous req/resp

- Closed communication
- Trusted/ “secure”
- High coupling (code/space/time)
- Works good when synchronous app request involved

Async messages

- Open ended communication (pub/sub)
- Low coupling (data only)
- “Insecure” from a dev perspective
Flow orchestration is complex
- Works bad when synchronous app request involved

Communication shapes coupling



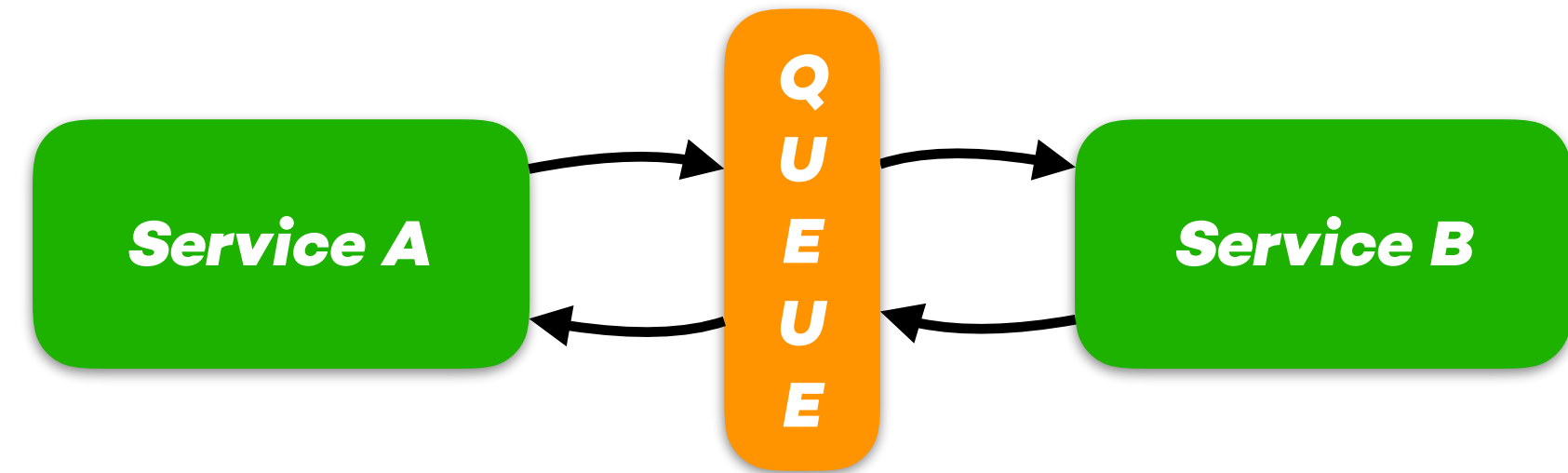
Synchronous req/resp

Temporal coupling

Spatial coupling

Behavioural coupling - Service A commands the behaviour of Service B

Example: Signup commands user service to CreateUser



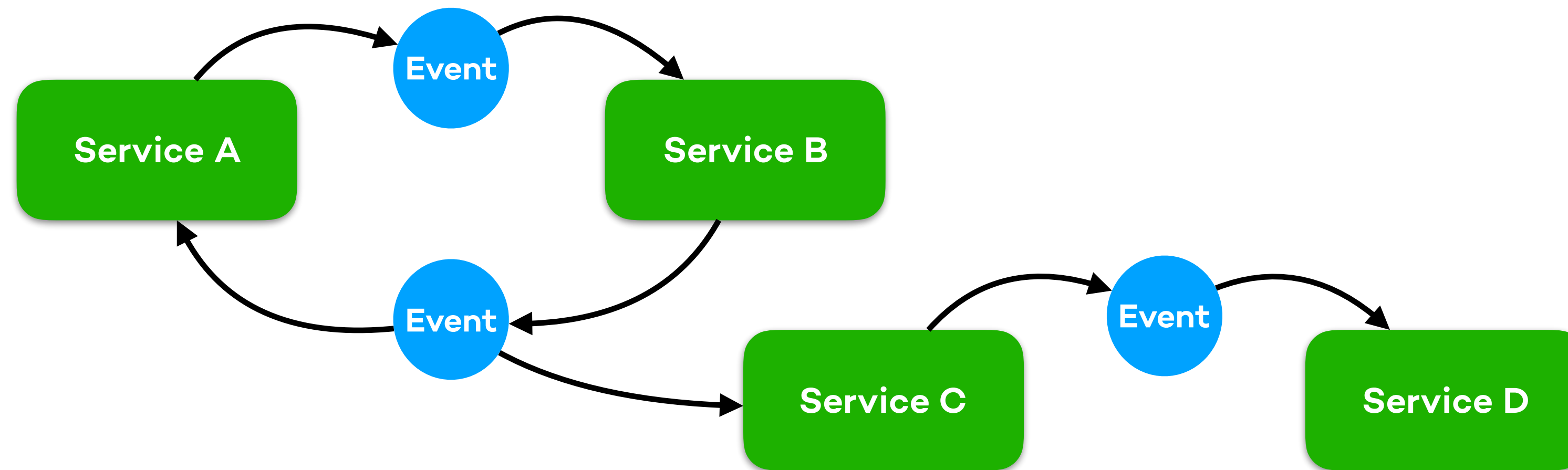
Async messages

No spatial and temporal coupling

No behavioural coupling - Service B determines its own behaviour based on the behaviour of Service A

Example: Signup publishes UserApplied. User service consumes event, creates user and publishes UserCreated. Signup service consumes and changes state

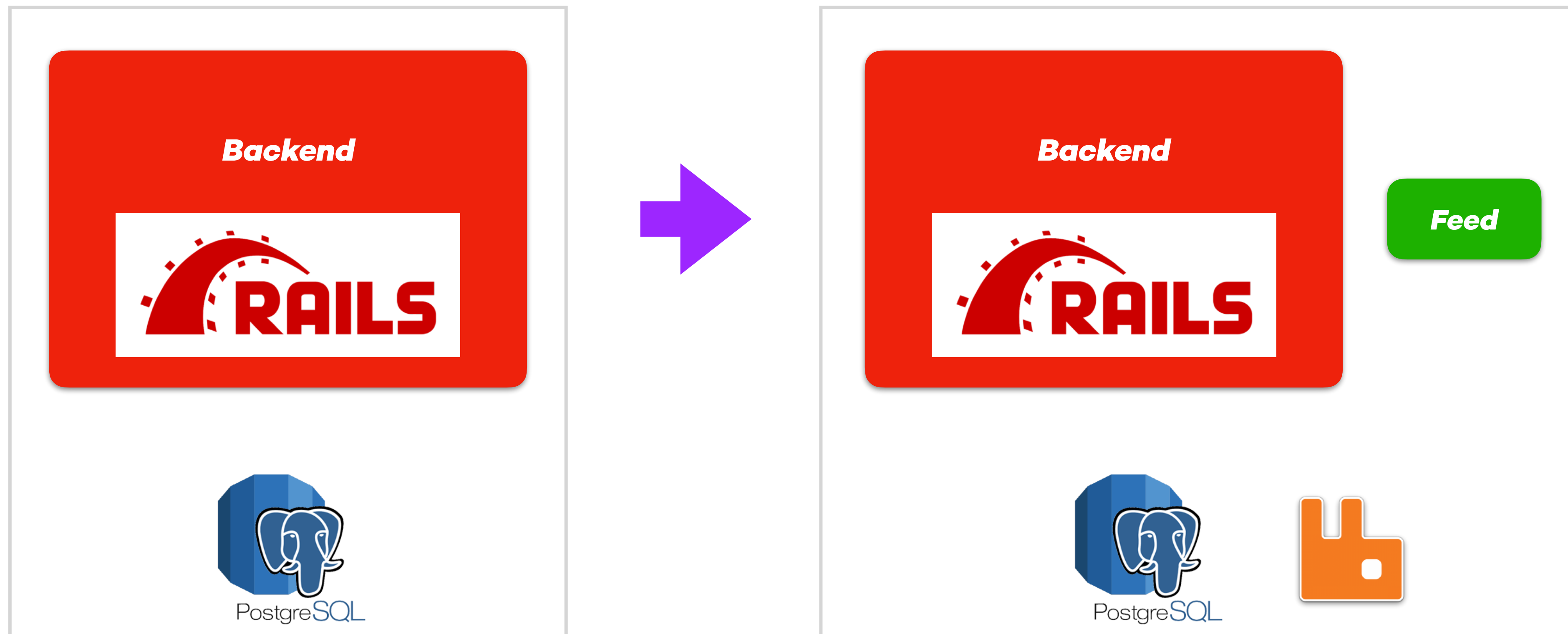
Event driven systems



- All changes published as events
- Events drive behaviour
- Traditional system design focus on only the state changes - events disappear after they happen
- Event driven systems complete the picture

Our Road to Microservices

The first microservice



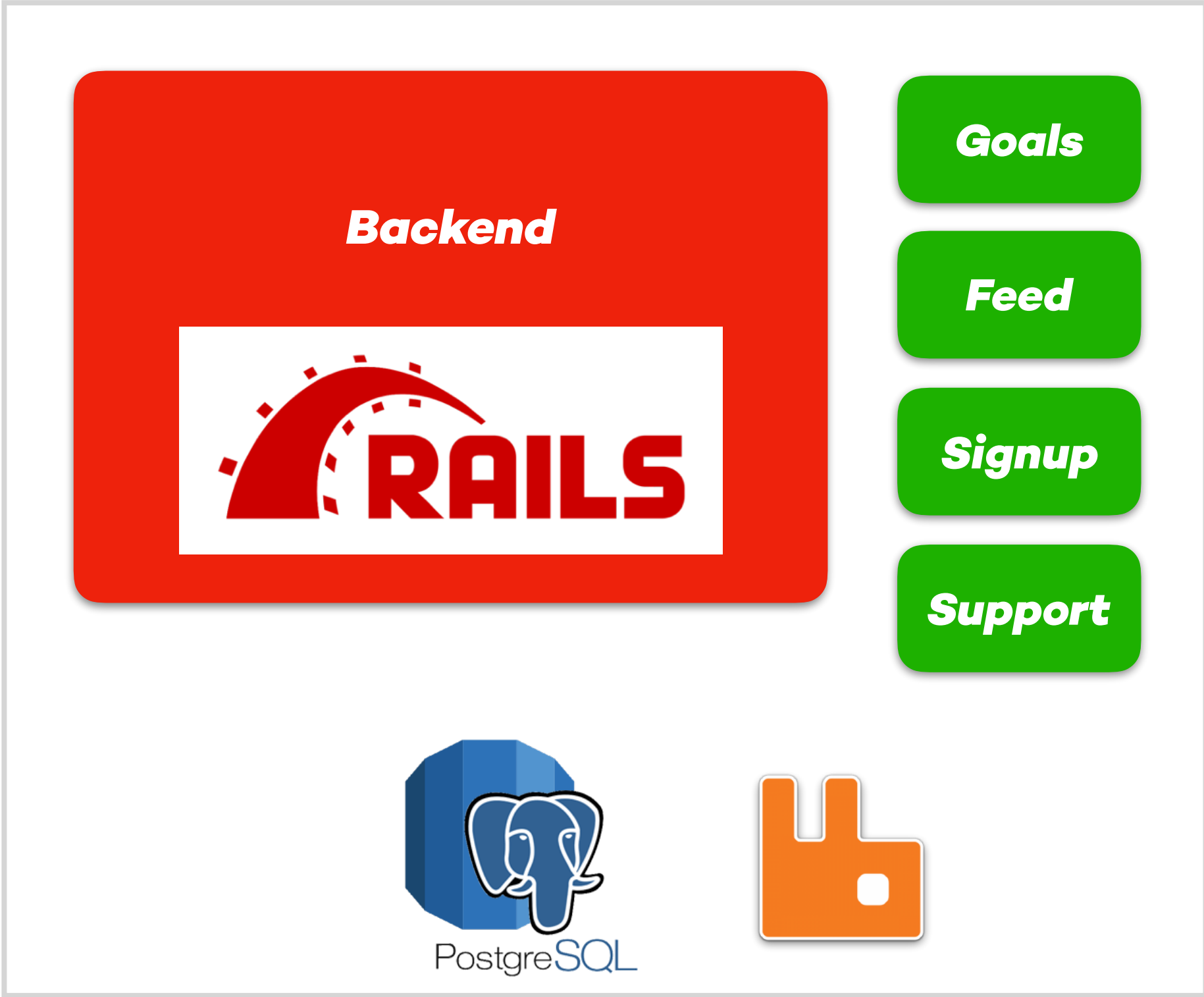
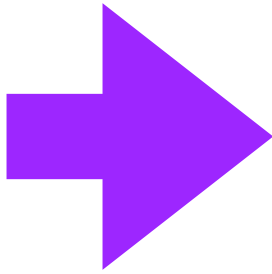
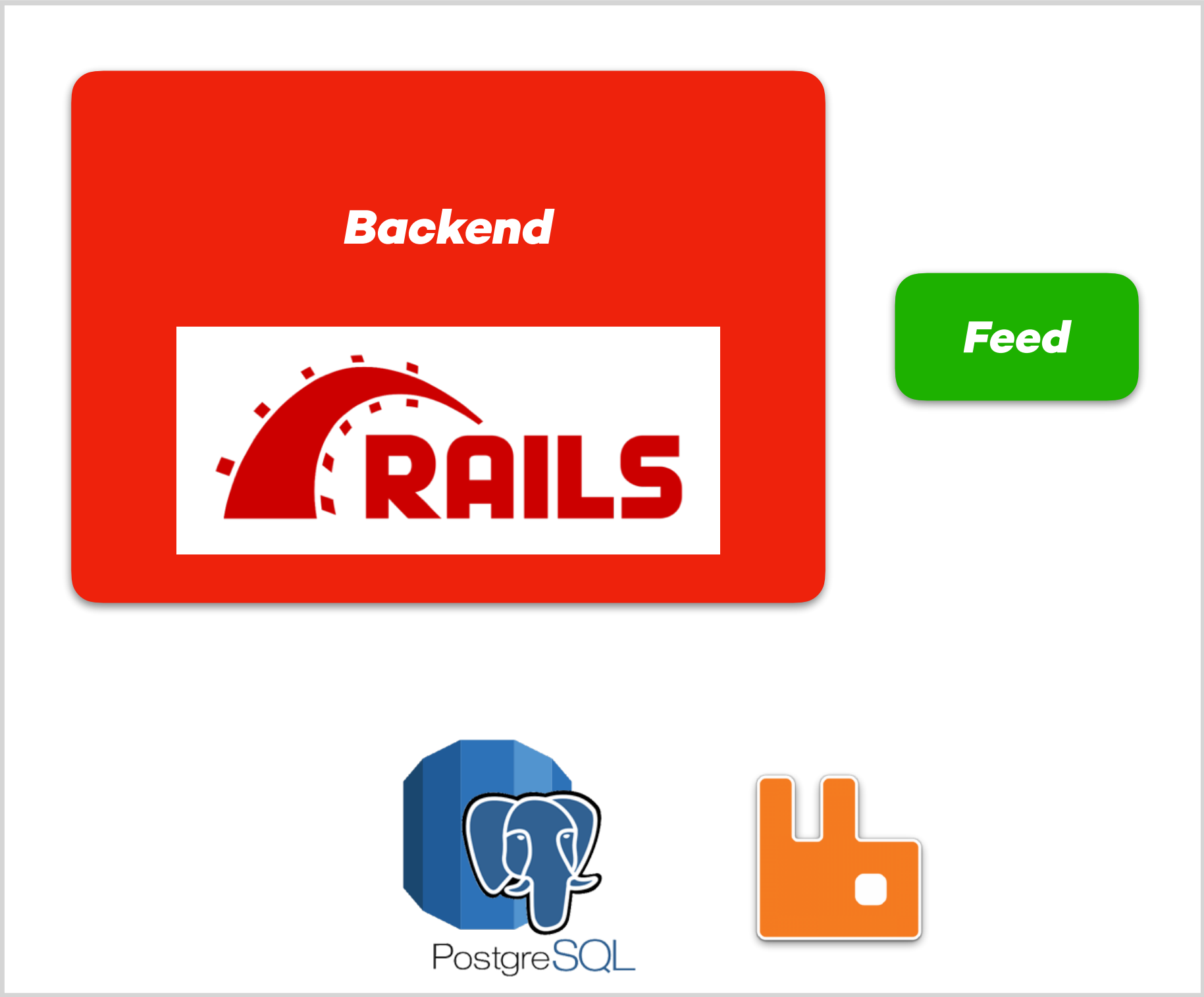
Learning 1

Never allow microservice A to access the data of microservice B directly

Learning 2

Reduce the number of
new technologies
introduced in one go

Microservice 2, 3 and 4



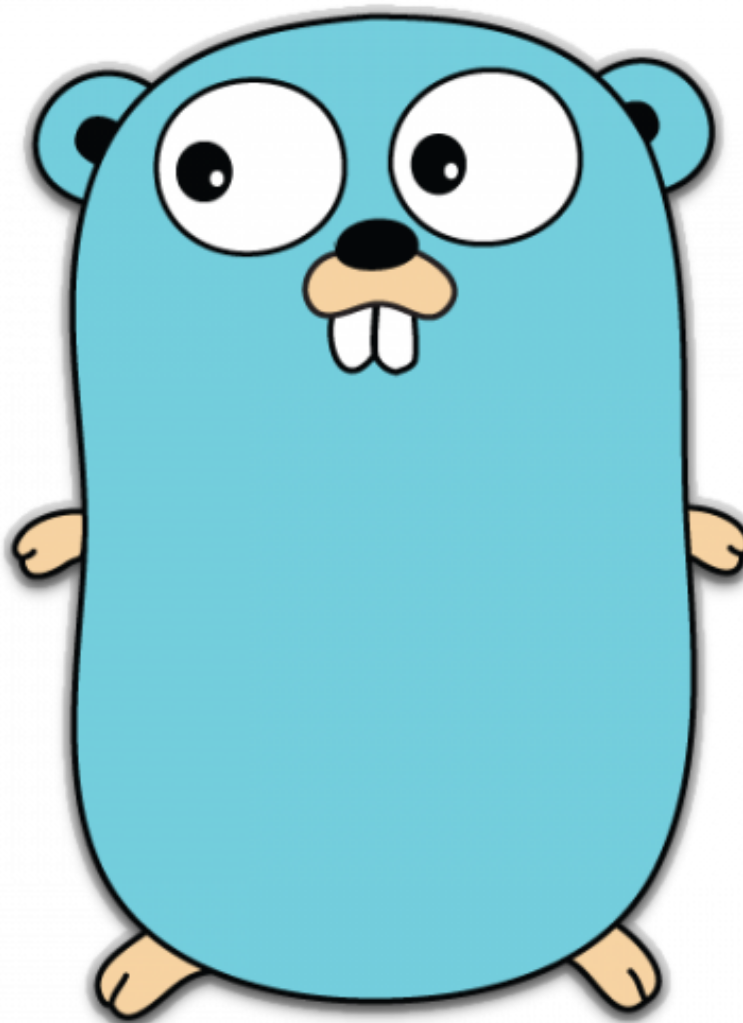
Learning 3

Prioritise your
deployment pipeline
and runtime platform

Microservice X, Y and Z



vs



Learning 4

Choose your
toolbox wisely

Learning 5

Insist on paying
off technical debt

**Microservice N,
N+1, N+2...**

**Beyond
counting...**

Learning 6

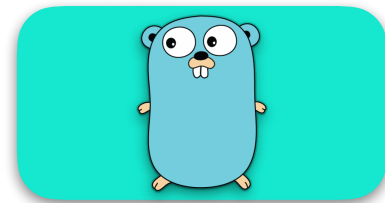
Be systematic!

Learning 7

DRY up your services
– factor out common
functionality into
new services

Learning 8

Appreciate the value
of publishing all
business model
changes as events



Native iOS
and Android
apps

App Facing

Push	Credit	Goals
Auth	Topup	Travel Card
User	Referral	Appsync
Feature	Move Money	Tracking
Feed	Social	User Settings
Stream	Signup	Split



Internal


Time	Locali- zation
Rules	Intercom Sync
KPI	KPI CLI
Insight	Promo
Support	TC Support

Houston
(Support)

Integrations

Bank Integration	DK Bank
NemID Integration	NemID
PayLike Integration	PayLike
PaySafe Integration	PaySafe
GPS Integration	GPS



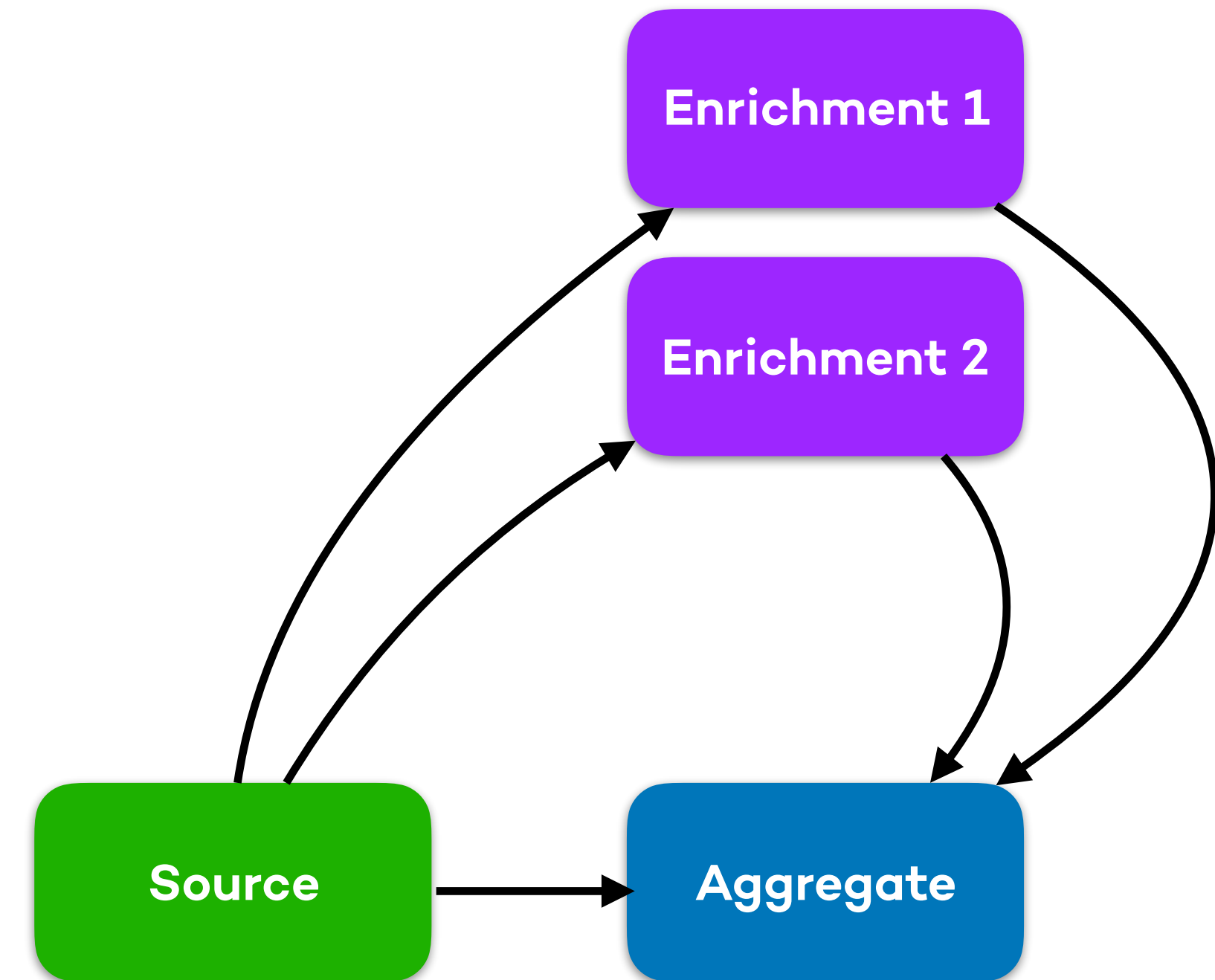
A glass bottle with a metal clasp is lying on a sandy beach. In the foreground, there is a seashell with many small, pointed teeth. The background is a blurred beach scene with other seashells.

Use Cases of Async Message Passing

Data enrichment

One source of data, multiple enrichments

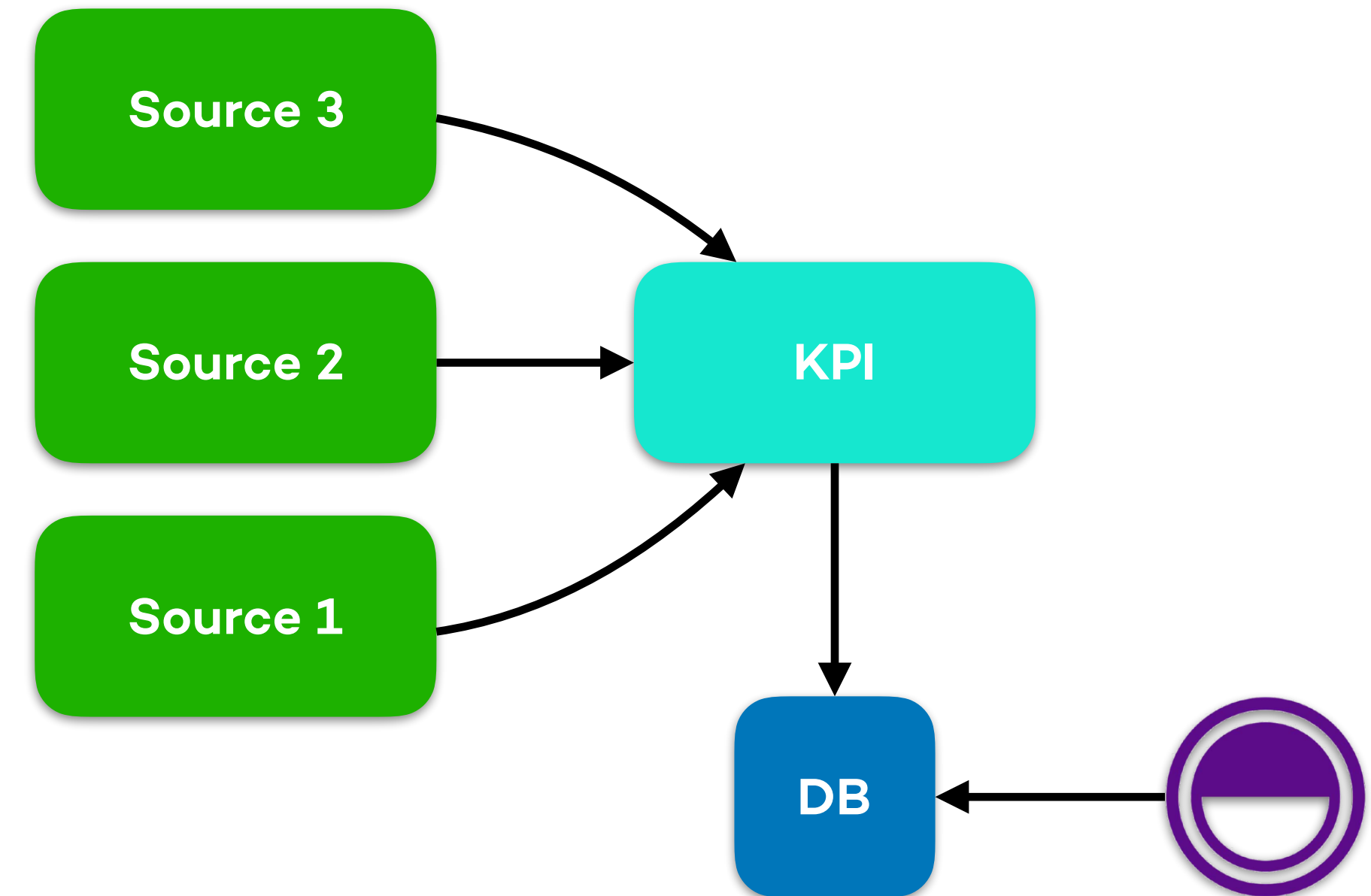
1. Source publishes event
2. Aggregate stores entity
3. Enrichments runs and publishes event
4. Aggregate updates aggregate with enrichment



Business insight

Business requires insight across services

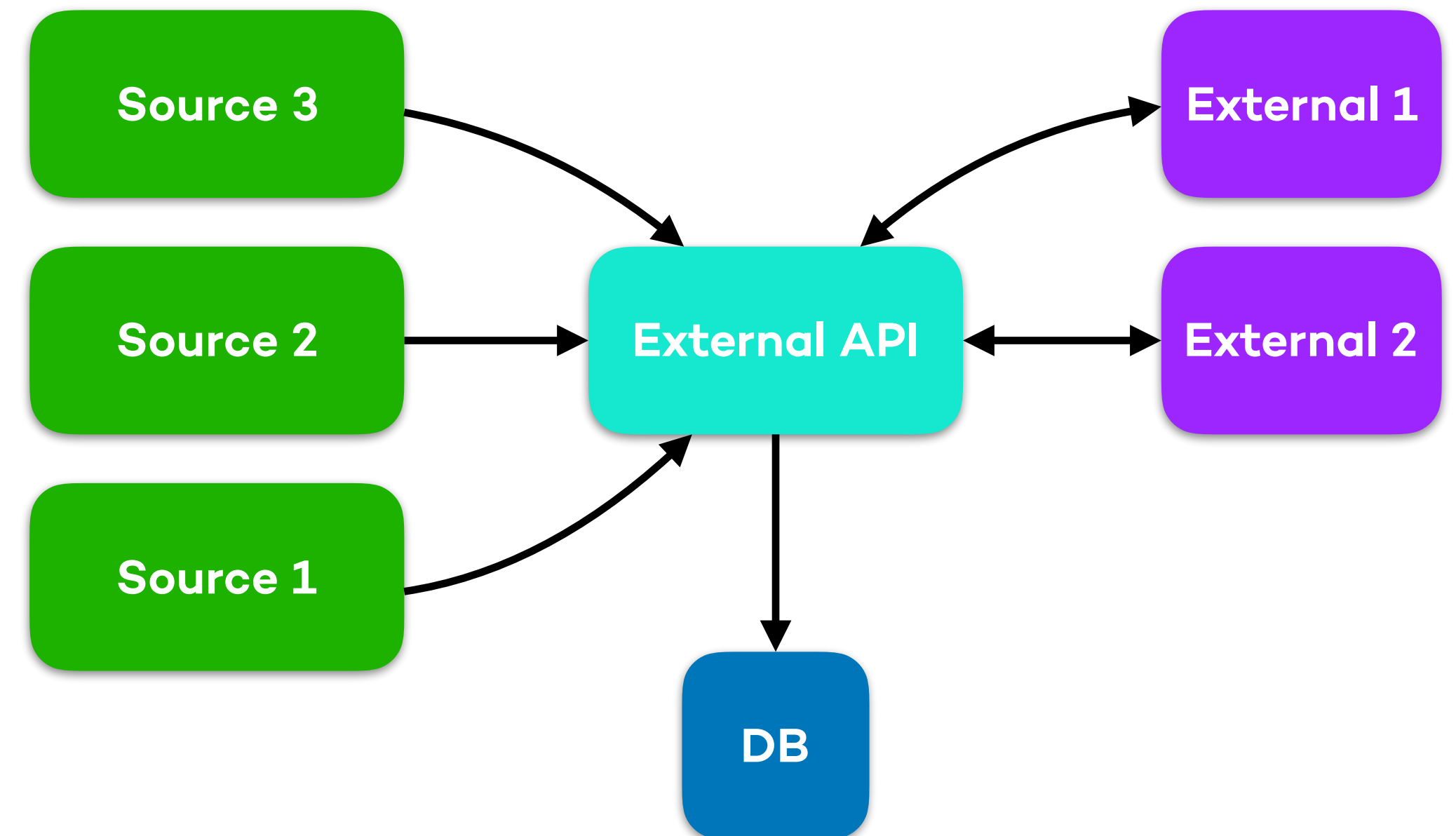
1. Sources publish events
2. KPI subscribes on events and converts to own model
3. Tooling on top to provide insight



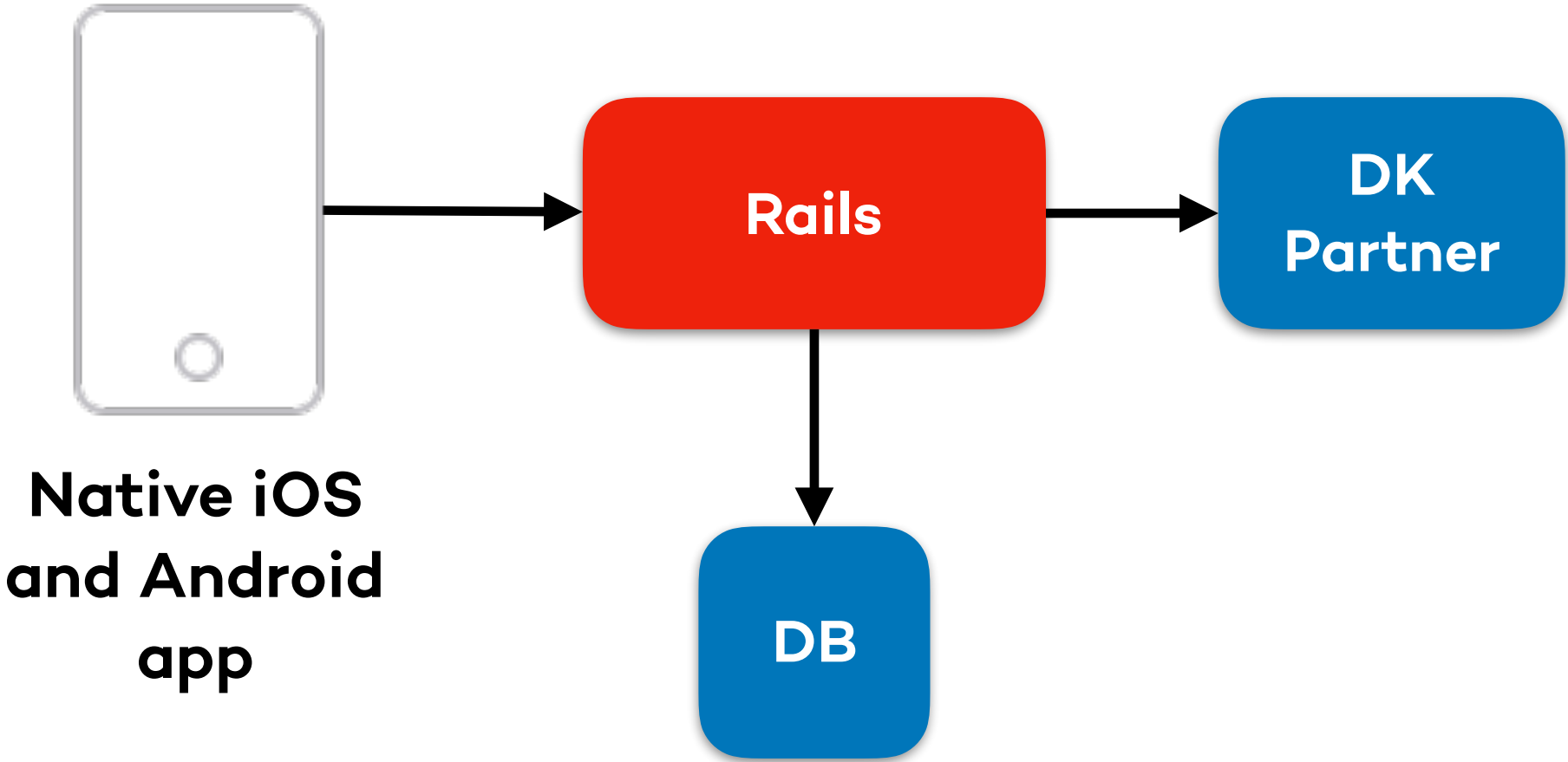
External APIs and web hooks

3rd parties require access to your data

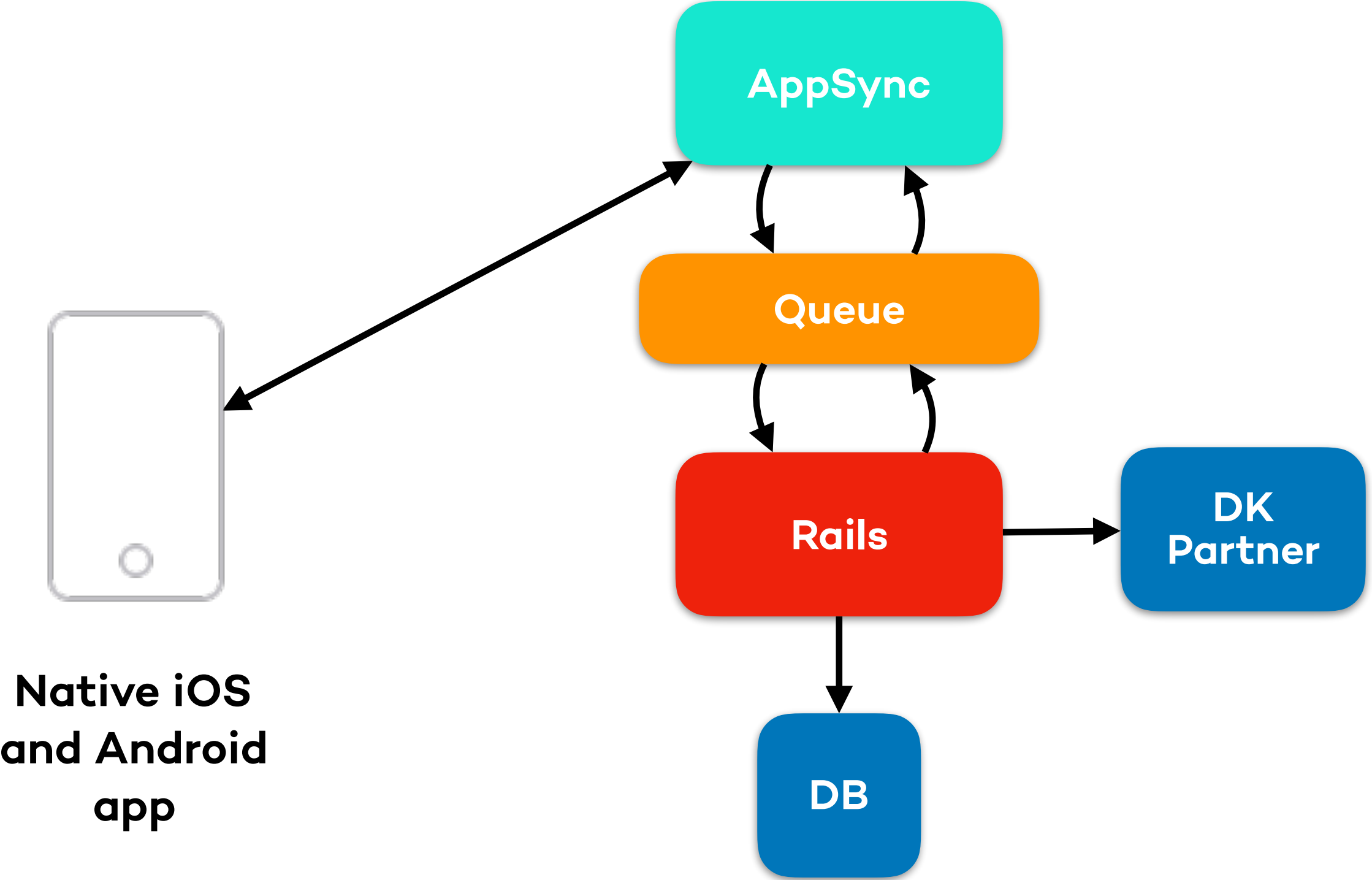
1. Sources publish events
2. External API subscribes on events and converts to own model
3. 3rd parties access external API and may register web hooks



From pull to push



App triggers a pull (3-5 requests ~ 1-2 seconds) from partner bank with a DB transaction open



Never let an app request trigger a sync! Control the sync process and use async events to push data to the app on a web socket

Wrapping up

Key takeaways if entering microservice land

Adapt to the size of your team

Use asynchronous communication
between services... preferably
event driven

Prioritise your deployment pipeline
and runtime platform from the
start

Be systematic!

lunar
way®