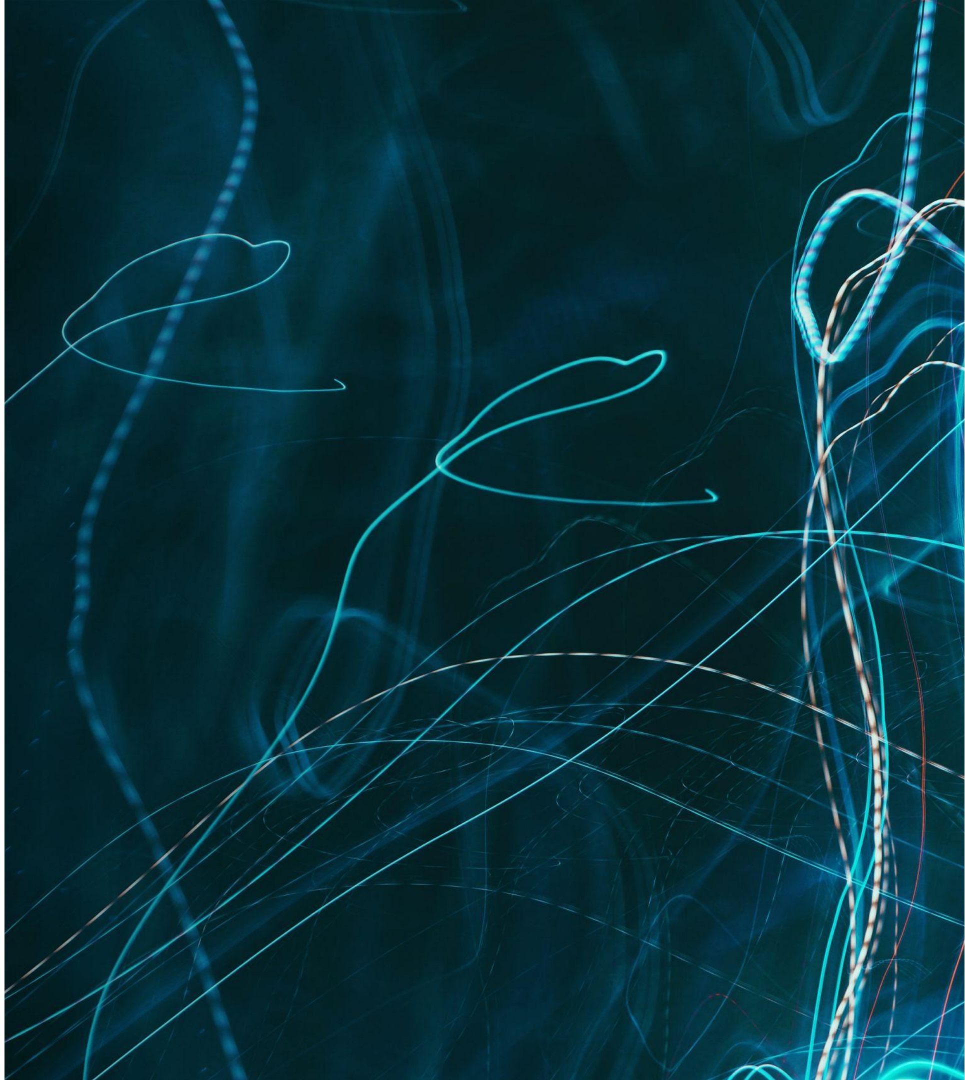
Blunders in event-driven architecture

By Simon Aubury

/thoughtworks



Mistakes - I’ve made a few

Blunders in event-driven architecture

/thoughtworks

Simon Aubury

Principal Data Engineer

/thoughtworks

Life isn’t lived in batches …

Event-driven architecture is a software architecture paradigm promoting the production .. consumption and reaction to events.

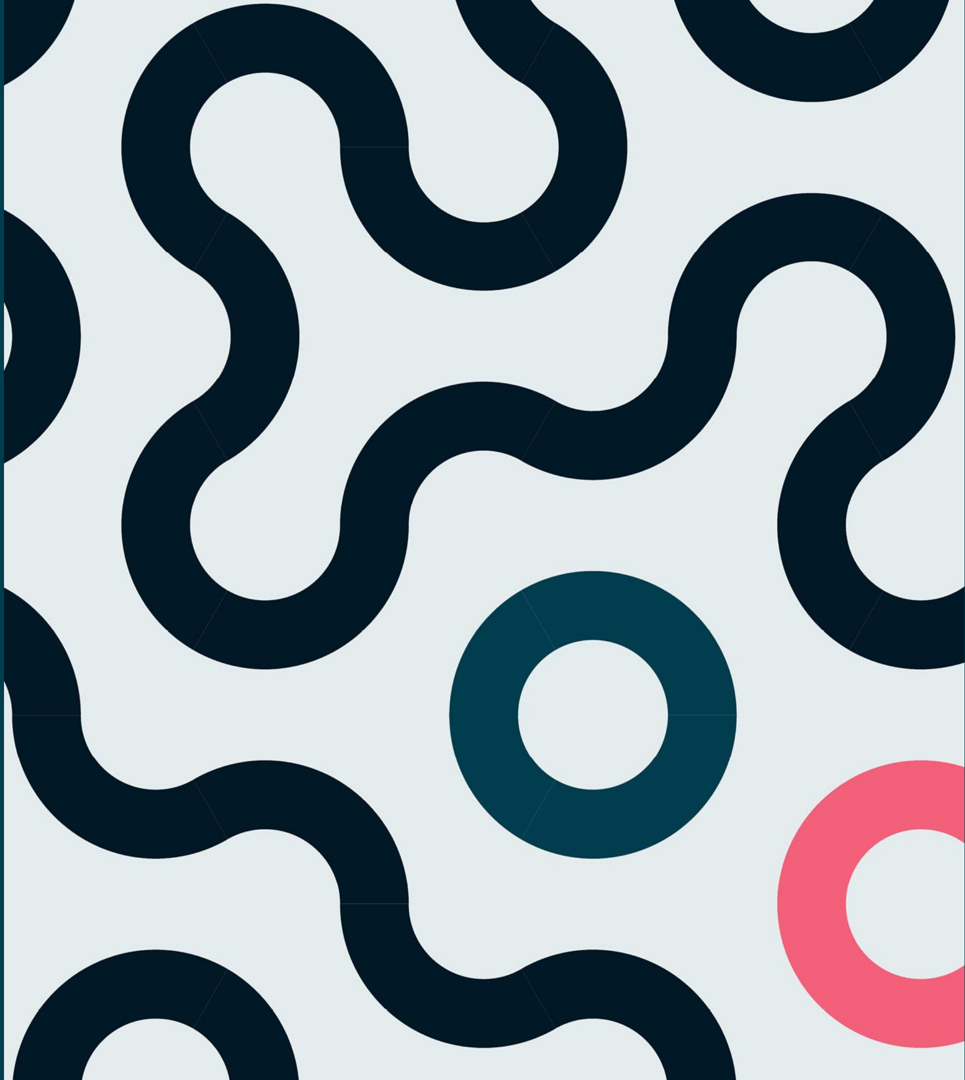
https://en.wikipedia.org/wiki/Event-driven_architecture

Let’s help a younger me

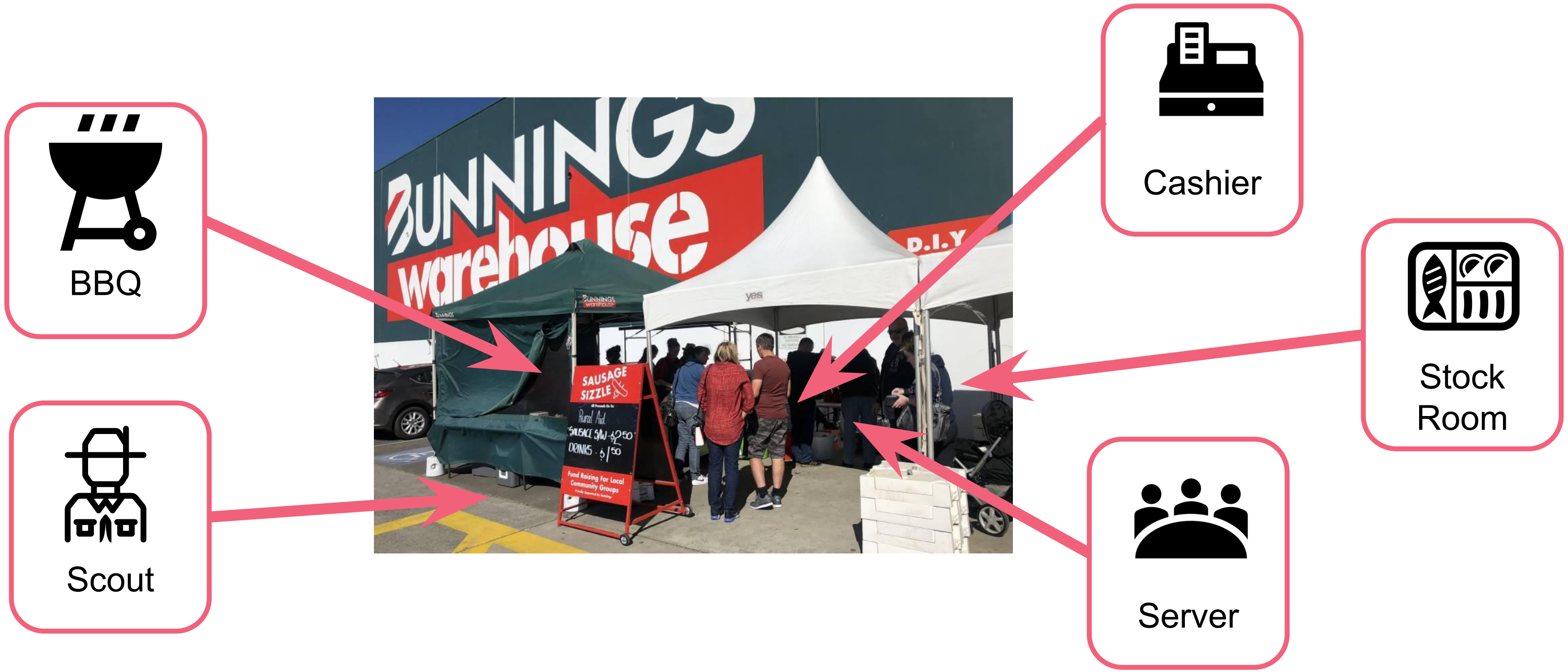
Mistake #1 ...

The “right size” for a Microservice ..

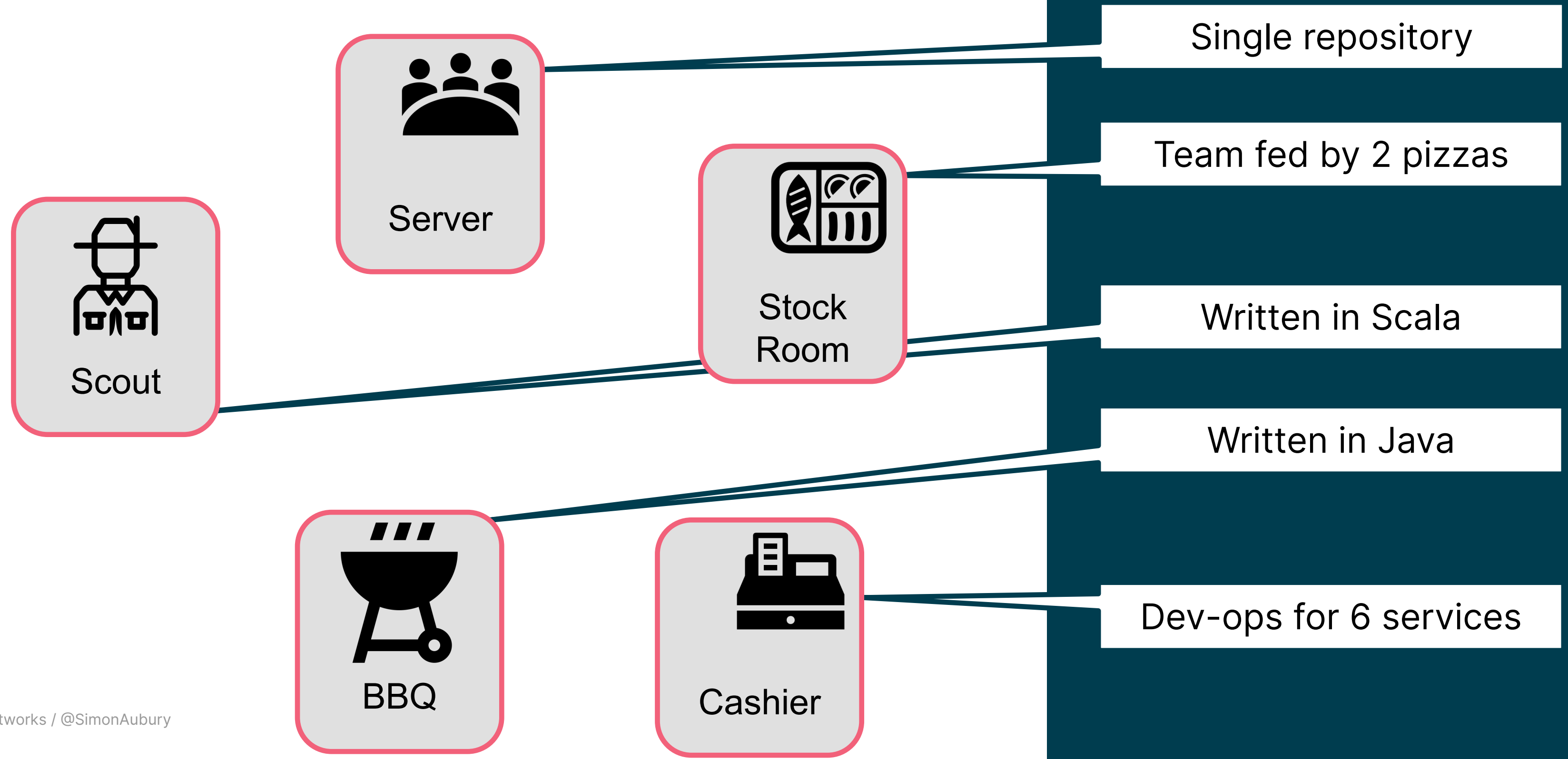
/thoughtworks



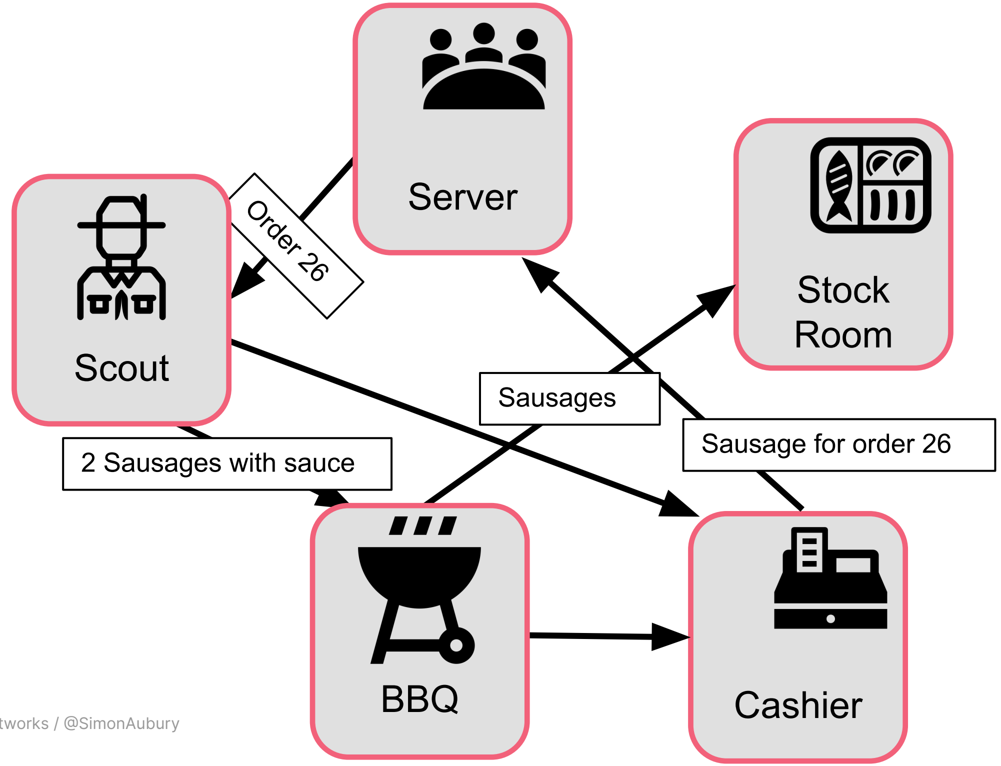
Thinking of events and boundaries



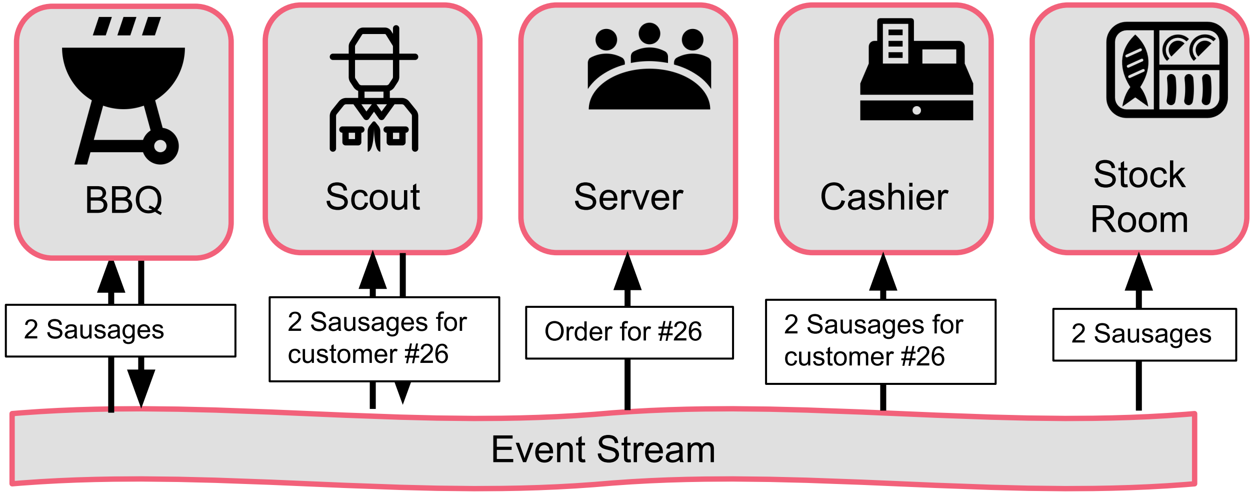
Too obsessed with microservices



Boundary between services



Understand event boundaries



Event Granularity

- Context boundaries
  - How big is a microservice?
- Record events as received
  - Avoid early splitting
- Capture broad categories
  - Producers shouldn’t discard

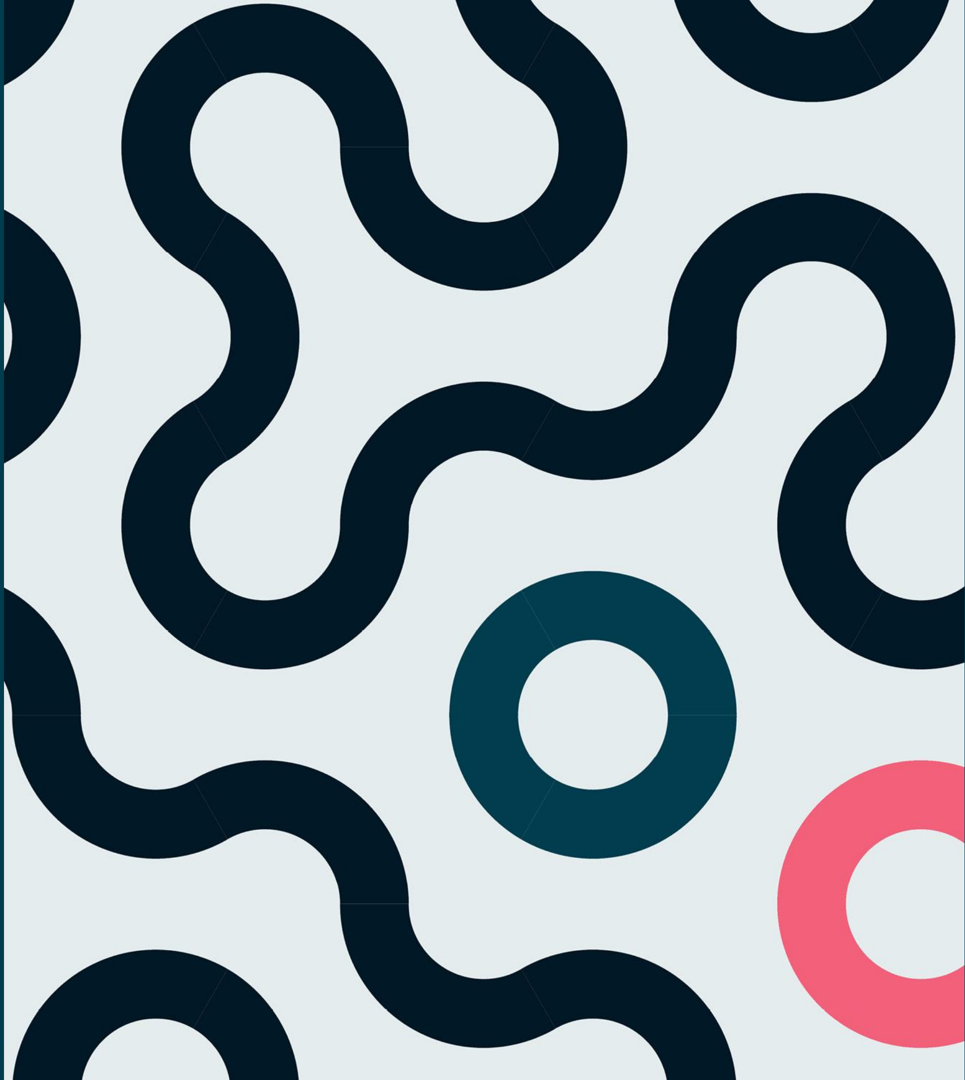
Rules are different within vs across boundaries

Favour asynchrony and eventual consistency at context boundaries, embrace the productive coupling of synchronous within the boundary

Mistake #2 ...

Messages != Events

/thoughtworks



Messages are not events

Message

"Charge Jane's credit card $20"

"Send an email to bob@example.com"

Event

"Jane ordered a burger"

"Bob viewed the new customer web page"

Why events?

Messages in **Job Queue**

Messages

- Persisted until consumed
- Targeted to the entity
- Couple producers and consumers

Events in an **Event Log**

Events

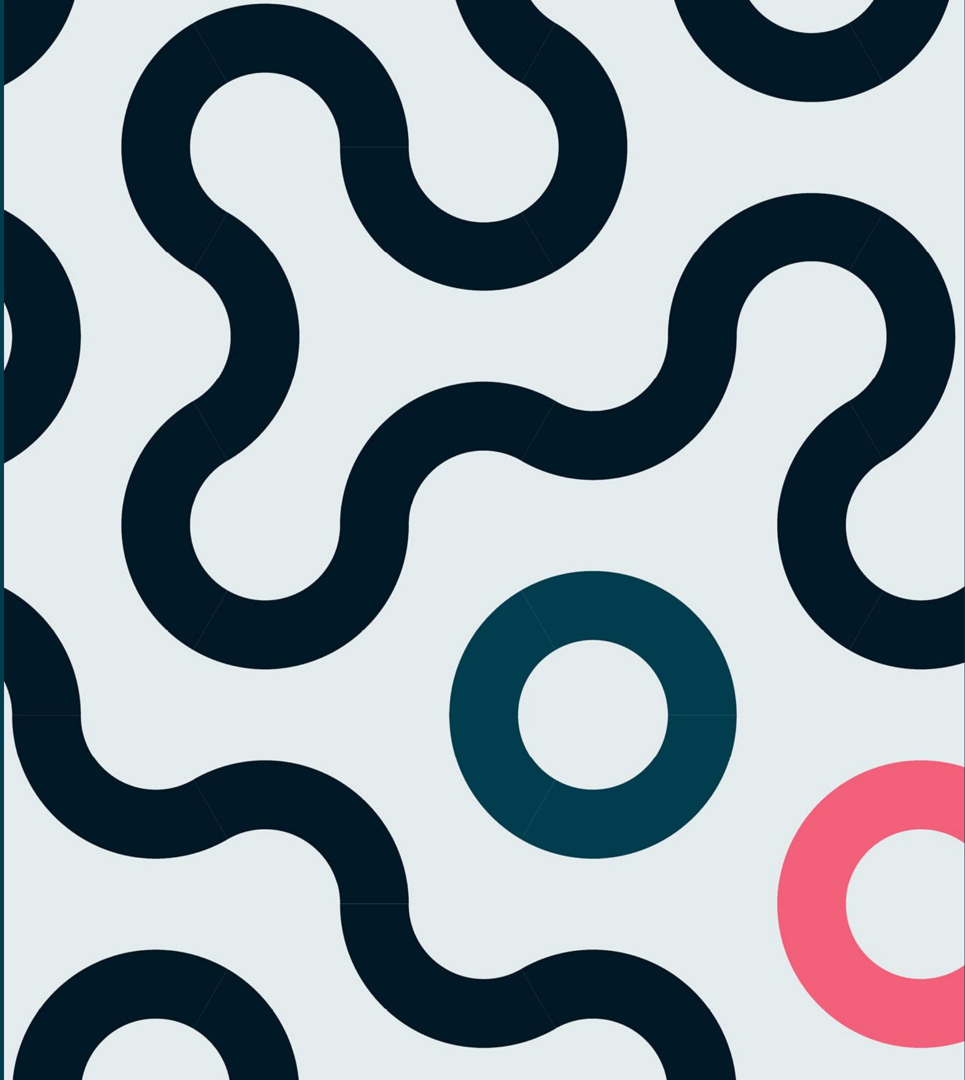
- Re-playable stream history
- Consume historic events
- Loose coupling consumers & producer

Beware the passive aggressive events

It's a "bad smell" if a source system expects the recipient to carry out an action yet styles the message as an event instead.

Mistake #3 ...

Event notification != Event sourcing

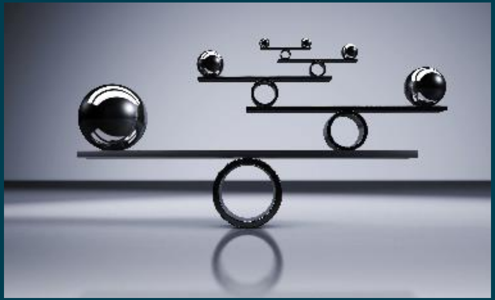


/thoughtworks

Choreography vs. orchestration

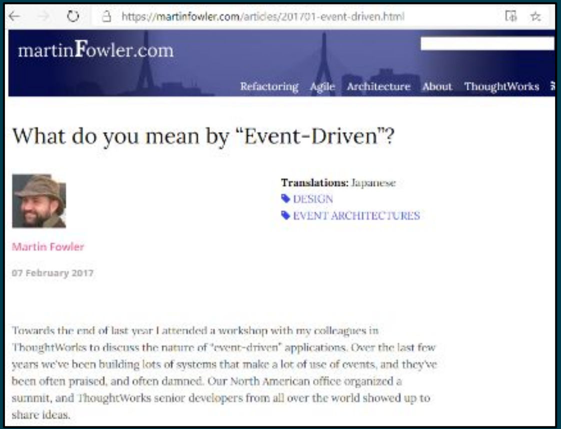
Which system decides that an action should be taken?

- **Orchestration** – system that will perform once told to do it by another system
- **Choreography** (event driven) - a system takes independent action



Many Meanings of Event Driven

- Event Notification
- Event-Carried State Transfer
- Event-Sourcing
- Command Query Responsibility Segregation (CQRS)



What do you mean by “Event-Driven”?

Martin Fowler

07 February 2017

Translations: Japanese
DESIGN
EVENT ARCHITECTURES

Towards the end of last year I attended a workshop with my colleagues in ThoughtWorks to discuss the nature of "event-driven" applications. Over the last few years we've been building lots of systems that make a lot of use of events, and they've been often praised, and often damned. Our North American office organized a summit, and ThoughtWorks senior developers from all over the world showed up to share ideas.

https://martinfowler.com/articles/201701-event-driven.html

Lesson: don’t over engineer

Pick an event driven approach – and be consistent and simple.

Don’t over-engineer an eventing solution using Event-Sourcing / CQRS

Mistake #4 …

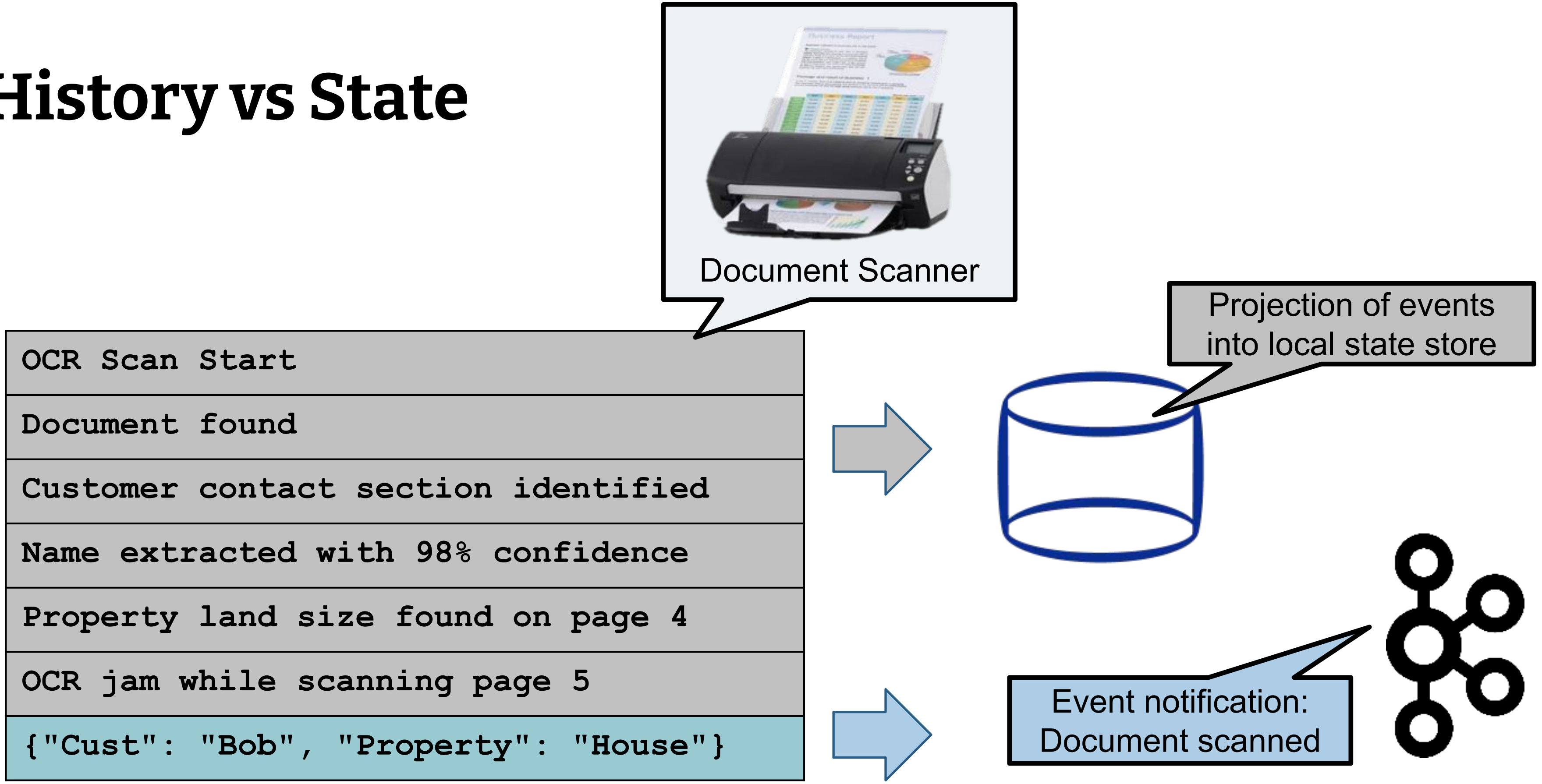
Building a message bus in Kafka

/thoughtworks

History vs State

OCR Scan Start
Document found
Customer contact section identified
Name extracted with 98% confidence
Property land size found on page 4
OCR jam while scanning page 5
{"Cust": "Bob", "Property": "House"}

History vs State



```
OCR Scan Start
Document found
Customer contact section identified
Name extracted with 98% confidence
Property land size found on page 4
OCR jam while scanning page 5
{"Cust": "Bob", "Property": "House"}
```

Document Scanner
Projection of events into local state store
Event notification: Document scanned

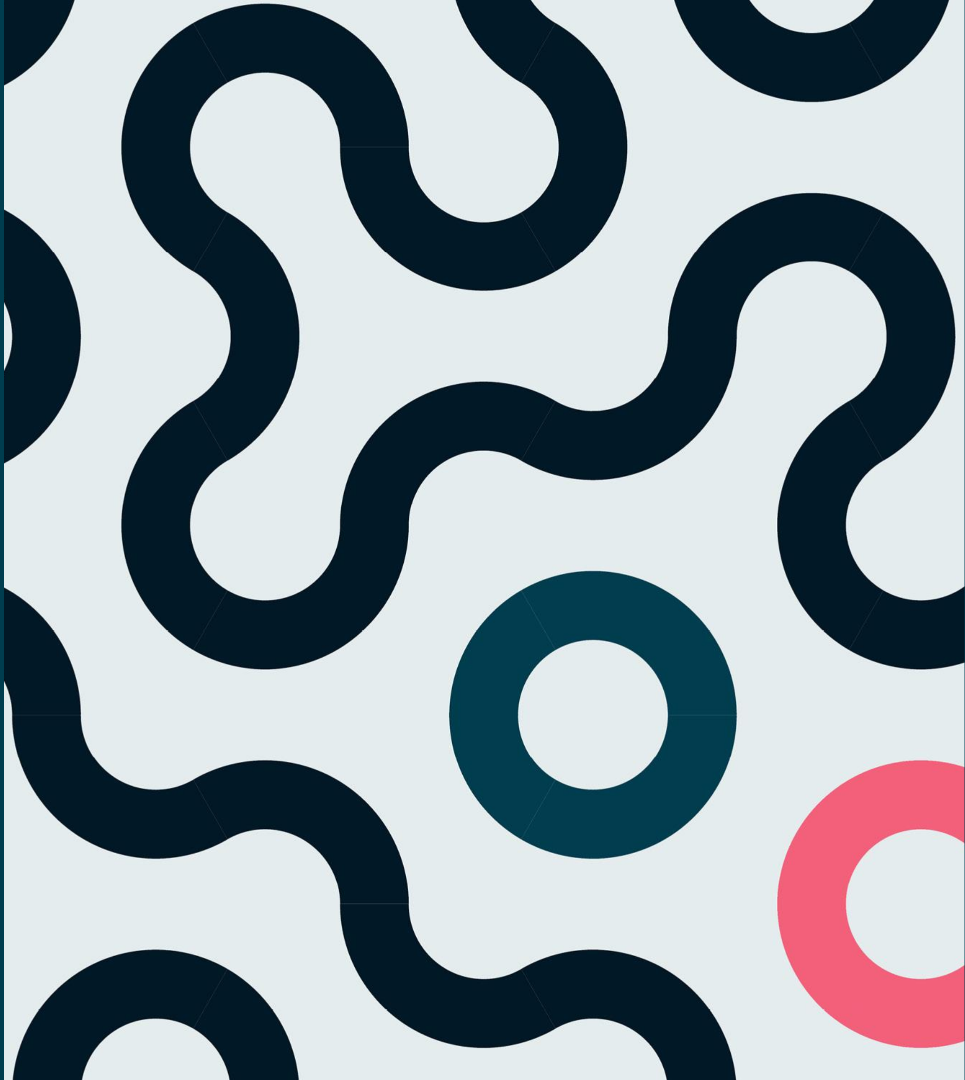
Lesson: Don’t build a message bus in Kafka

It’s a slippery slope – remember to produce & be reactive to events ... and not messages.

Mistake #5 …

Producers are not children

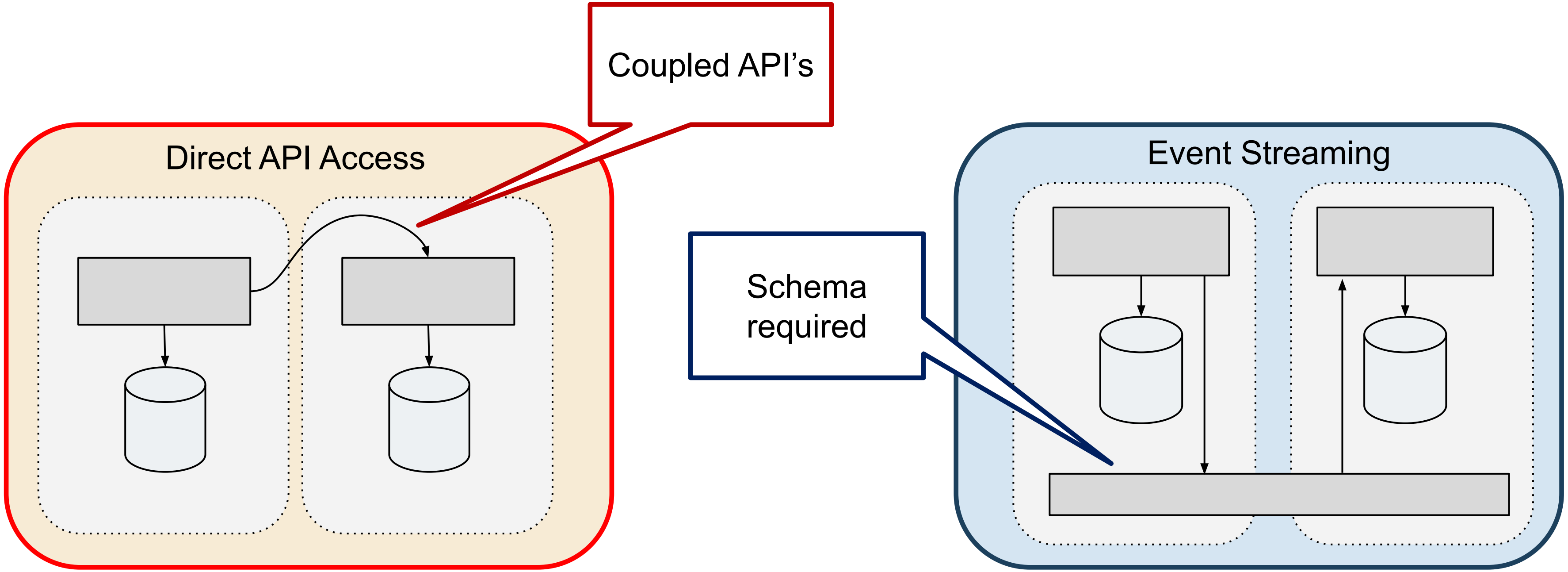
thoughtworks



Producers are not children

- Basic rules for events
  - EDA guard-rails
  - Data Governance & Security
  - Good citizen rules
  - Naming scheme for topics
- Support & training
  - Opinioned framework approach

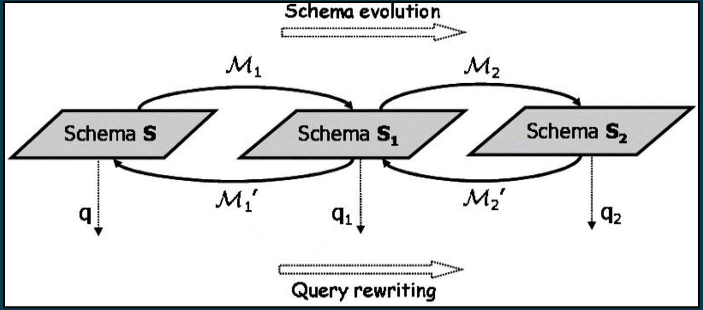
Cross bounded context ...

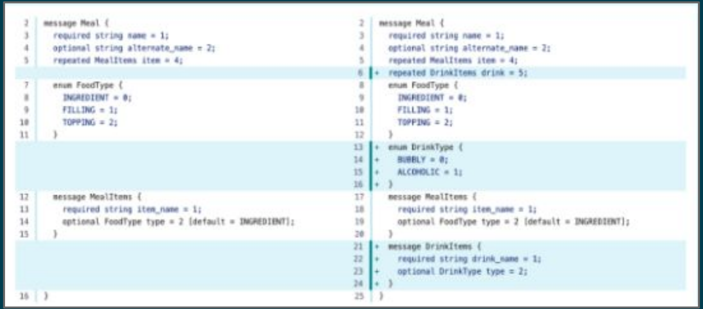


Plan for schema evolution

Support change - data domains need to evolve at their own rate ... without breaking consumers.

TL;DR - Use a schema registry





```
message Meal {
  required string name = 1;
  optional string alternate_name = 2;
  repeated MealItems item = 4;

  enum FoodType {
    INGREDIENT = 0;
    FILLING = 1;
    TOPPING = 2;
  }
  message MealItems {
    required string item_name = 1;
    optional FoodType type = 2 [default = INGREDIENT];
  }
}
```

```
message Meal {
  required string name = 1;
  optional string alternate_name = 2;
  repeated MealItems item = 4;
+ repeated DrinkItems drink = 5;
  enum FoodType {
    INGREDIENT = 0;
    FILLING = 1;
    TOPPING = 2;
  }
+ enum DrinkType {
+   BUBBLY = 0;
+   ALCOHOLIC = 1;
+ }
  message MealItems {
    required string item_name = 1;
    optional FoodType type = 2 [default = INGREDIENT];
  }
+ message DrinkItems {
+   required string drink_name = 1;
+   optional DrinkType type = 2;
+ }
}
```

There is a process to find the events that matter

- Use everyone to identify the events that matter
- Understand the systems
- Start with broad categories

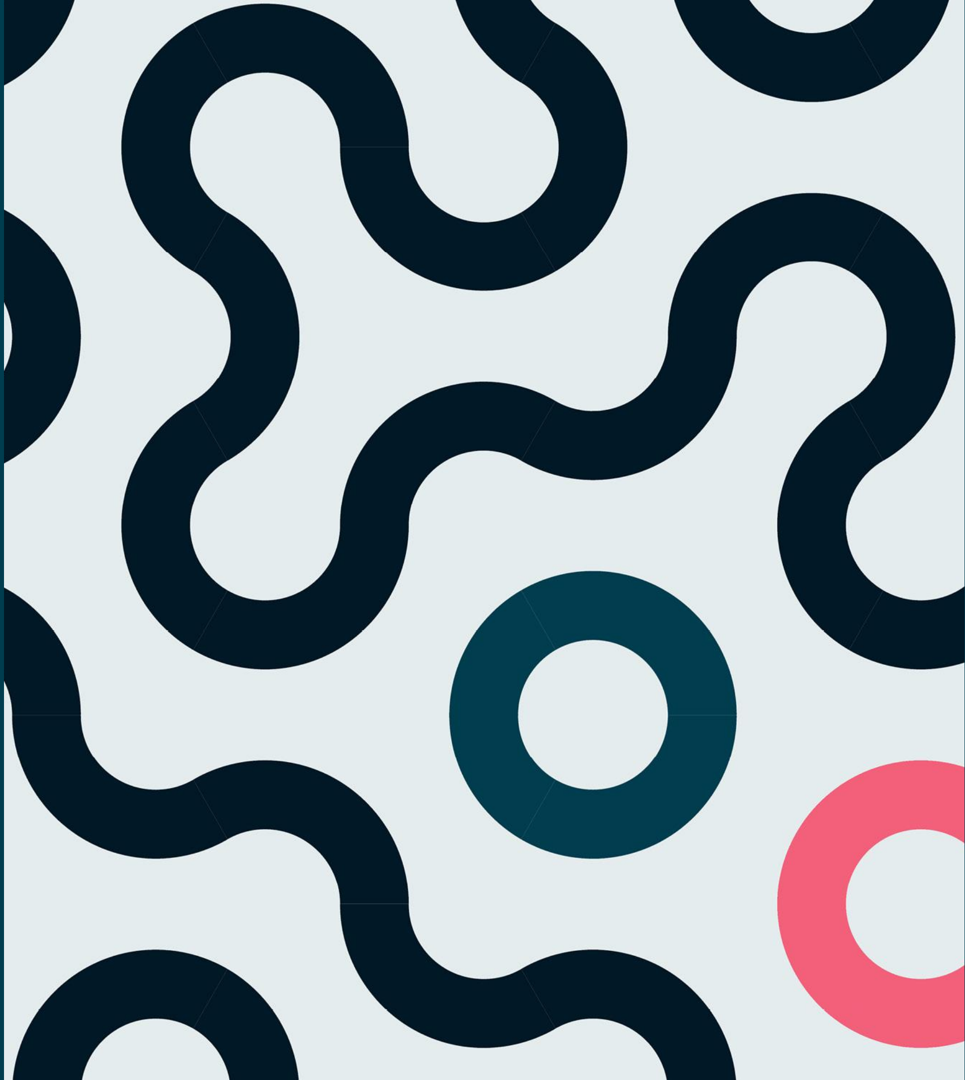
Event must have's

- Name – past tense
- Correlation ID
- Event production time
- Originating system
- Event creation system (may be different)
- A payload of stuff

Mistake #6 …

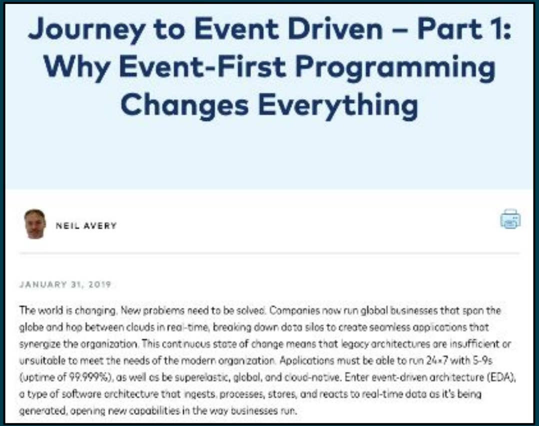
Don’t reinvent the EDA wheel

/thoughtworks



Event first thinking

- Capture facts & behaviour
- Represent the real world
- Model use cases of how we think
- Repeatability & scaling
- Common language



Journey to Event Driven – Part 1: Why Event-First Programming Changes Everything

NEIL AVERY

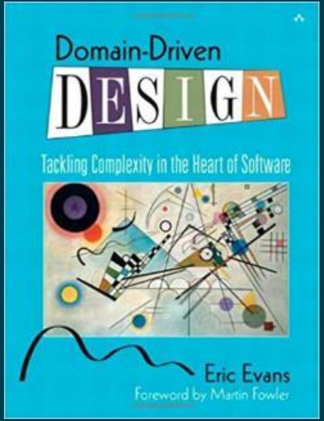
JANUARY 31, 2019

The world is changing. New problems need to be solved. Companies now run global businesses that span the globe and hop between clouds in real-time, breaking down data silos to create seamless applications that synergize the organization. This continuous state of change means that legacy architectures are insufficient or unsuitable to meet the needs of the modern organization. Applications must be able to run 24×7 with 5-9s (uptime of 99.999%), as well as be superelastic, global, and cloud-native. Enter event-driven architecture (EDA), a type of software architecture that ingests, processes, stores, and reacts to real-time data as it's being generated, opening new capabilities in the way businesses run.

https://www.confluent.io/blog/journey-to-event-driven-part-1-why-event-first-thinking-changes-everything/

DDD - existing practices

- Problem modelling
  - Contexts - delineate boundary of consistency
- Separate our business logic from other application concerns
- Reduce complexity
  - More effective software delivery
- Communicate better / A common language


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Know your events

Modelling - discovery & integration

Use simple language ..
  solicit everyone's input.

Develop your system inside out,
focus on the domain

It’s hard work without DDD

Domain Driven Design gives us the tools to define our bounded contexts, which give us our services.

What did I learn?

- Messages != Events
- Rules are different within vs across boundaries
- Don’t over engineer
- Don’t build a message bus in Kafka
- EDA guardrails
- Know your events

Lesson: start .. now

Event Driven Architecture adoption should start now.

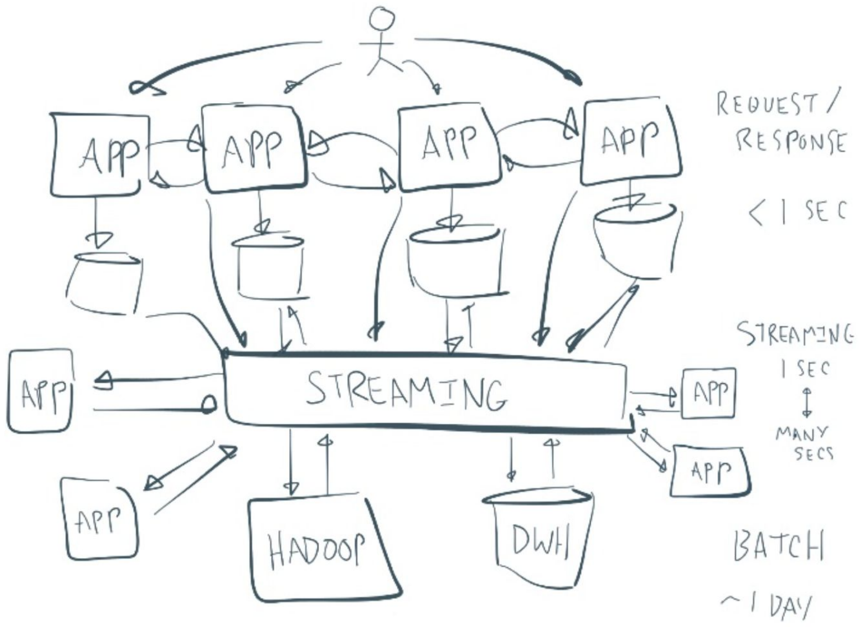
Starting the first leads to the next transformational opportunity.

Make (and share) your own mistakes ...

Q & A

/thoughtworks

Many options...



amazon KINESIS

kafka

PULSAR

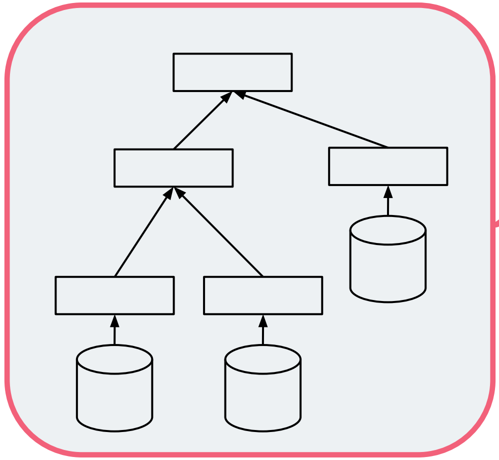
Microsoft Azure Event Hubs

amazon SQS

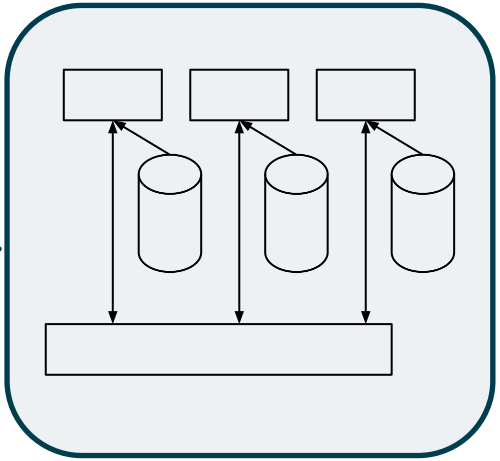
EVENT STORE®

Take a bunch of events

Monolith / Microservices



Microservices + Events



11:23

Wednesday, October 21

Uber

Rates have dropped at 350 5th Ave. Request a ride as soon as possible to avoid surge pricing.