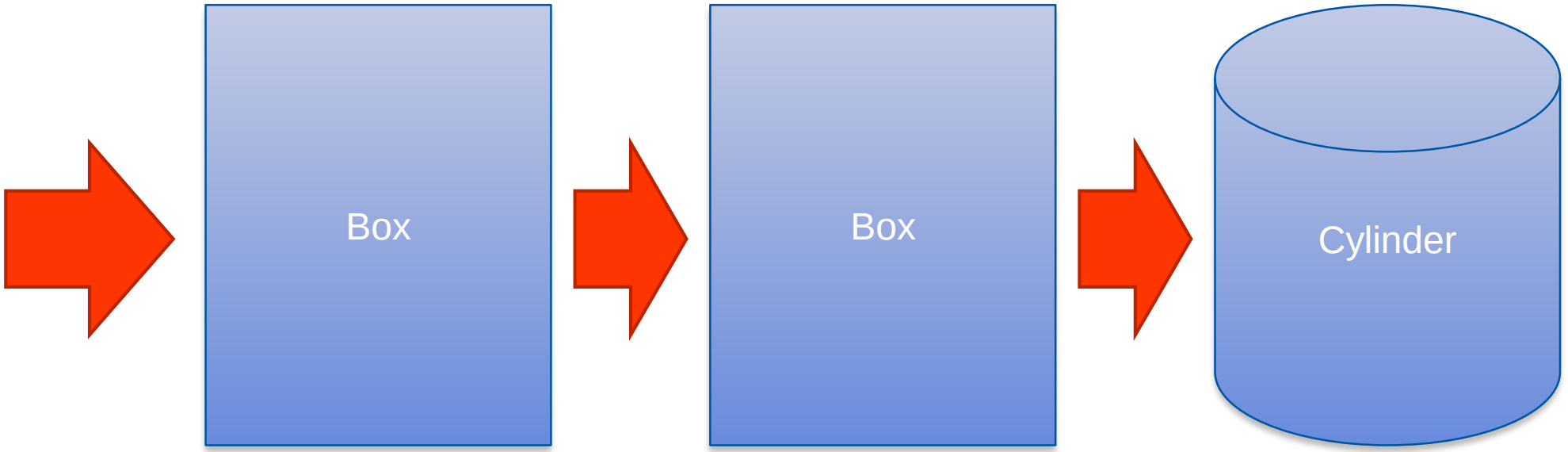
Event Driven Microservices

The sense, the non-sense and a way forward

Once upon a time...

“Universal ‘BBC’ architecture”



Source: Ted Neward

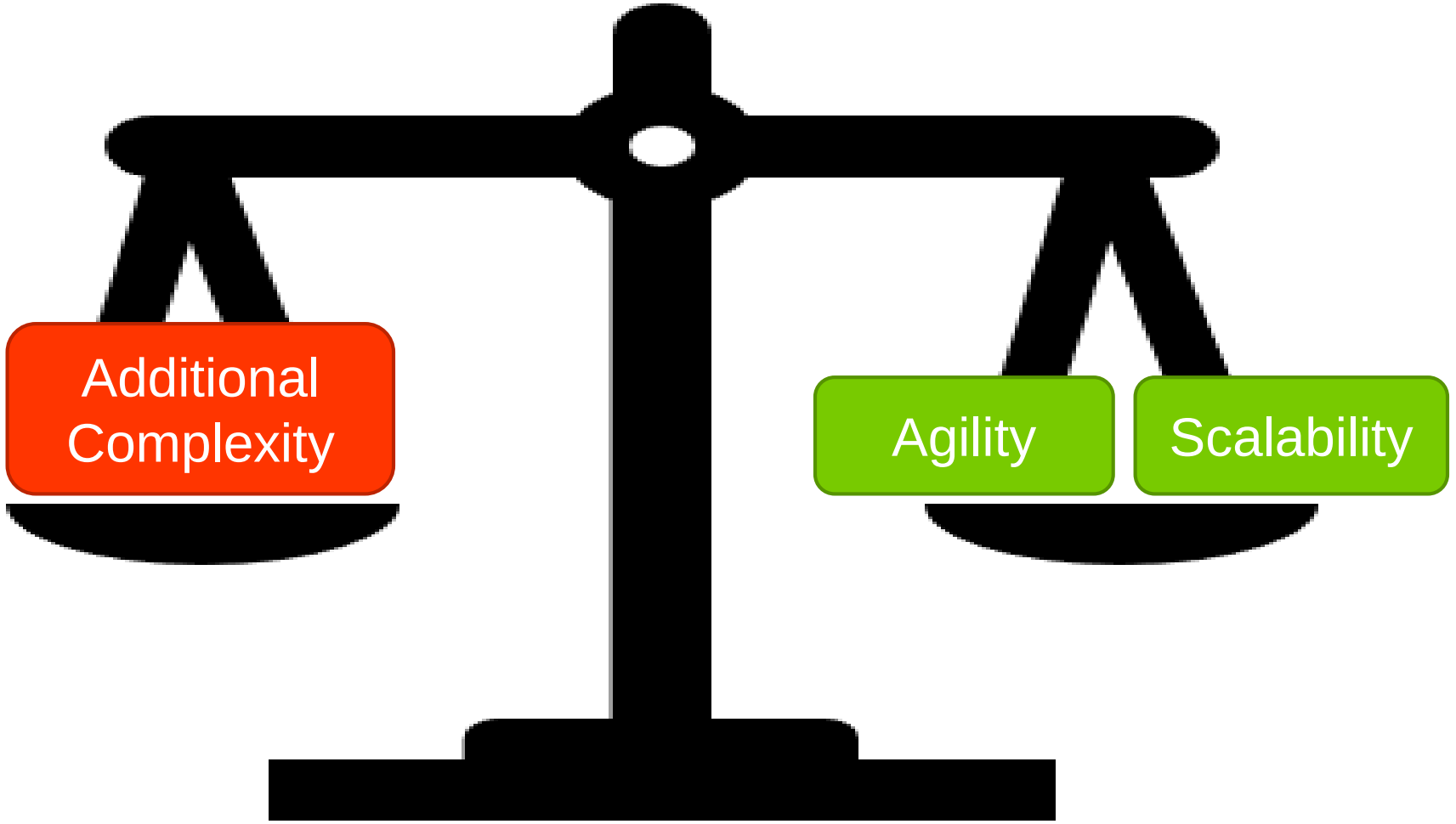
Source: http://www.sabisabi.com/images/DungBeetle-on-dung.JPG

Microservices to the rescue!

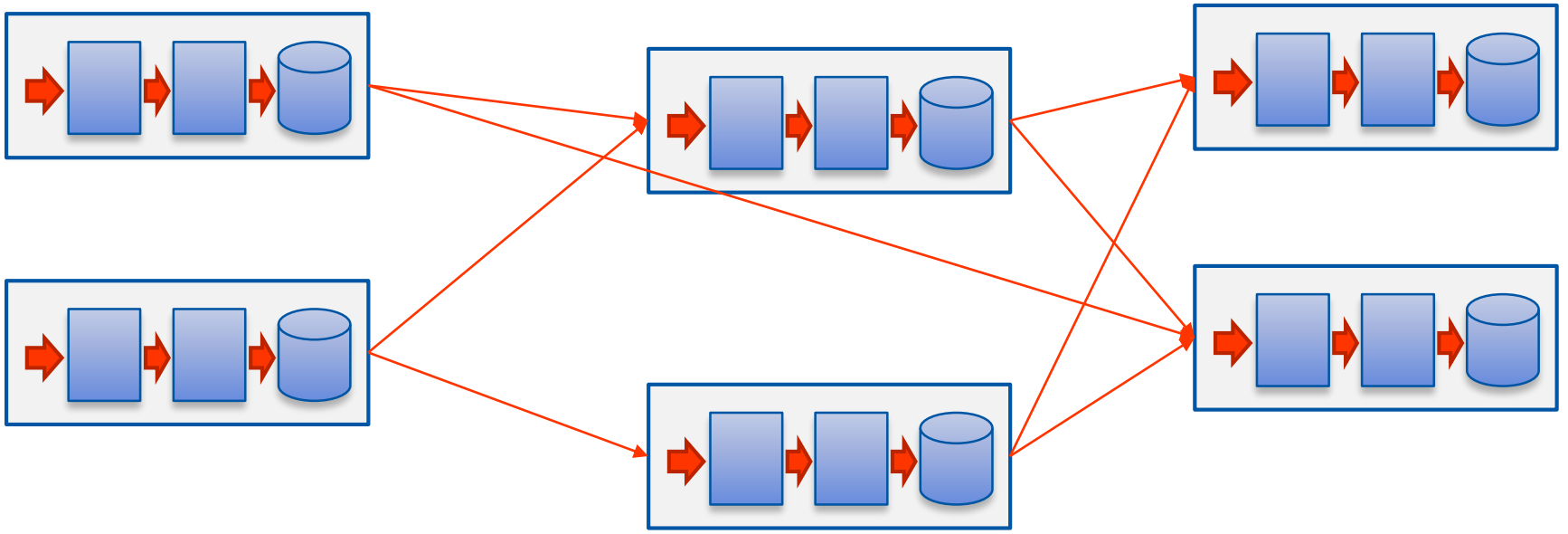
Why microservices?

Agility

(Team) Scalabilty


Additional Complexity
Agility
Scalability

“Universal Microservices architecture”



Noun Driven Design

Noun? → Service!

Noun Driven Design

OrderService

Noun Driven Design

CustomerService

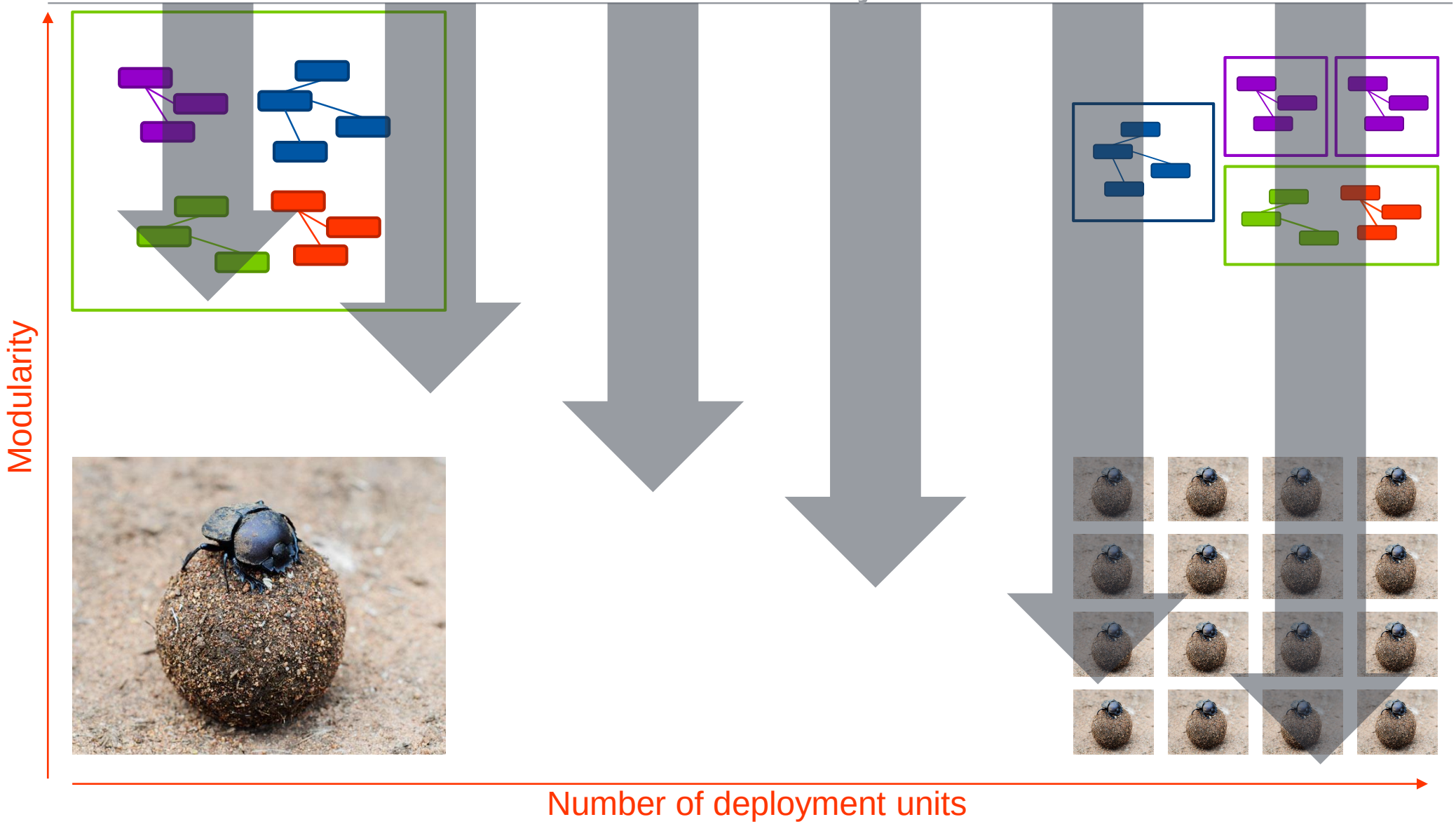
Noun Driven Design

ProductService

Noun Driven Design

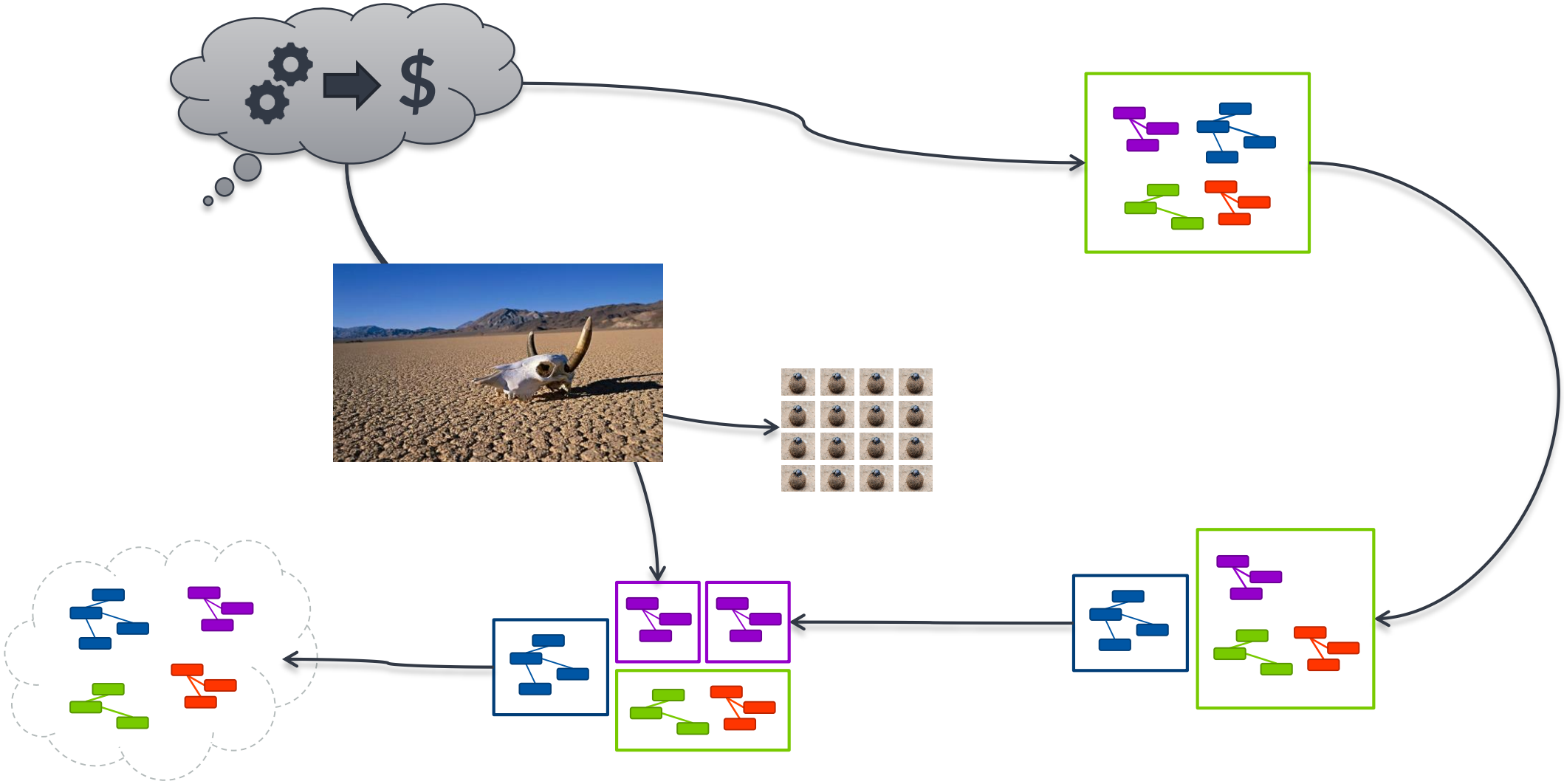
InventoryService

“Evil anti-modularity forces”

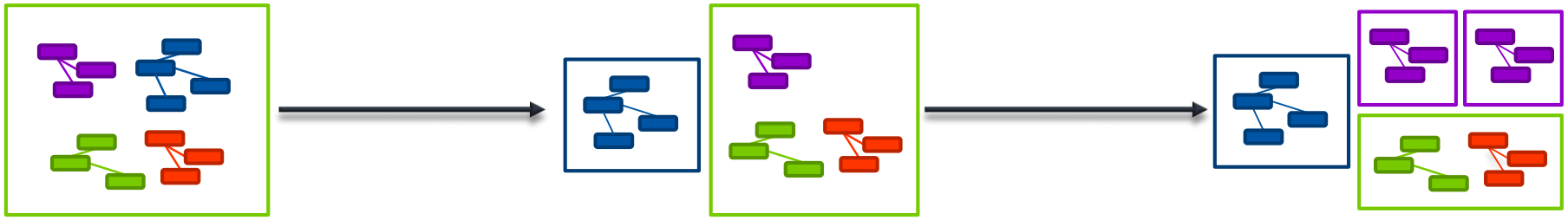


Want to build microservices?

Learn to build a decent monolith first!



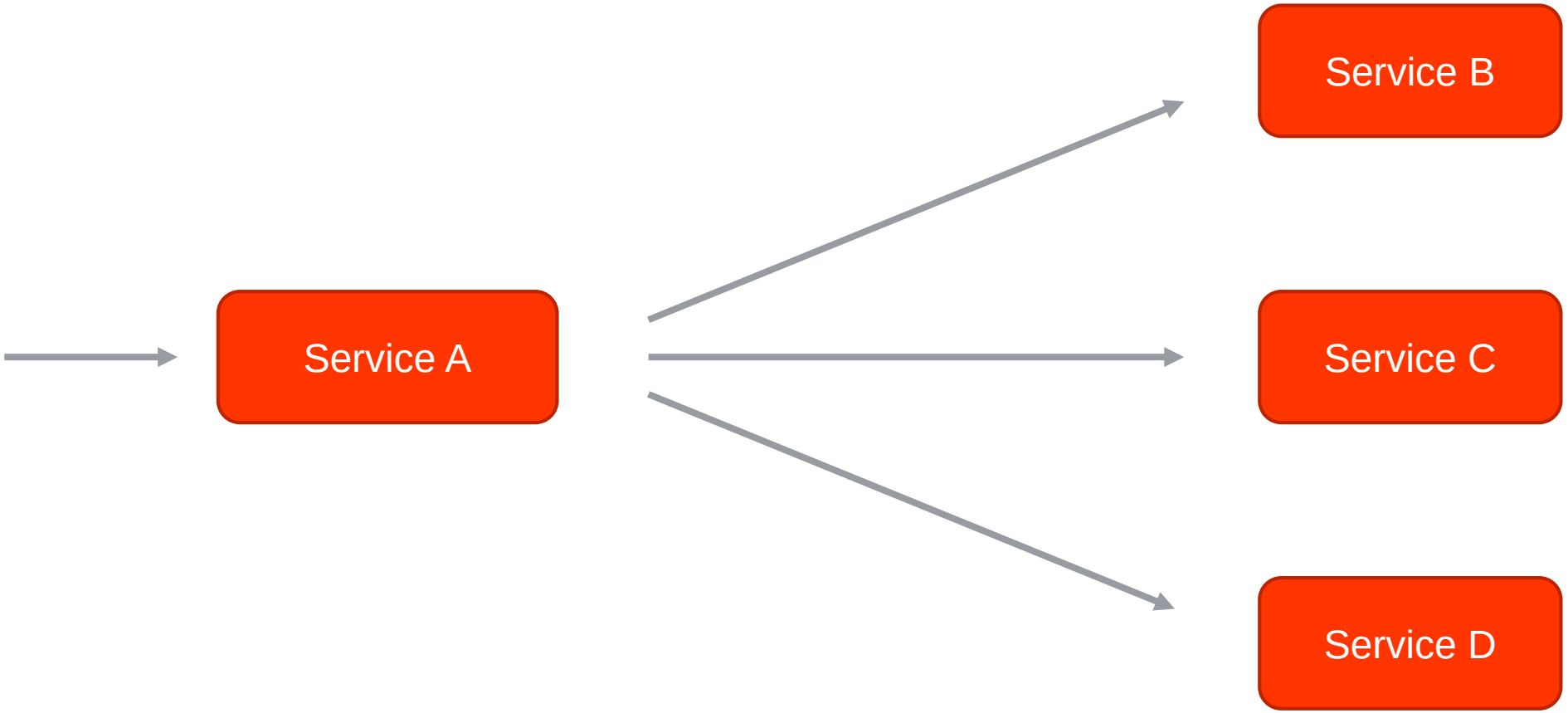
Location transparency

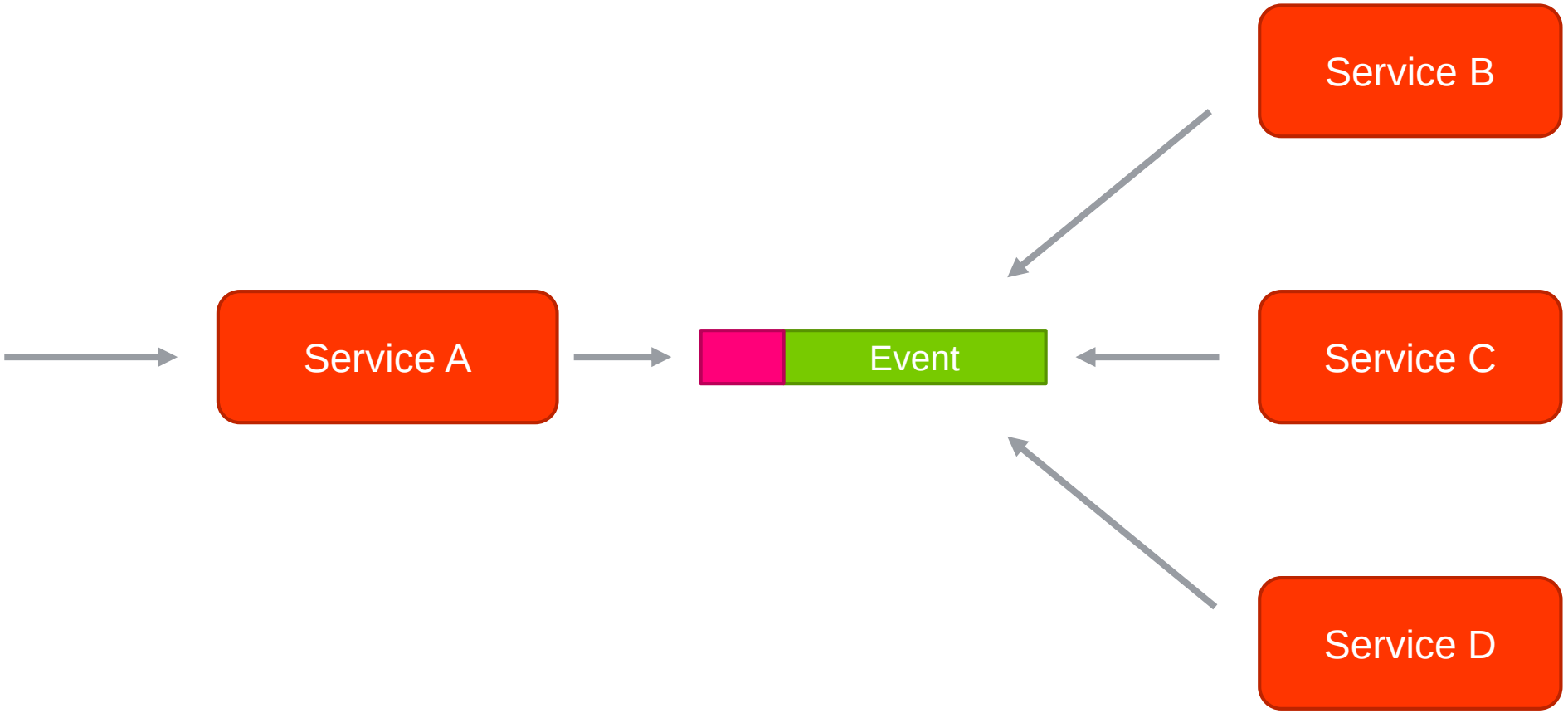


A component should neither be aware of nor make any assumptions about the location of components it interacts with.

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

Events


Service A
Service B
Service C
Service D


Service B
Service A
Event
Service C
Service D

“Event” all the things!

Maslow's Hammer

Birmingham Screwdriver

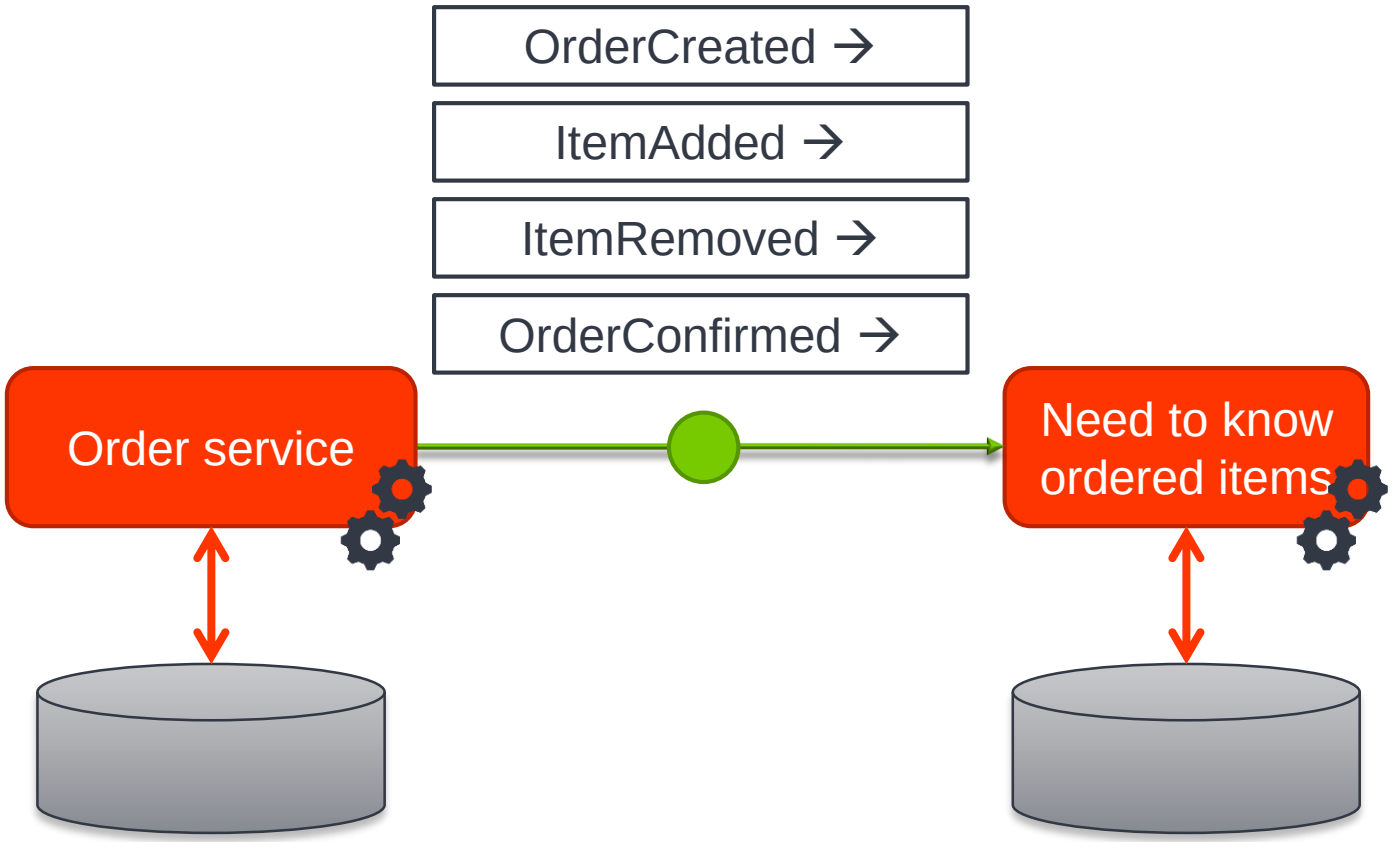
“Maslow Syndrome”

Event Notification

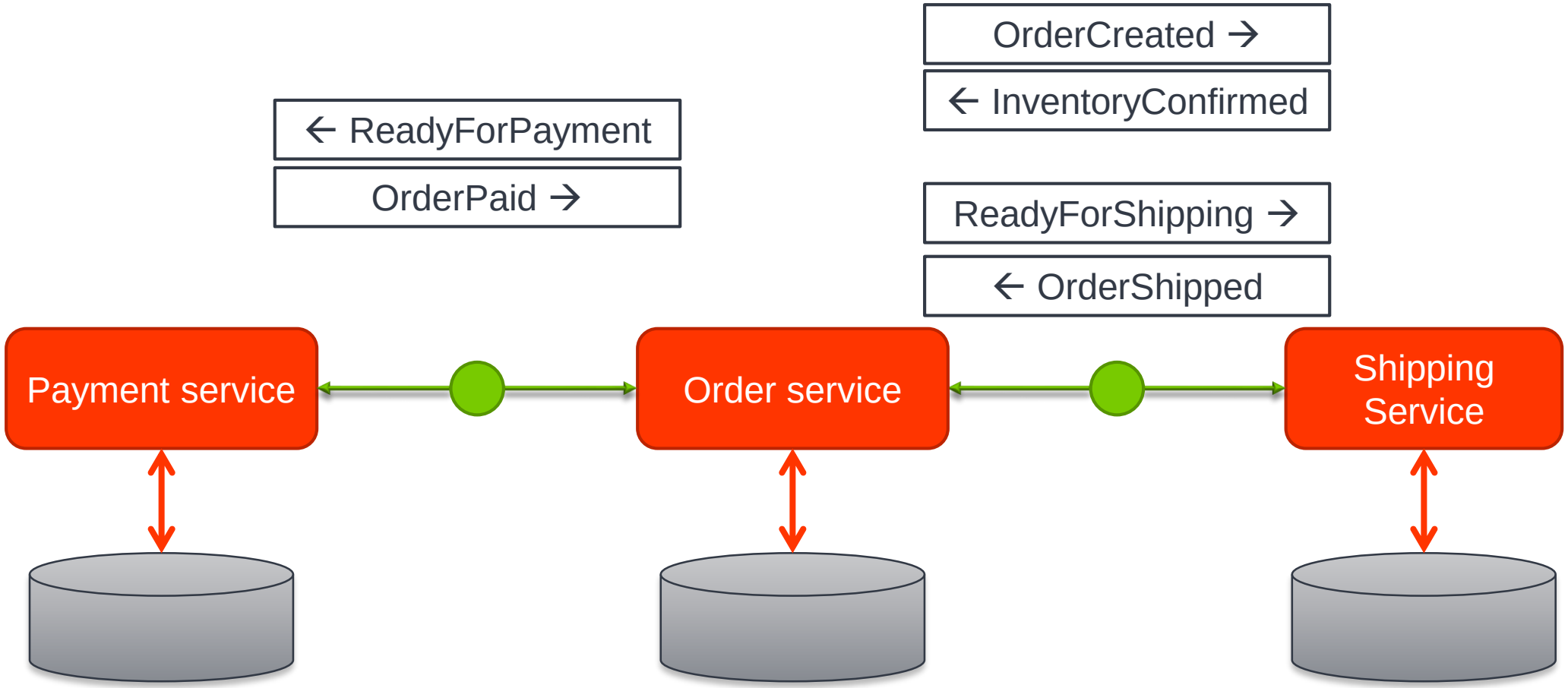
Event-carried State Transfer

Event Sourcing

‘Event-Driven’ Microservices

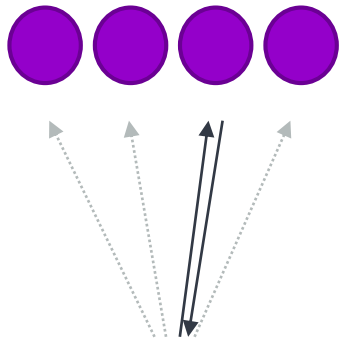


Or worse…


OrderCreated →
← InventoryConfirmed
← ReadyForPayment
OrderPaid →
ReadyForShipping →
← OrderShipped
Payment service
Order service
Shipping Service

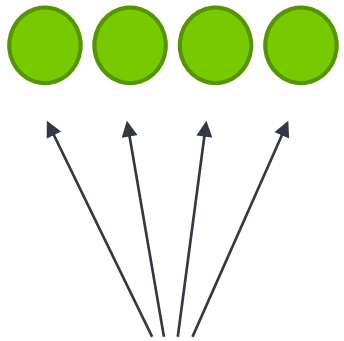
Microservices Messaging

Commands



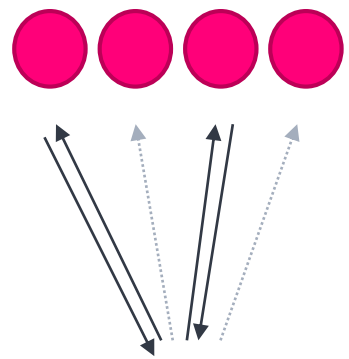
Route to single handler
Use consistent hashing
Provides confirmation/result

Events



Distribute to all logical handlers
Consumers express ordering req's
No results

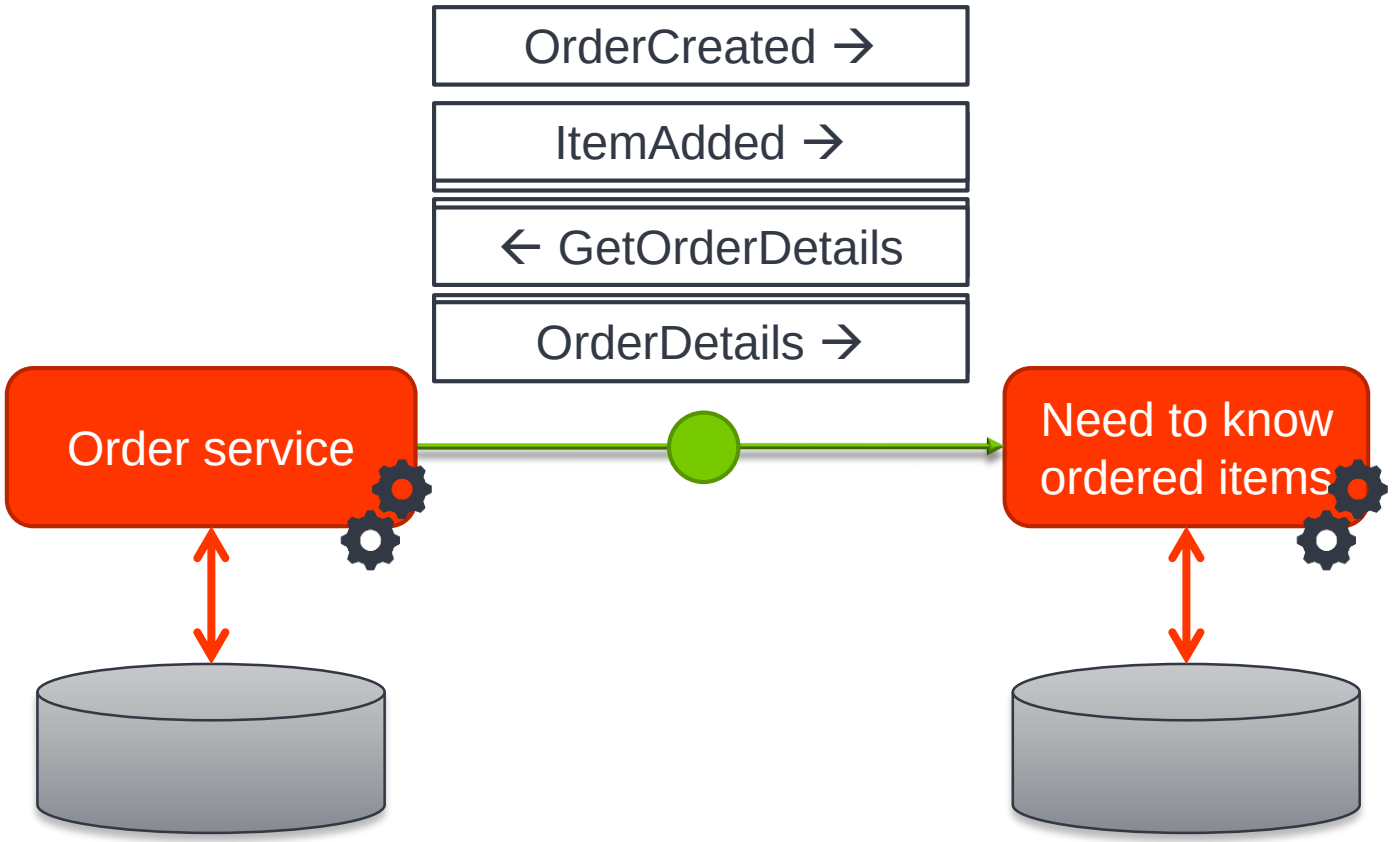
Queries



Route with load balancing
Sometimes scatter/gather
Provides result

"Event" and "Message" is not the same thing

‘Event-Driven’ Microservices



Event Sourcing:

the truth,
the whole truth,
nothing but the truth

Event Sourcing

State storage

id: 123

items

1x Deluxe Chair - € 399

status: return shipment rcvd

Event Sourcing

OrderCreated (id: 123)

ItemAdded (2x Deluxe Chair, €399)

ItemRemoved (1x Deluxe Chair, €399)

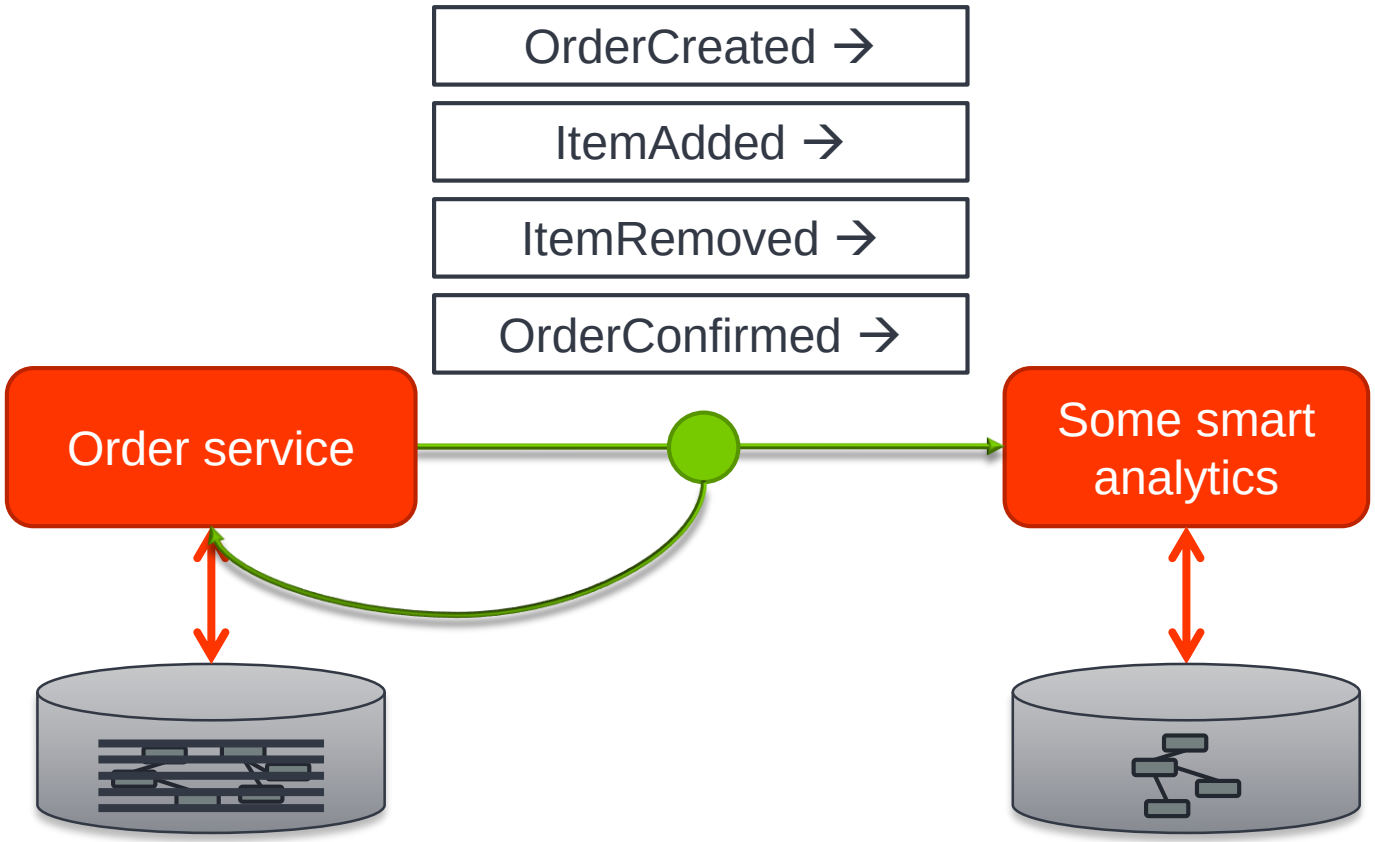
OrderConfirmed

OrderShipped

OrderCancelledByUser

ReturnShipmentReceived

Event Sourcing



Why use event sourcing?

Business reasons

- Auditing / compliance / transparency
- Data mining, analytics: value from data

Technical reasons

- *Guaranteed completeness of raised events*
- *Single source of truth*
- *Concurrency / conflict resolution*
- *Facilitates debugging*
- *Replay into new read models (CQRS)*
- *Easily capture intent*
- *Deal with complexity in models*

The challenges

~~Dealing with increasing storage size~~

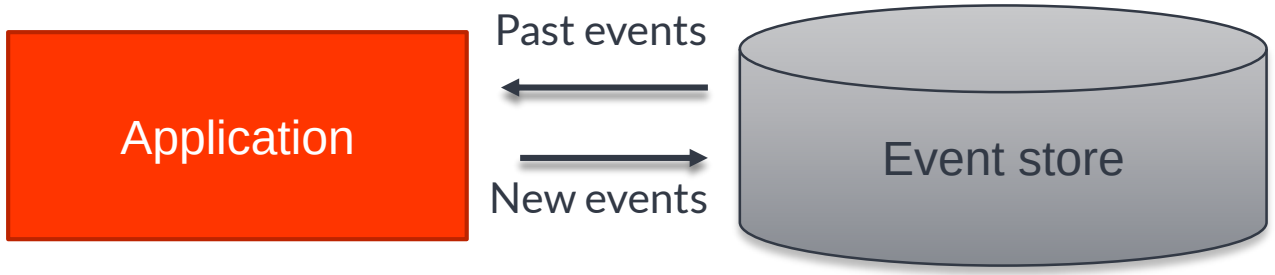
~~Complex to implement~~

“Event Thinking”

Source

"Event" all the things!

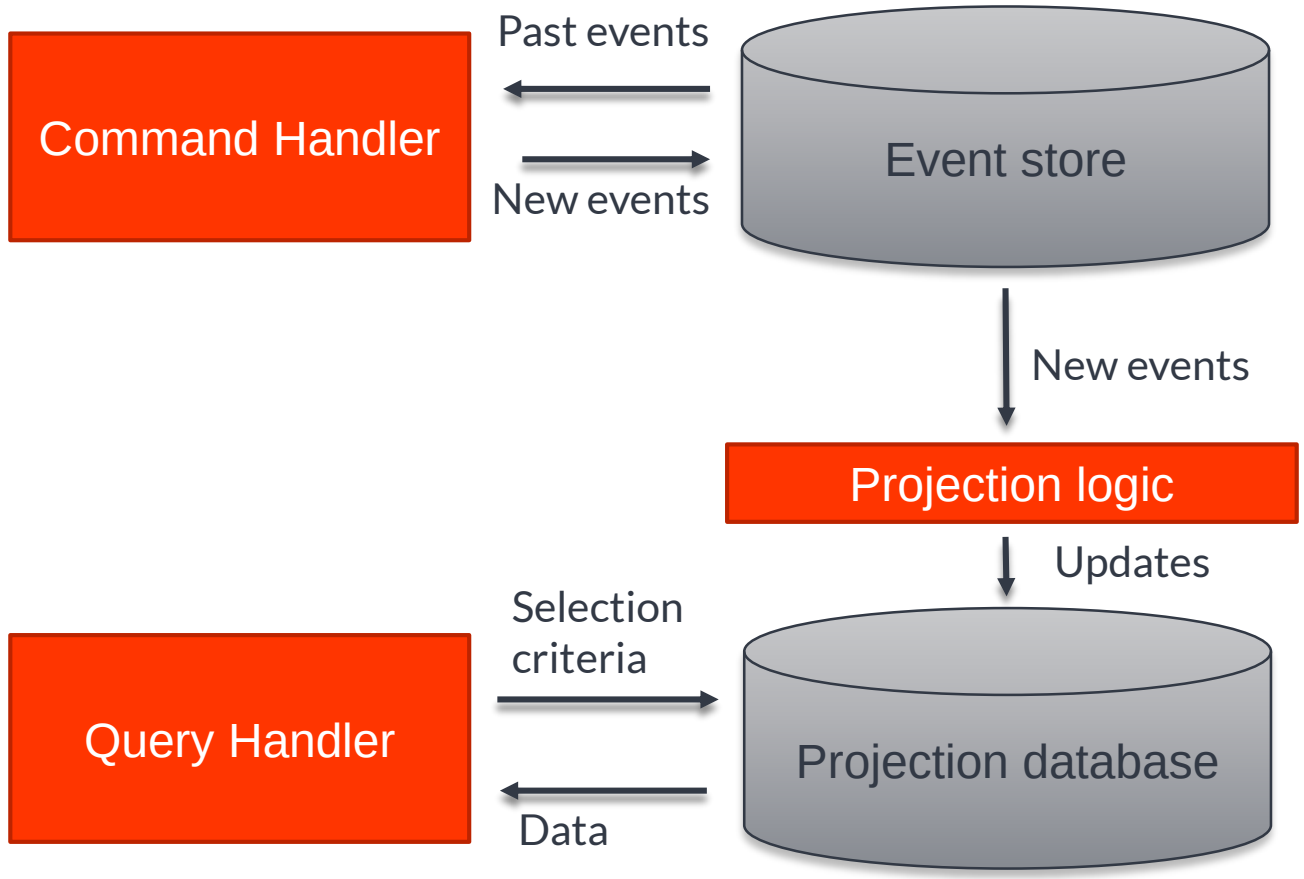
Event store in context



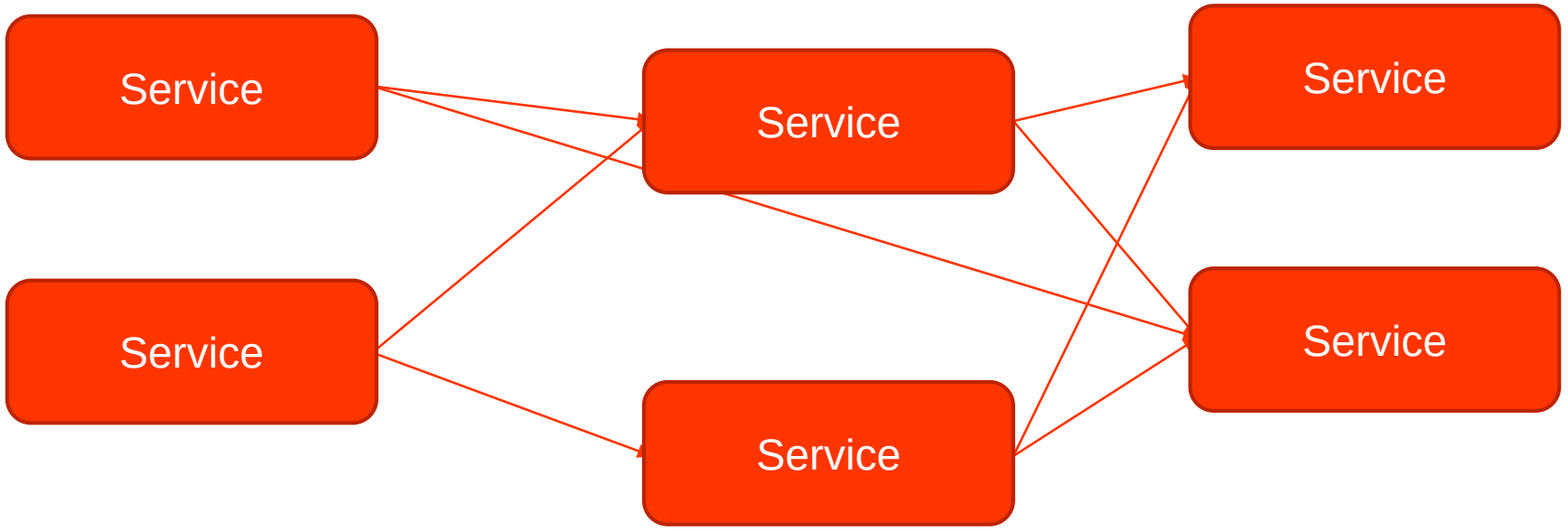
- Works well for processing changes on single entities/aggregates (Commands)
- Does ***not*** work well for generic queries

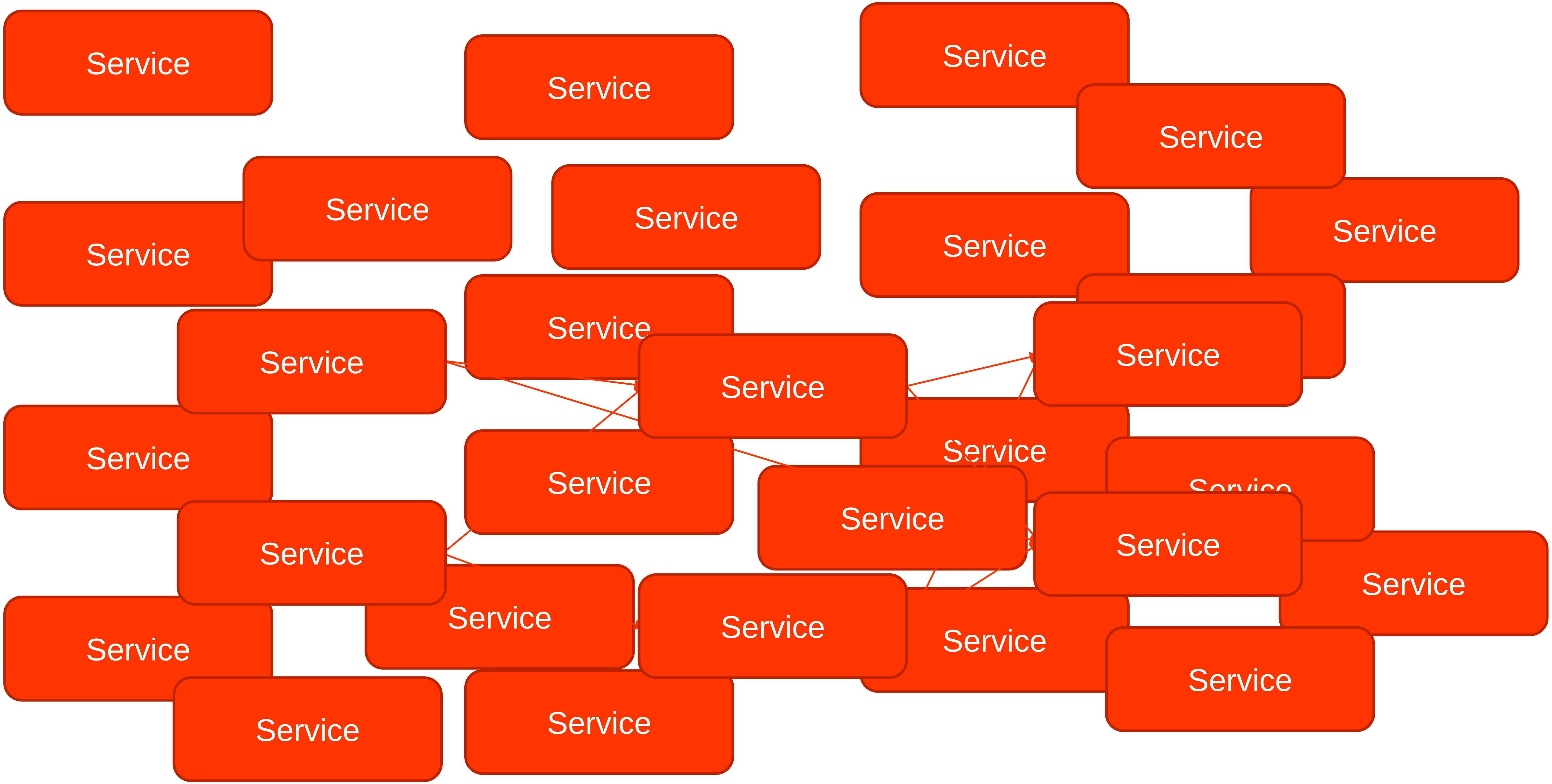
CQRS

Command-Query Responsibility Segregation

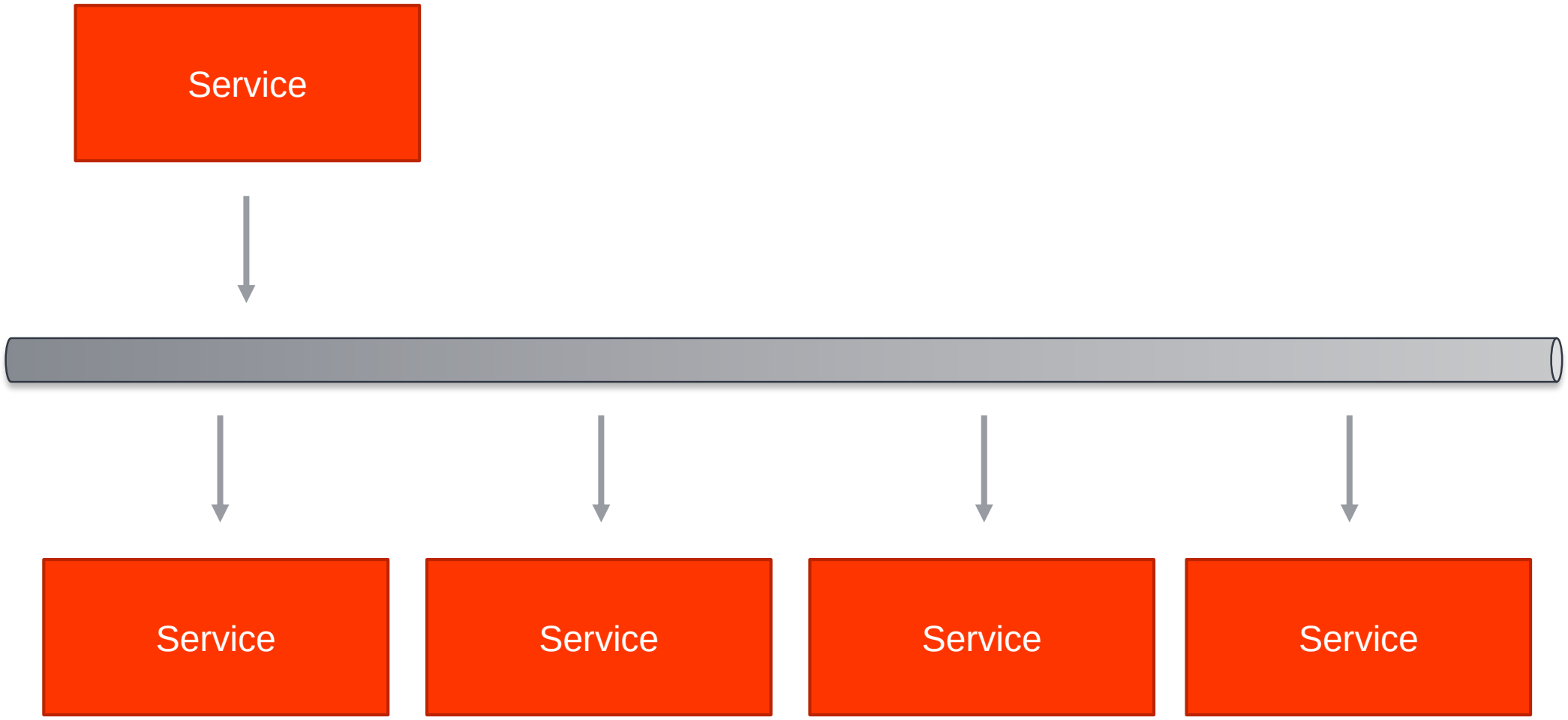


“CQRS” all the things?


Service
Service
Service
Service
Service
Service

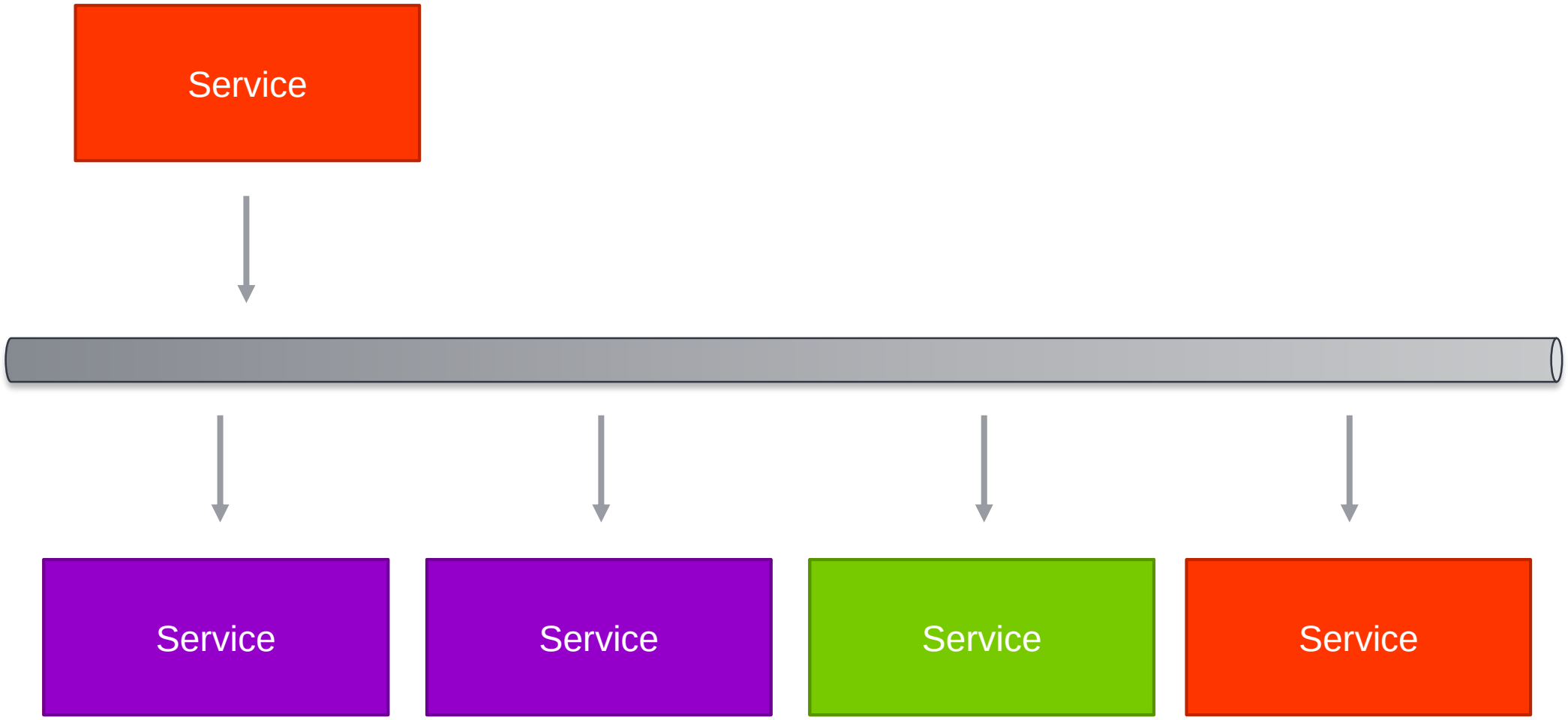

Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service

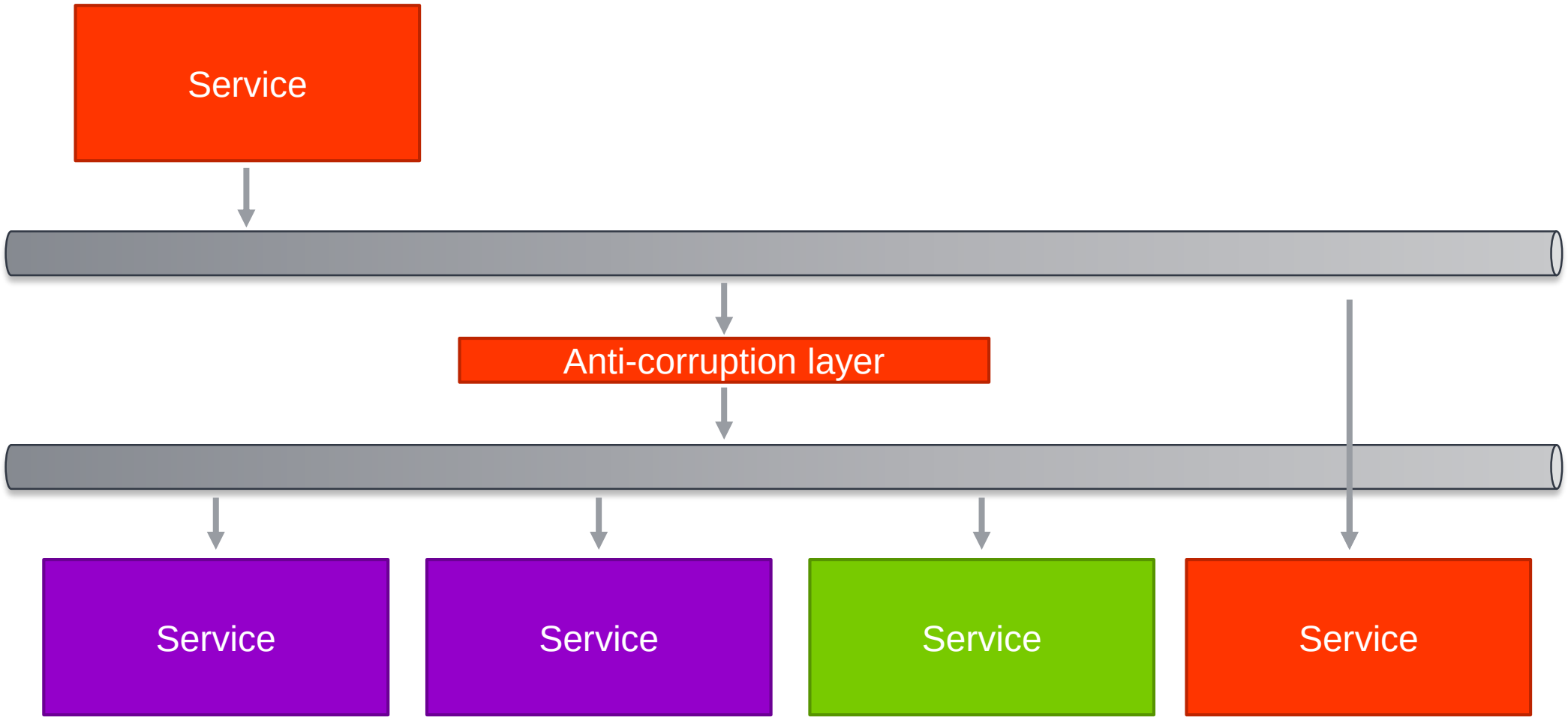
Communication = Contract

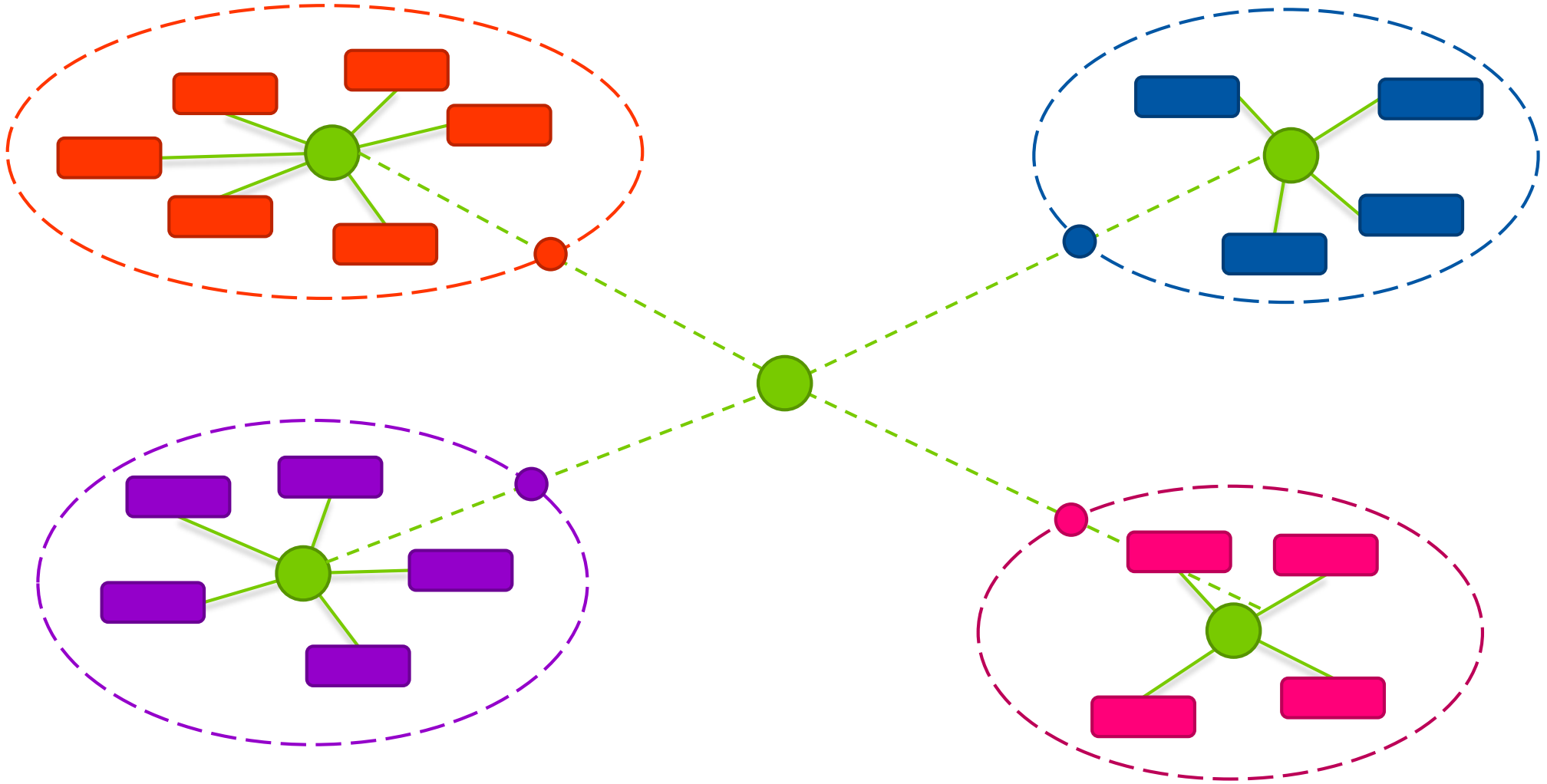

Service
Service
Service
Service
Service

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.


Service
Service
Service
Service
Service


Service
Anti-corruption layer
Service
Service
Service
Service



In closing....

Consider *commands*
and *queries*
as much as *events*

Sharing is caring

Beware coupling across bounded contexts

“Microservice Journey”

Monolith first

wax-on, wax-off!