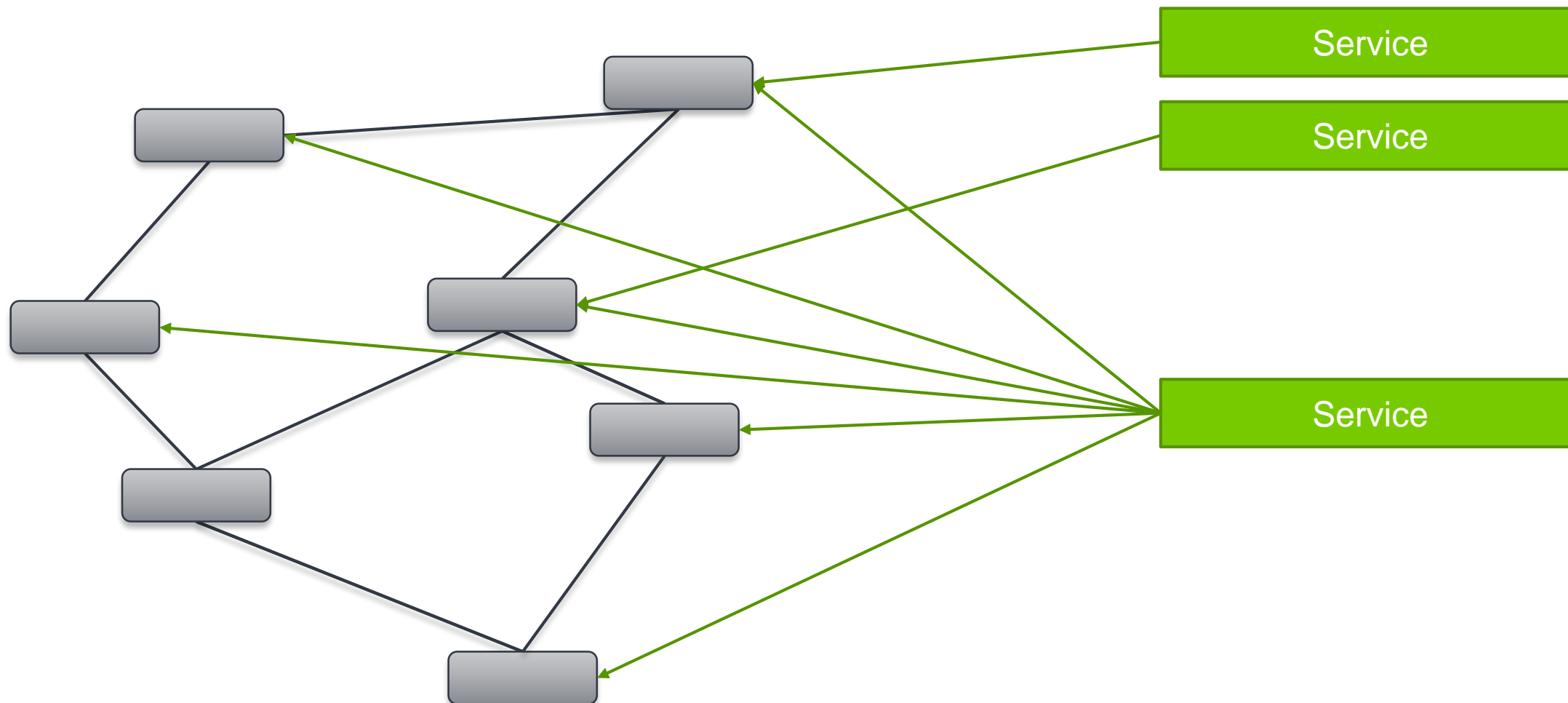


Pragmatic Event-Driven Microservices

Allard Buijze

@allardbz – allard@axoniq.io



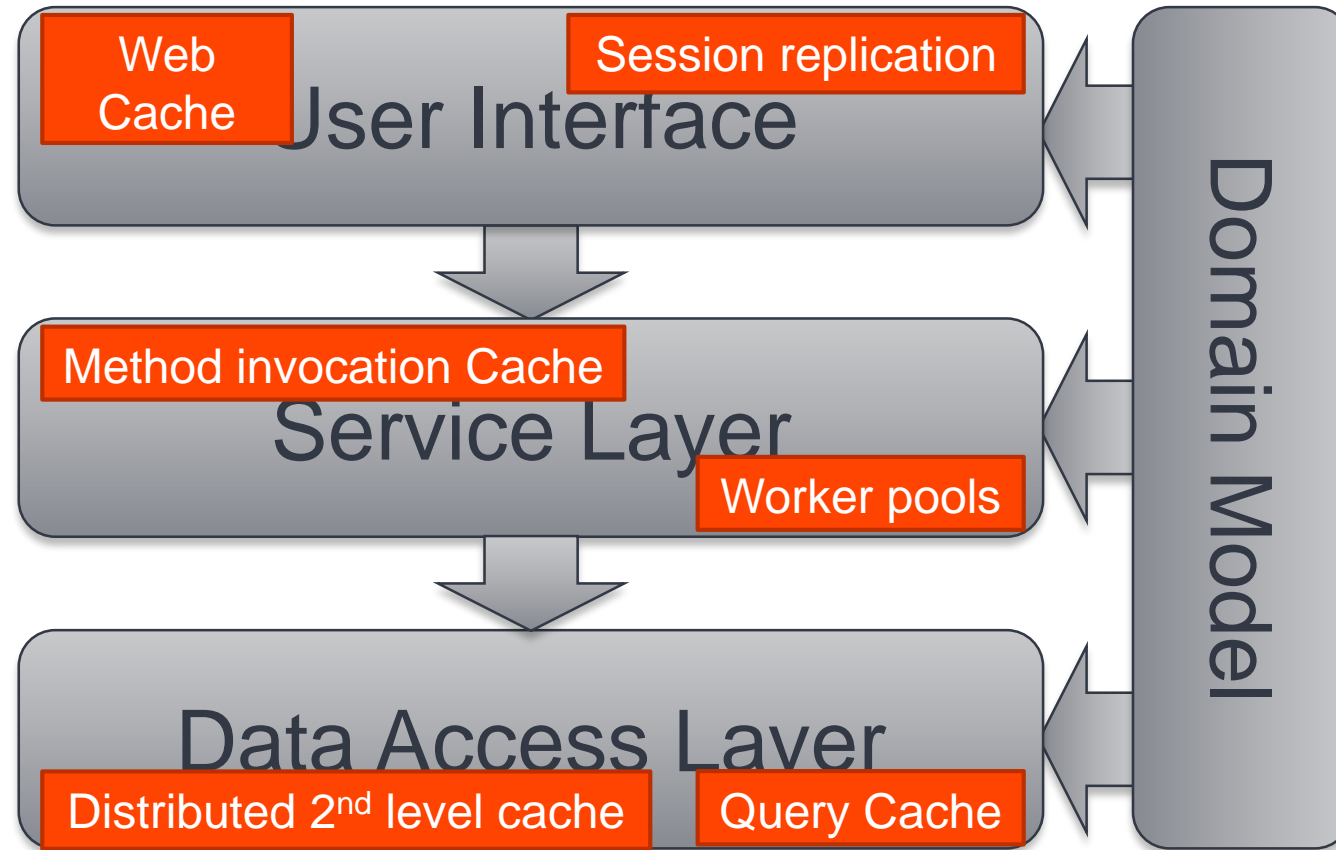
'Normal' SQL QUERY

22 JOINS
 6 SUBQUERIES

22 JOINS

6 SUBQUERIES

Layered architecture









Source: <http://www.sabisabi.com/images/DungBeetle-on-dung.JPG>

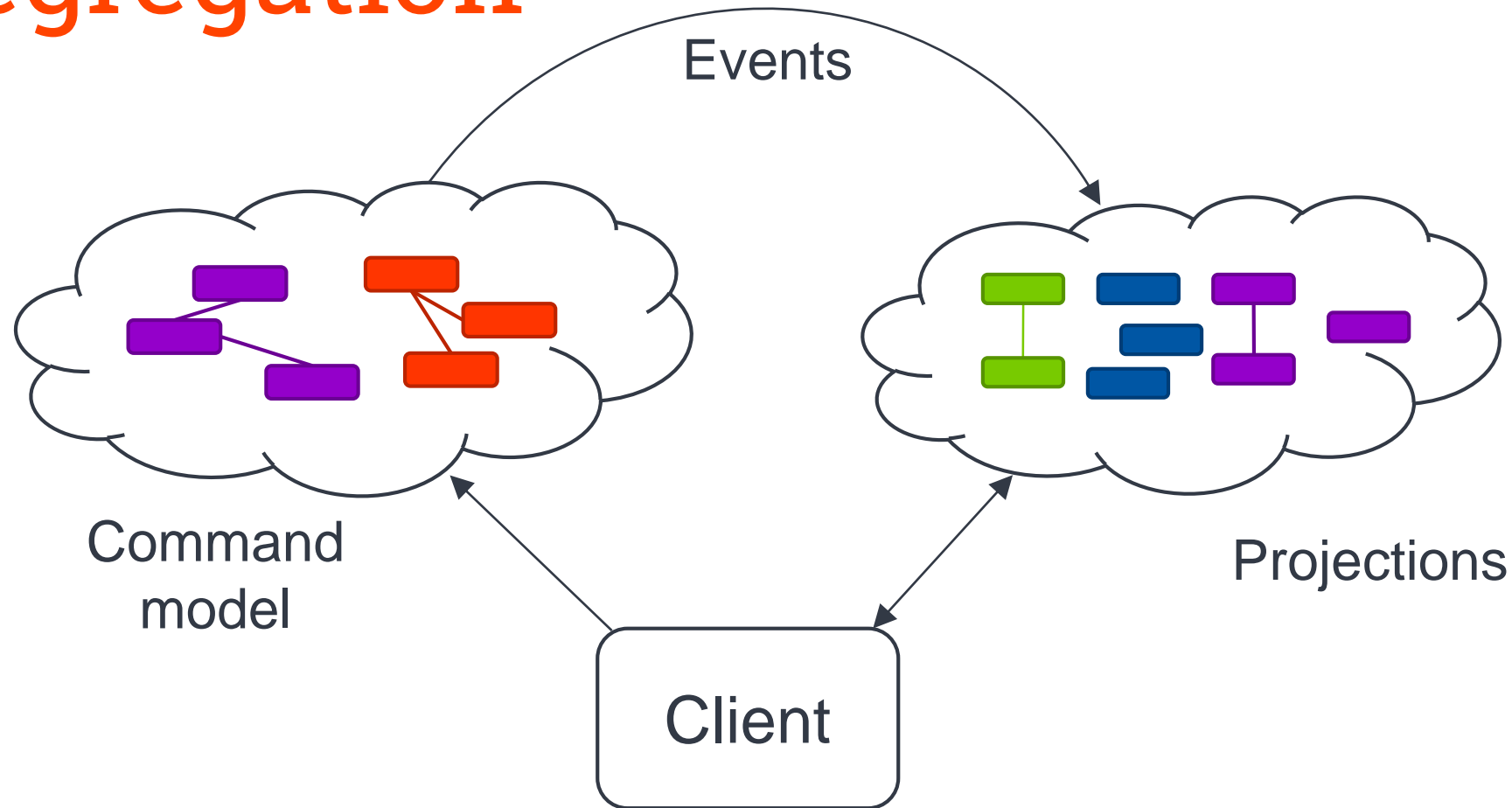


GitHub



AxonFramework

Command Query Responsibility Segregation



Monoliths



St Breock Downs Monolith - www.cornwalls.co.uk



Microservices vs Monoliths

Monoliths

Almost all the successful microservice stories have started with a monolith that got too big and was broken up

Microservices system

Almost all the cases where I've heard of a system that was built as a microservice system from scratch, it has ended up in serious trouble.

Martin Fowler

Are you tall enough?

You must be
this tall to use
microservices



Noun Driven Design

Noun? → Service!

Noun Driven Design

OrderService

Noun Driven Design

CustomerService

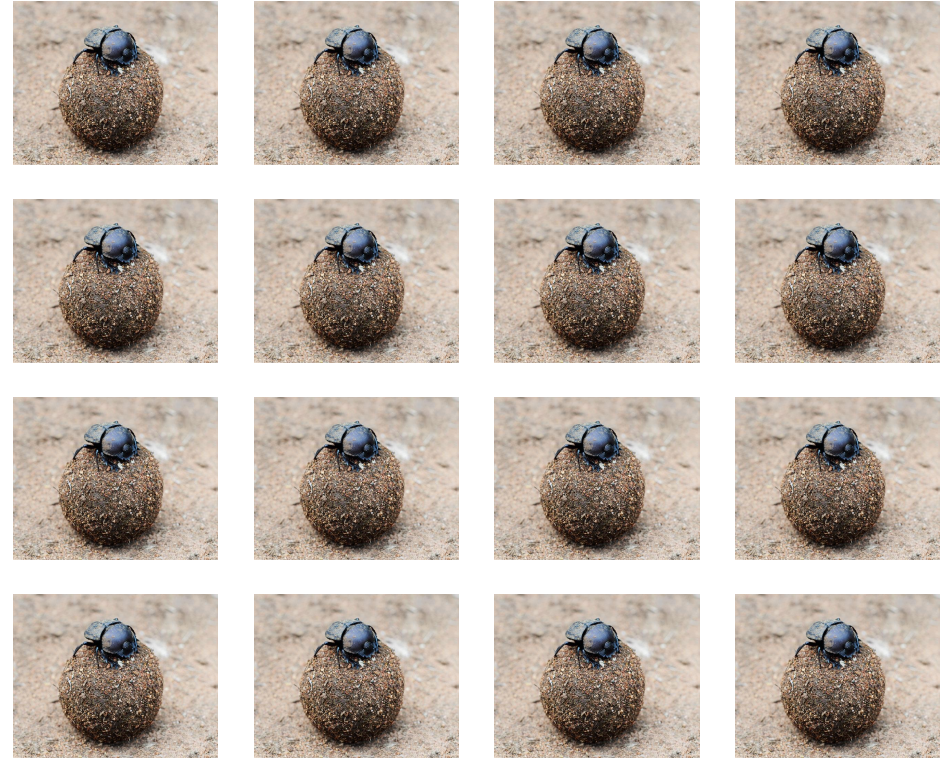
Noun Driven Design

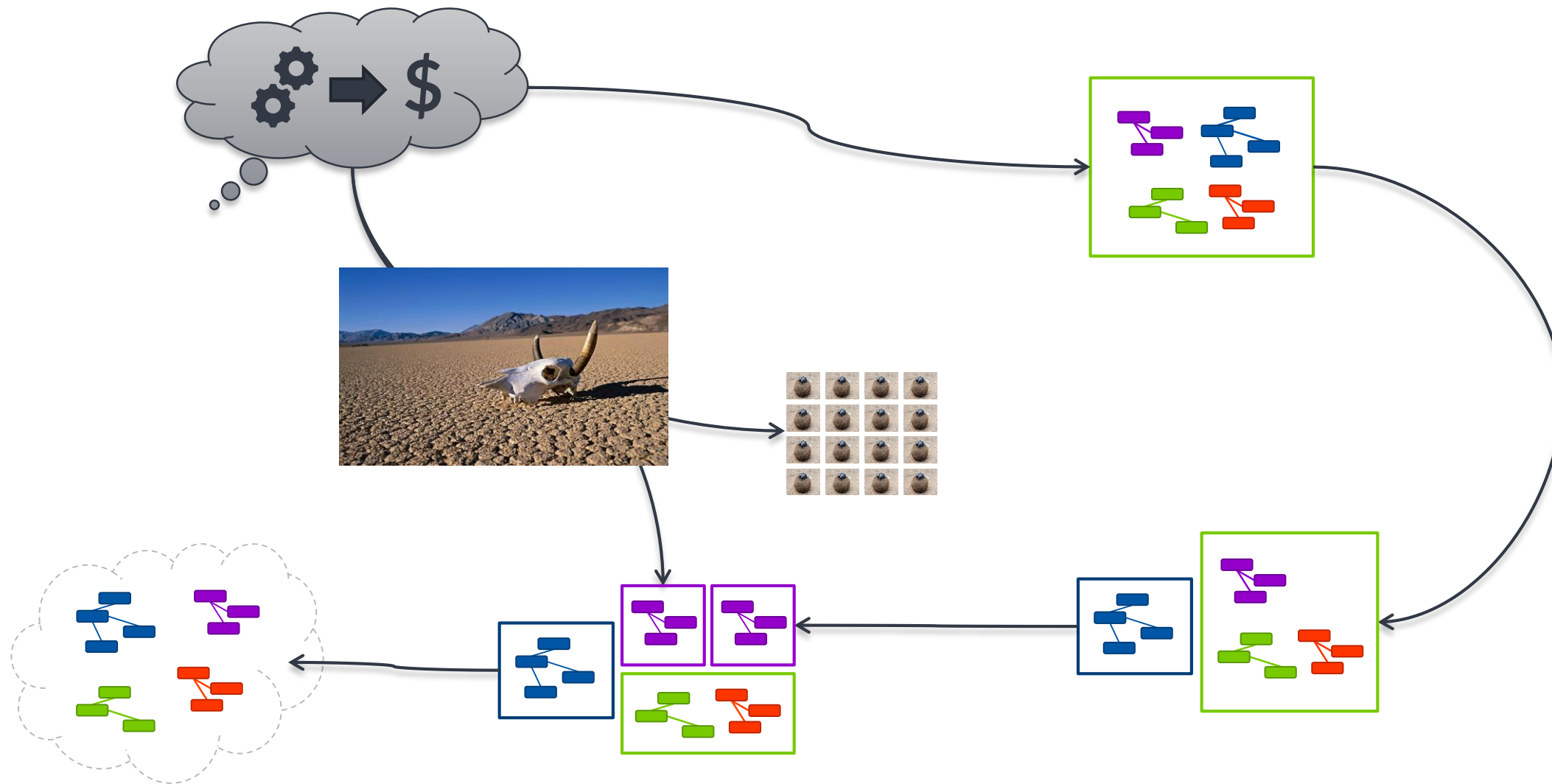
ProductService

Noun Driven Design

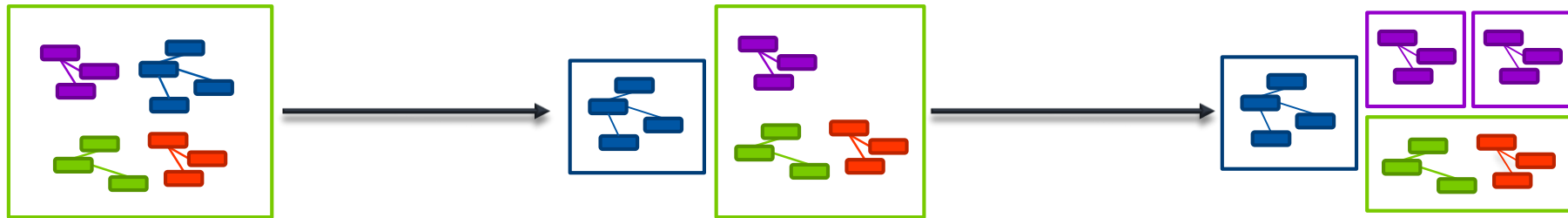
InventoryService

Noun Driven Design





Location transparency



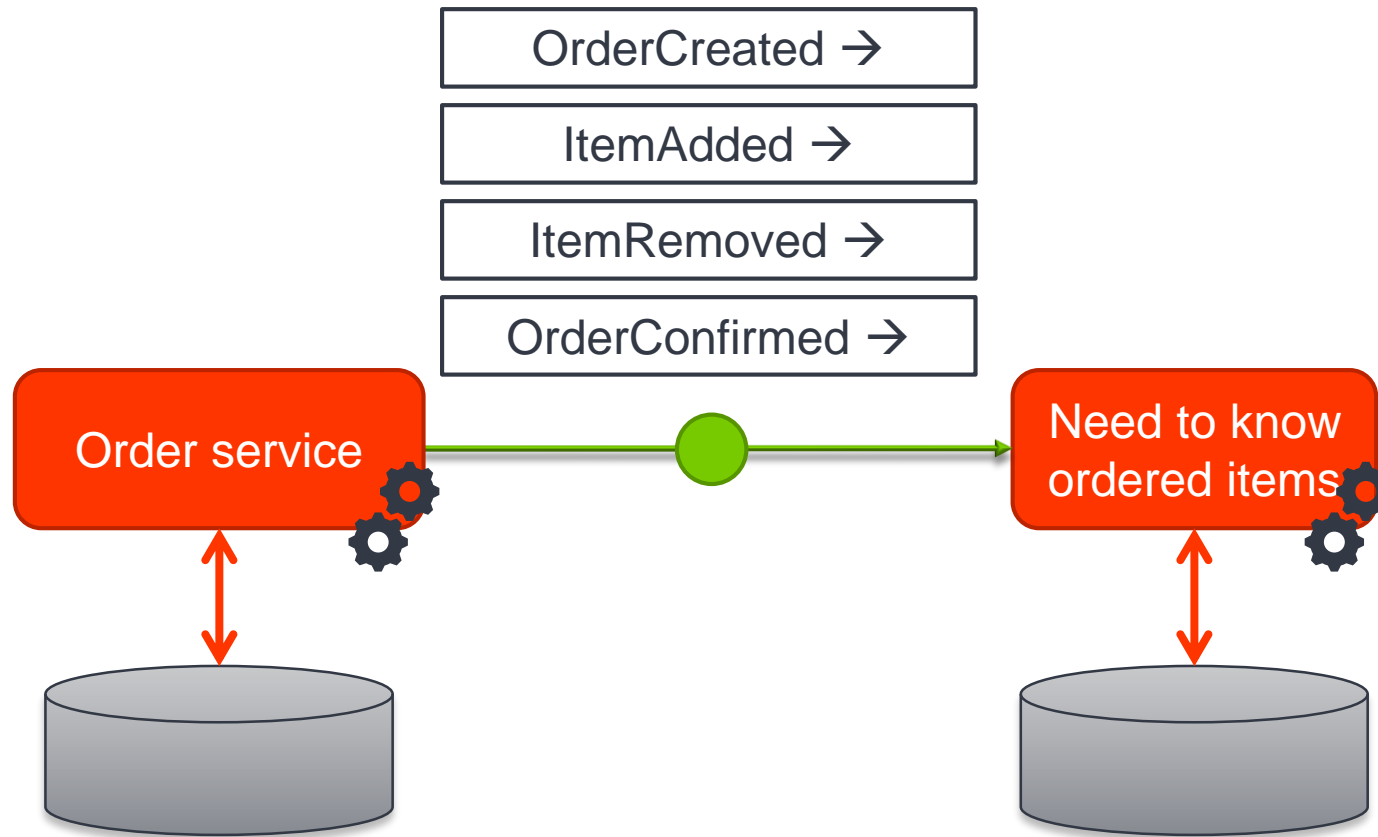
A component should neither be aware of nor make any assumptions about the location of components it interacts with.

Location transparency starts with good API design
(but doesn't end there)

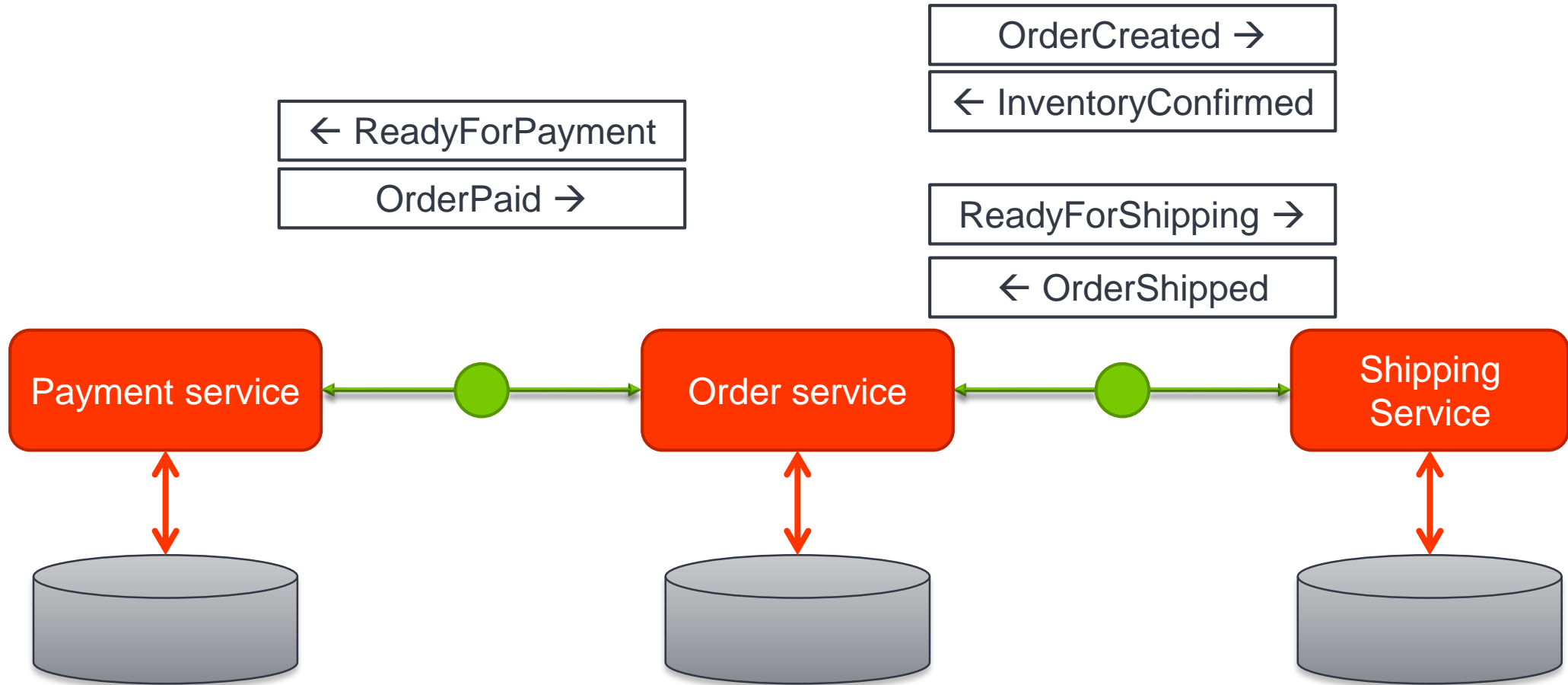
'Event-Driven' Microservices

- Event Something has happened

'Event-Driven' Microservices



Or worse...



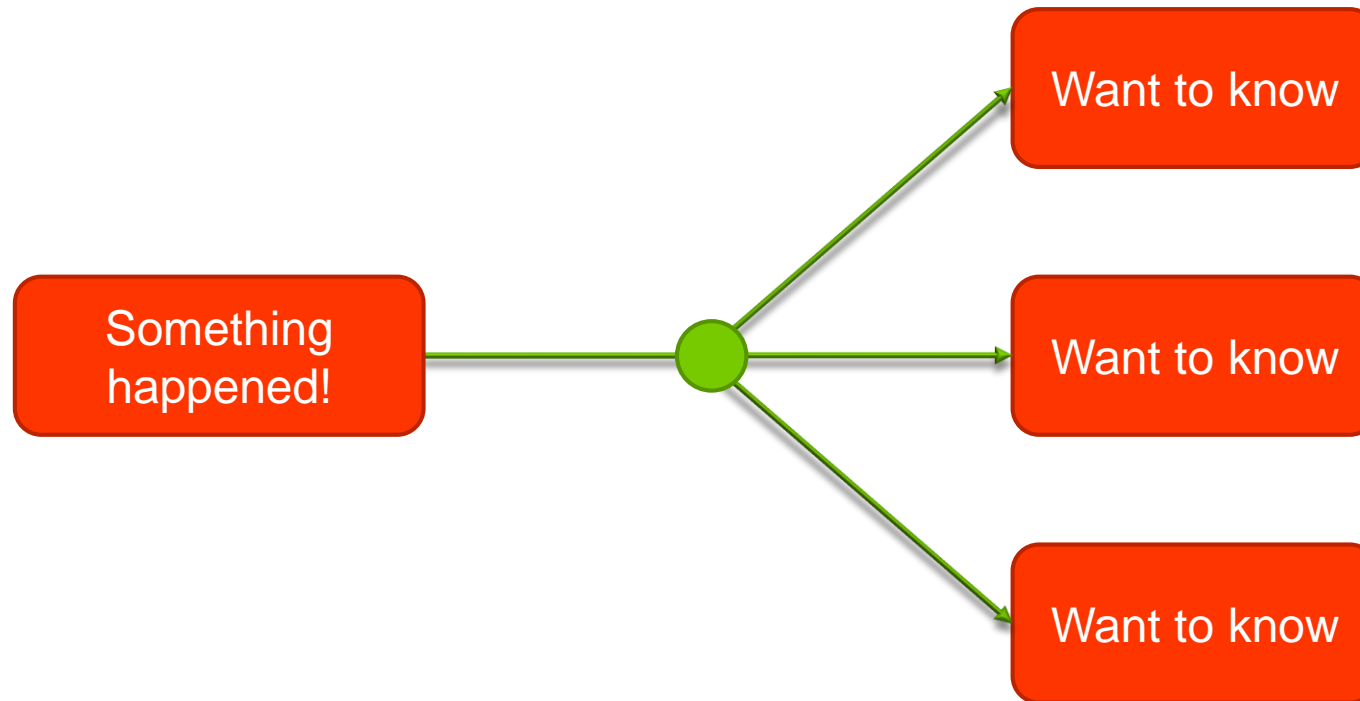
Reasons to send a message

- Event Something has happened
- Command I want the system to do something
- Query I want to know something

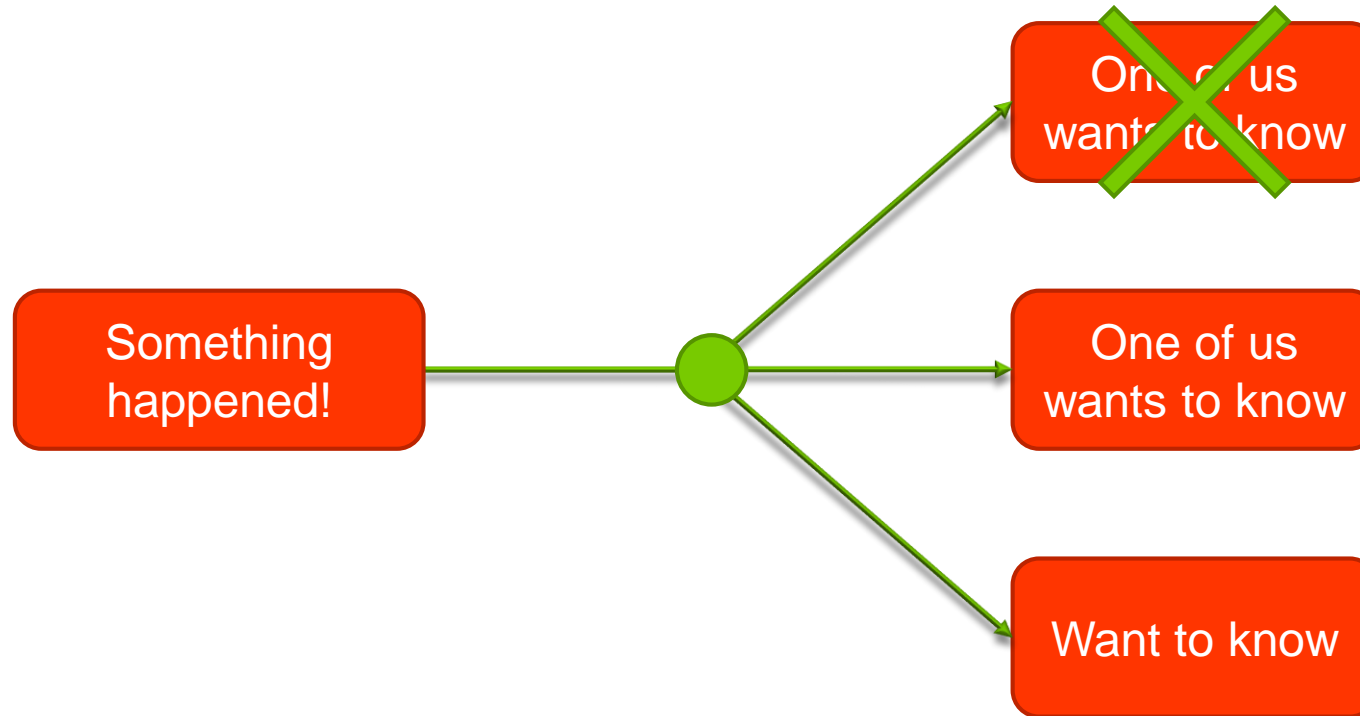
Something has happened - Event

- Data change
- Deadline passing
- Or anything else that's relevant in the domain

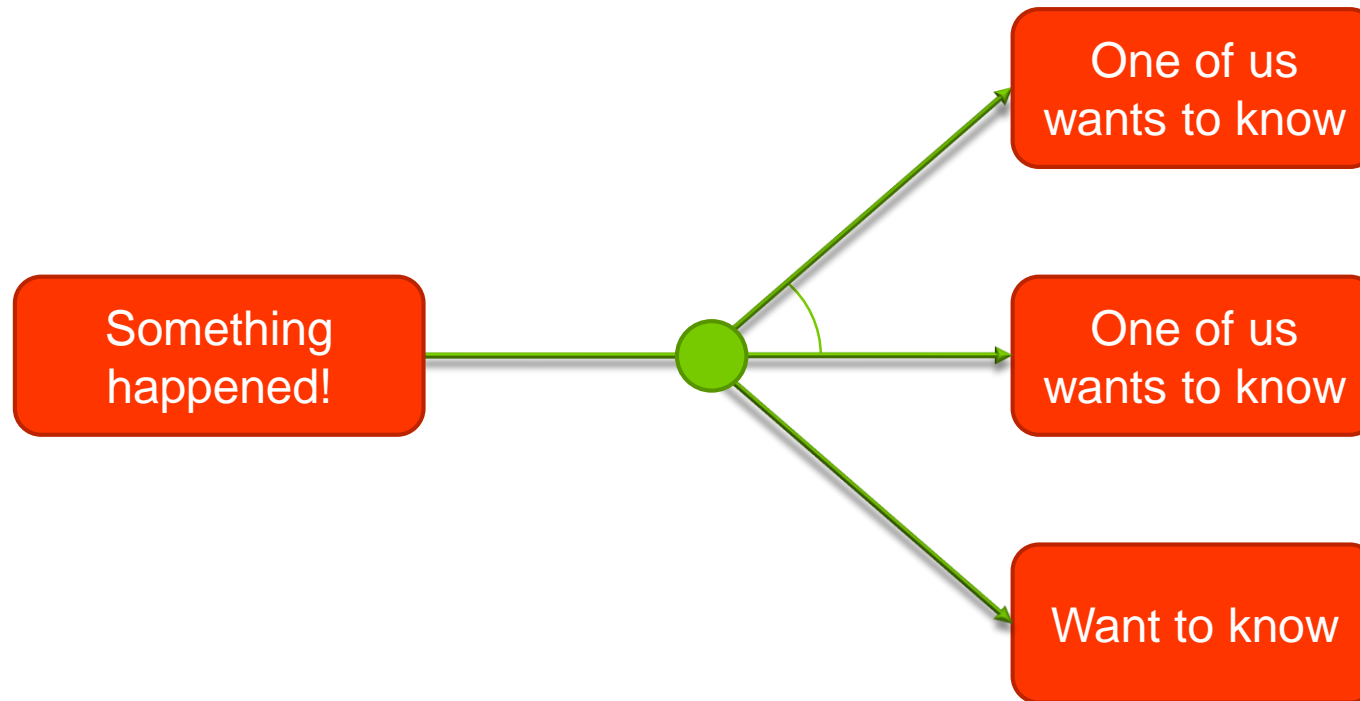
Publish-subscribe



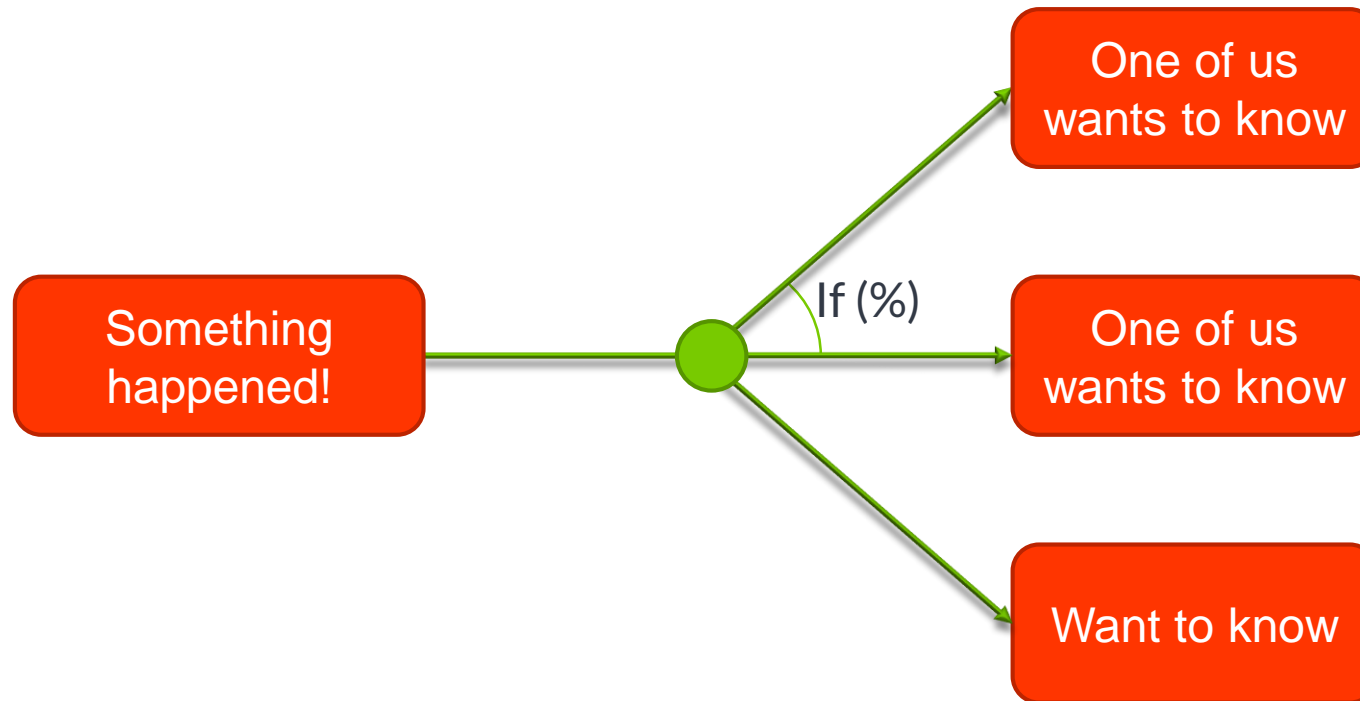
Exclusive consumers



Competing consumers



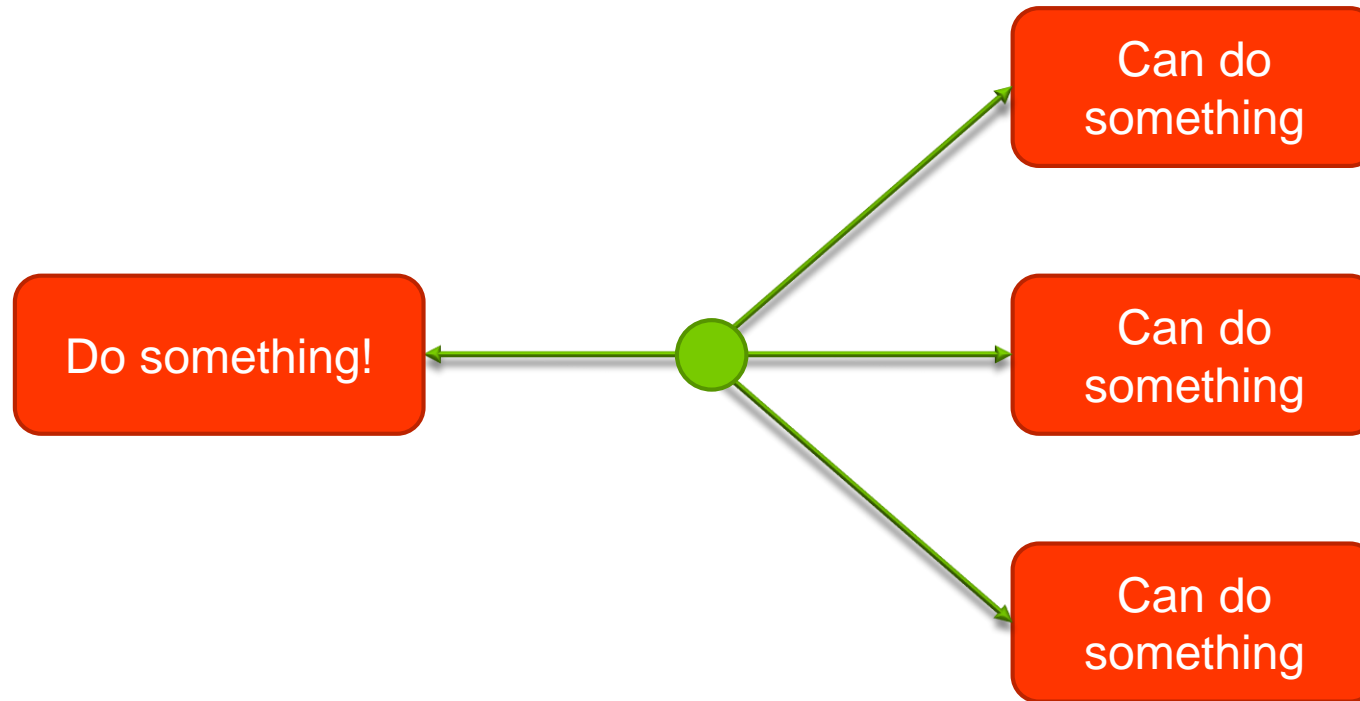
Balanced consumers



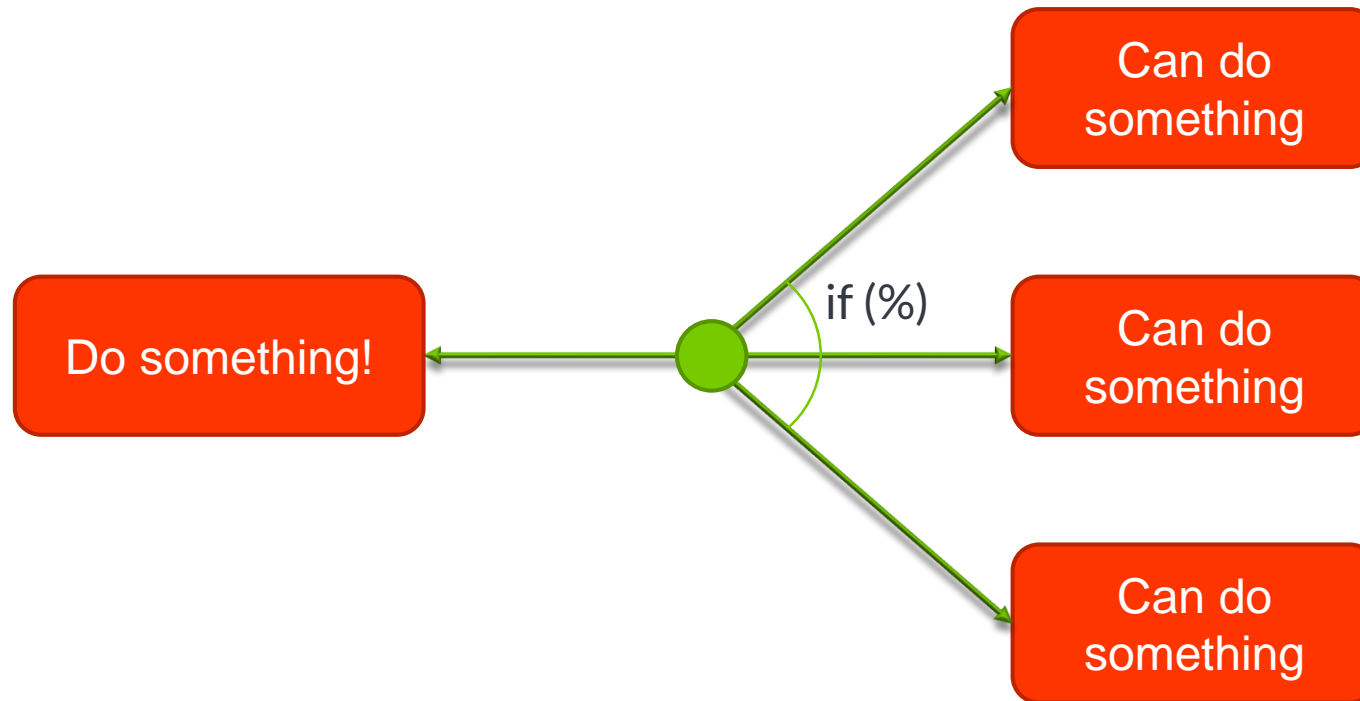
I want something done - Command

- Request-a-side-effect
 - Change data / application state
 - Send email
- Exactly 1 destination
- OK / NOK reply
 - Maybe some data

Command Routing



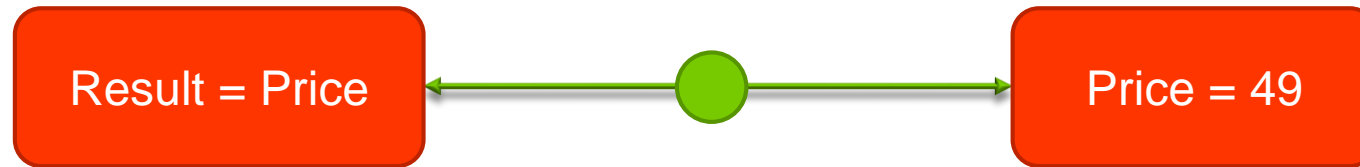
Command Routing



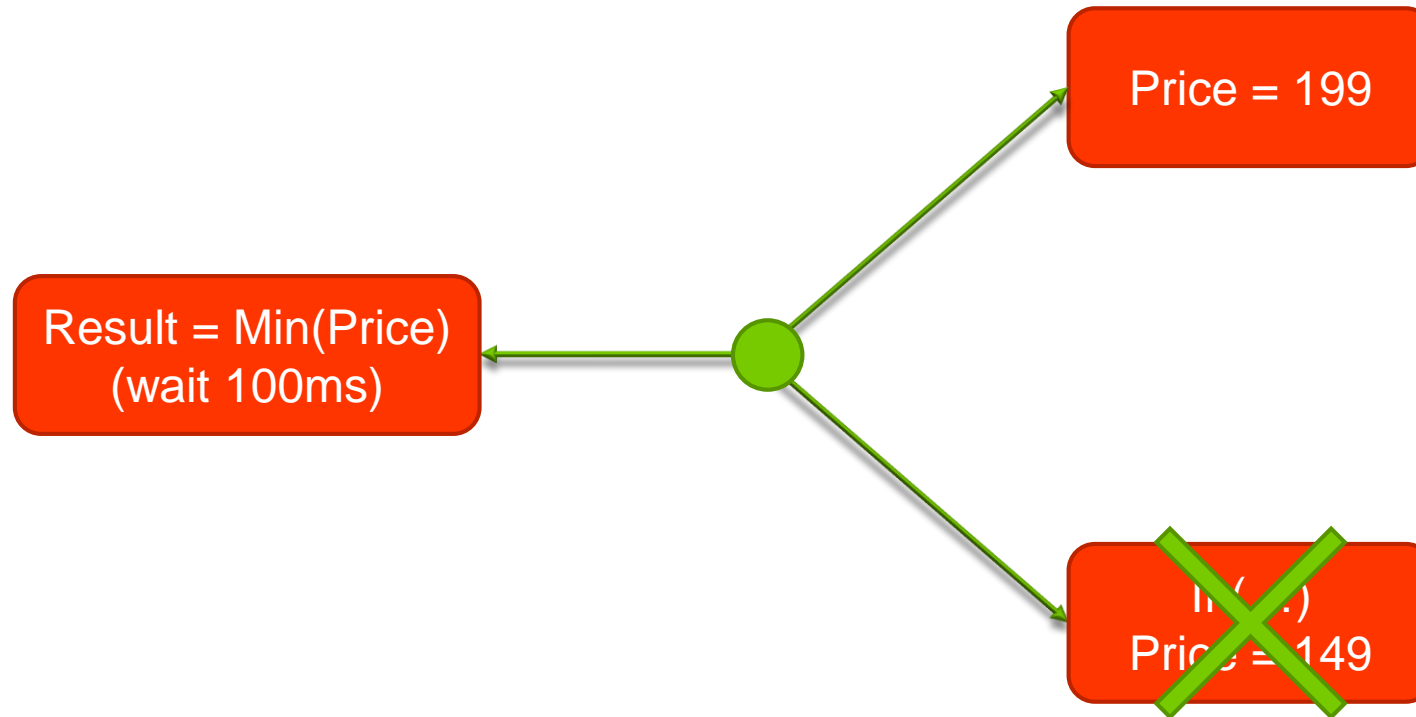
I want to know something - Query

- Desire for information
 - The response has more value than the question
 - (Usually) side-effect free
- Different messaging patterns
 - Single destination
 - Scatter – gather query
 - Subscription

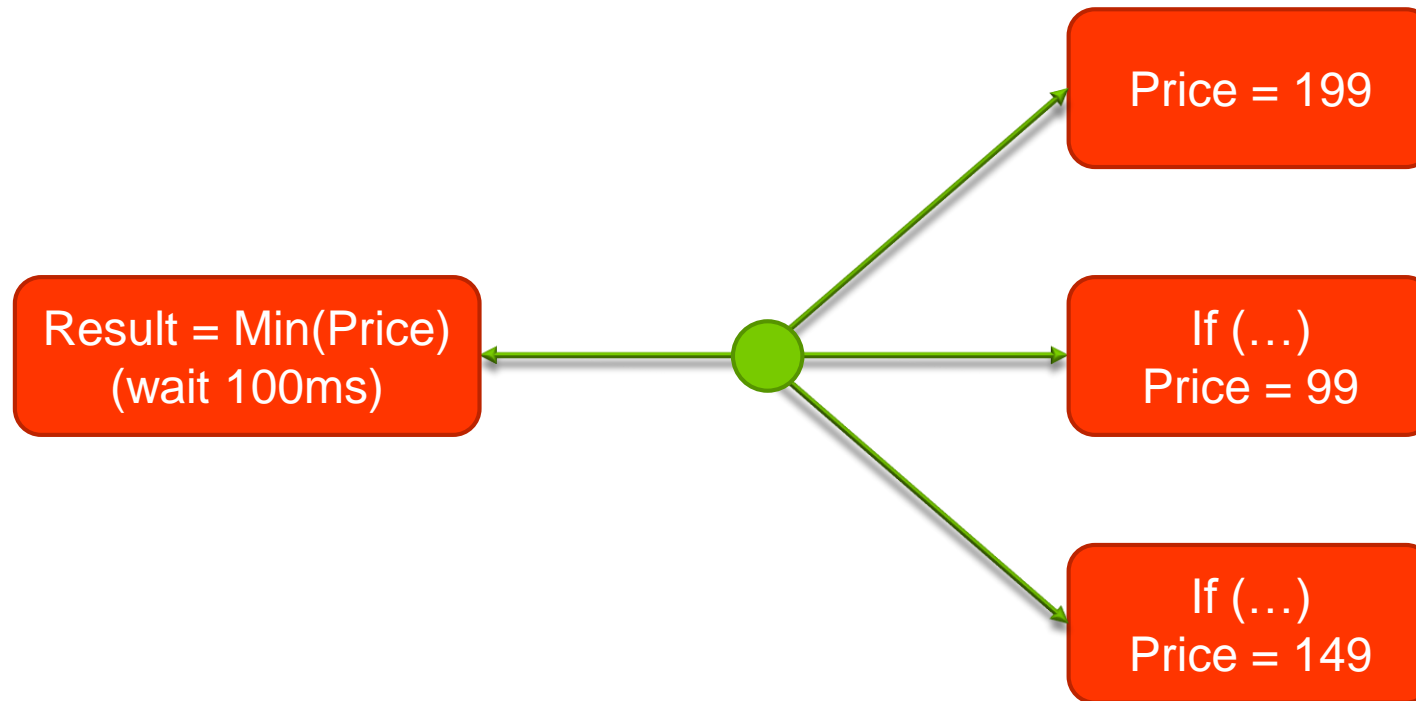
Query – point to point



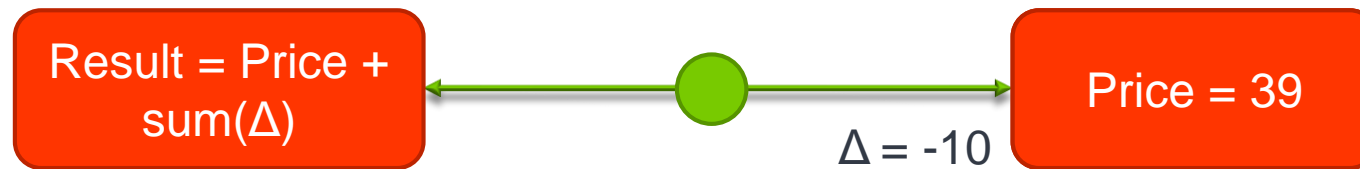
Query – scatter-gather



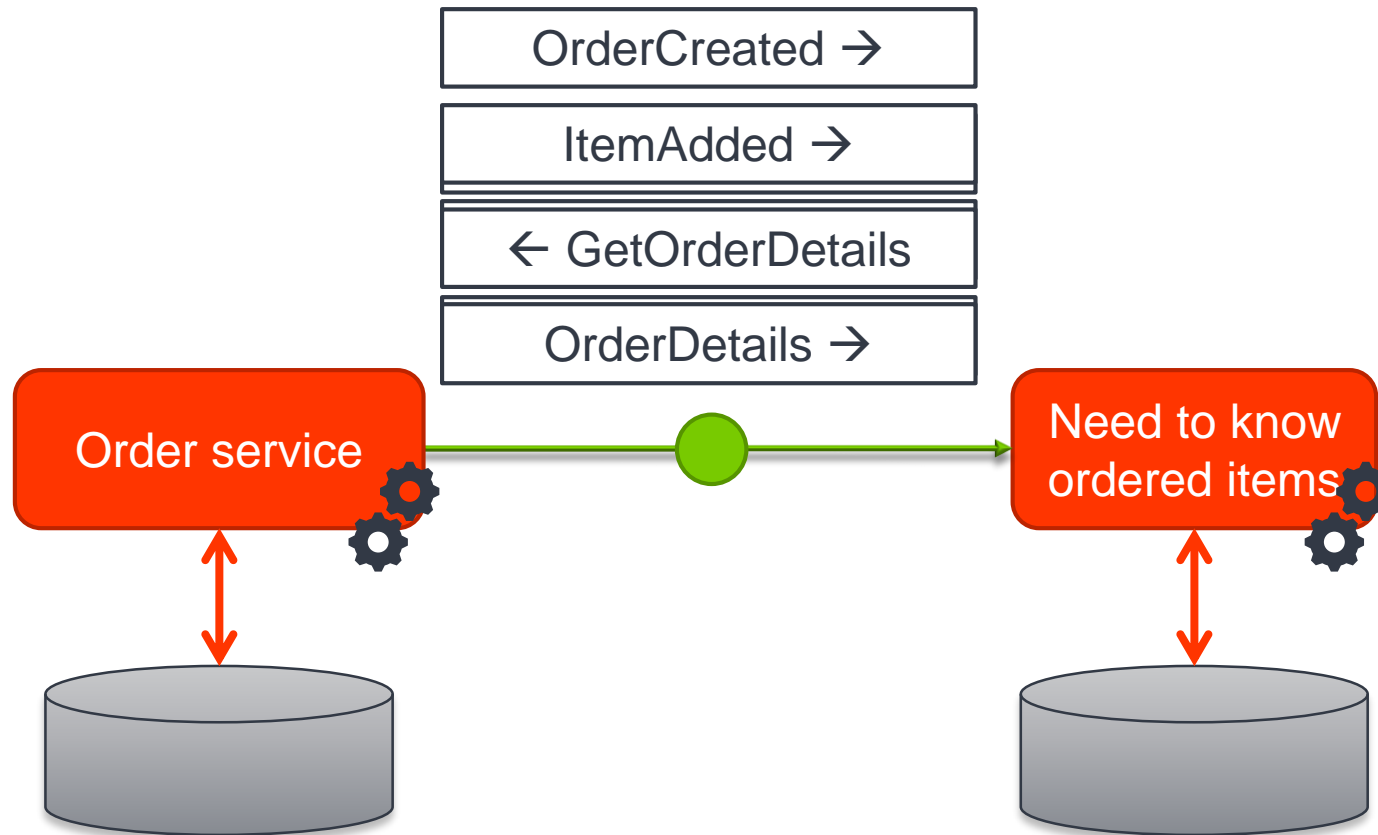
Query – scatter-gather



Query – subscription



'Event-Driven' Microservices



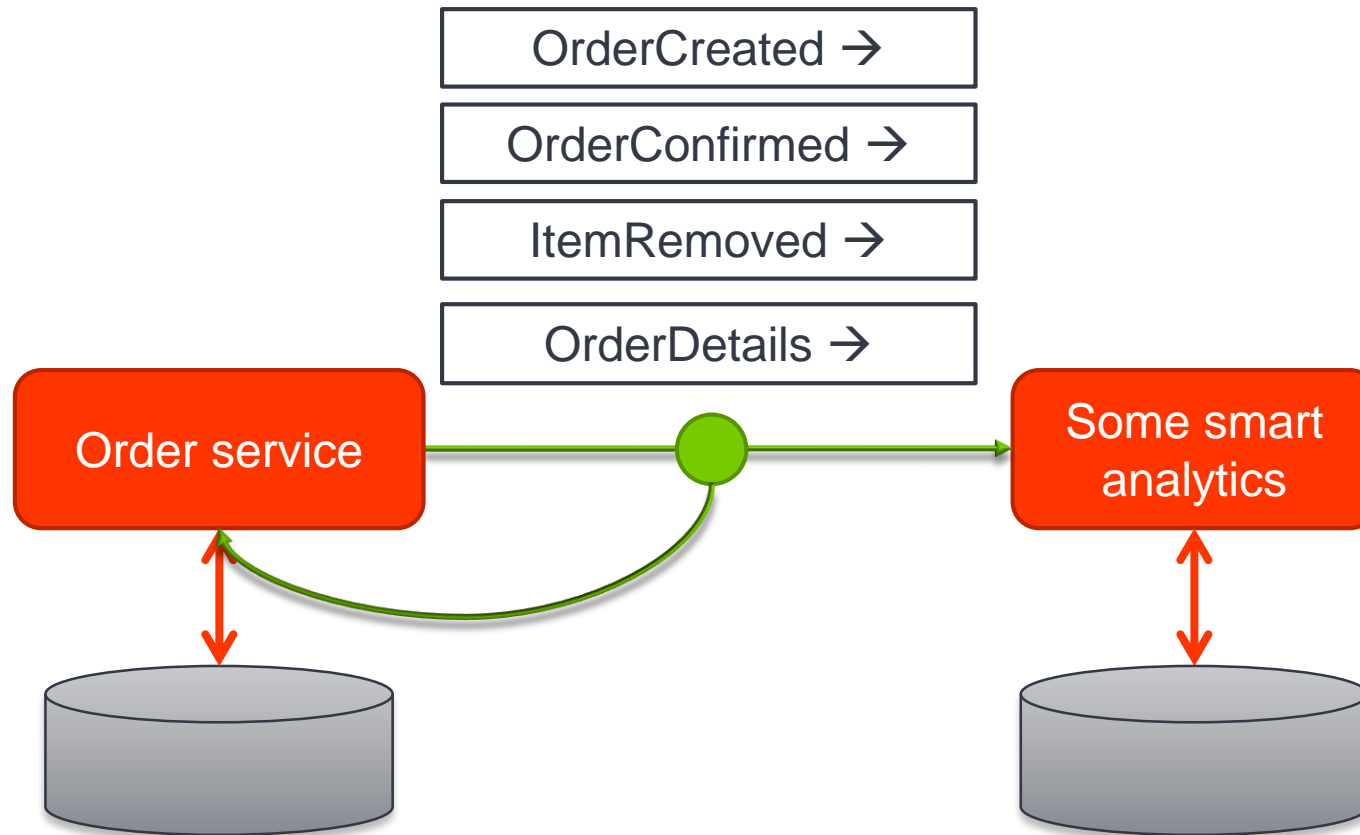
Events retain value

Event Sourcing is an Architectural pattern in which Events are considered the “source of truth”, based on which components (re)build their internal state.

Event Store

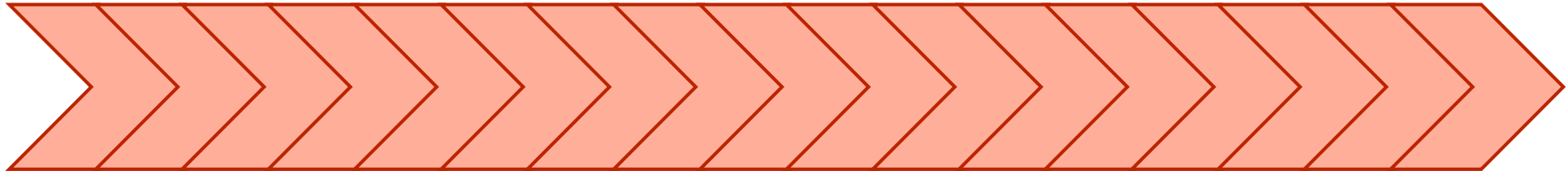
An Event Store stores the published events to be retrieved both by consumers as well as the publishing component itself.

Event Sourcing



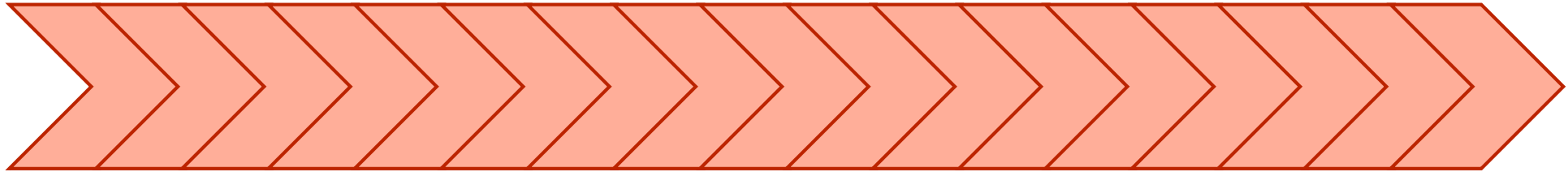
Event Store operations

- Append
- Validate 'sequence'



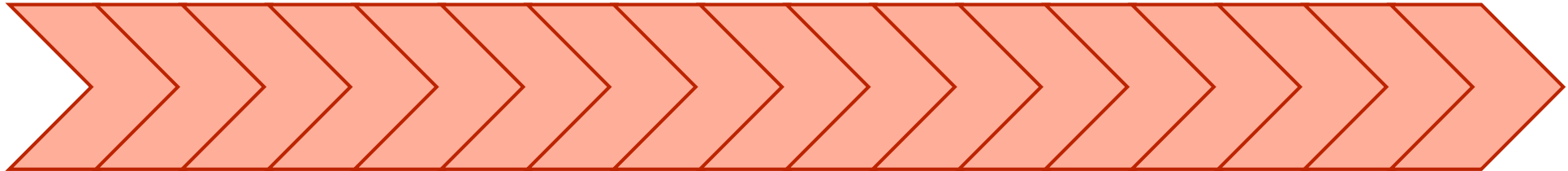
Event Store operations

- Full sequential read



Event Store operations

- Read aggregate's events



1. Define which routing patterns to apply
2. Choose technology/protocol accordingly



AxonDB

AxonHub



HTTP

HTTP/2



At scale, different rules apply

How do you route all these events to all components?

How will this scale?

You Don't!

It Won't!

Unmanageable mess

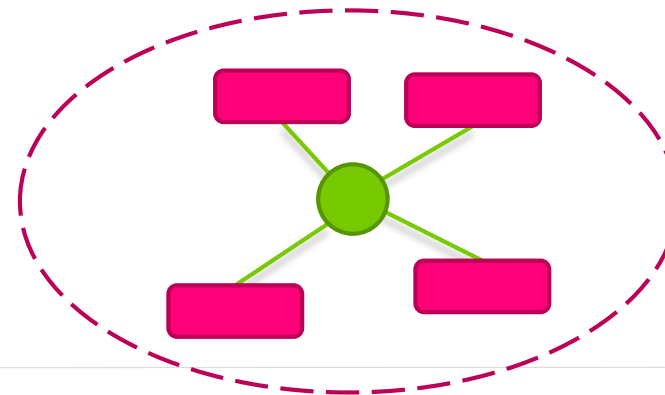
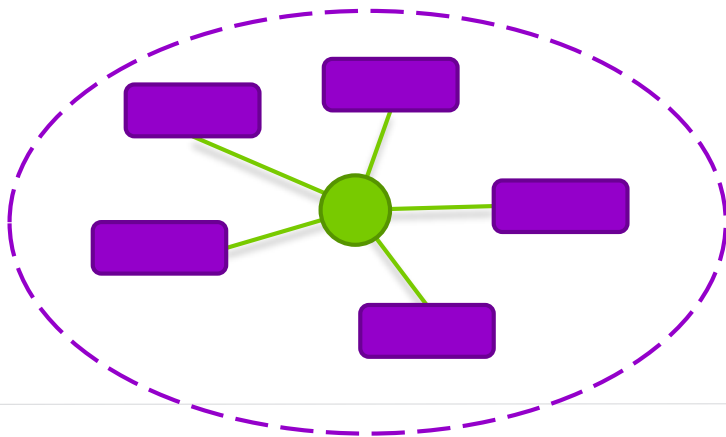
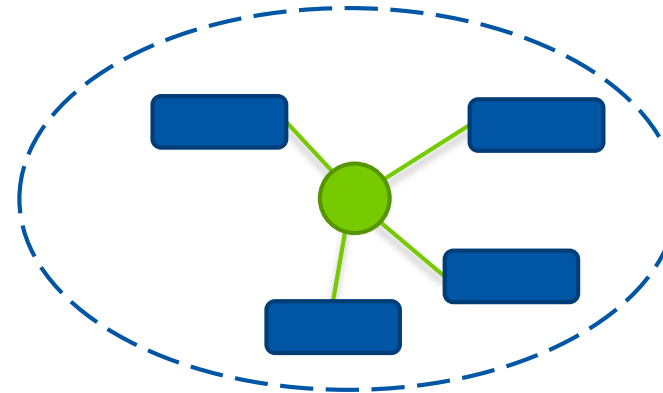
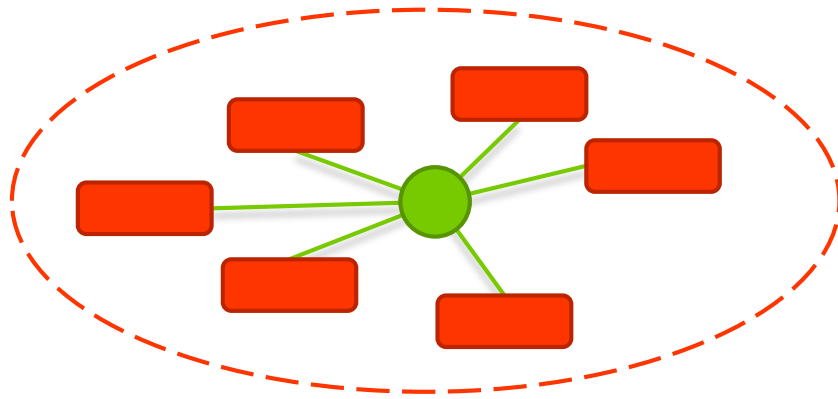


Bounded context

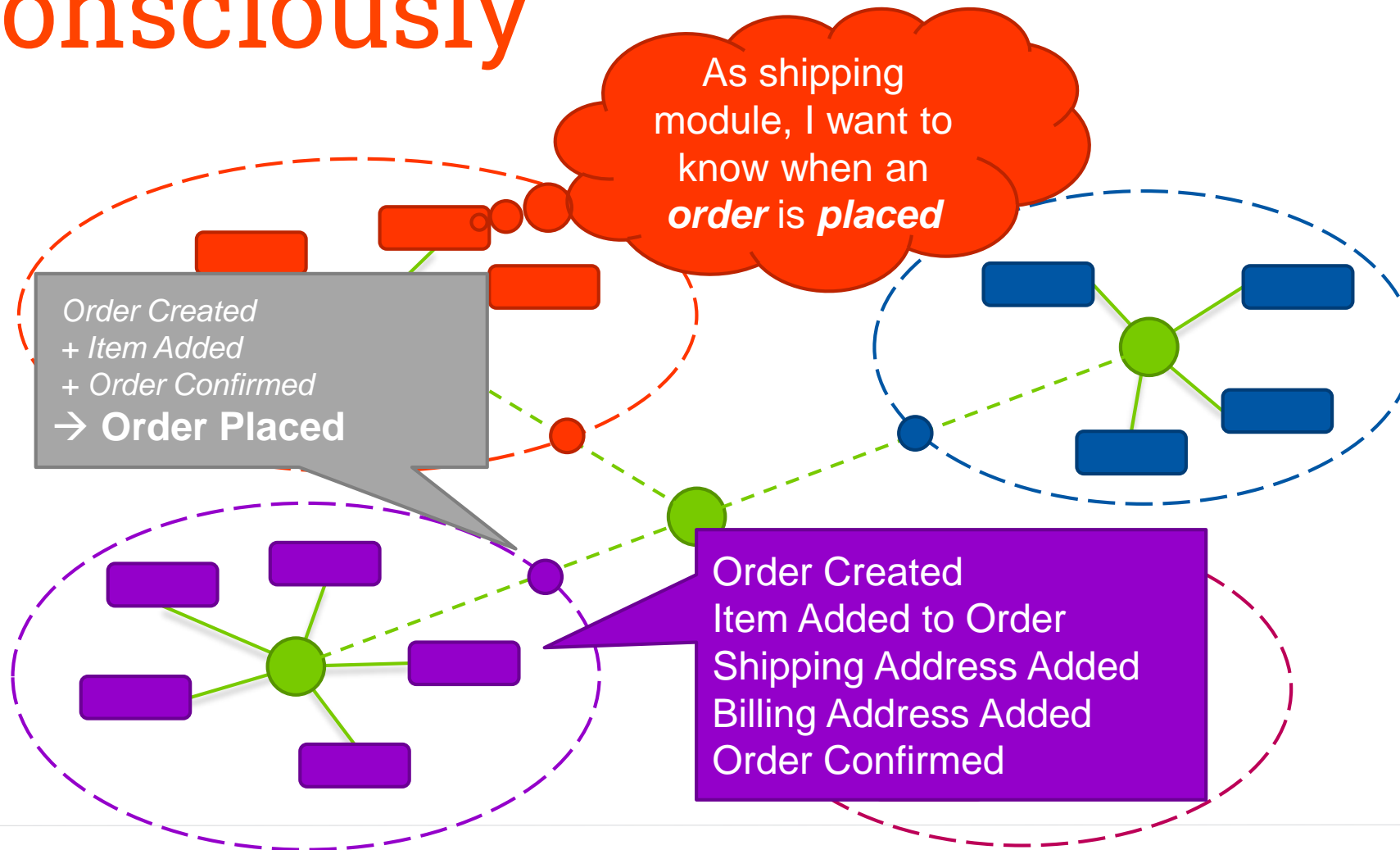
Explicitly define the context within which a model applies.

Explicitly set boundaries in terms of team organization, usage within specific parts of the application, and physical manifestations such as code bases and database schemas. Keep the model strictly consistent within these bounds, but don't be distracted or confused by issues outside.

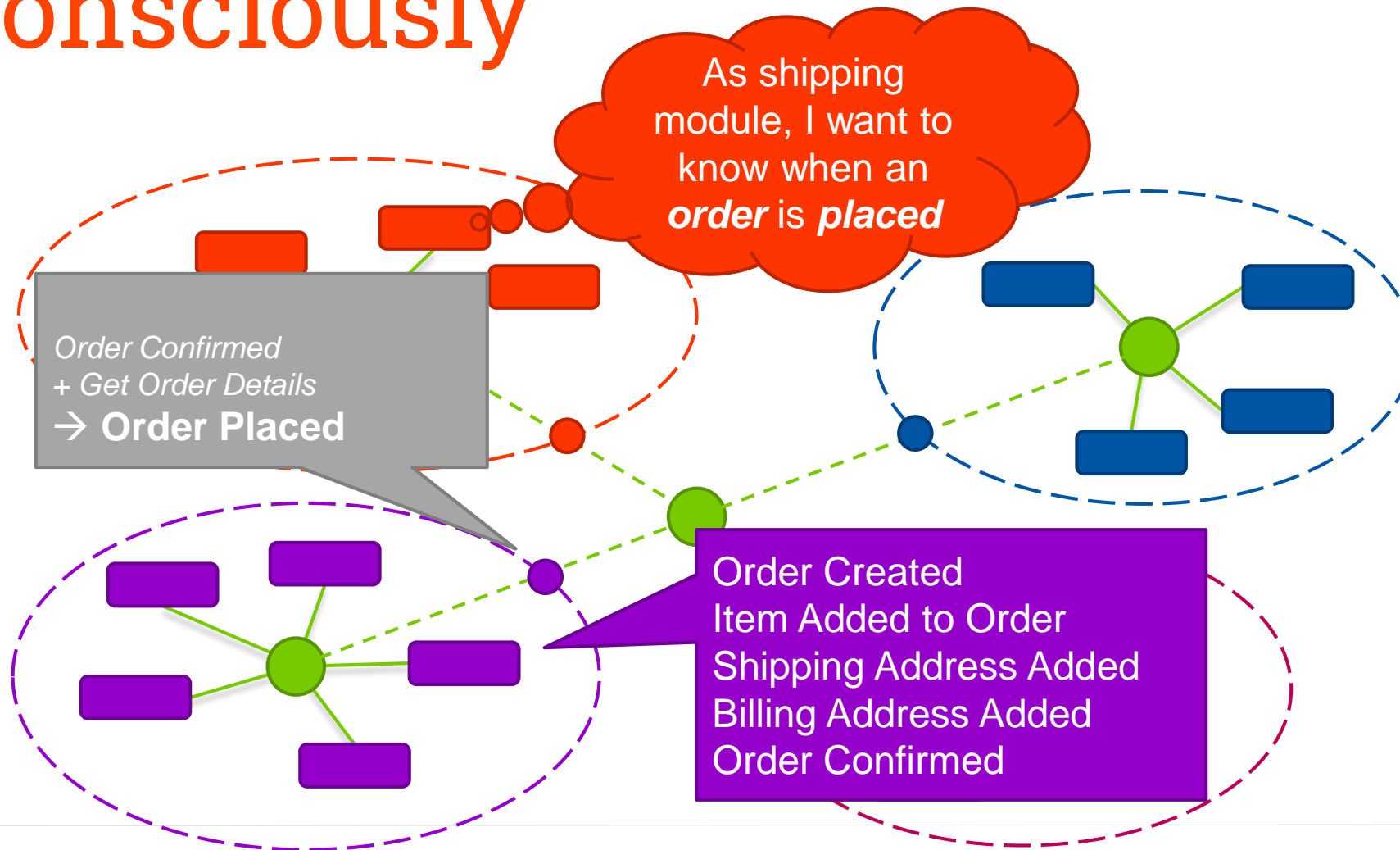
Within a context, share 'everything'



Between contexts, share 'consciously'



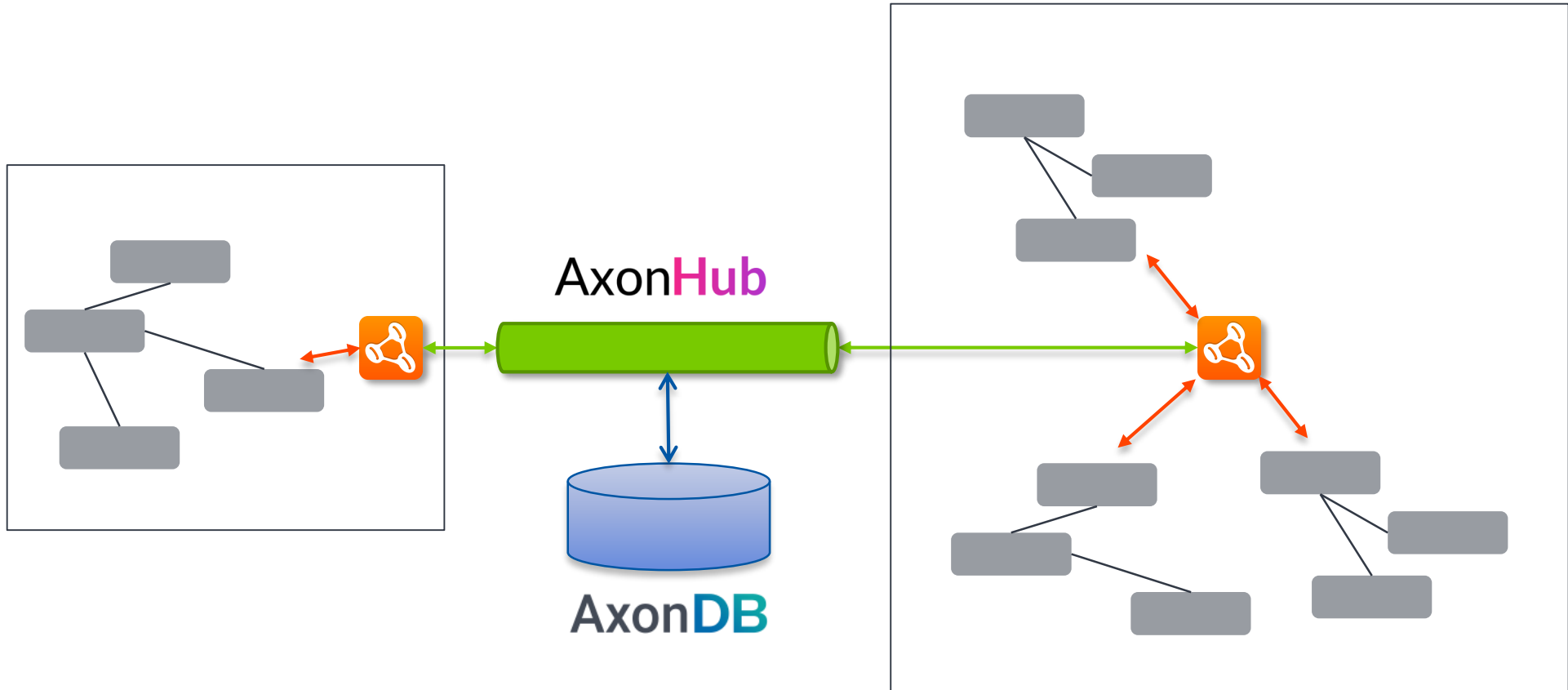
Between contexts, share 'consciously'



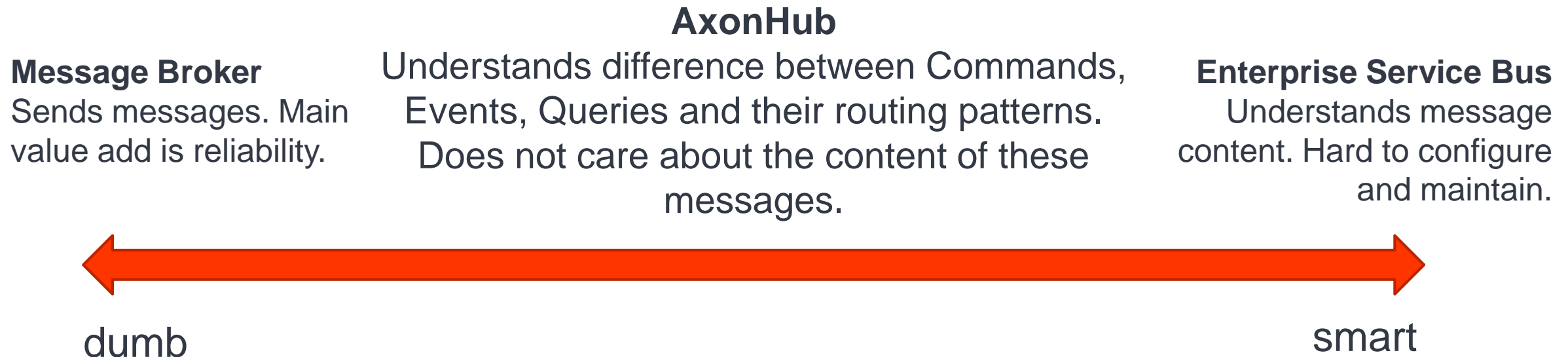
Where does AxonFramework fit?

- Inside each component in bounded context
- Axon provides the Java APIs towards platform
 - EventBus, CommandBus, QueryBus
- Separation of business logic and infrastructure logic

Microservices Messaging



“Just enough” intelligence



Our mission

Provide the APIs and implementations necessary for event-driven microservices to cooperate harmoniously, allowing each of them to focus on the business logic.

