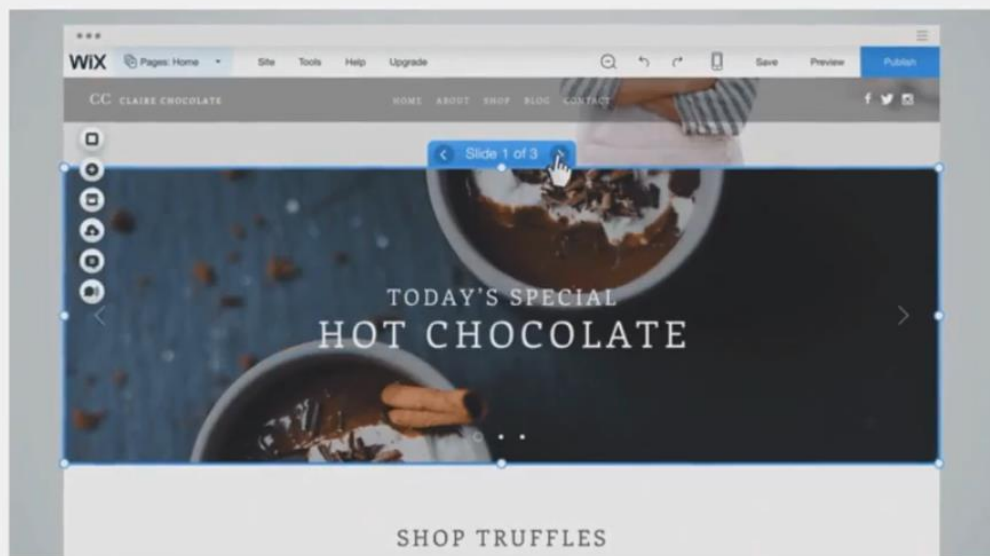


LESSONS LEARNED FROM WORKING WITH 2000 EVENT-DRIVEN MICROSERVICES

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.



Lessons Learned from 2000 Event-driven Microservices @NSilnitsky

WIX Engineering



~1B

Unique visitors use Wix platform every month



~500B

Daily HTTP Transactions



~70B

Kafka messages a day



> 600

GAs every day



2300

Microservices in production

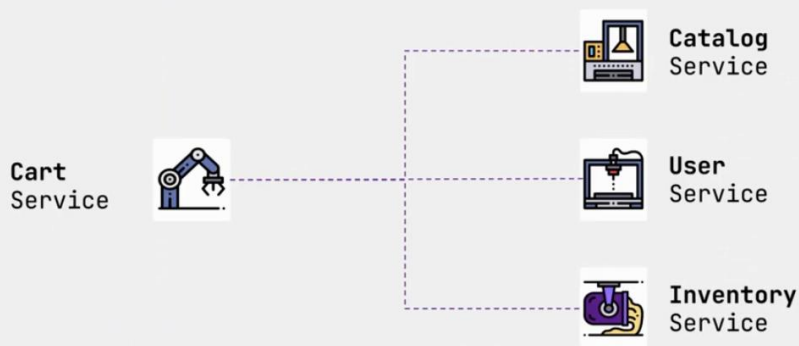
Challenges

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

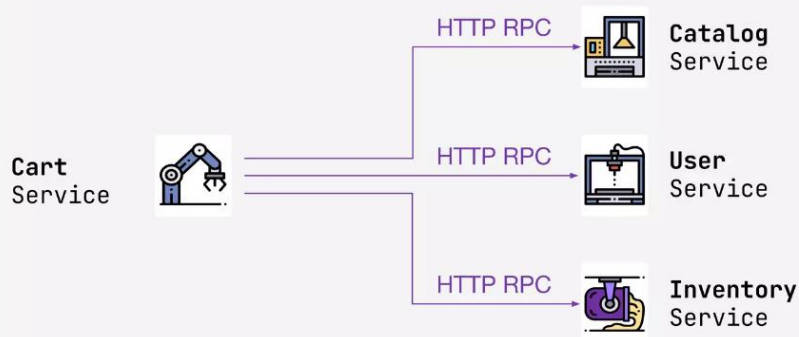
- 1 Producing message failures
- 2 Processing out-of-order & duplicates
- 3 Sending large payloads
- 4 Troubleshooting production

How Event-driven Architecture Works BEGADOL

Service-to-Service Communication

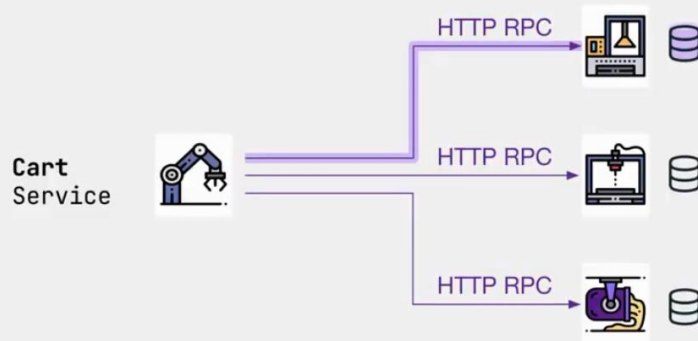


Request-Reply Communication

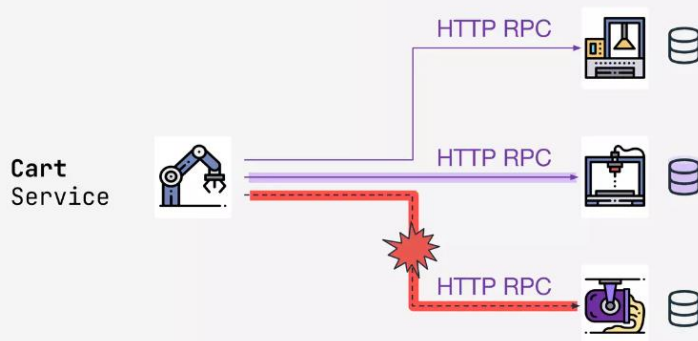


* issue scale

slow

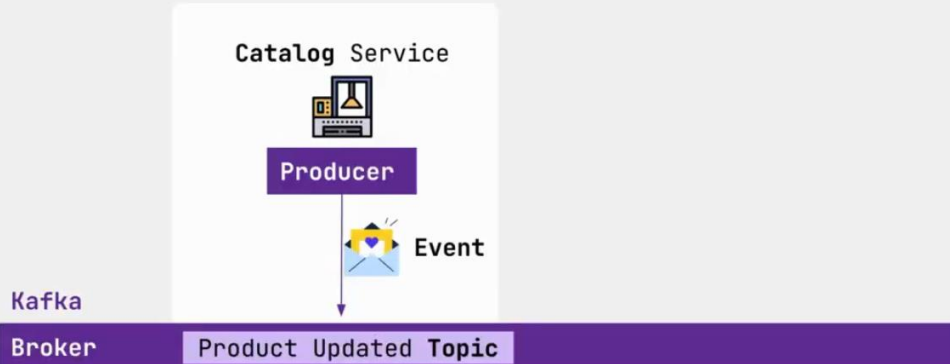


unreliable

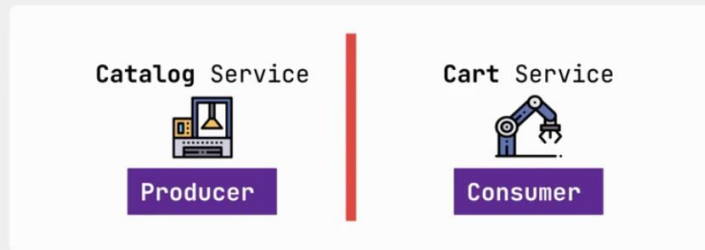


*unreliable, cascade, reti

Event-driven Communication



more robust



Kafka

Broker

Product Updated Topic

* DB, decoupling, no impact

Event processing is guaranteed

Catalog Service



Producer

Cart Service



Consumer

Kafka

Broker

Product Updated Topic

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



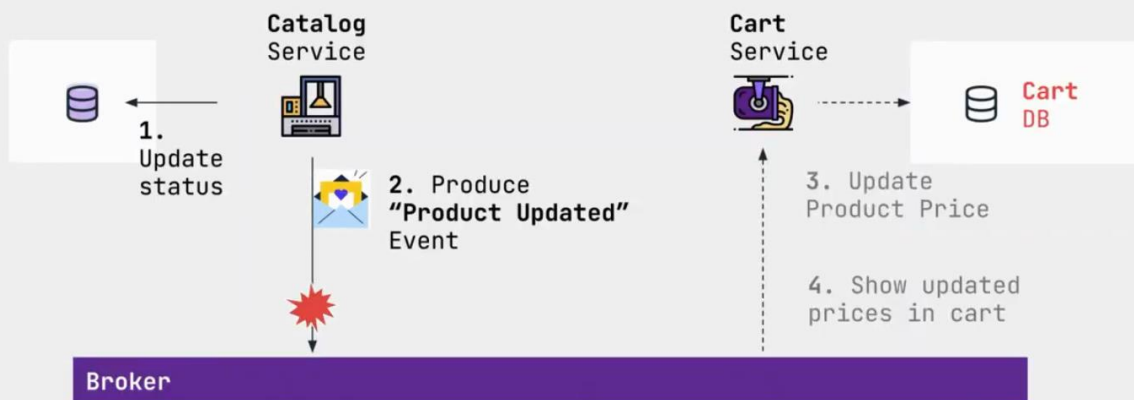
It all began when
Ecom experienced
data issues

Cart
DB

Data does NOT reflect
actual catalog

Risk: show wrong
prices in cart

After investigating



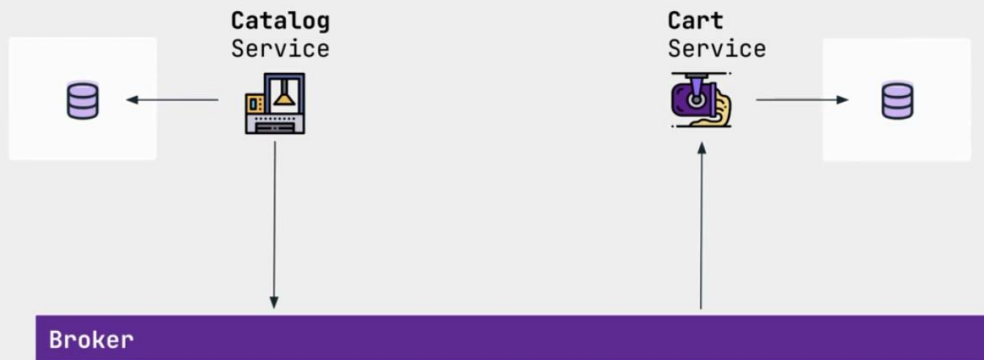
Challenge #1

Producing message failure

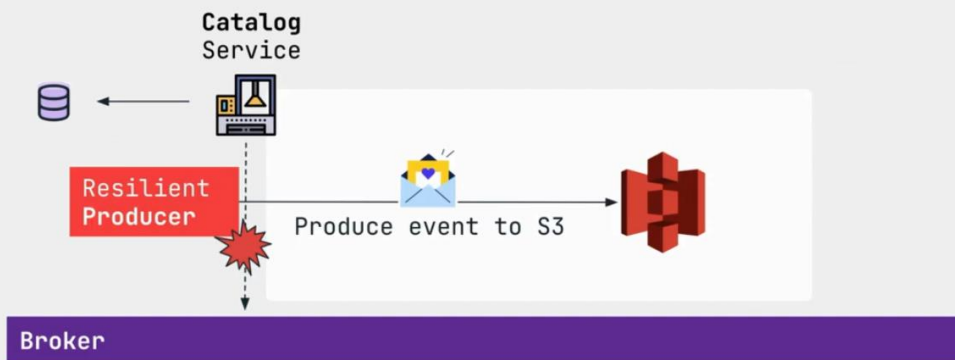


Kafka

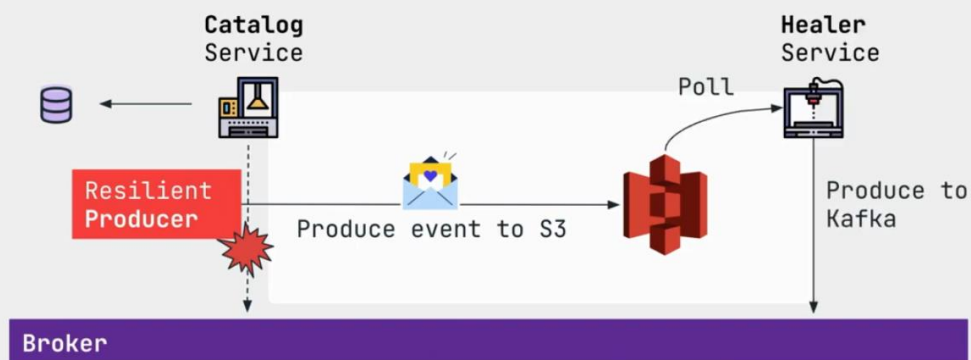
Make DB Update & Event Producing Atomic



Catch Unsent Events

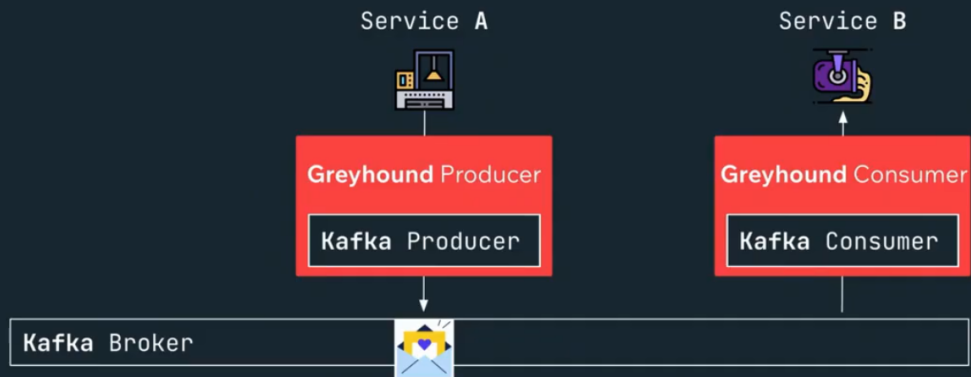


Fallback to S3 and Heal



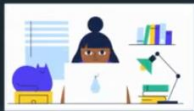
Wrap Kafka with Greyhound*

* 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

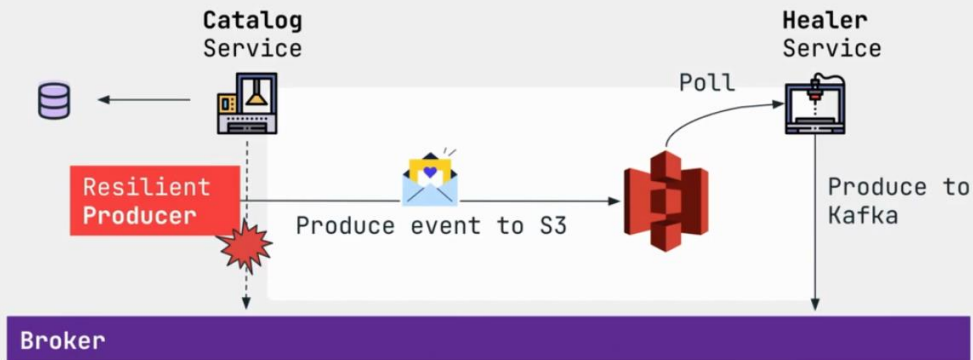
*Wix starts using
event-driven*

2018

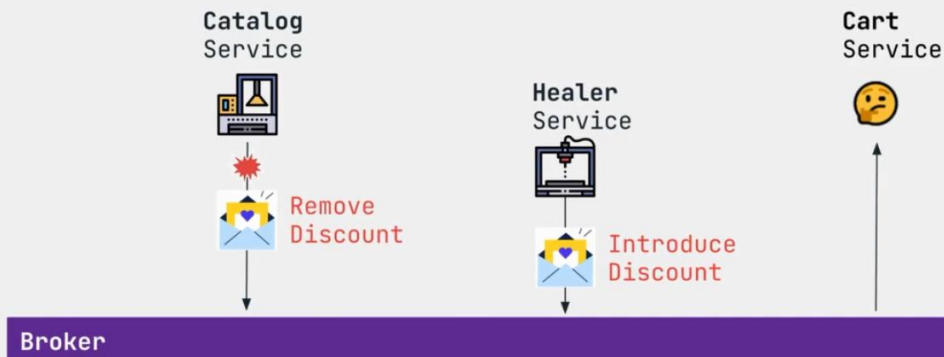
*Greyhound
Resilient producer
& Consumer
retries*



Fallback to S3 and Heal



Then 'out-of-order' happened

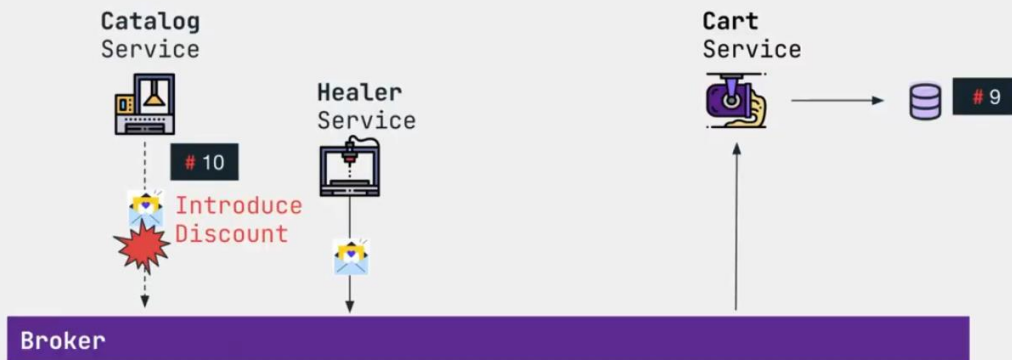


Challenge #2

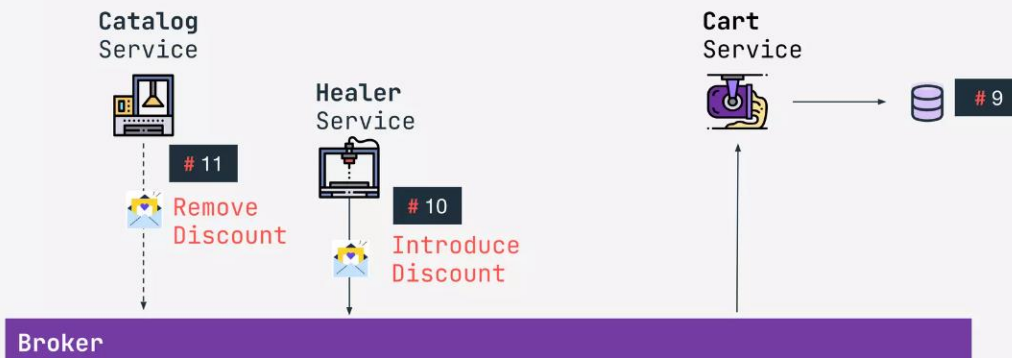
Out-of-order & duplicates processing



Mitigating out-of-order with revision ID



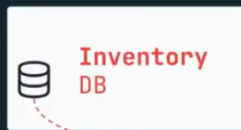
Mitigating out-of-order with revision ID



Mitigating out-of-order with Debezium connector



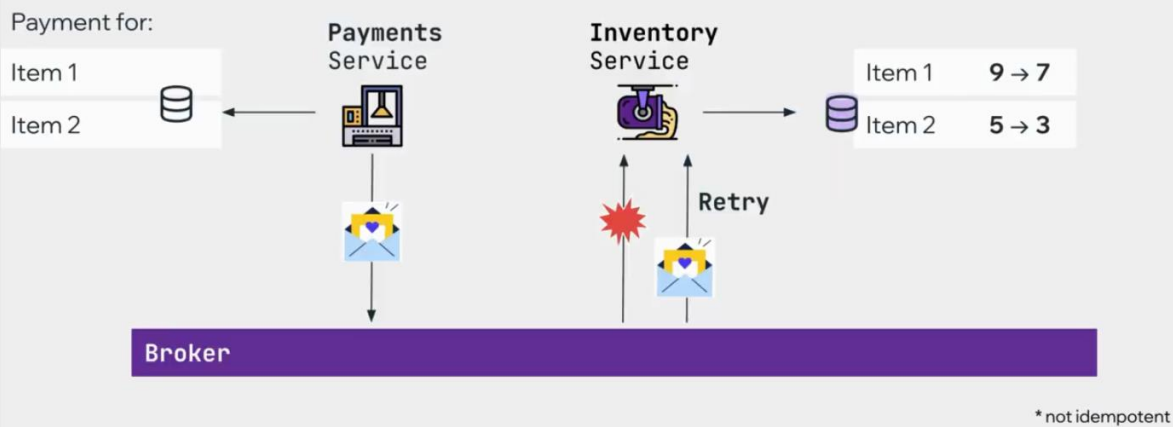
More Ecom data issues



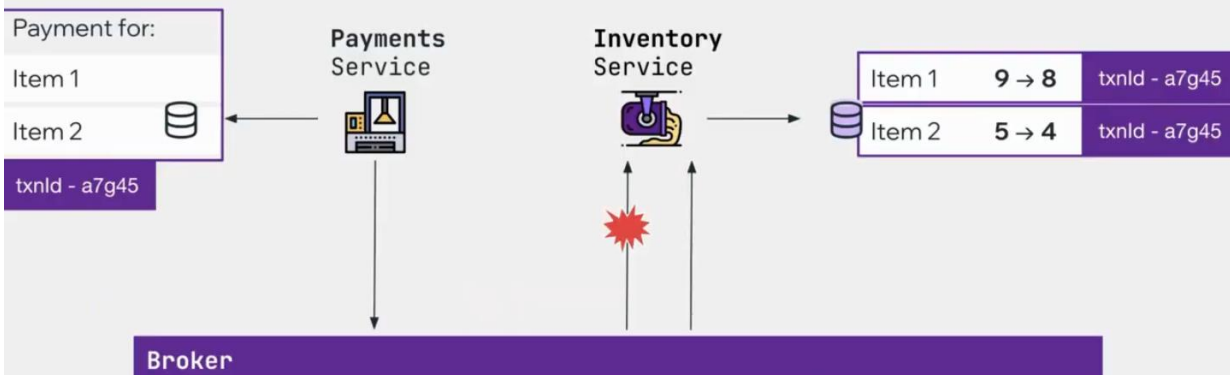
Data does NOT reflect actual inventory

Risk: lose potential customers

Investigation leads to duplicate processing



Mitigating duplicates with Transaction ID



- 2016
Wix starts using
event-driven
- 2018
Greyhound
Resilient
producer &
Consumer retries
- 2019
Revisions &
Transaction IDs

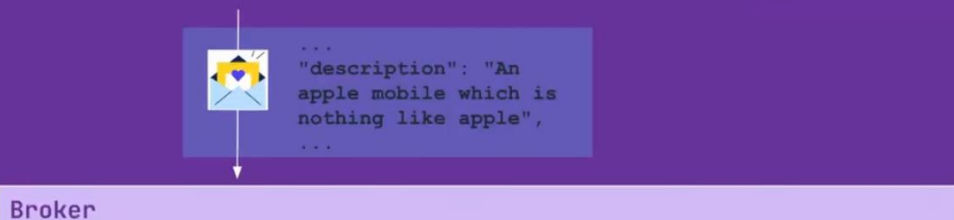


“Dude, I can’t produce large payloads”



Challenge #3

Failure to send large payloads



* 1MB

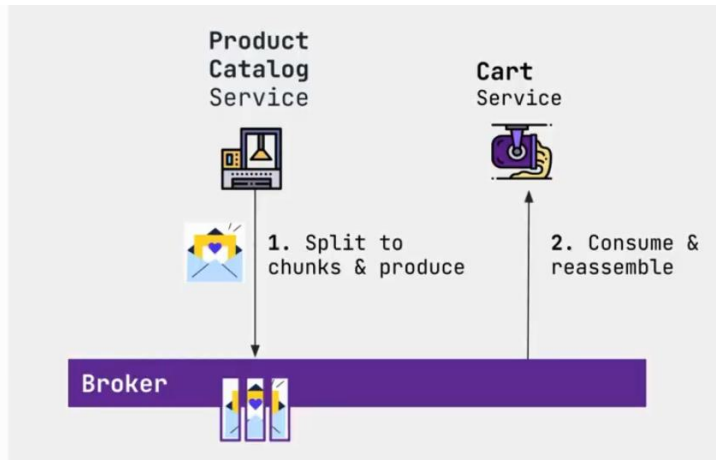
Large Payloads Remedy I

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

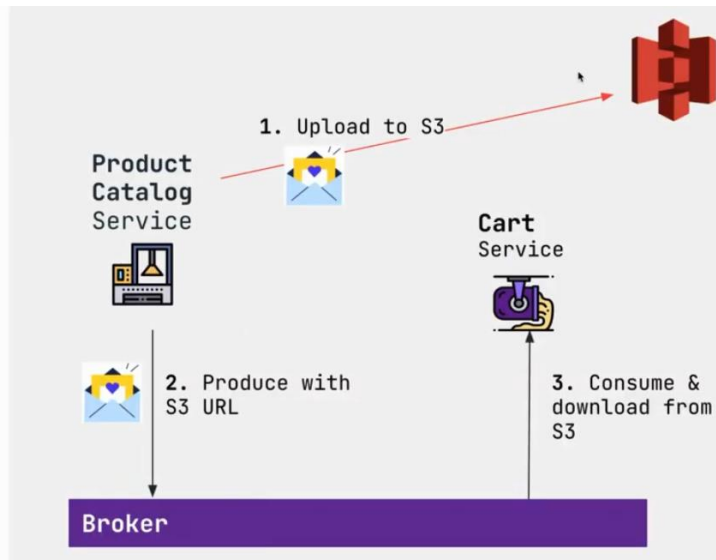
Large Payloads Remedy II

Chunking



Large Payloads Remedy III

Reference to
Object Store



2016

Wix starts using
event-driven

2019

We use IDs for
ooo & duplicates

2018

Greyhound Resilient
producer &
Consumer retries

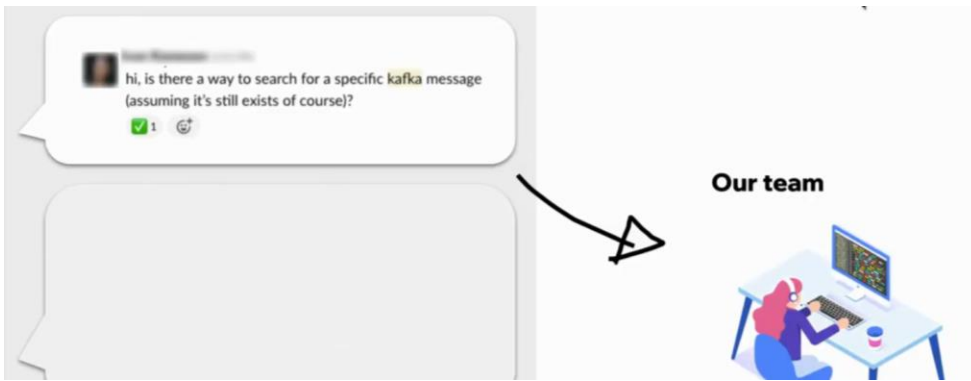
2020

Added
compression
by default



Challenge #4

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



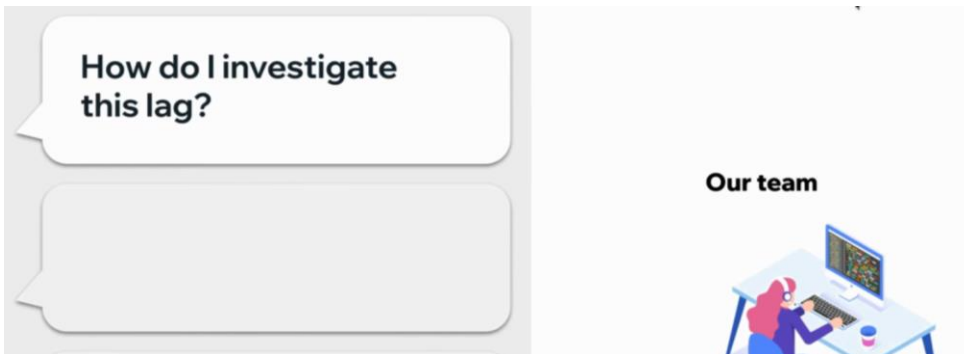
Stream events with various filters

Live Stream natans@wix.com

From Time 20:00 Sep 12, 2022 Partition Offset Filter key=message/header

Processed: 20 Last at: 20:00:35 Passed Filter: 20

Offset	Produced	Key	Value
1843615530	2022-09-12 20:00:32.636		{ "name": ["X", "Y", "Z"], "grams": [300, 200, 500], "qty": [4, 5, null], "new": [true, false, true] }
4593601520	2022-09-12 20:00:32.675		
2643617758	2022-09-12 20:00:32.817		
2643617759	2022-09-12 20:00:32.817		
2643617760	2022-09-12 20:00:32.817		
9943585051	2022-09-12 20:00:33.001		
10843587045	2022-09-12 20:00:34.358		
16743614823	2022-09-12 20:00:34.358		



Investigate consumer lag per partition

Buffering Telemetry Data...

ARTIFACT: GROUP: CLUSTER:

Partition	Last offset	Committed offset	Produce RPM	Consume RPM	Lag +	Host	Last Consumed	Top Keys
2(200)			28.08K	27.98K	42	15 pods(j)		
152	3615721	3615716	79.54	79.54	5		Last message	Copy Command
136	3614581	3614577	111.13	86.44	4		Last message	Copy Command
197	3626557	3626553	48.95	48.95	4		Last message	Copy Command
34	3601629	3601626	190.71	211.15	3		Last message	Copy Command
69	3594152	3594149	161.12	179.71	3		Last message	Copy Command

View a “stuck” event in some partition

Message Details

This message was the first one that hasn't already been committed by consumer group (this could have changed by now). Use this to identify 'stuck' messages (when the consumer is not progressing at all on 1 or more partitions).

Partition: 136 | Offset: 3614577 | Produced: 2022-09-12 19:51:19.926

```
{
  "root": {
    {
      "name": ["X", "Y", "Z"],
      "grams": [300, 200, 500],
      "qty": [4, 5, null],
      "new": [true, false, true]
    }
  }
}
```

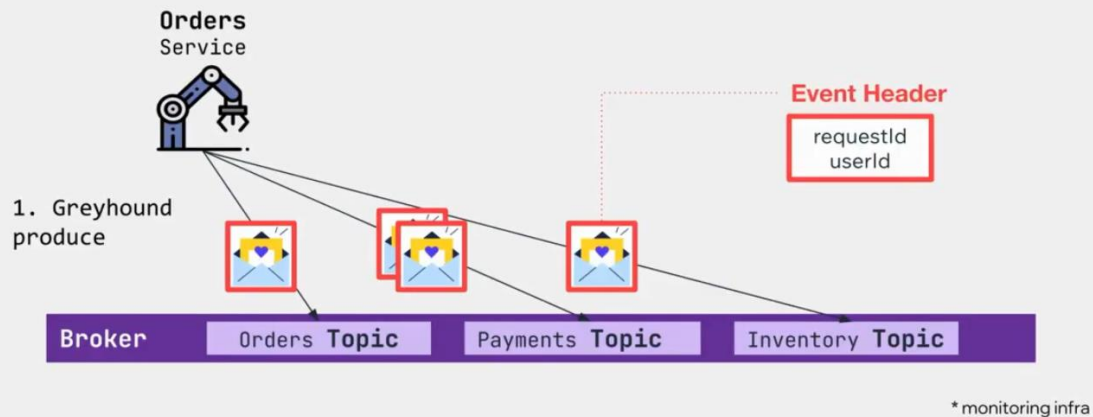
Top Keys
Copy Command
Copy Command
Copy Command
Copy Command
Copy Command

How come this side-effect didn't happen?

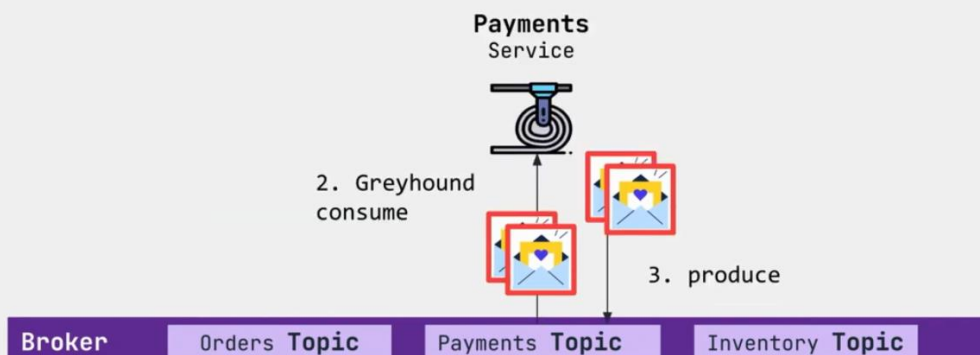
Our team



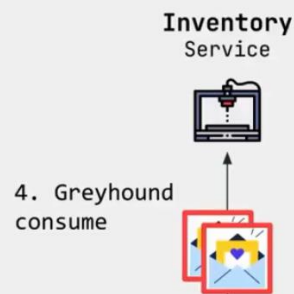
Propagate the Context



Propagate the Context



Propagate the Context



Broker

Orders Topic

Payments Topic

Inventory Topic

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_i	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

Trace View

Trace Start: 2022-09-06 09:47:03.083 Duration: 3.33s Services: 1 Depth: 2 Total Spans: 2

Service & Operation

0µs 833.12ms 1.67s 2.5s 3.33s

Service: [redacted] Duration: 3.33s Start Time: 0µs Child Count: 1

Tags: dc = [redacted] error = true span.kind = internal status.code = 2

Process: service.name = [redacted]

Logs (1)

Service: [redacted] Duration: 3.33s Start Time: 66.25µs

Tags: client.method = TracingDemo client.protocol = [redacted] client.service = [redacted] dc = [redacted] error = true span.kind = client status.code = 2

Process: service.name = [redacted]



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>

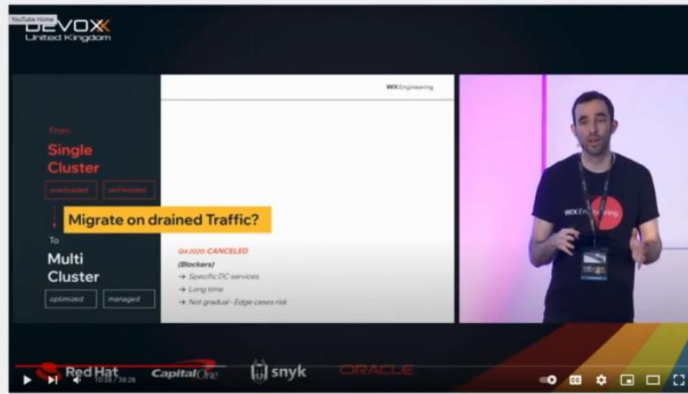
The screenshot shows a Medium article by Natan Silnitsky, dated August 14, with a 10-minute read time. The title is "Event Driven Architecture — 5 Pitfalls to Avoid". The text states: "Event driven architecture is very powerful and well suited for a distributed microservices environment. By introducing a broker intermediary, event driven architecture offers a decoupled architecture, easier scalability, and a much higher degree of resiliency."

Below the text is a diagram illustrating the transition from a synchronous client-server model to an event-driven architecture. On the left, a "client" icon sends a request to a "Server" icon. On the right, separated by a dashed line, a "Message Consumer" icon sends a message to a "Broker" icon, which then routes it to another "Message Consumer" icon.

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

