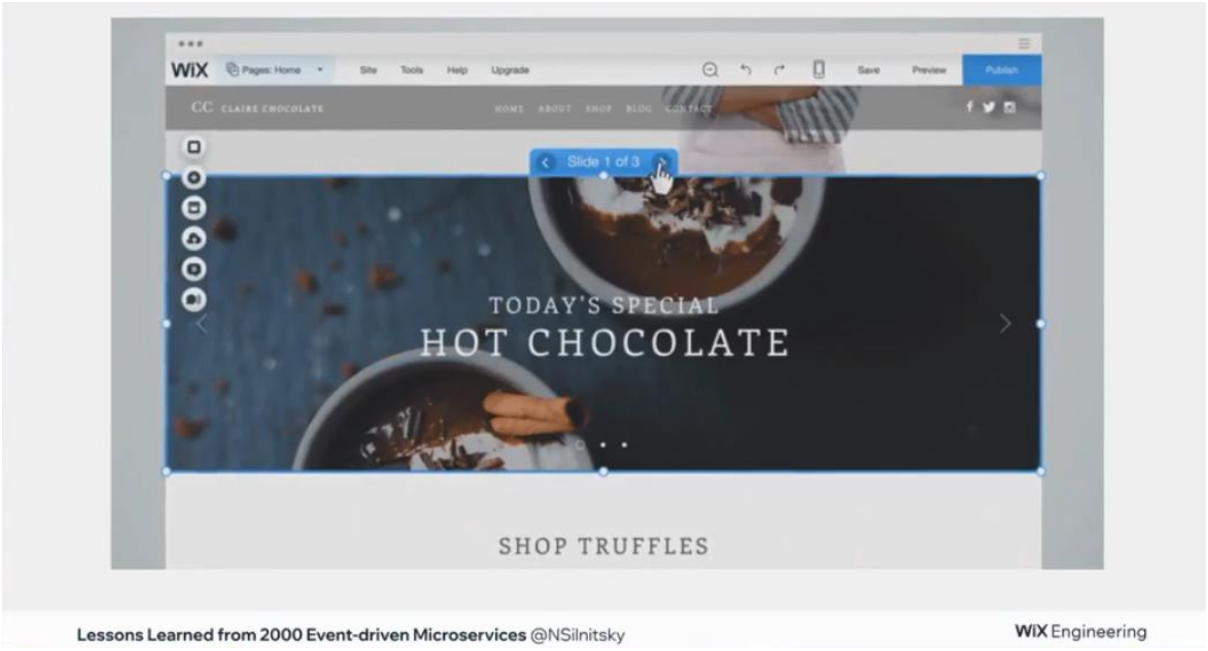
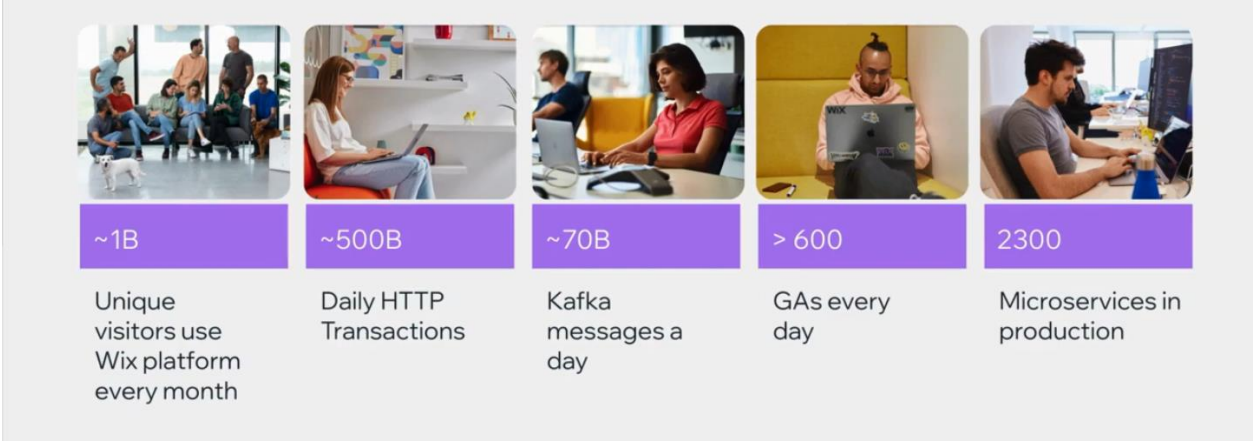
During this session developers and architects will get many concrete takeaways about how to improve their event-driven design with proven battle-tested methods or convince them to switch from request-reply architecture. Wix has a huge scale of event driven traffic. More than 70 billion Kafka business events per day. Over the past few years Wix has made a gradual transition to an event-driven architecture for its 2000 microservices. We have made mistakes along the way but have improved and learned a lot about how to make sure our production is still maintainable, performant and resilient. In this talk you will hear about the lessons we learned including: 1. The importance of atomic operations for databases and events 2. Avoiding full-blown event sourcing, instead taking CRUD to the next level 3. Having essential events debugging and quick-fix tools in production and a few more.





Challenges

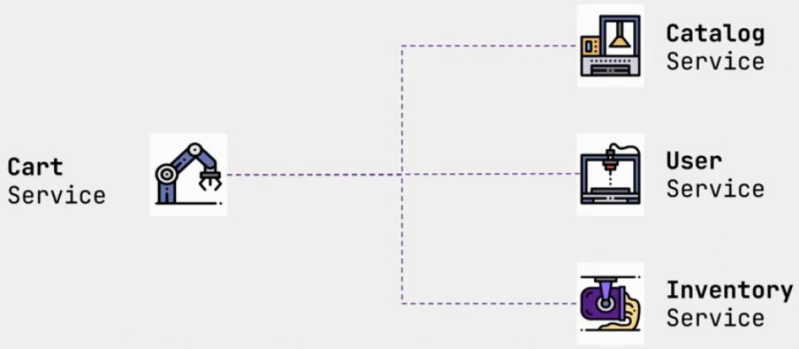
of event-driven architecture,
that we've bumped into

1. Producing message failures
2. Processing out-of-order & duplicates
3. Sending large payloads
4. Troubleshooting production

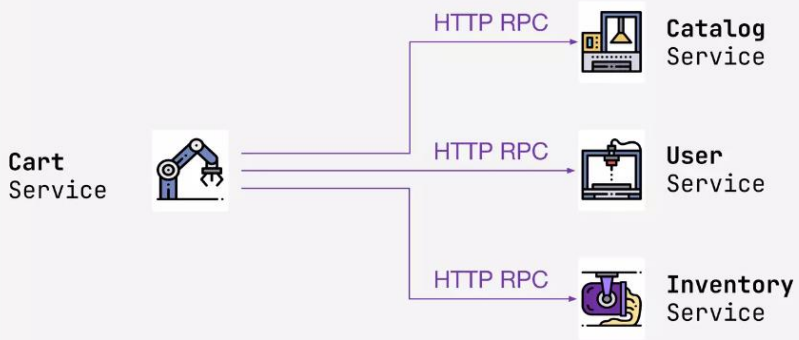
How Event-driven Architecture Works

BEGADOL

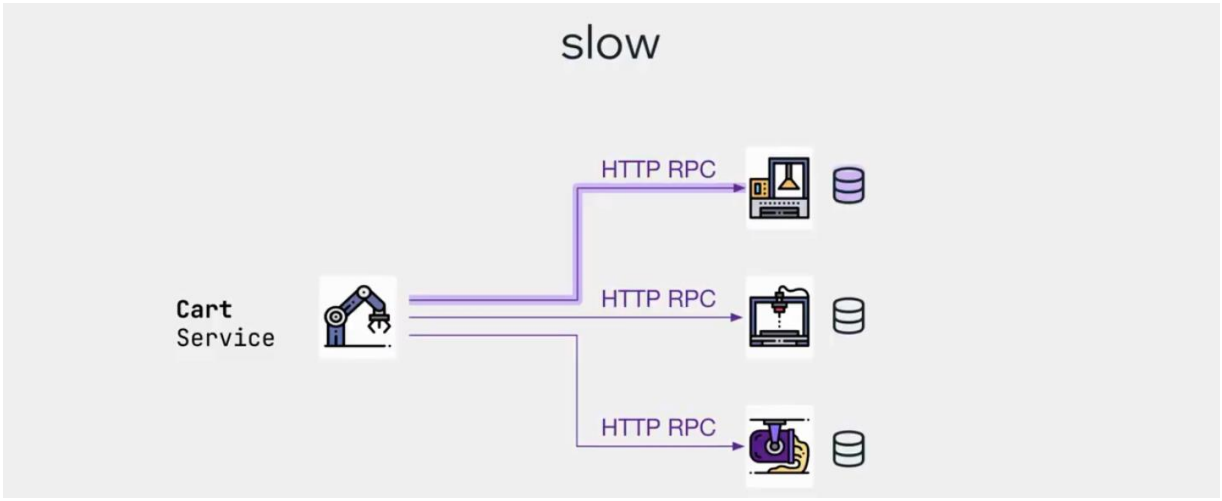
Service-to-Service Communication

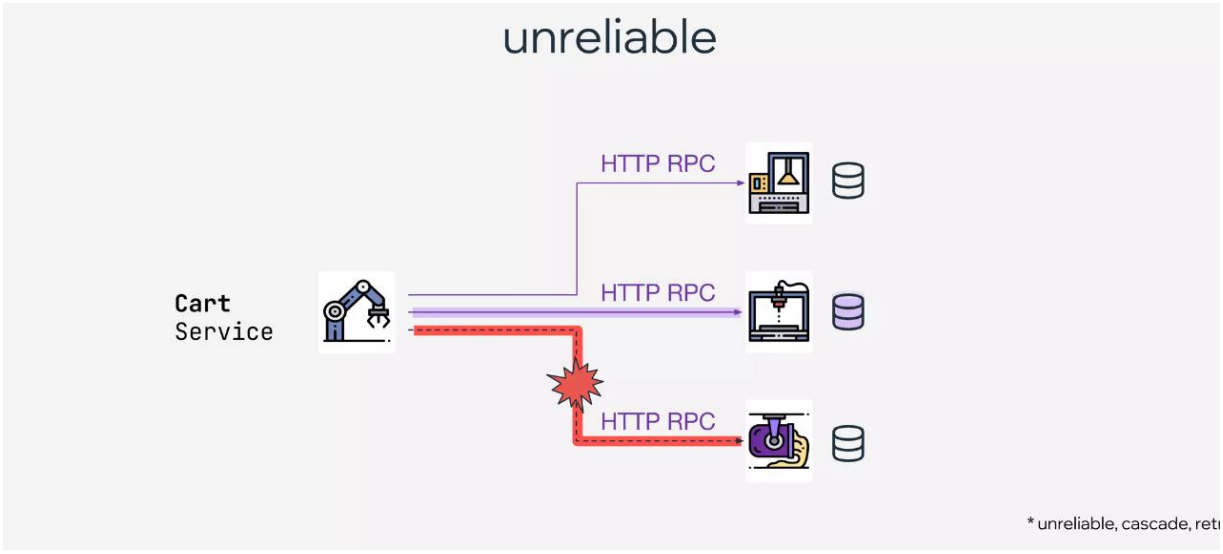


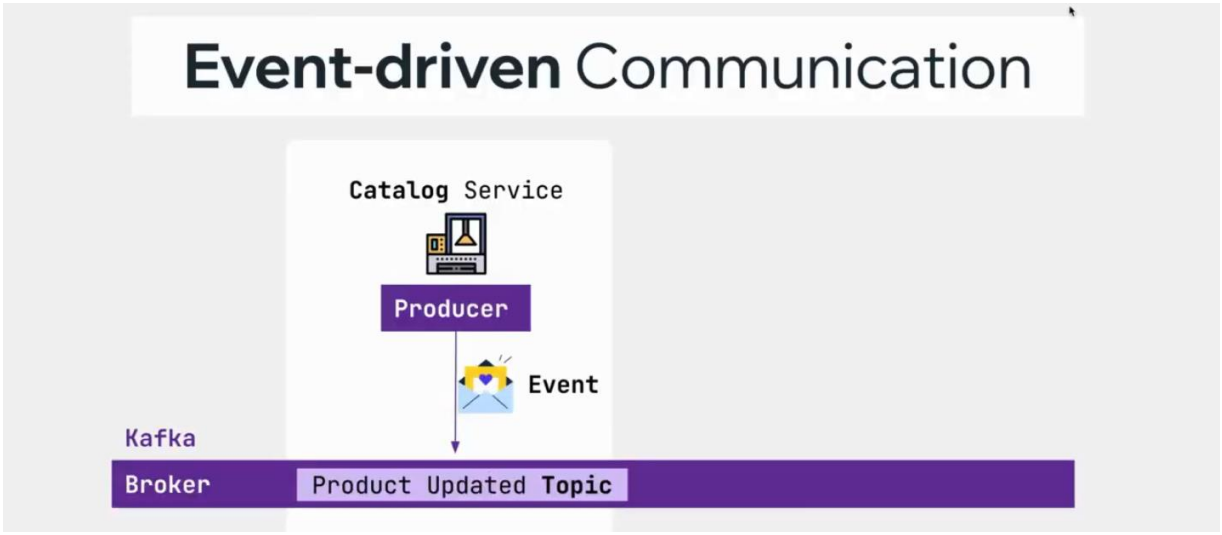
Request-Reply Communication



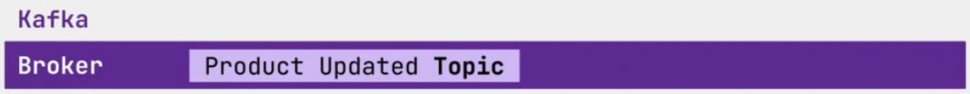
* issue scale


slow
HTTP RPC
HTTP RPC
HTTP RPC
Cart
Service


unreliable
HTTP RPC
HTTP RPC
HTTP RPC
Cart
Service
* unreliable, cascade, ret

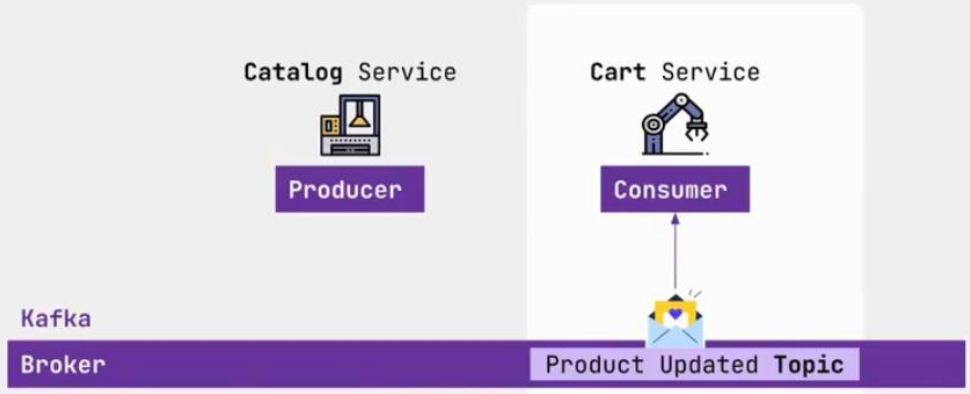

Event-driven Communication
Catalog Service
Producer
Event
Kafka
Broker
Product Updated Topic

more robust



* DB, decoupling, no impact

Event processing is guaranteed



The following is based on a true story

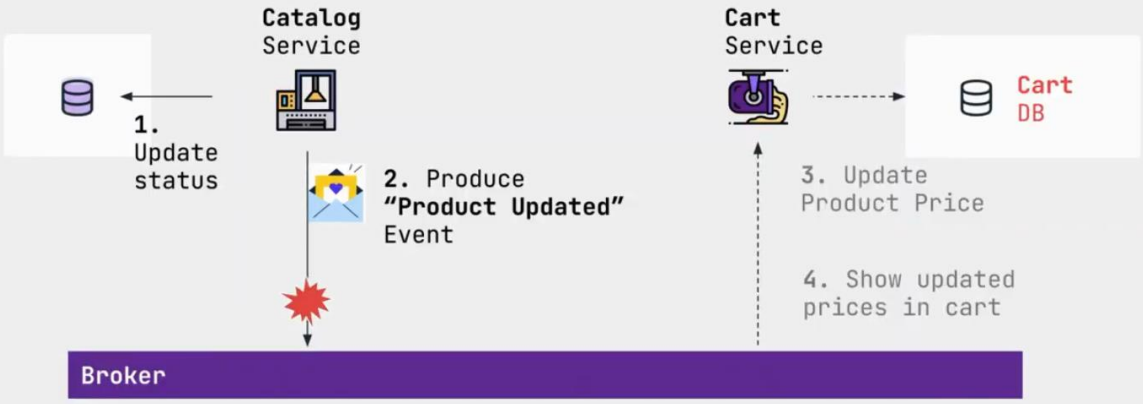
*Dates and products were changed for clarity :)

We can work event-driven!!

2016
Wix starts using event-driven

It all began when Ecom experienced data issues

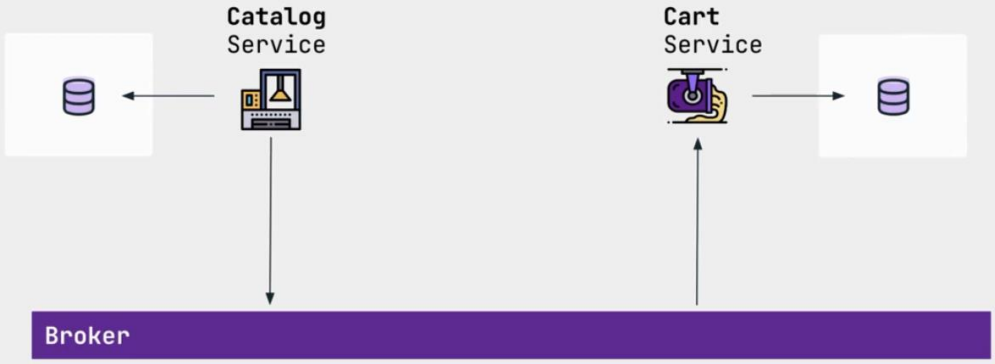
After investigating



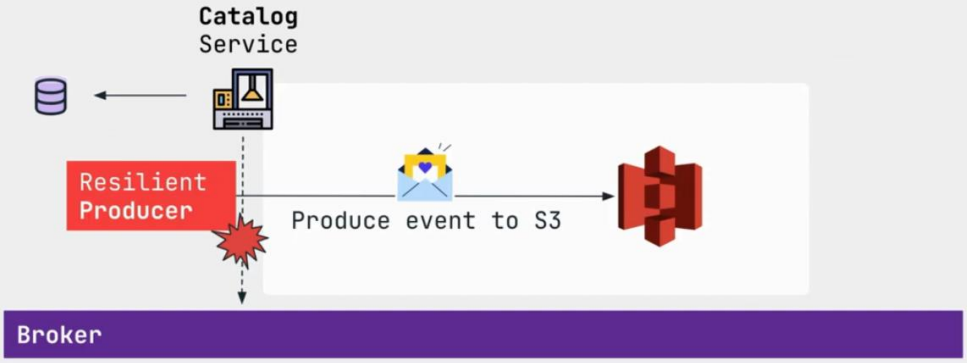
Challenge #1

Producing message failure

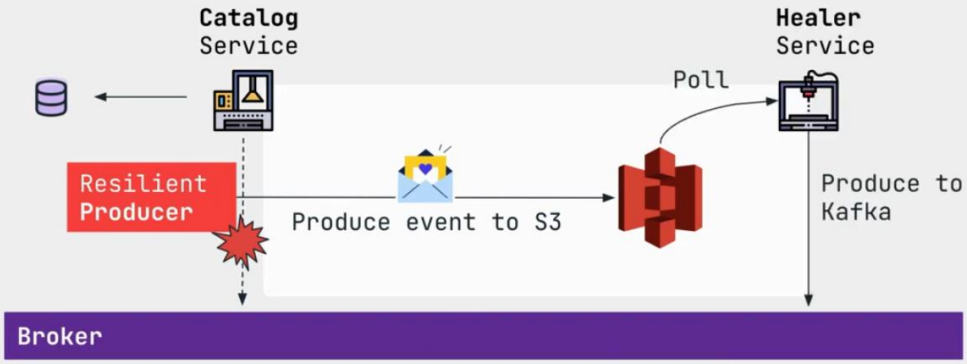
Make DB Update & Event Producing Atomic



Catch Unsent Events

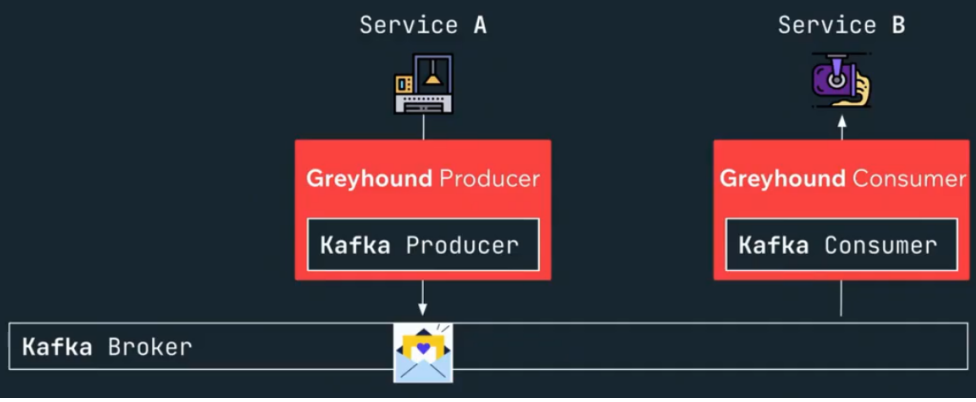


Fallback to S3 and Heal



Wrap Kafka with Greyhound*

* Open source: https://github.com/wix/greyhound



Wrap Kafka with Greyhound*

* Open source: https://github.com/wix/greyhound

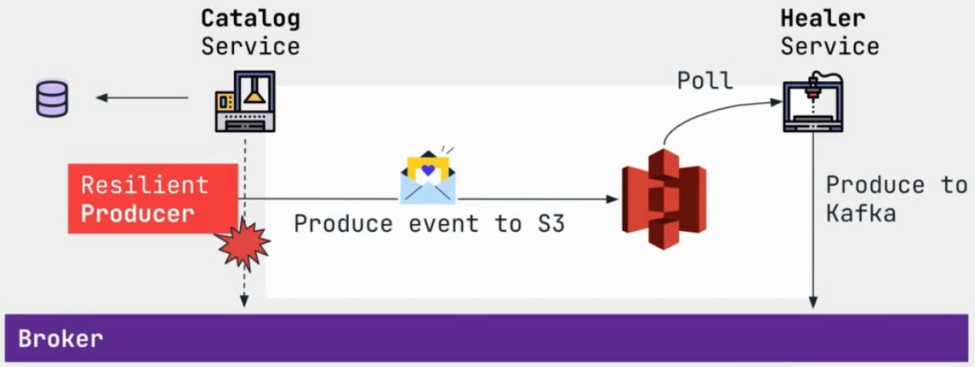
Developer Self-Service:

- Resilient Producer
- Parallel Consumption
- Batch Consumer
- Consumer Retry Strategies
- Context Propagation
- Metrics reporting

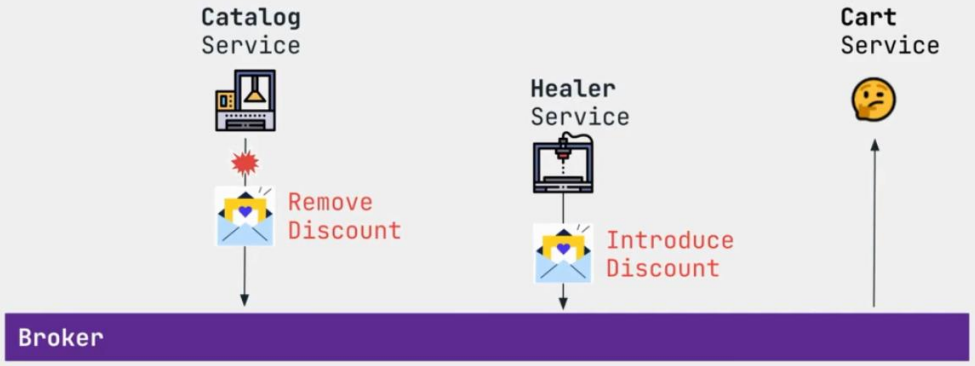
2016
Wix starts using event-driven

2018
Greyhound Resilient producer & Consumer retries

Fallback to S3 and Heal



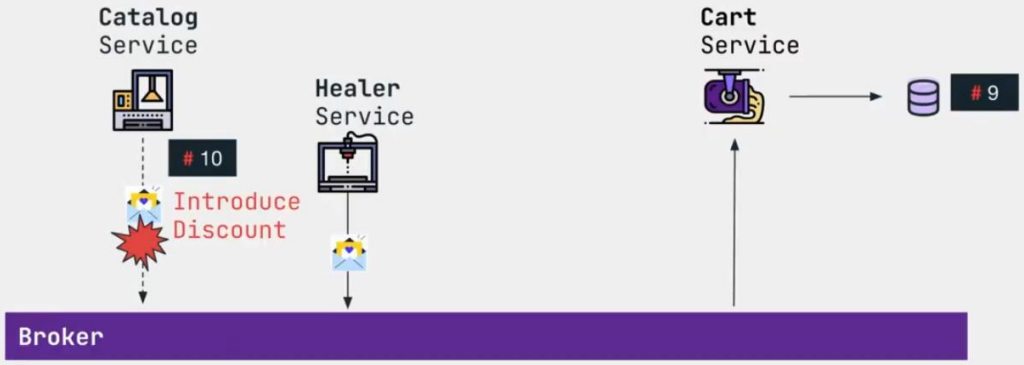
Then 'out-of-order' happened



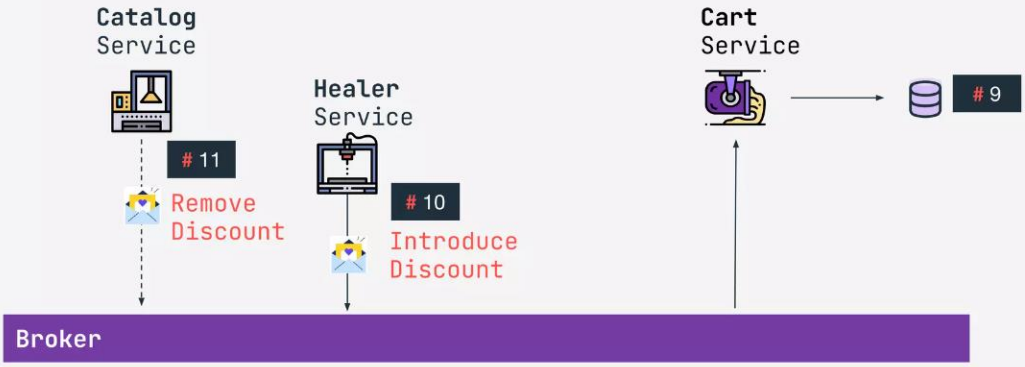
Challenge #2

Out-of-order & duplicates processing

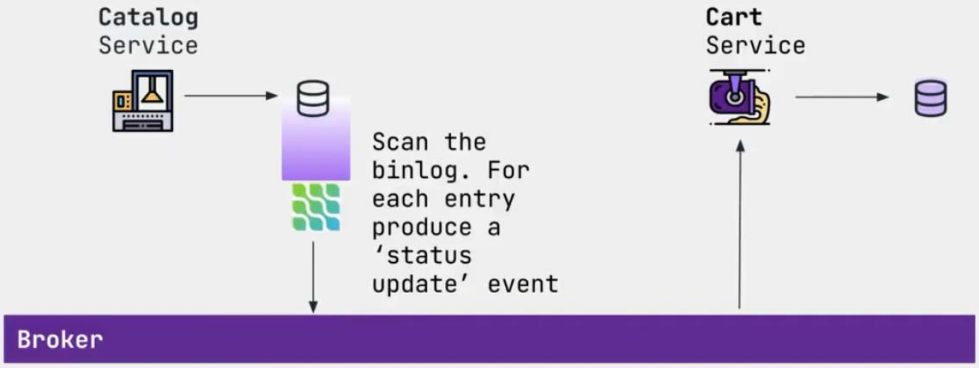
Mitigating out-of-order with revision ID



Mitigating out-of-order with revision ID

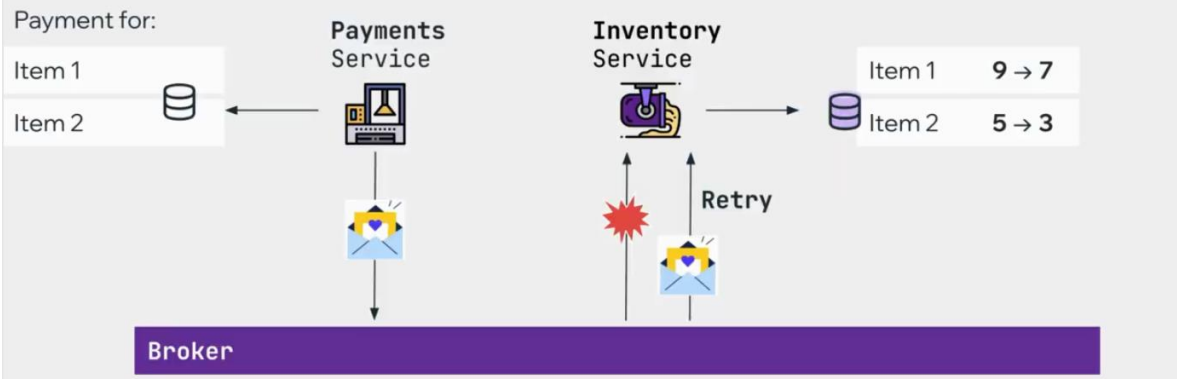


Mitigating out-of-order with Debezium connector



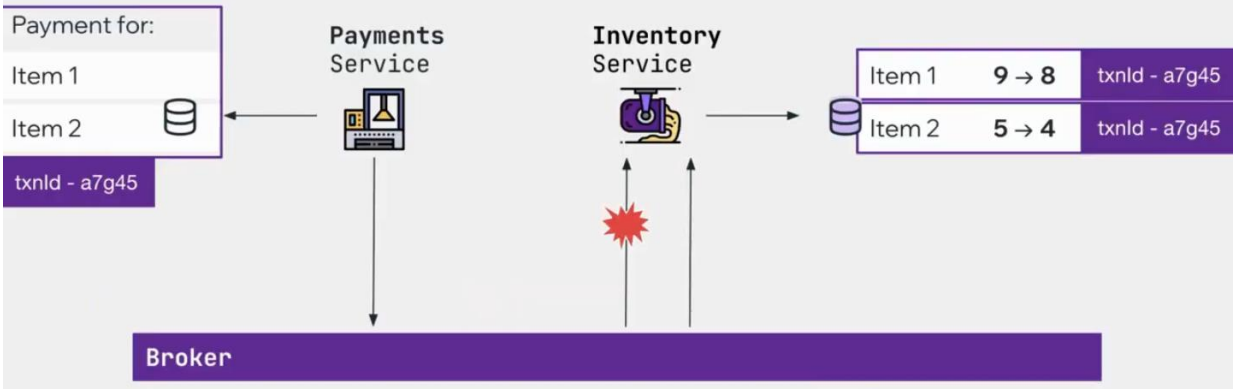
More Ecom data issues

Investigation leads to duplicate processing



* not idempotent

Mitigating duplicates with Transaction ID



2016
Wix starts using event-driven

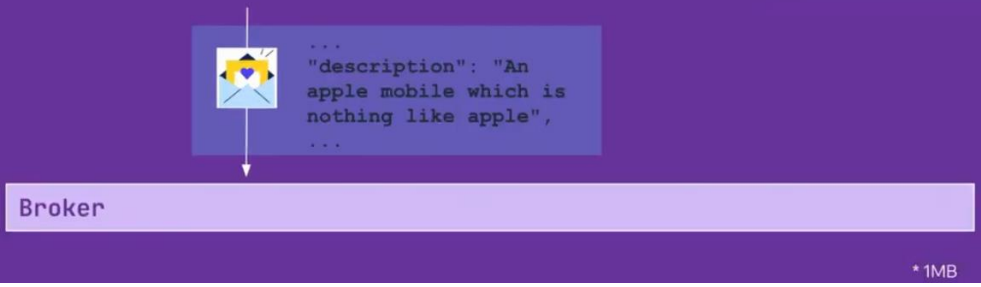
2018
Greyhound Resilient producer & Consumer retries

2019
Revisions & Transaction IDs

"Dude, I can't produce large payloads"

Challenge #3

Failure to send large payloads



* 1MB

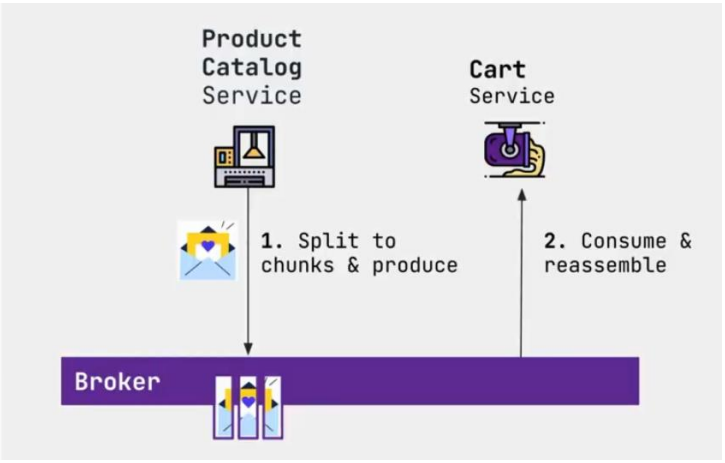
Large Payloads Remedy I

Compression

- → Try several compression types (lz4, snappy, etc.)
- → Compression on Kafka level is usually better than application level, as payloads can be compressed in batches

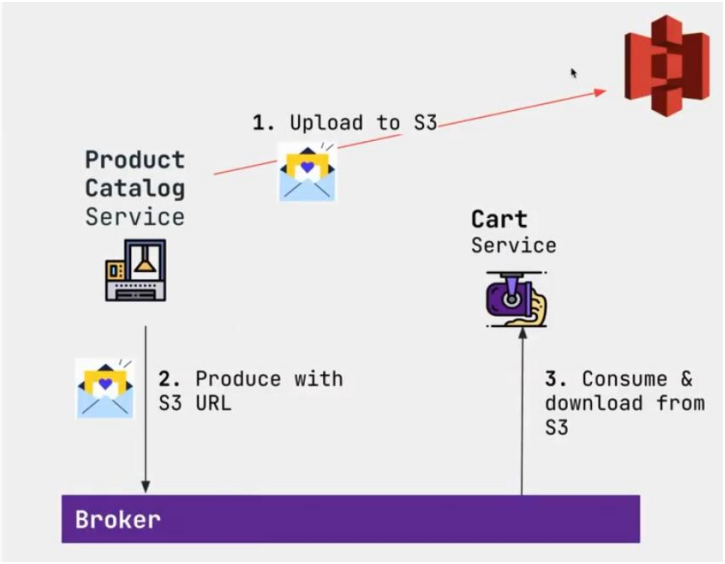
Large Payloads Remedy II

Chunking



Large Payloads Remedy III

Reference to Object Store



2016 *Wix starts using event-driven*

2018 *Greyhound Resilient producer & Consumer retries*

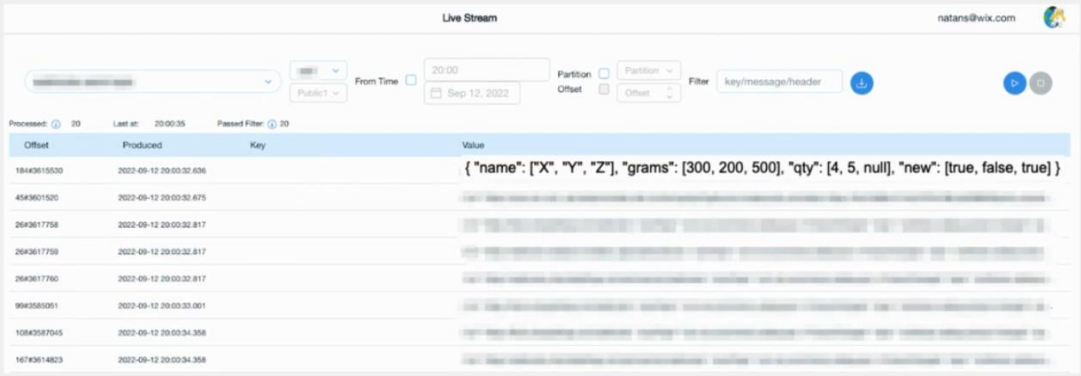
2019 *We use IDs for ooo & duplicates*

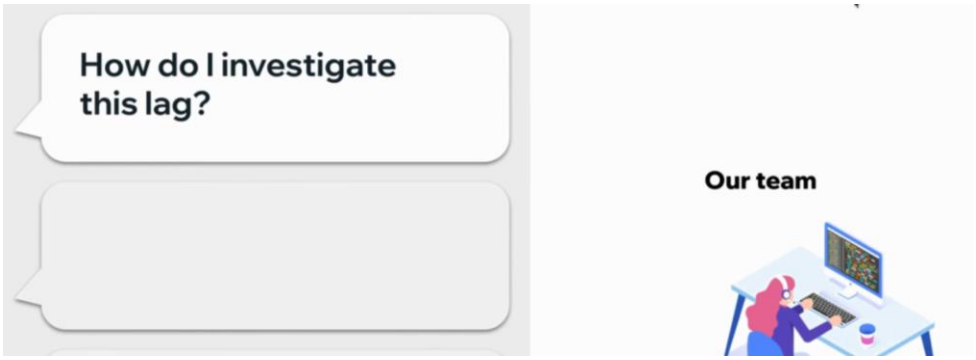
2020 *Added compression by default*

Challenge #4

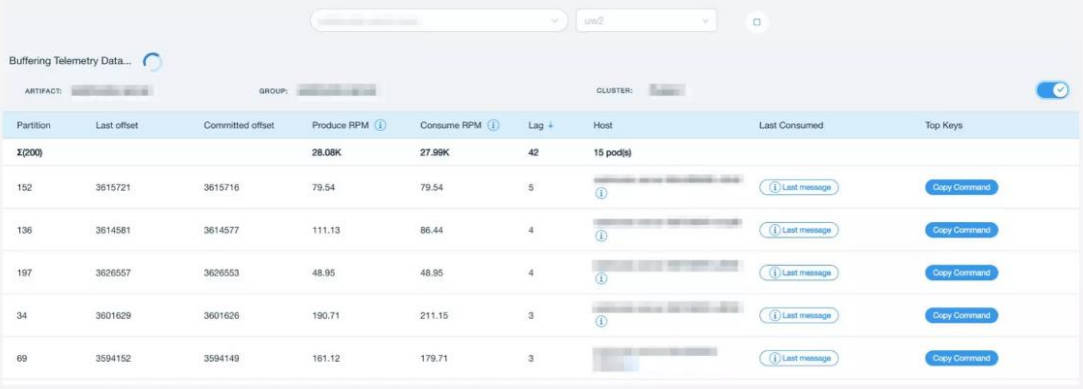
It's hard for developers to debug and maintain event-driven microservices at scale in production

Stream events with various filters

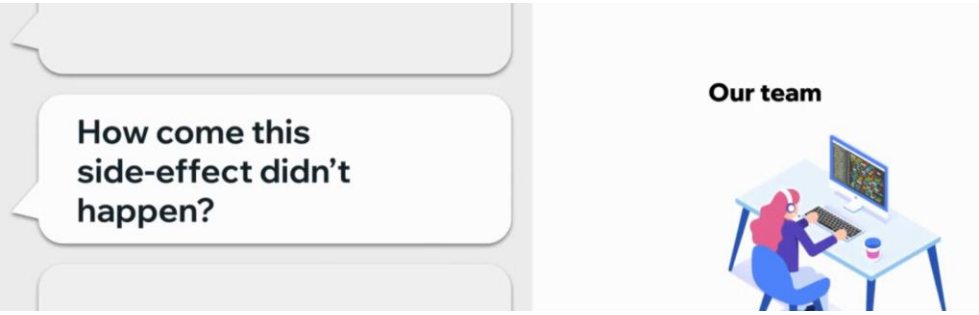




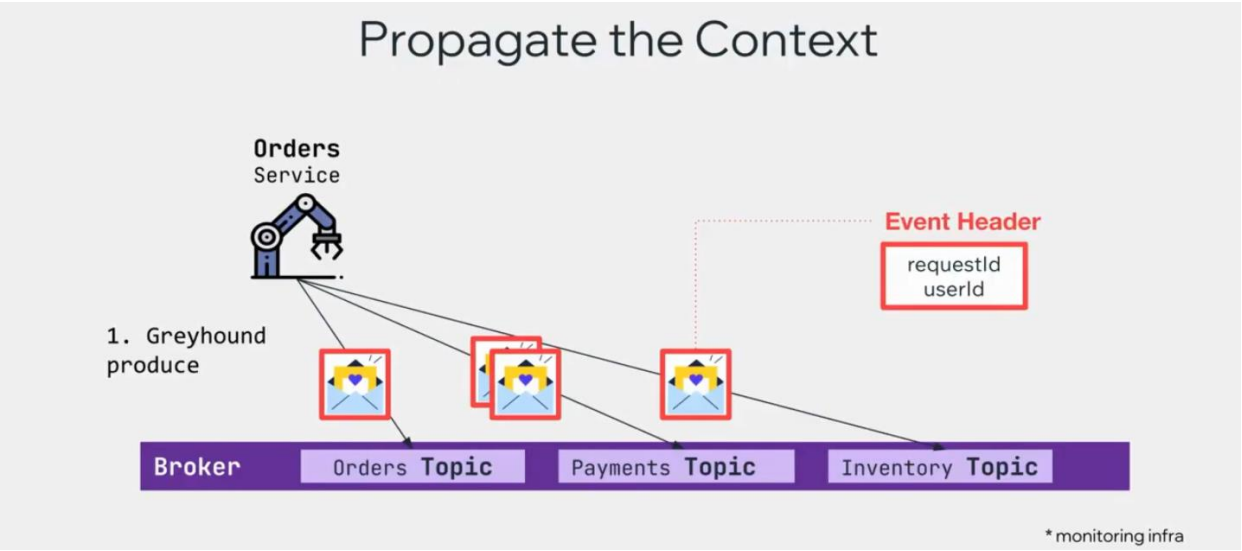
Investigate consumer lag per partition



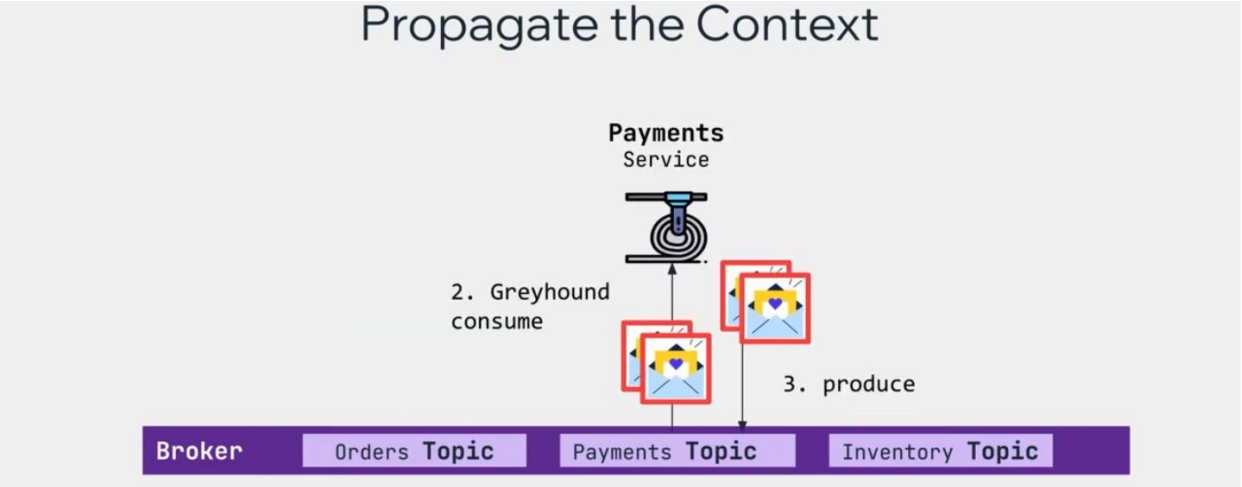
View a “stuck” event in some partition



Propagate the Context



Propagate the Context



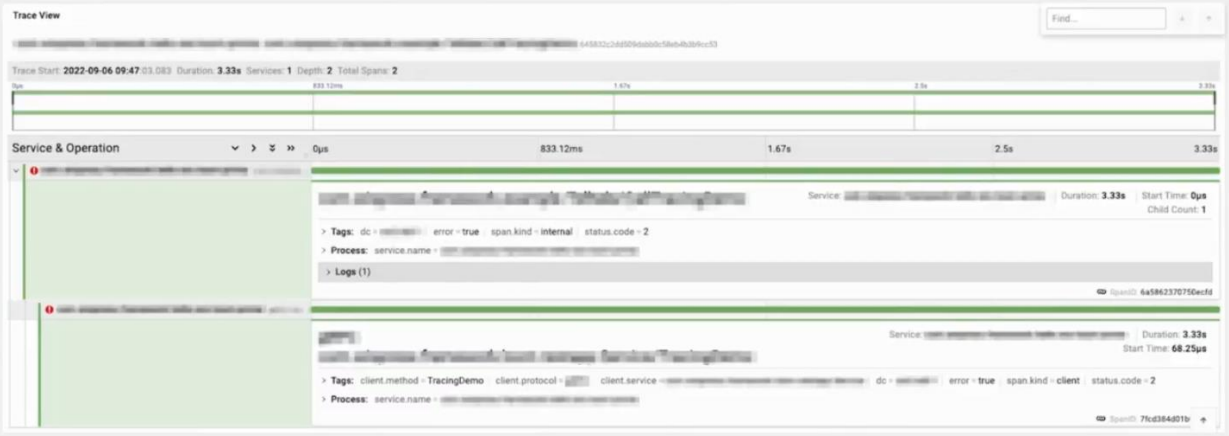
Propagate the Context

So developers can track events' route

Trace in Access logs of 1662997682.19964775441831922504 (LIMIT 5000)

timestamp	nginx_artifact_name	dc	status	time(ms)	flags	grpc_r	pipeline	hostname
2022-09-12 15:48:02.224		42	200	218	-		egress-di...	
2022-09-12 15:48:02.225		42	200	216	-		ingress	
2022-09-12 15:48:02.318		42	200	121	-		egress	
2022-09-12 15:48:02.318		42	200	120	-		ingress	
2022-09-12 15:48:02.288		42	200	25	-		egress	
2022-09-12 15:48:02.290		42	200	23	-		ingress	
2022-09-12 15:48:02.245		42	200	19	-		egress	
2022-09-12 15:48:02.247		42	200	15	-		egress	
2022-09-12 15:48:02.248		42	200	14	-		ingress	
2022-09-12 15:48:02.352		42	200	13	-		egress	

View event details



2016
Wix starts using event-driven

2018
We open source Greyhound

2019
We use IDs for ooo & duplicates

2020
Added compression by default

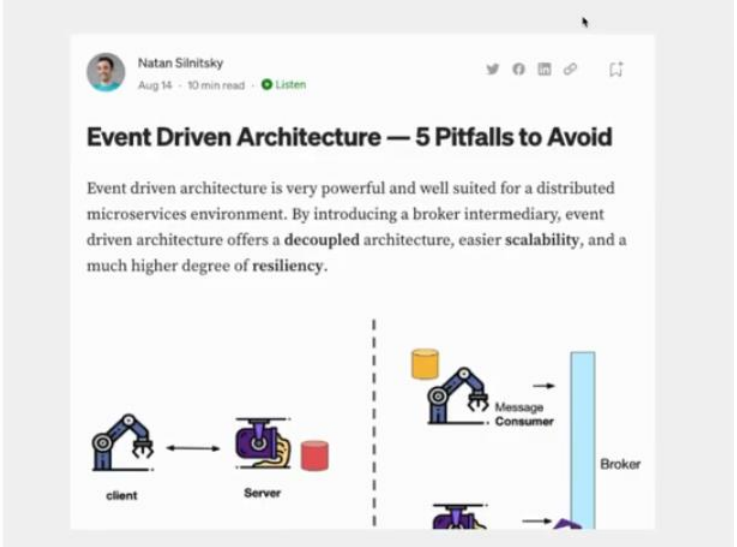
2021-22
Tools in Production

Wix developers have embraced event-driven architecture.

Meeting these challenges made our microservices more decoupled, resilient and scalable, while keeping complexity low and data consistent.

The Blog Post

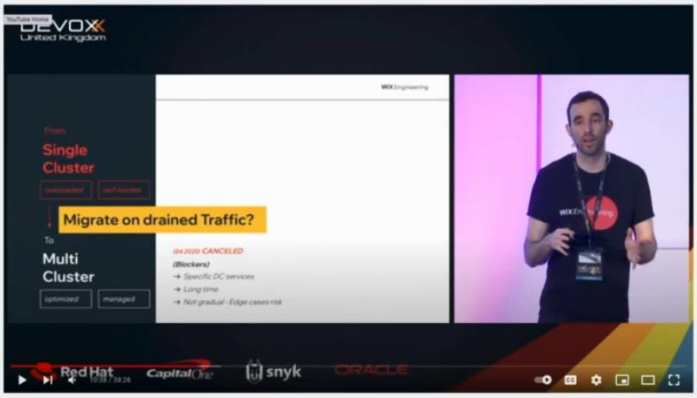
https://medium.com/wix-engineering/event-driven-architecture-5-pitfalls-to-avoid-b3ebf885bdb1



The Next Step

https://www.youtube.com/watch?v=XKbG8a-9NRE

How to migrate 2000 microservices to Multi Cluster Managed Kafka with 0 Downtime



Greyhound

github.com/wix/**greyhound**