Events on the outside, on the inside and at the core

Chris Richardson

Founder of Eventuate.io

Founder of the original CloudFoundry.com

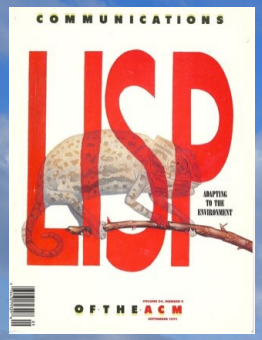
Author of POJOs in Action and Microservices Patterns

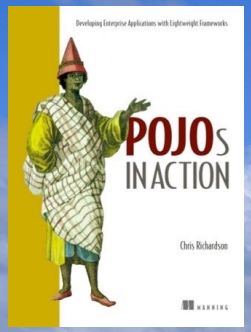
@crichardson

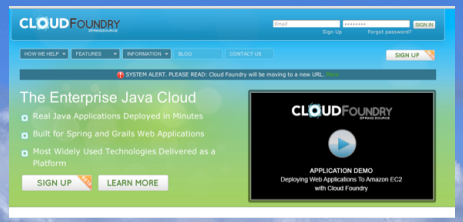
chris@chrisrichardson.net

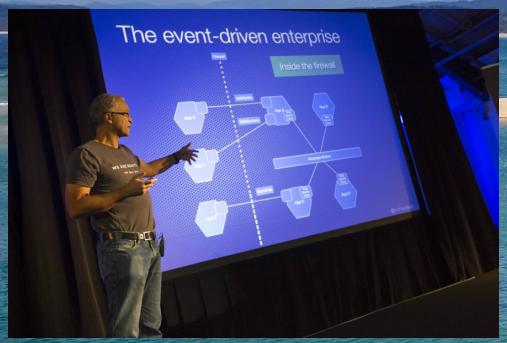
http://learn.microservices.io

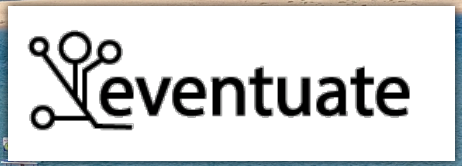
About Chris

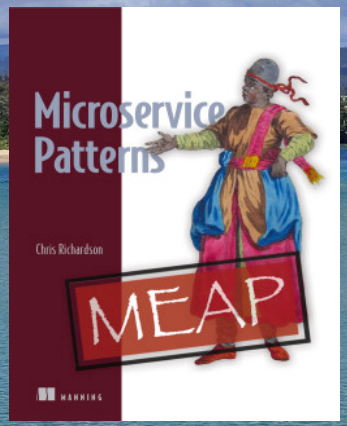








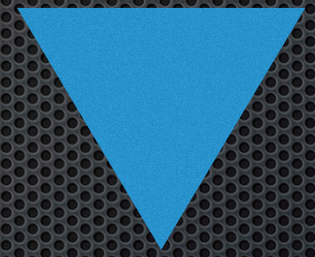


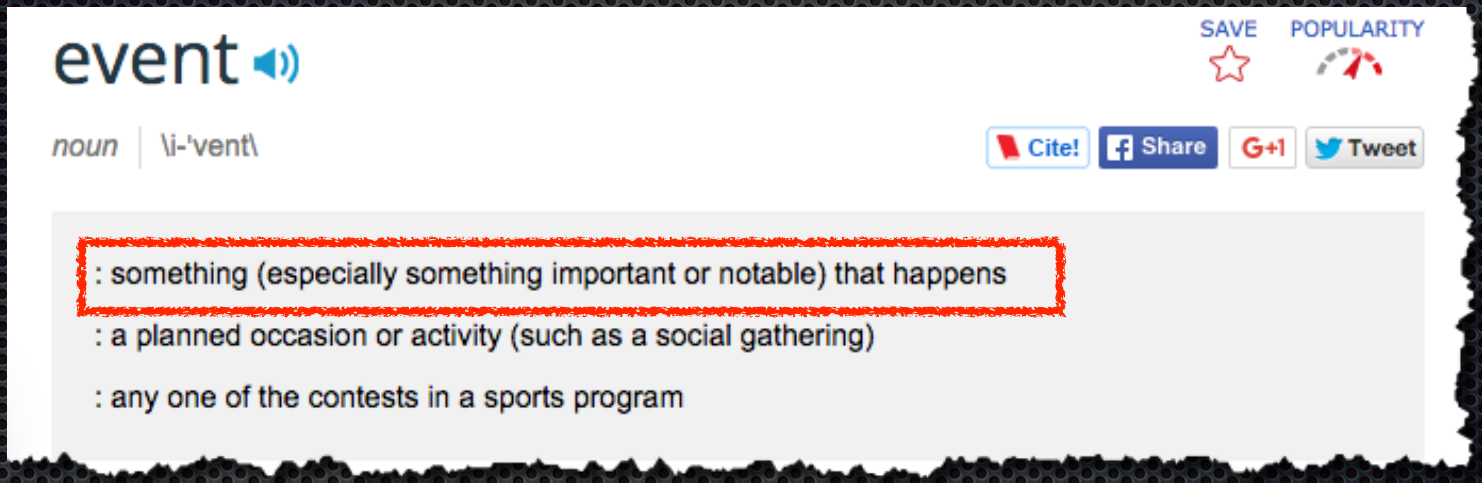


http://learn.microservices.io

ctwsoftarchconf18

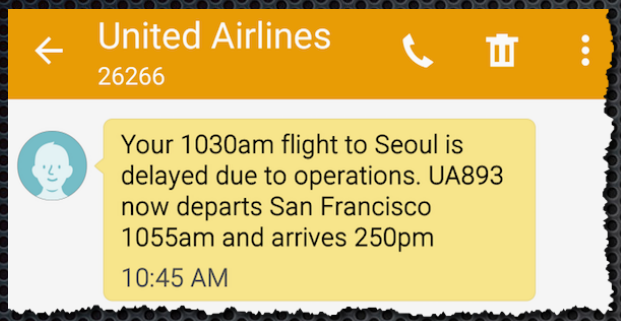
What's an event?

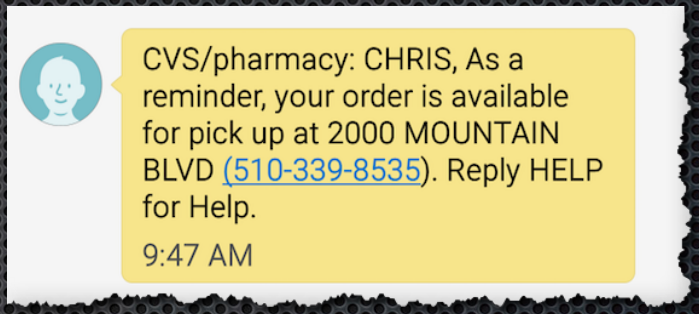


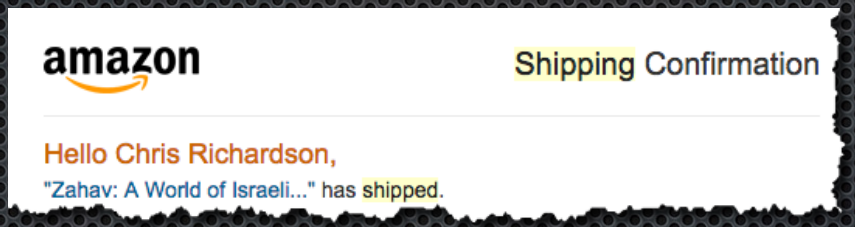


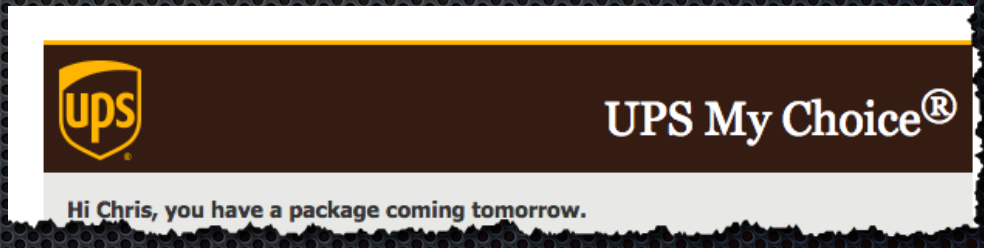
http://www.merriam-webster.com/dictionary/event

Examples of events



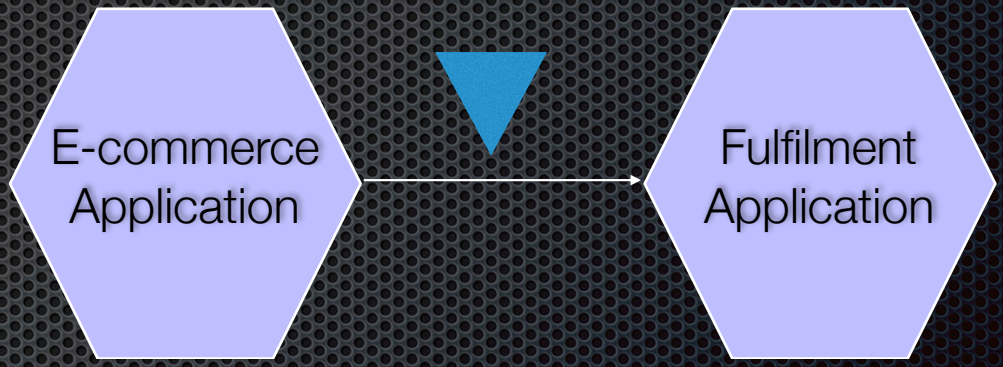




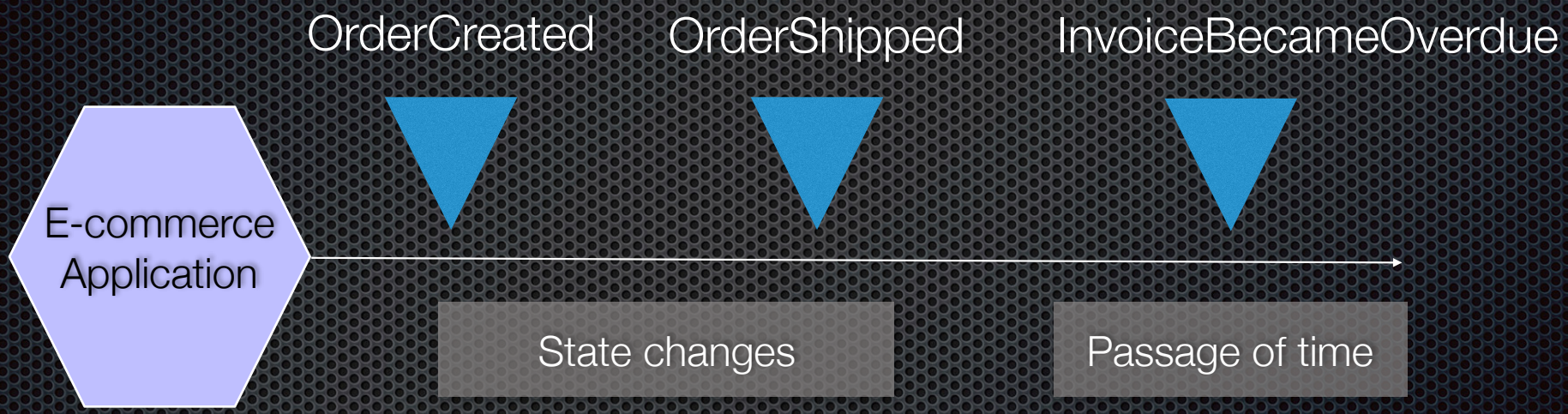


Events play a role at every level of an architecture

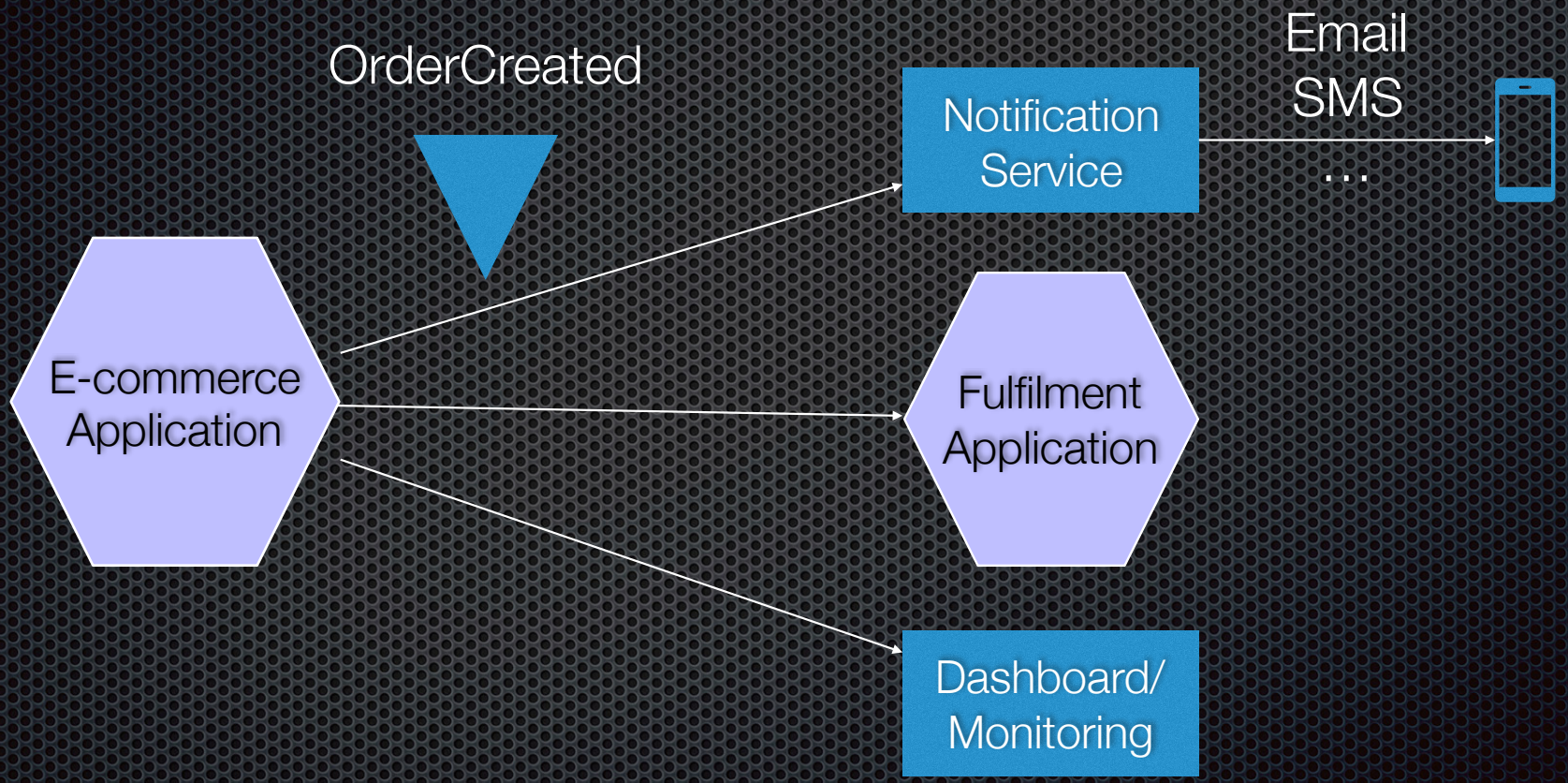
Events on the outside



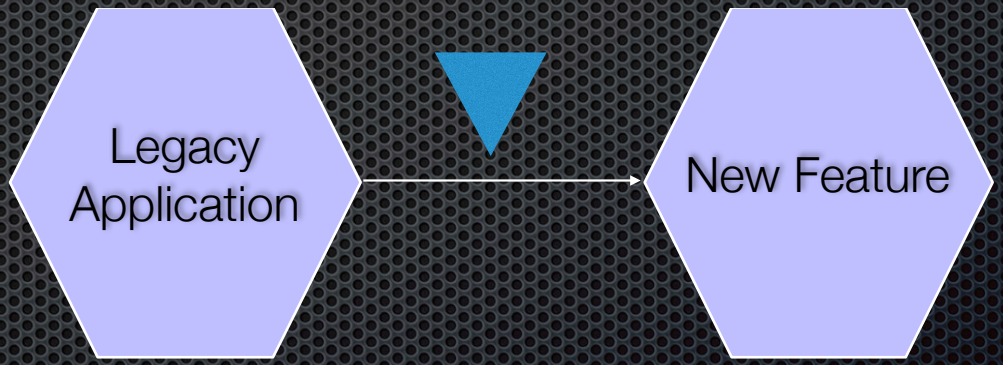
An application emits an event when…



Who consumes an event?

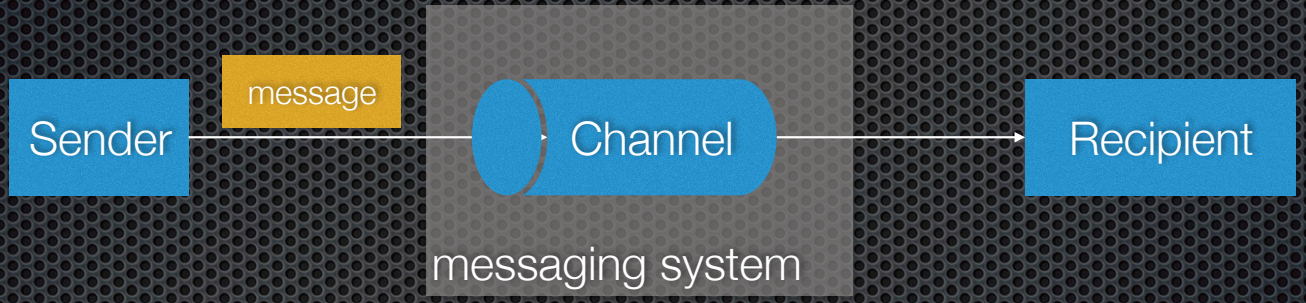


Implements the
Open for extension/**Closed** for
modification principle for
applications



How to transport events?

Inside the firewall: Messaging-based IPC



RabbitMQ

ActiveMQ

Apache Kafka
A high-throughput distributed messaging system.

AWS Kinesis

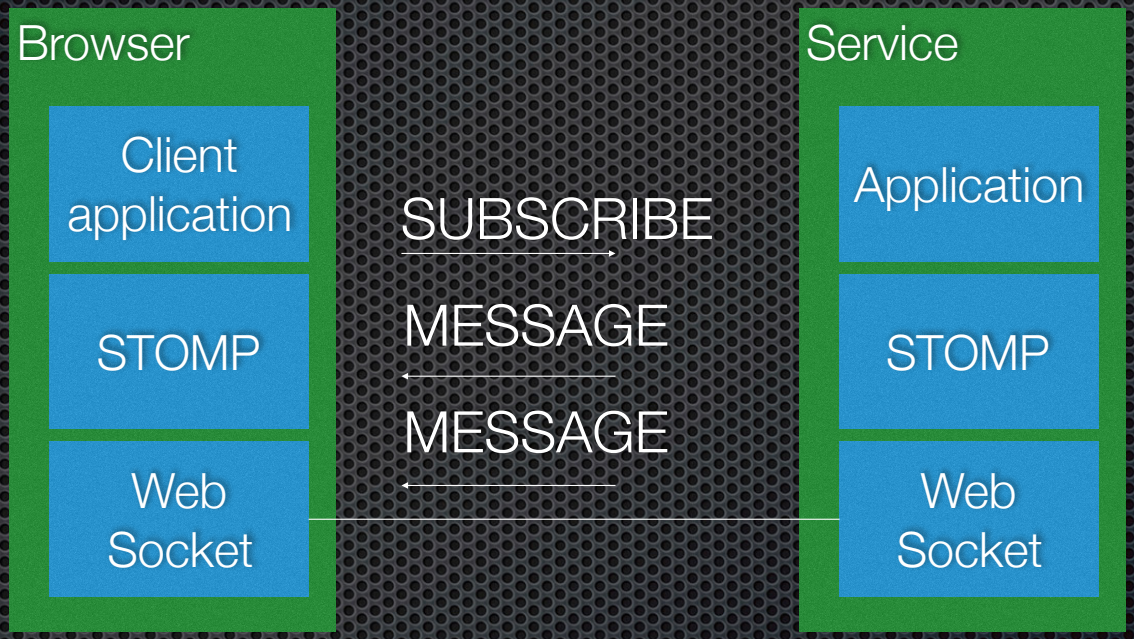
Outside the firewall

Polling for events

- HTTP
  - Periodically poll for events
- Atom Publishing Protocol (AtomPub)
  - Based on HTTP
  - Head is constantly changing
  - Tail is immutable and can be efficiently cached

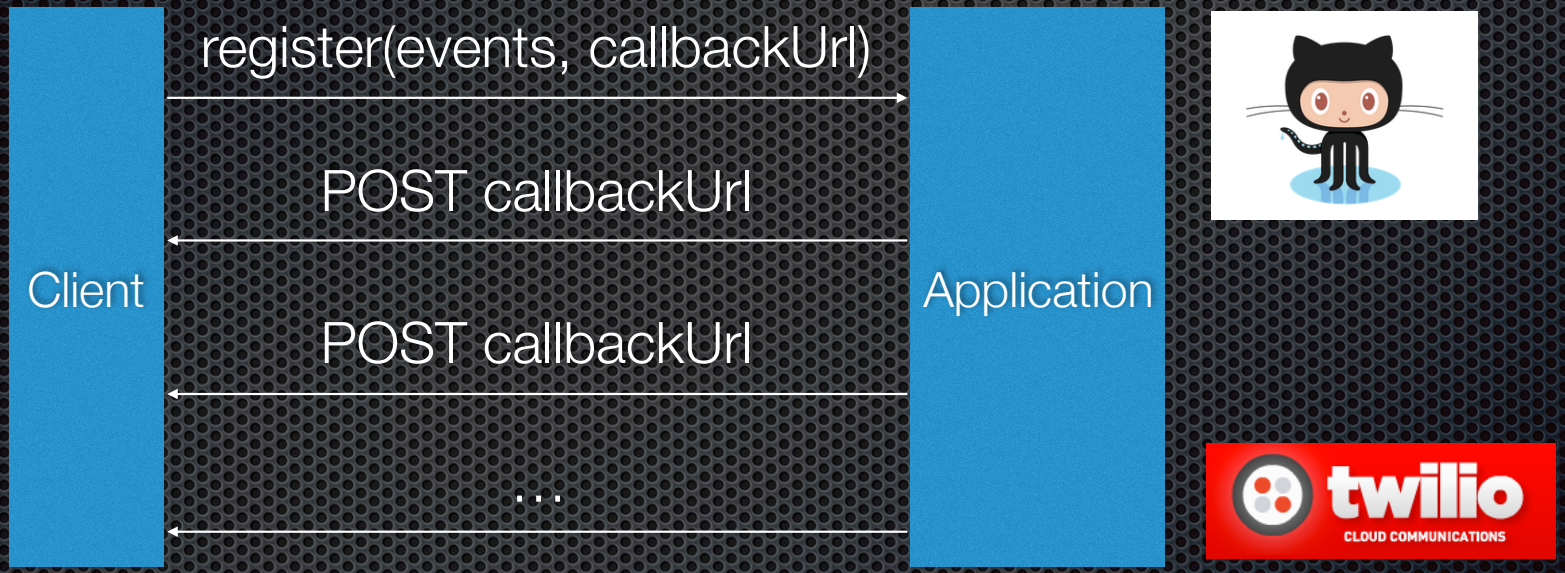
High-latency, inefficient

Using WebSockets



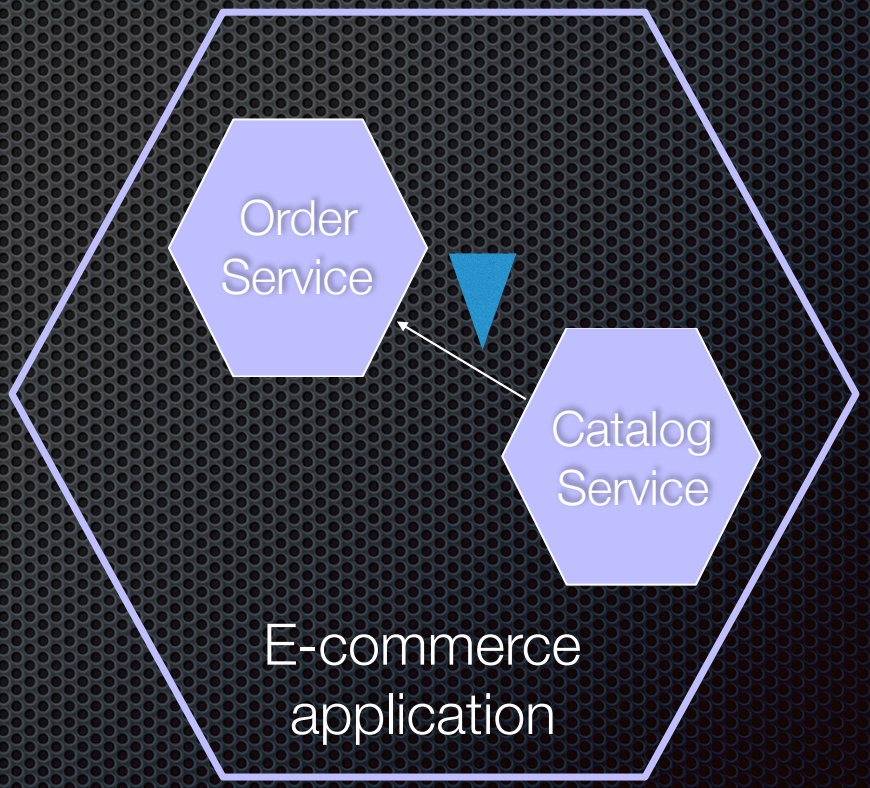
Low latency, more efficient, but what about past events?

Webhooks = web friendly publish/ subscribe



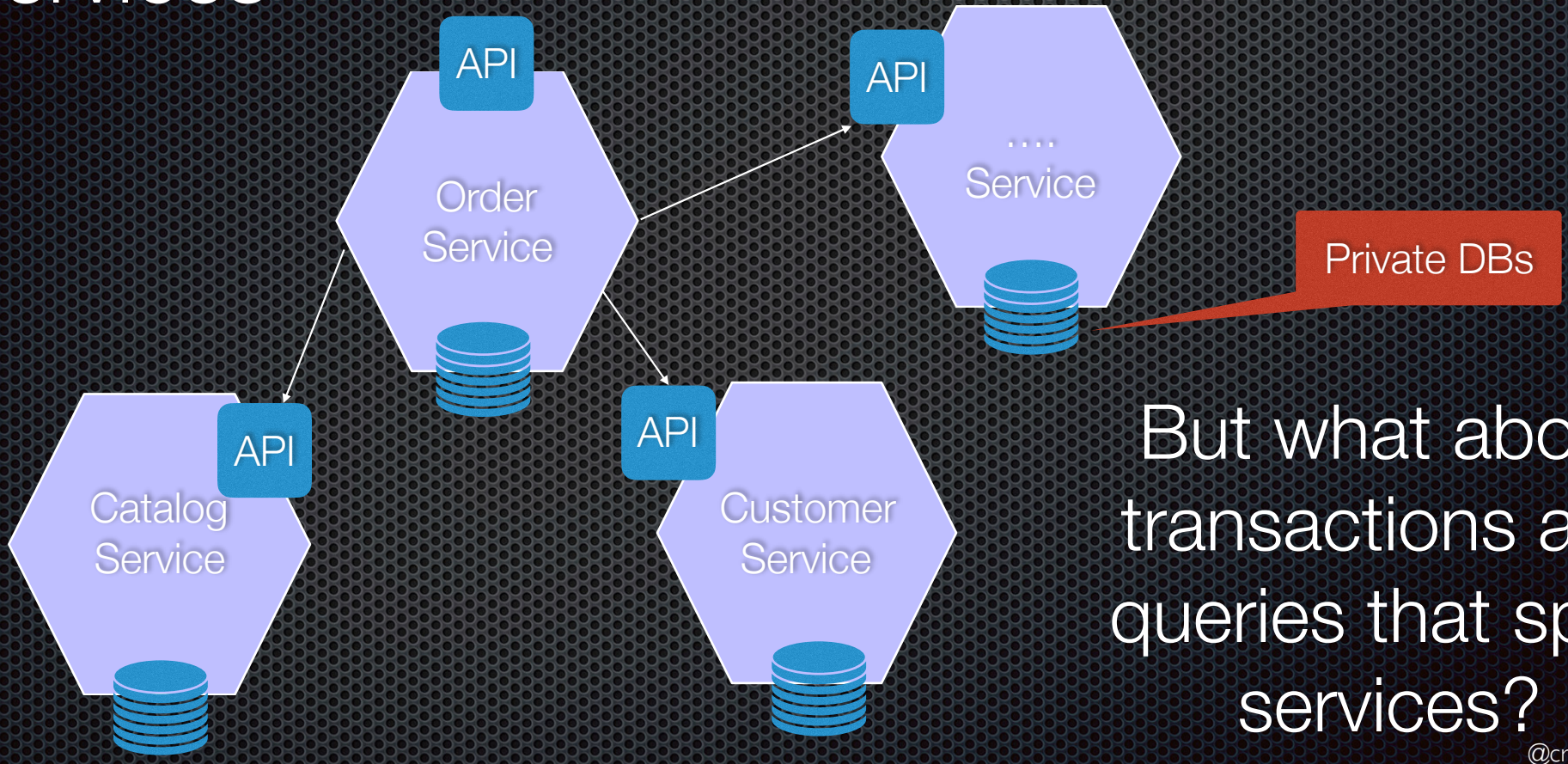
Low latency, more efficient, but what about past events?

Events on the inside



The Microservice architecture enables the rapid, safe delivery of large, complex applications

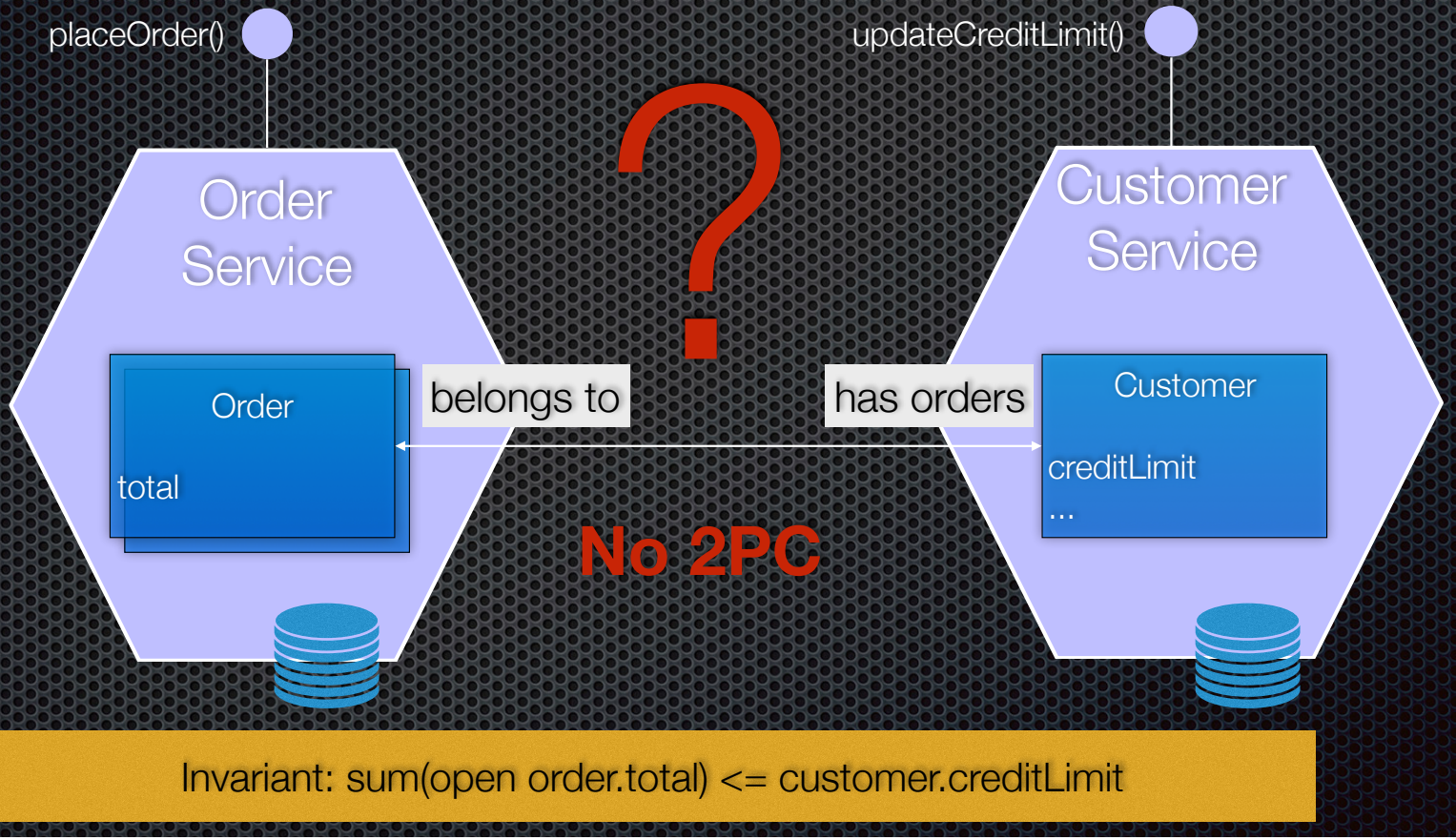
E-commerce application - refactored to services



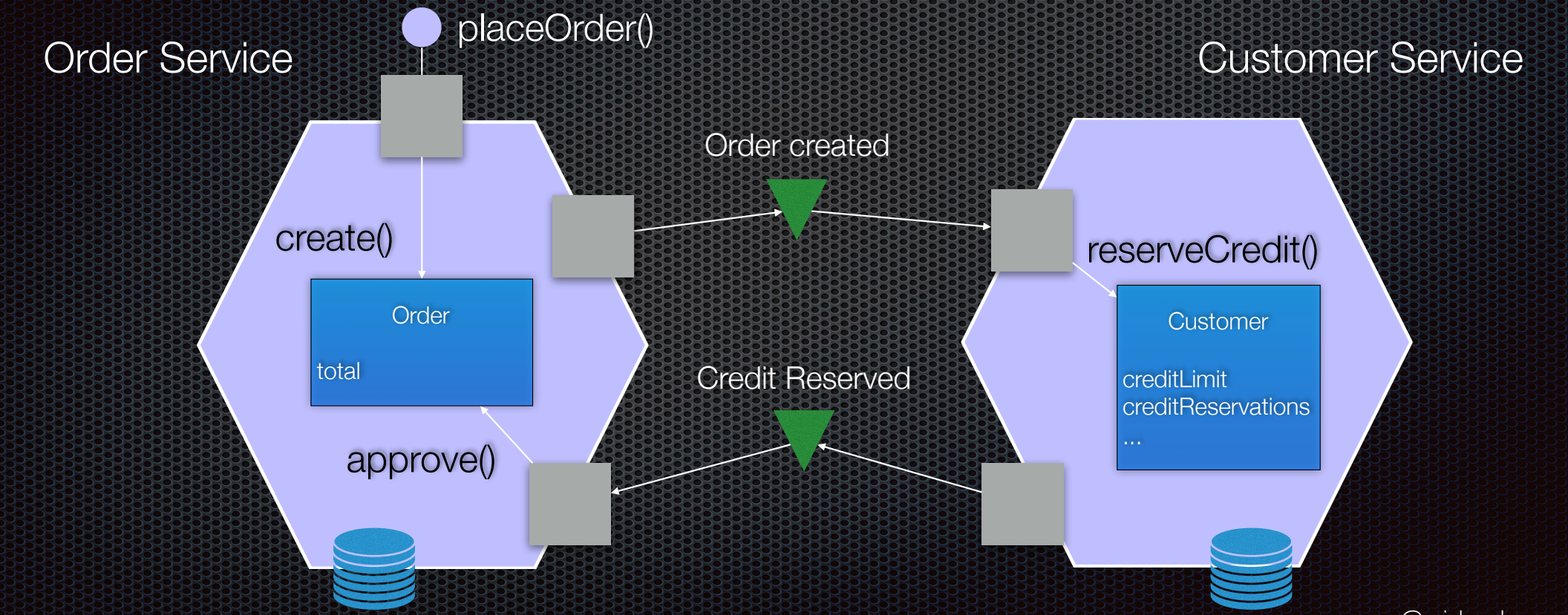
But what about transactions and queries that span services?

How to implement transactions that span services?

How to enforce the customer's credit limit?



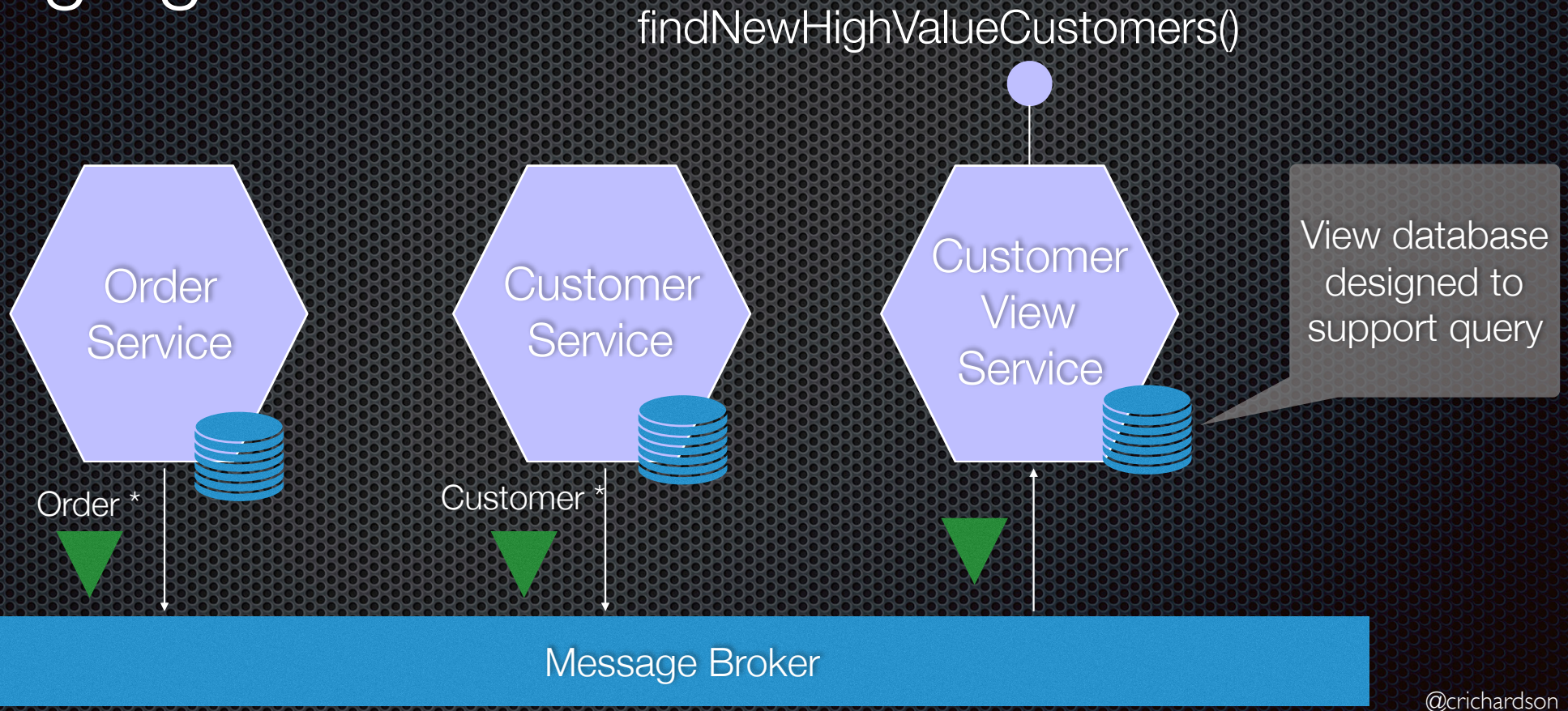
Use event-driven, choreography-based sagas



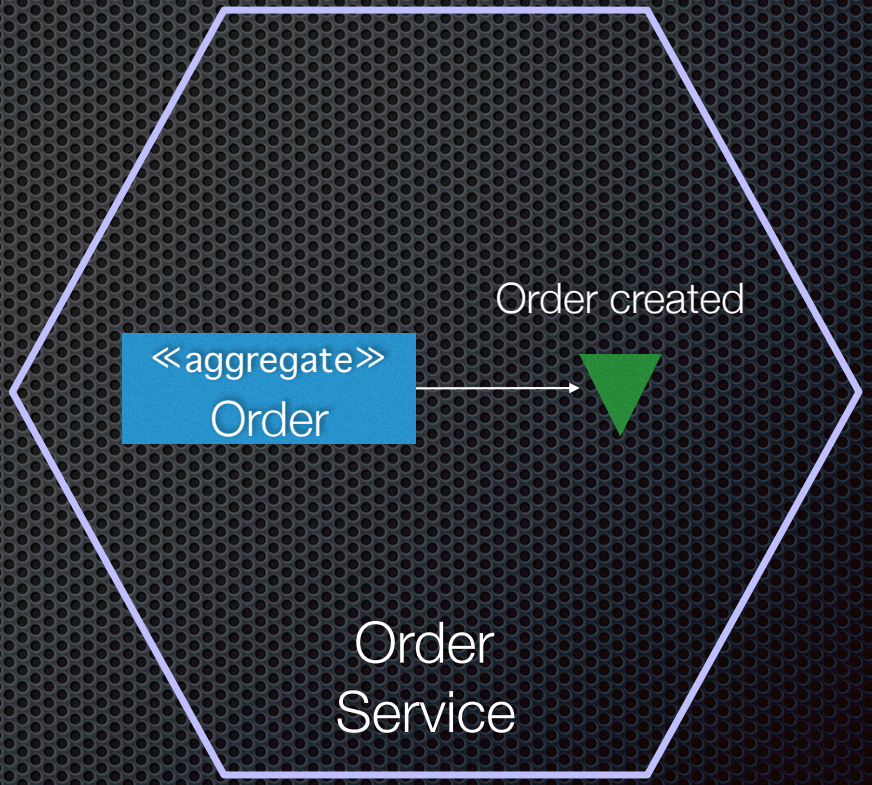
How to implement queries that span services?

Find new **customers** who have placed high value **orders** that have shipped

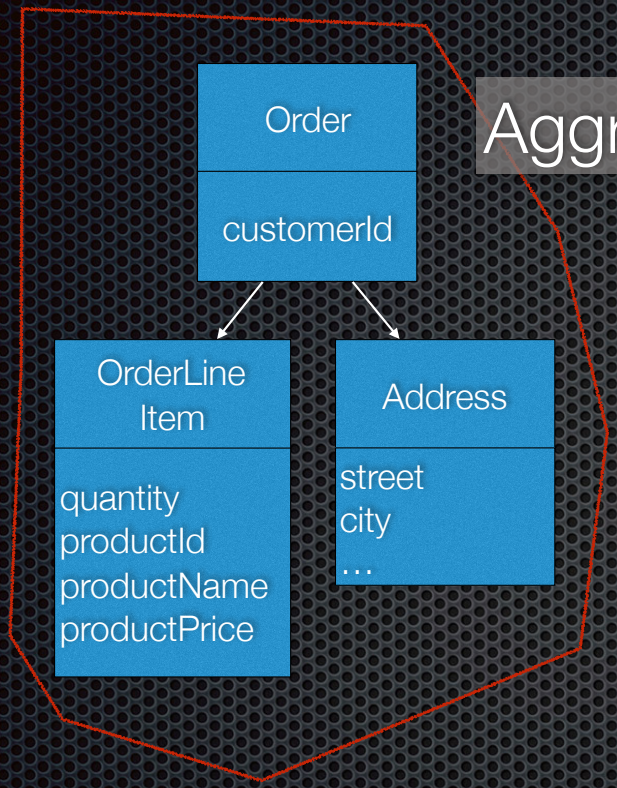
Use Command Query Responsibility Segregation



Events at the core



DDD: Aggregates publish domain events



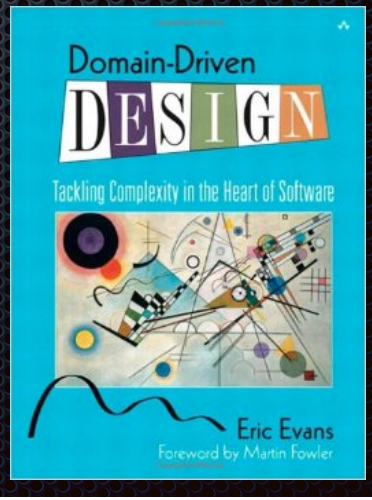
Aggregate

Domain event

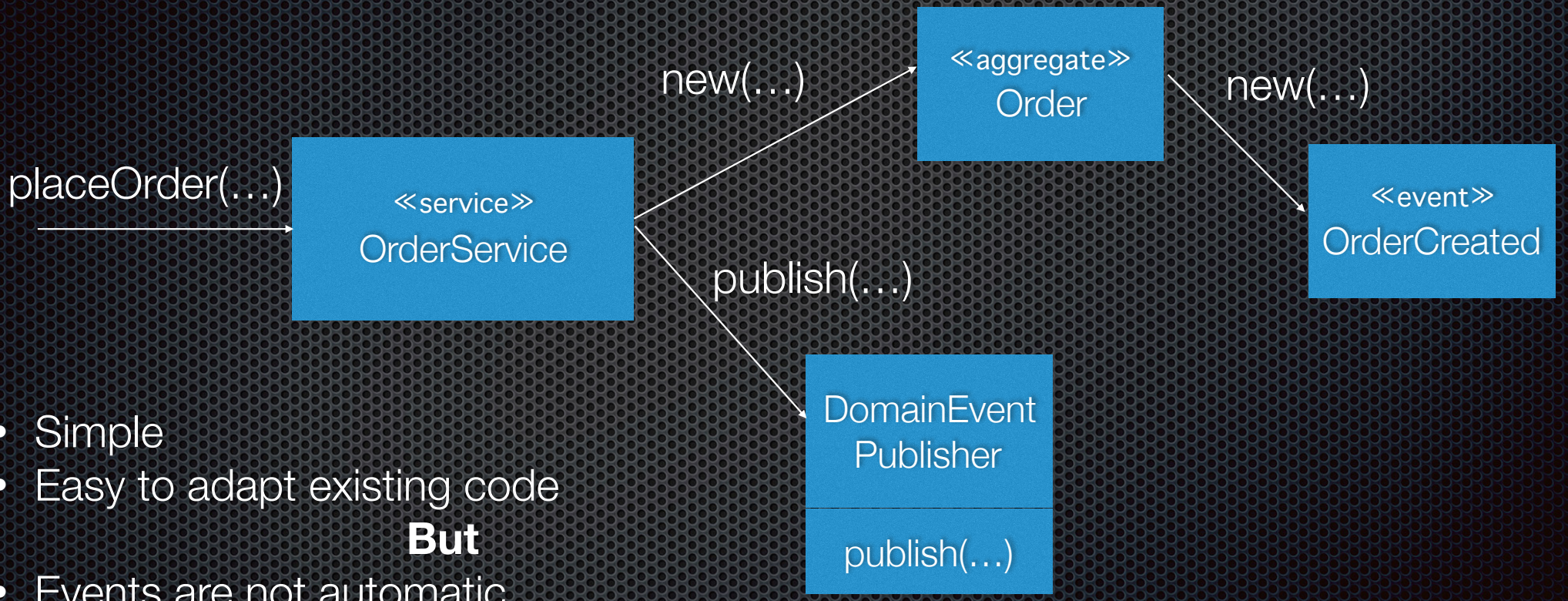
Order created

Publishes

when created or state changes



Business logic explicitly invoking event publishing API



- Simple
- Easy to adapt existing code

But

- Events are not automatic
- Tricky to publish messages transactionally

Use event sourcing

Event table

Entity id	Entity type	Event id	Event type	Event data
101	Order	901	OrderCreated	…
101	Order	902	OrderApproved	…
101	Order	903	OrderShipped	…

Event-based persistence

+

«aggregate»
Order

List<Event> process(CreateOrderCommand)
void apply (OrderCreatedEvent)

List<Event> process(CancelOrderCommand)
void apply (OrderCanceledEvent)

Event-based business logic

Reliable publishing, aggregate history, auditing, …. **YET** radically different

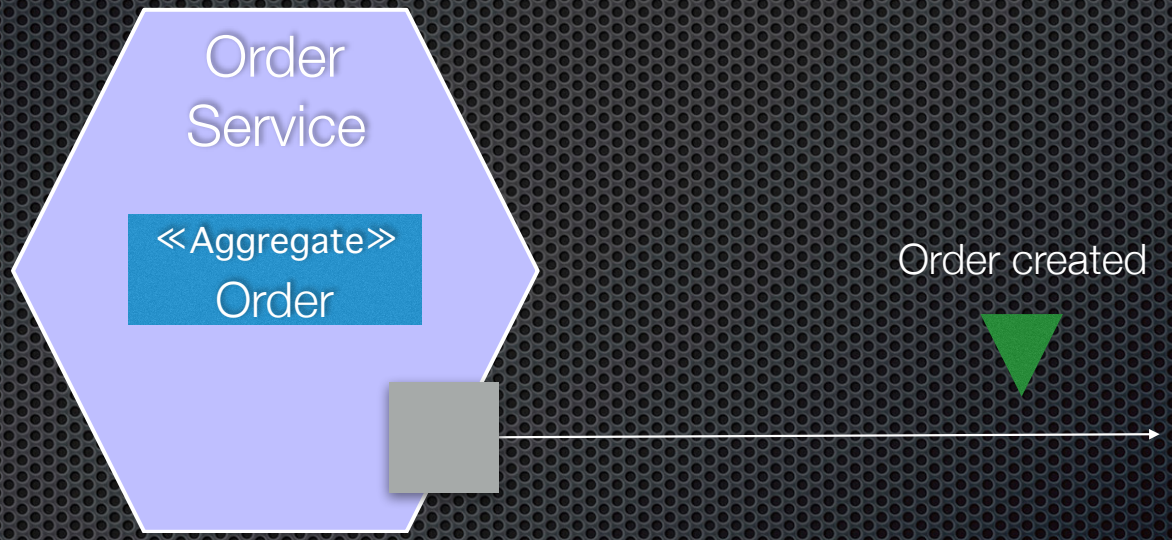
Summary

Events play a role at every level of an architecture

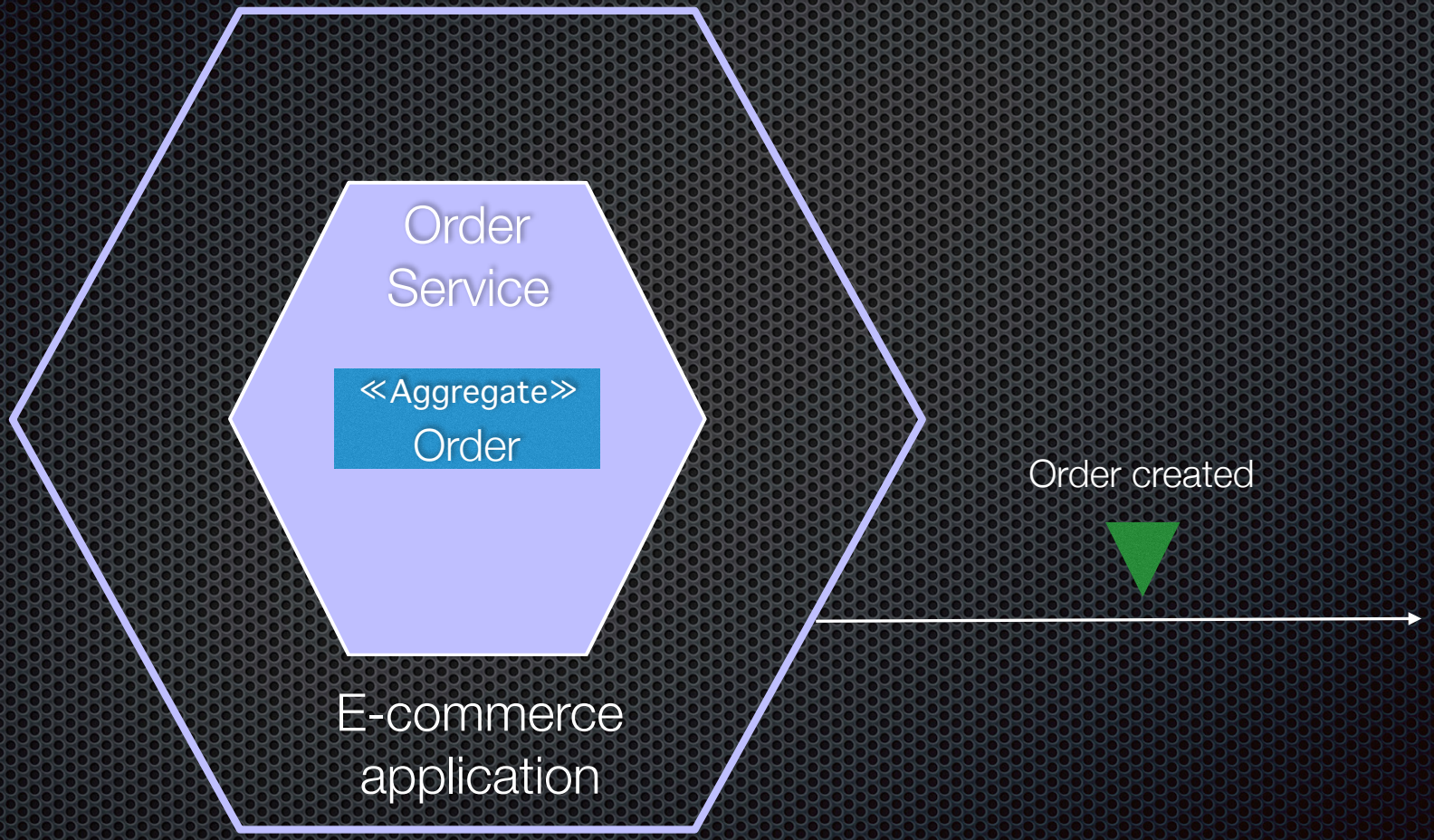
Order created

Summary: aggregates publish events

Summary: services publish events



Summary: applications publish events




@crichardson chris@chrisrichardson.net
Microservice Patterns
Chris Richardson
MEAP
MANNING
ctwsoftarchconf18
Questions?
http://learn.microservices.io