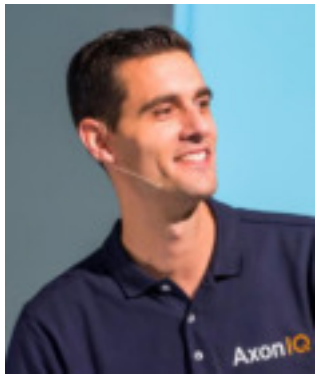
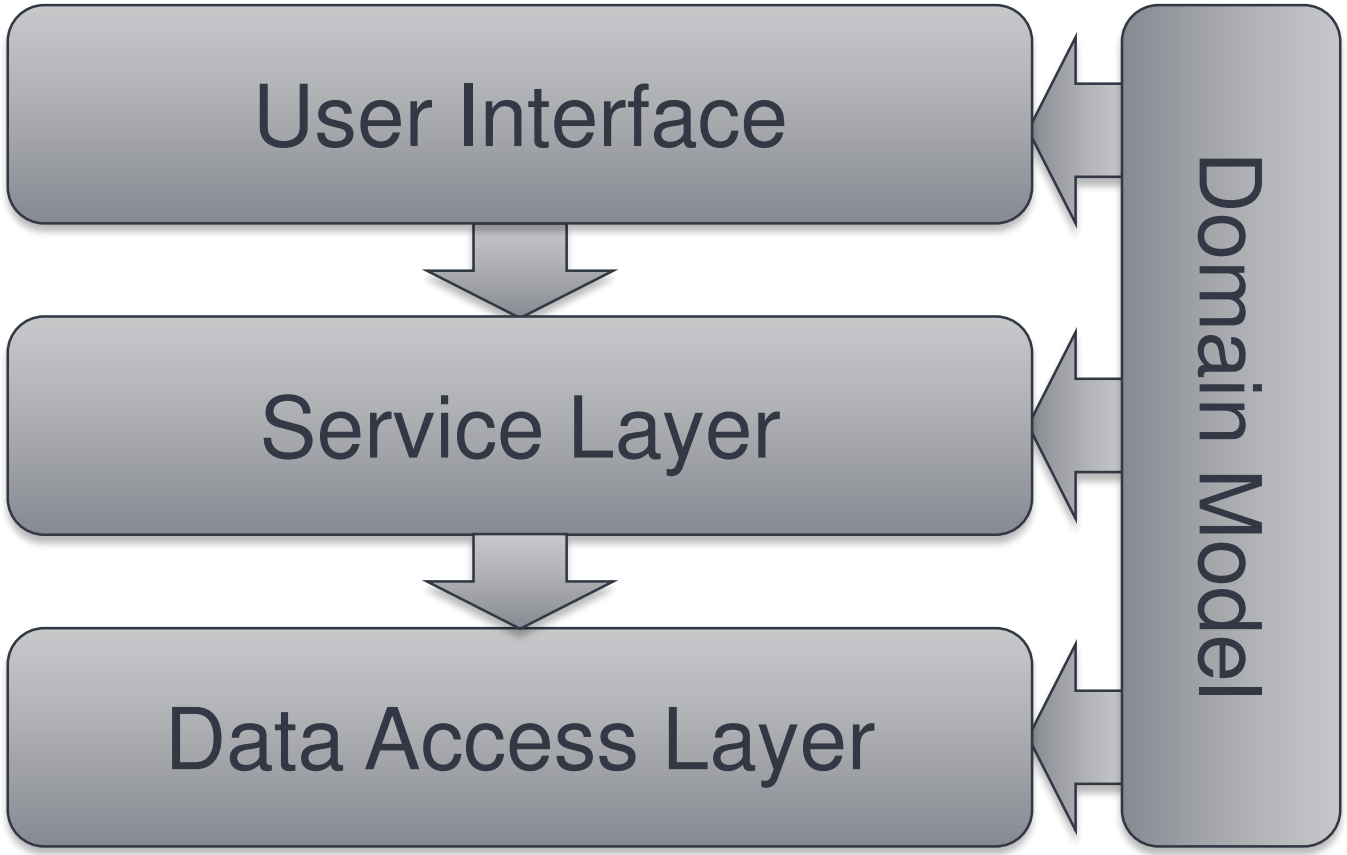
Pragmatic Microservices
with DDD, CQRS and Event Sourcing using Axon Framework

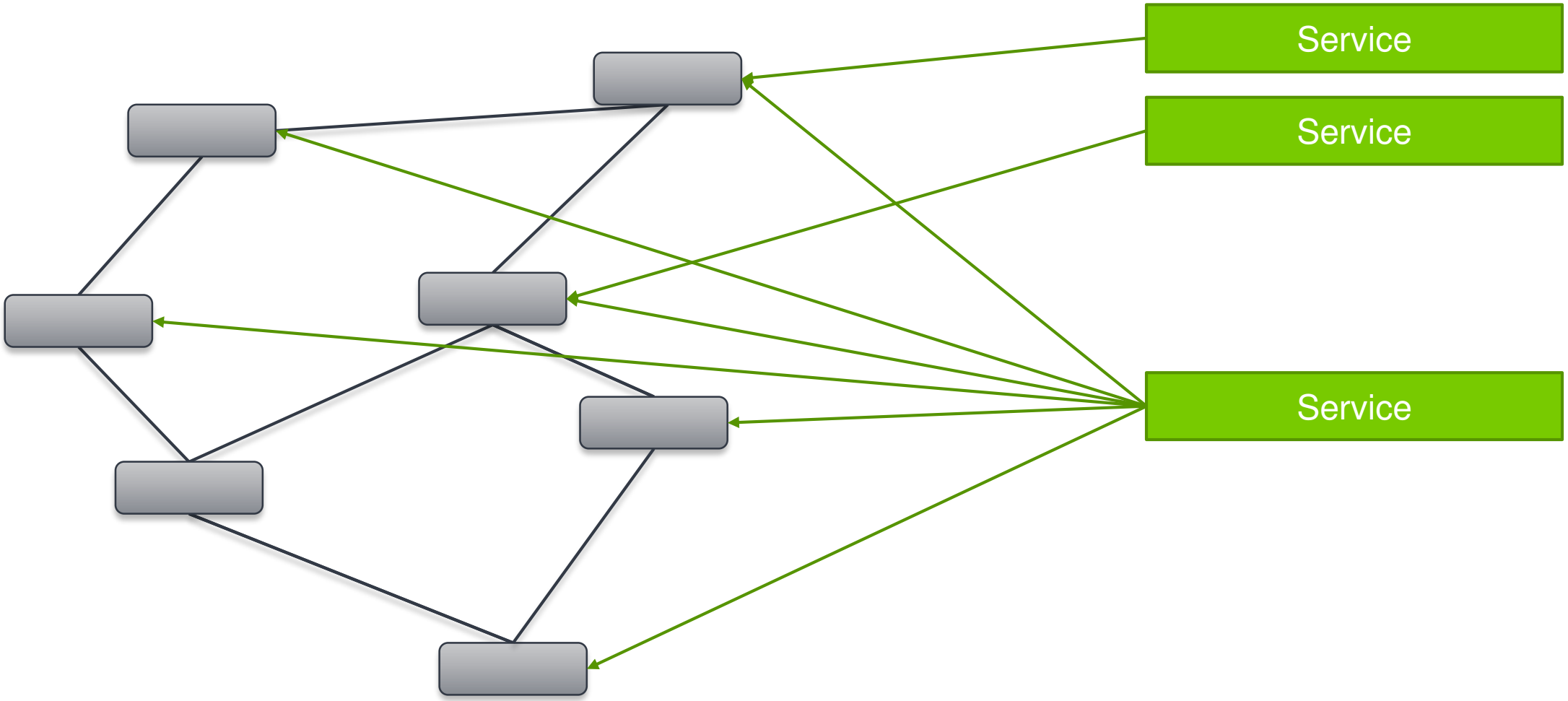


Allard Buijze
Founder & CTO, AxonIQ
Creator of AxonFramework

✉ allard@axoniq.io
@allardbz

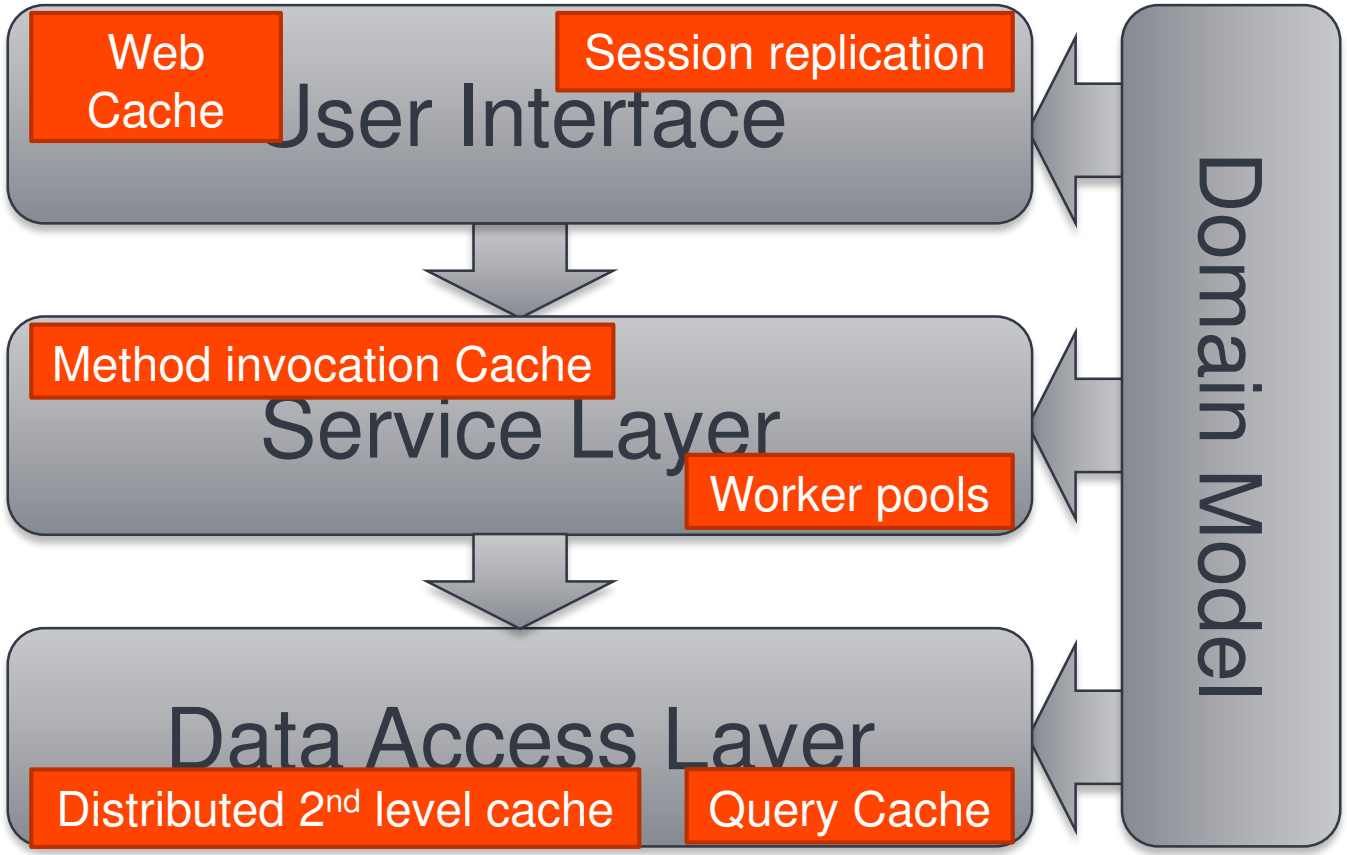
Layered architecture




Service
Service
Service

‘Normal’ SQL QUERY

Layered architecture



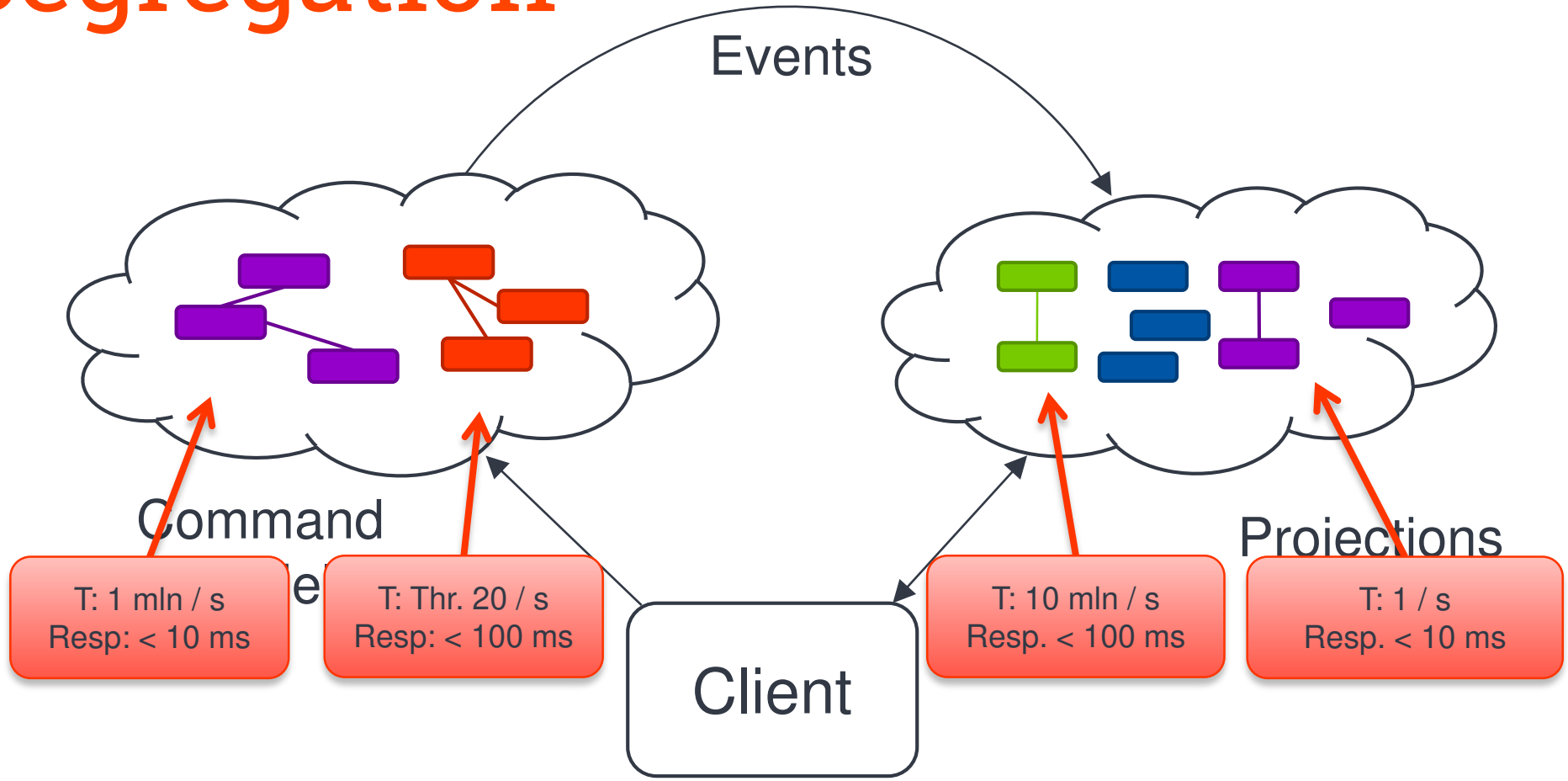

PORSCHE
20
MICHELIN
MODIFIEDPLANET.COM


352 KTH
MODIFIEDPLANET.COM

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

AxonFramework

Command Query Responsibility Segregation



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 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

The value of Event Sourcing

- Audit trail
- Analytics

- Improve modeling
- Given-when-testing

- …

The power of...

Not now

Live coding…

Event Sourcing

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

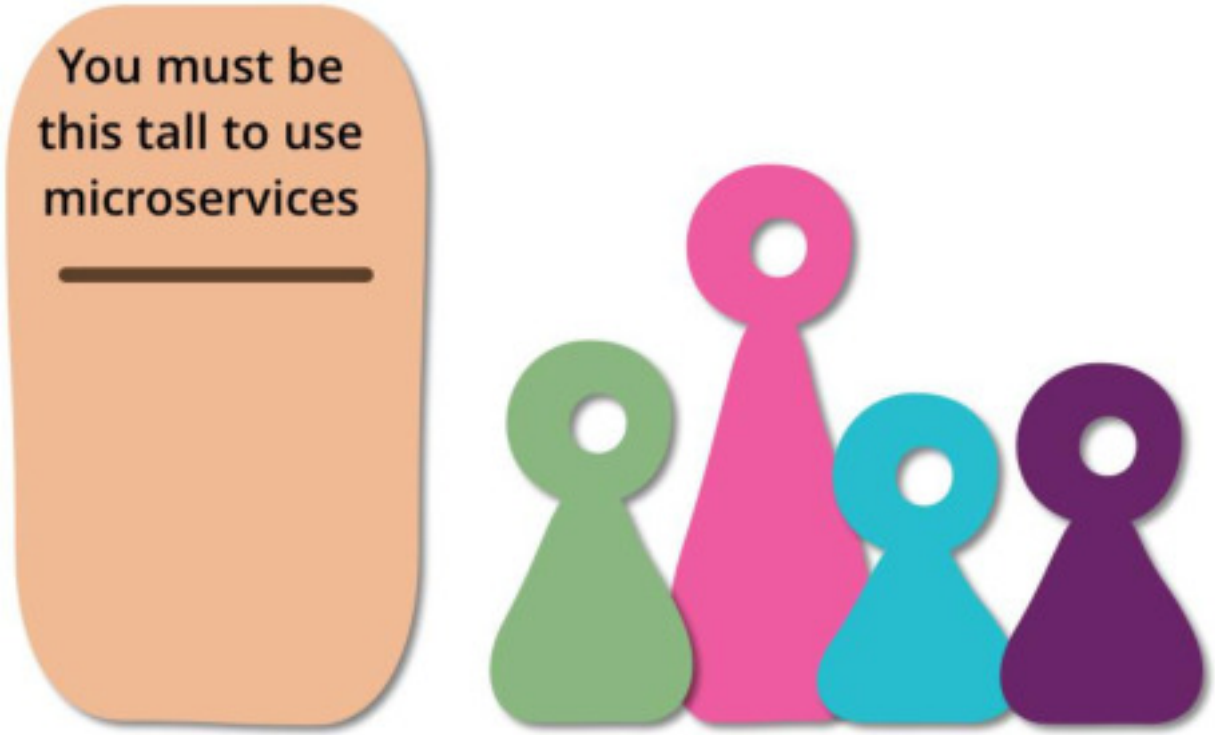
Microservices system

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

Source: http://martinfowler.com/bliki/MonolithFirst.html

Are you tall enough?

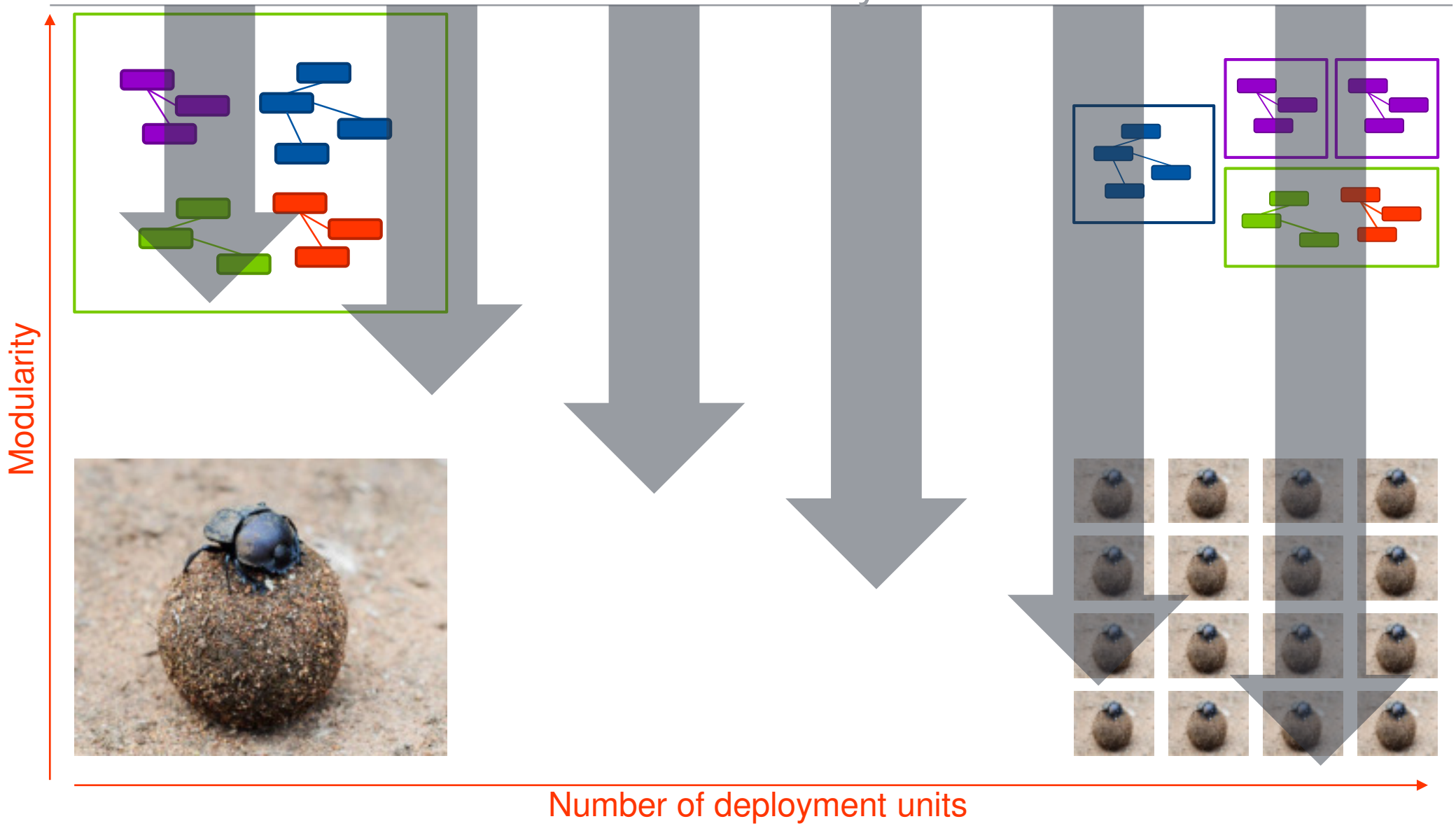


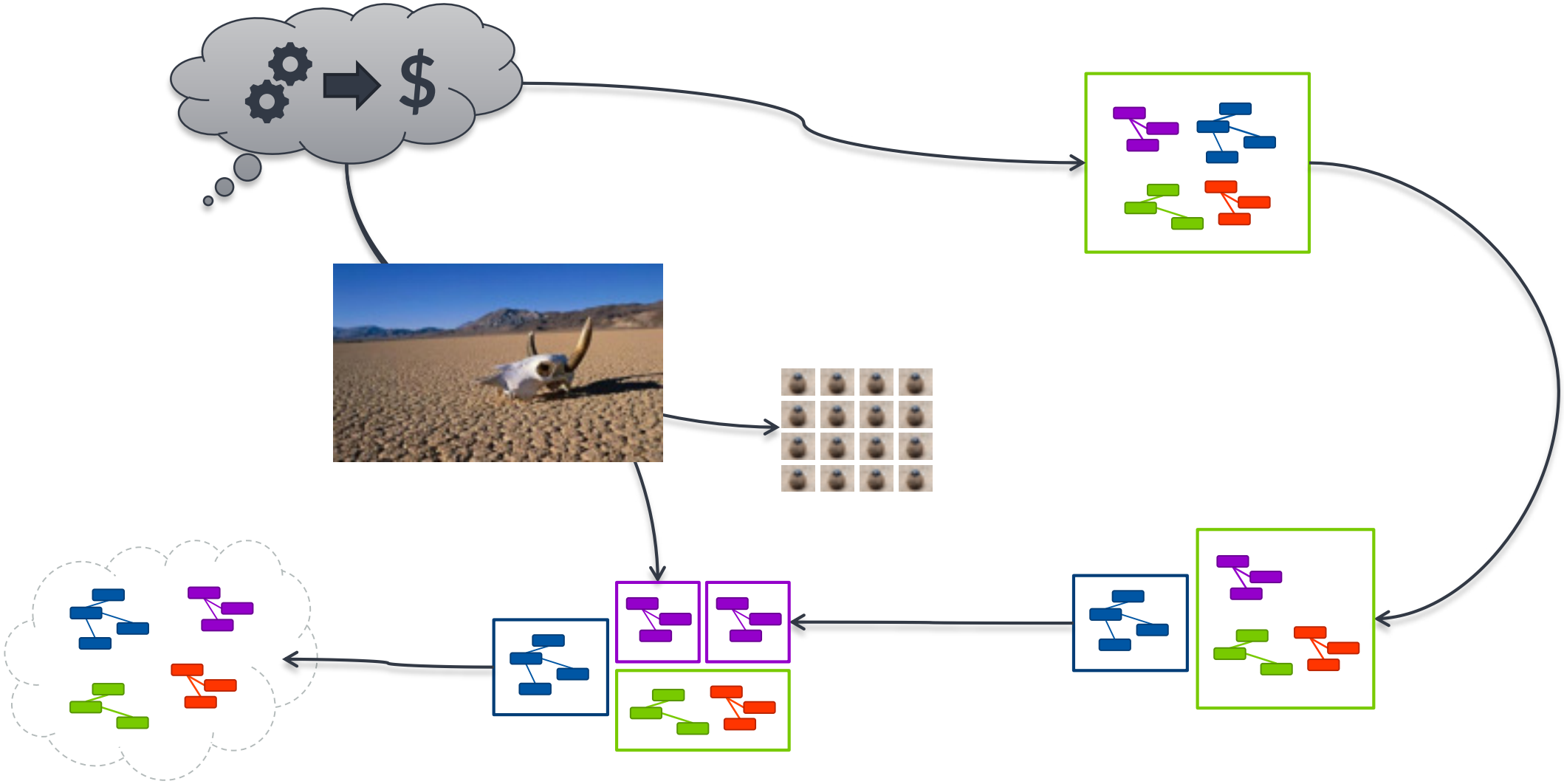
Source: martinfowler.com/bliki/MicroservicePrerequisites.html

“Noun Driven Design”

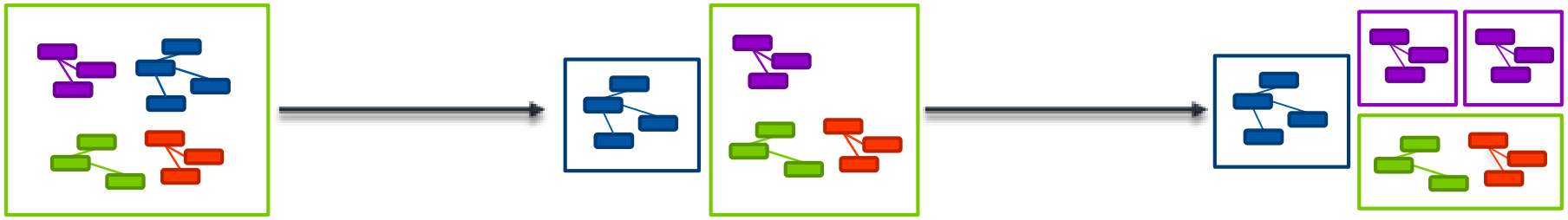
“Entity Services”

“Evil anti-modularity forces”





Location transparency

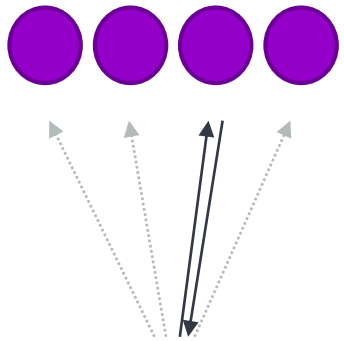


A component should neither be aware of nor make any assumptions about the location of components it interacts with.

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

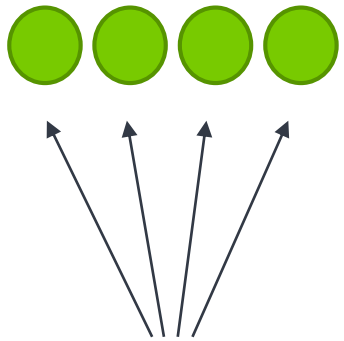
Microservices Messaging

Commands



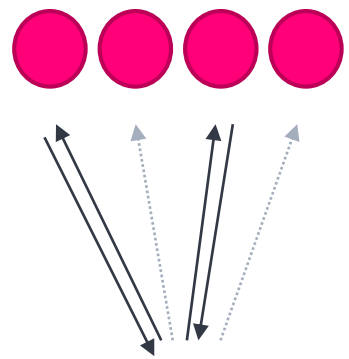
Route to single handler
Use consistent hashing
Provides confirmation/result

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!

Live coding…

Distributing components

Rule #1 of distributed computing:

Don't

Try it yourself

- Sources:
  - https://github.com/abuijze/bike-rental-demo ('devdays' branch)
  - https://github.com/AxonFramework/AxonFramework

- Download Axon
  - axoniq.io/download (includes quick-start-guide)