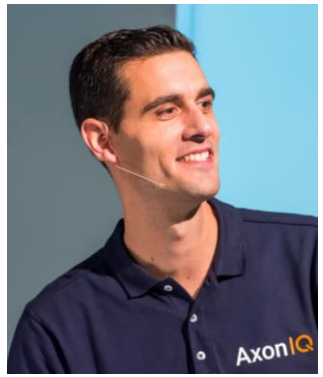
Scaling CQRS in Theory, Practice and Reality



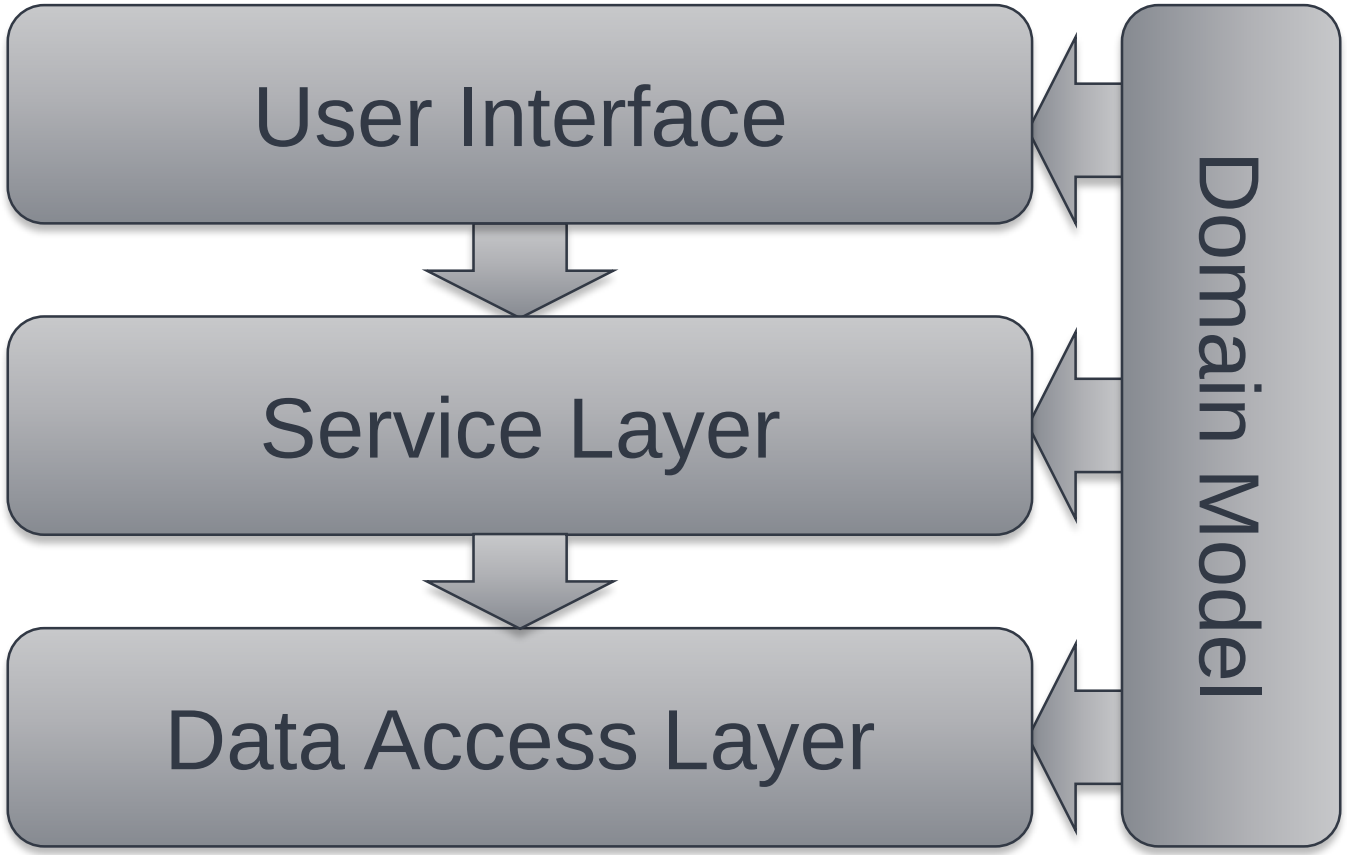
Allard Buijze
Founder & CTO, AxonIQ

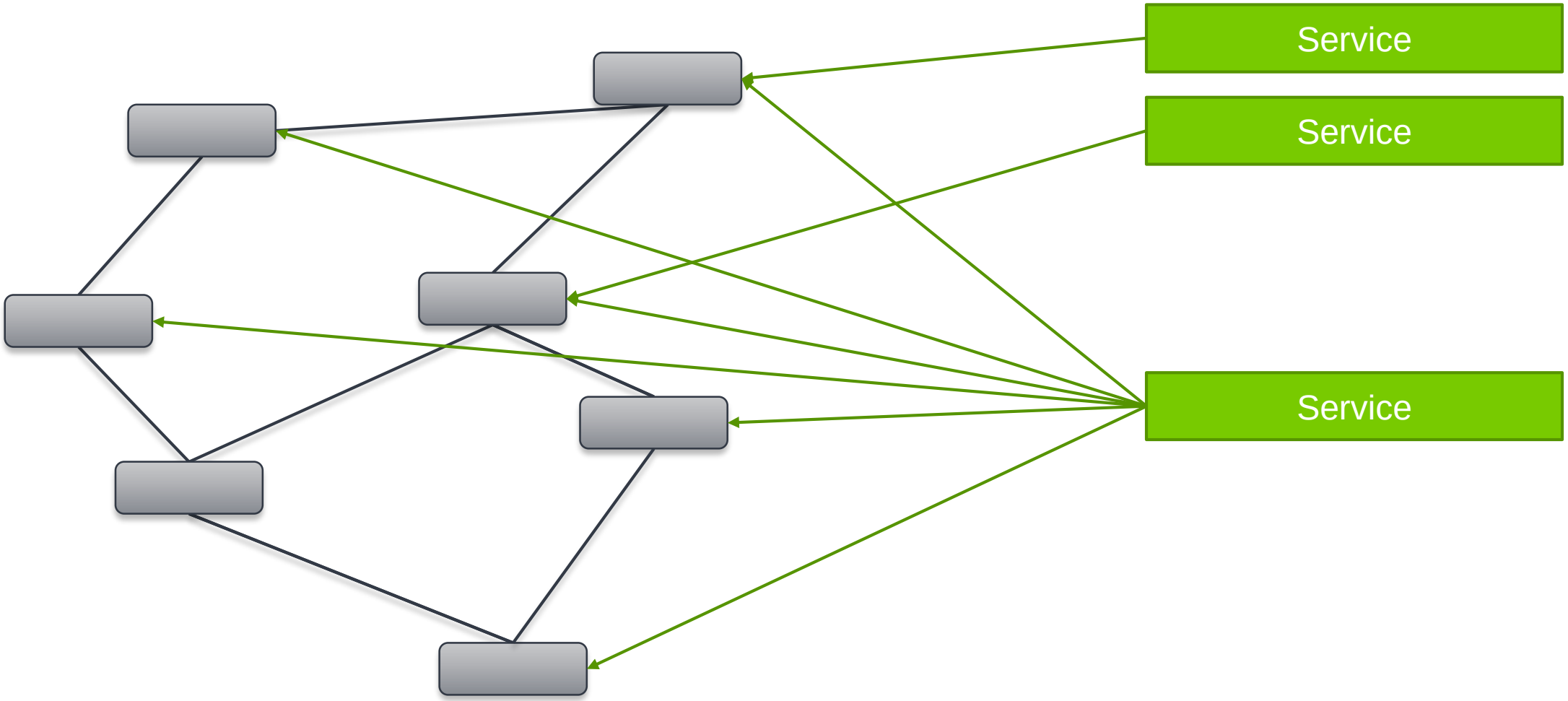
✉ allard@axoniq.io
allardbz

Nakul Mishra
Sr Engineer & Architect, Casumo

✉ nklmish@protonmail.com
nklmish

Layered architecture

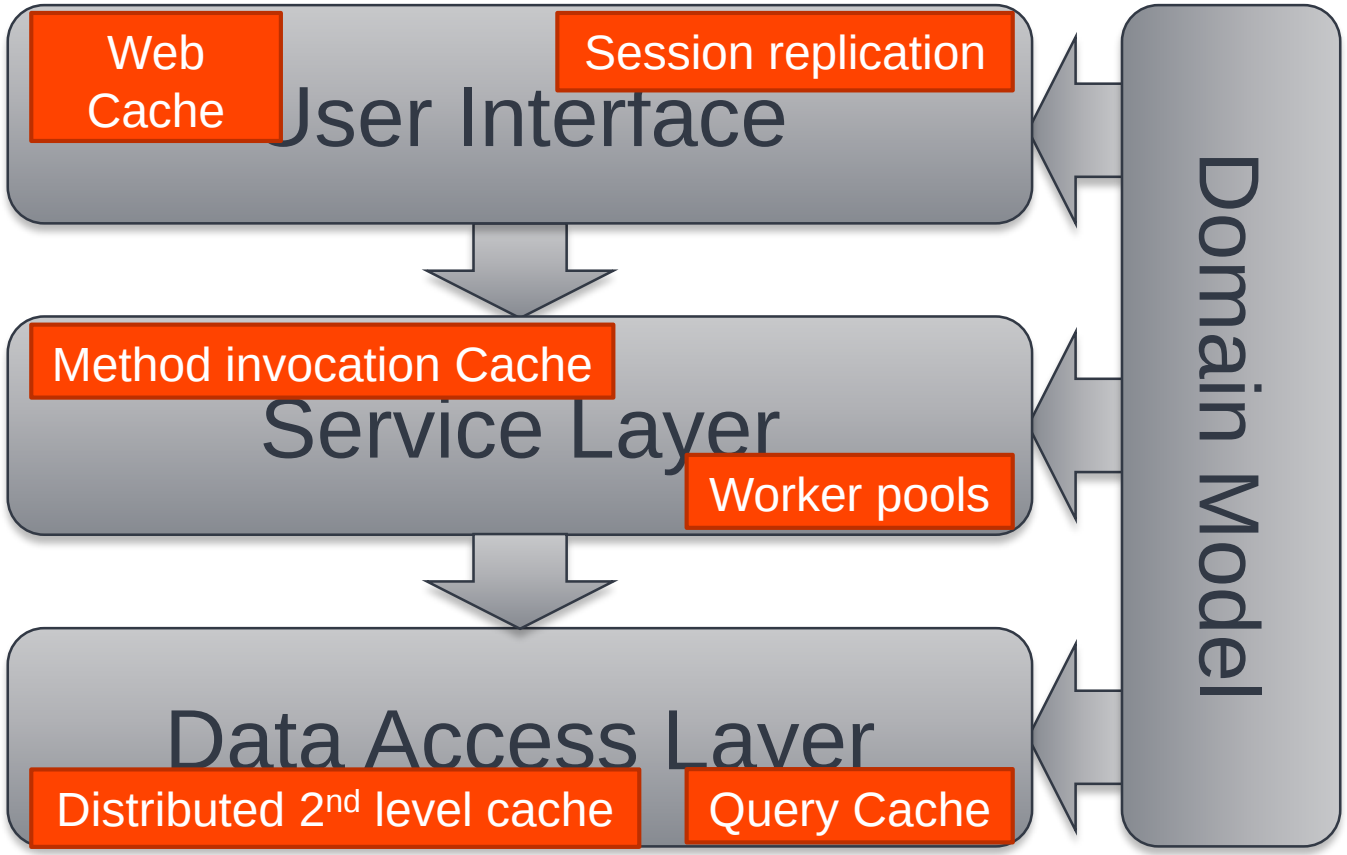



Service
Service
Service

‘Normal’ SQL QUERY

Layered architecture

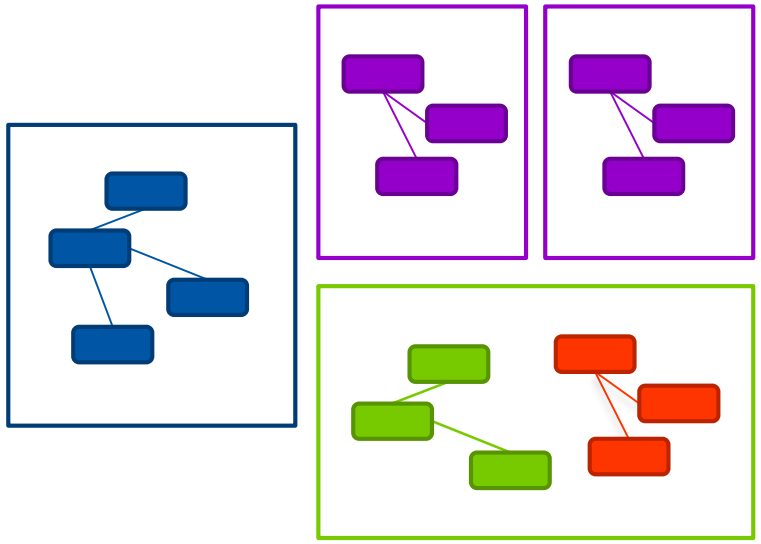
Ponder and deliberate before you make a move. —Sun Tzu



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

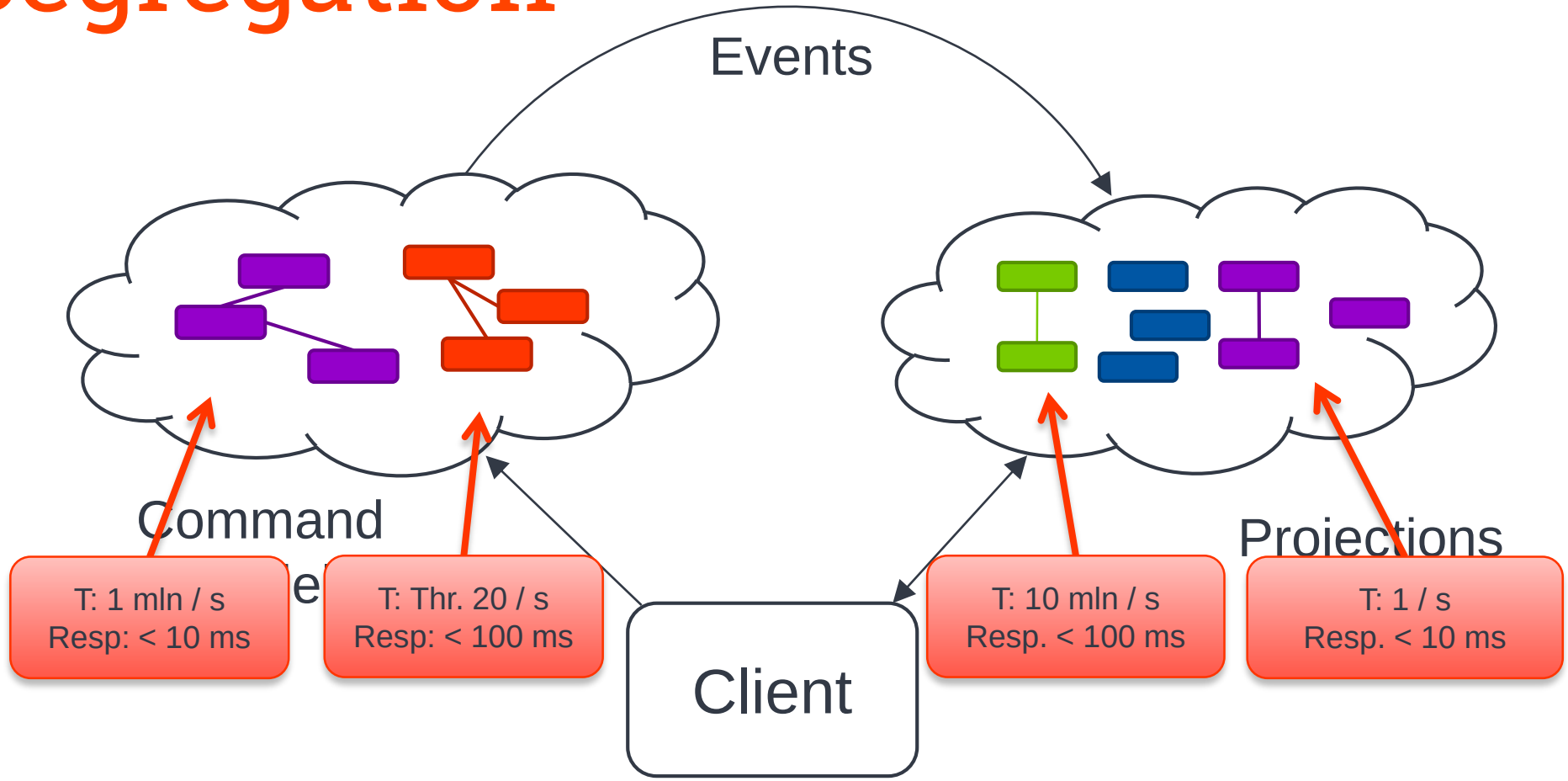
Microservices systems

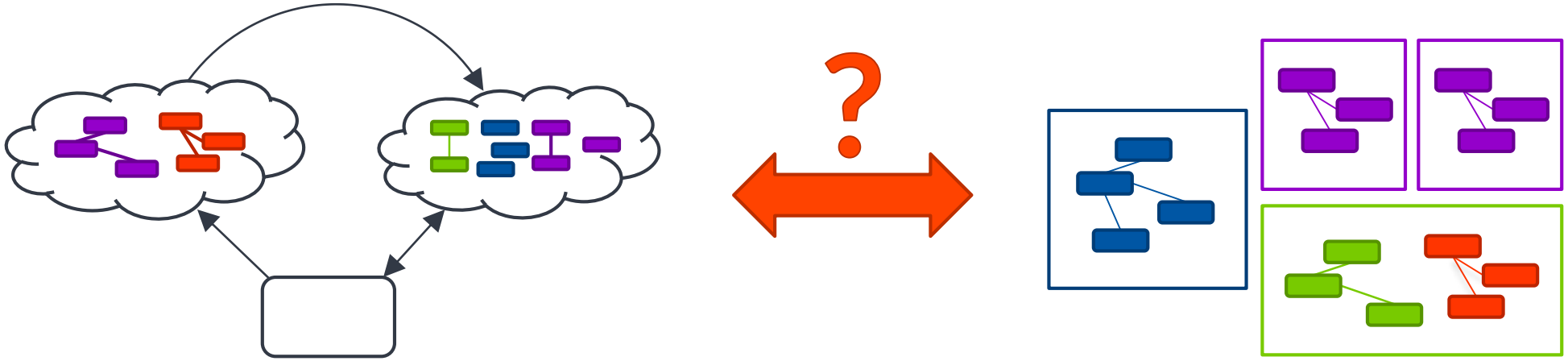
- Splitting up systems into smaller, simpler components
  - Agility
  - Scalability



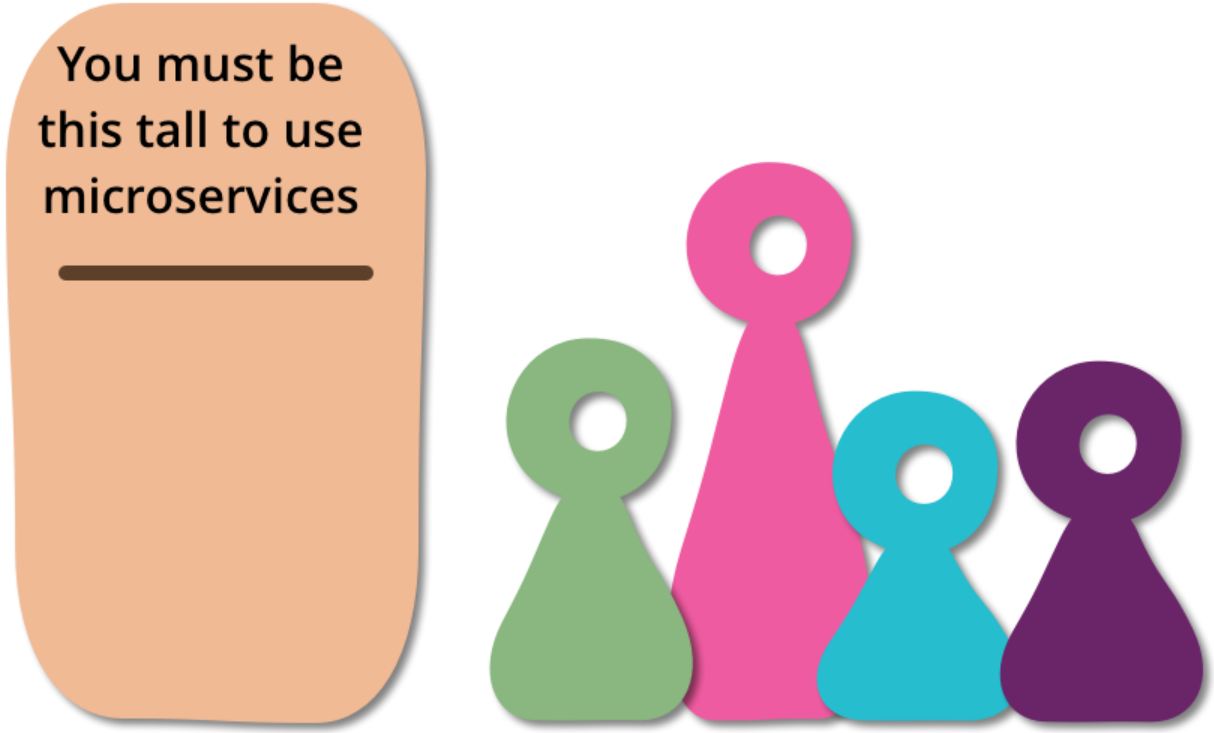
CQRS

Command Query Responsibility Segregation





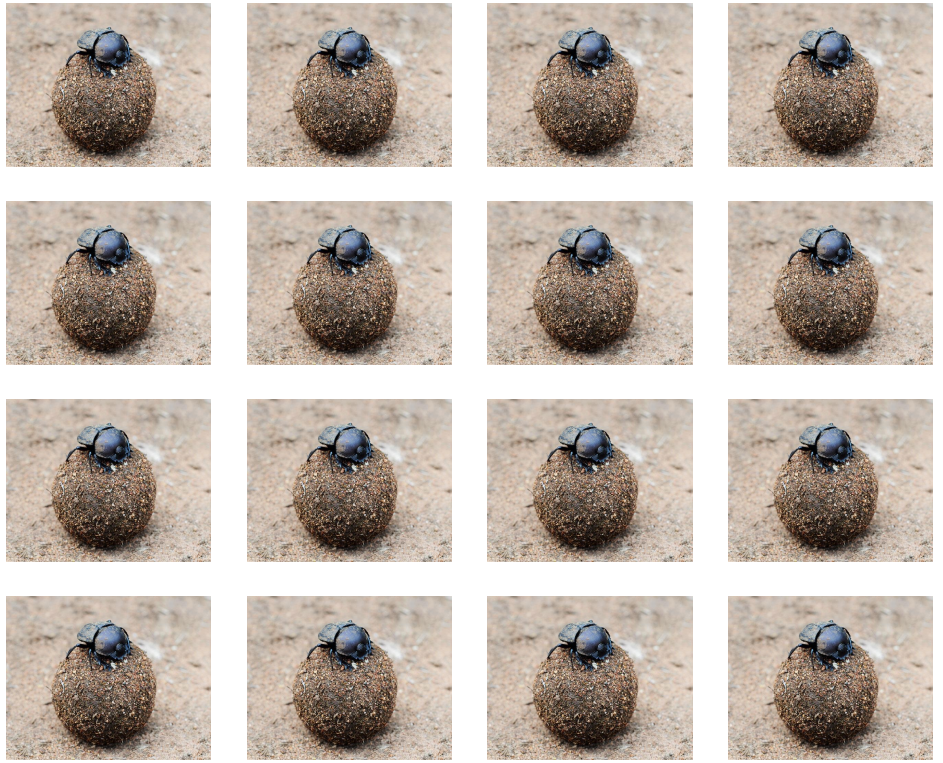
Are you tall enough?



Source: martinfowler.com/bliki/MicroservicePrerequisites.html

“Noun Driven Design”

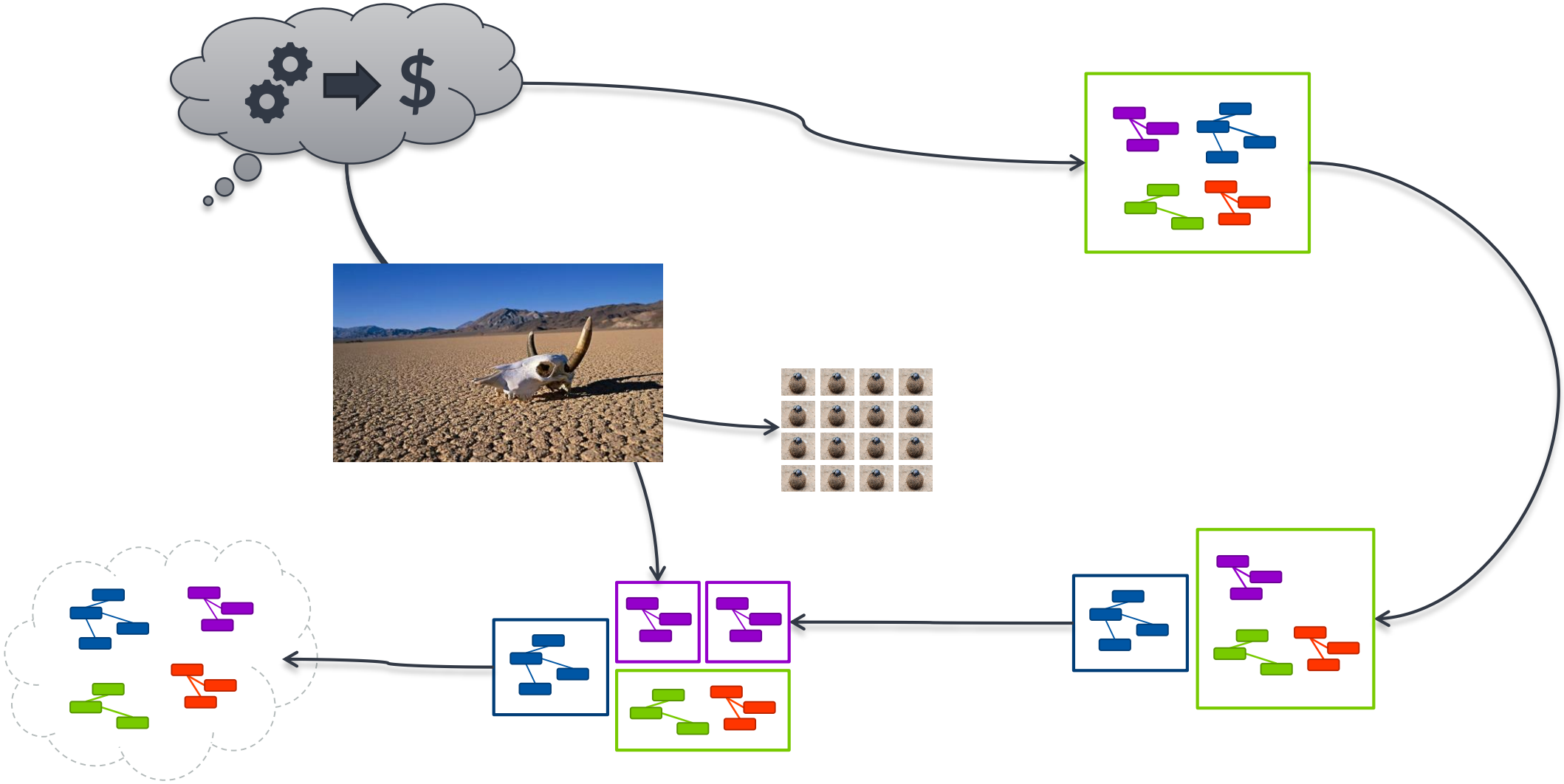
“Entity Services”



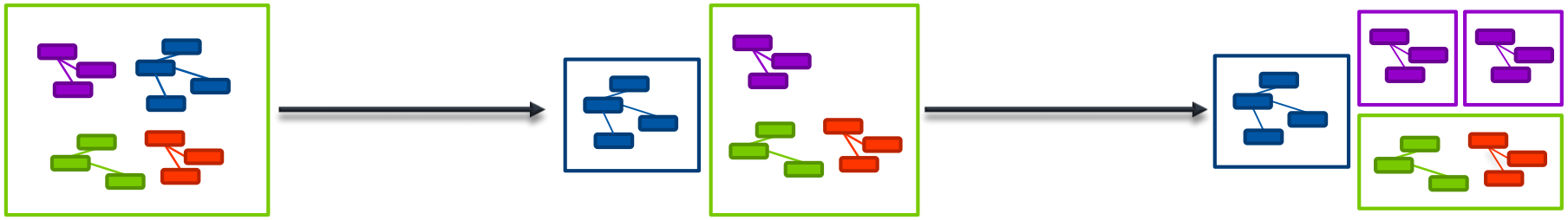
Monoliths

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

≠



Location transparency

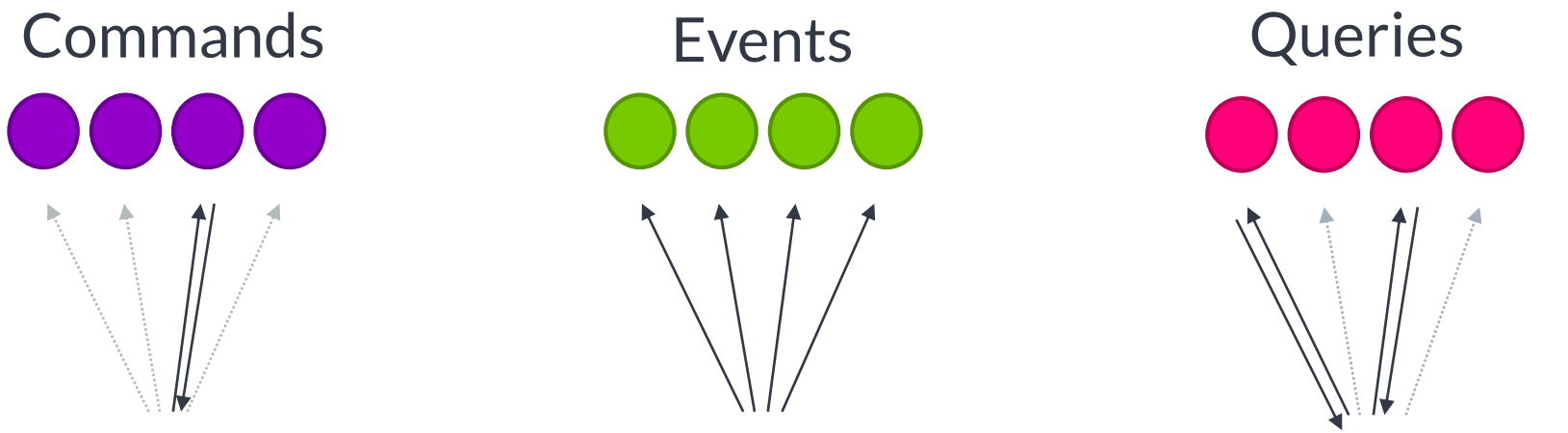


A component should neither be aware of nor make any assumptions about the location of components it interacts with.

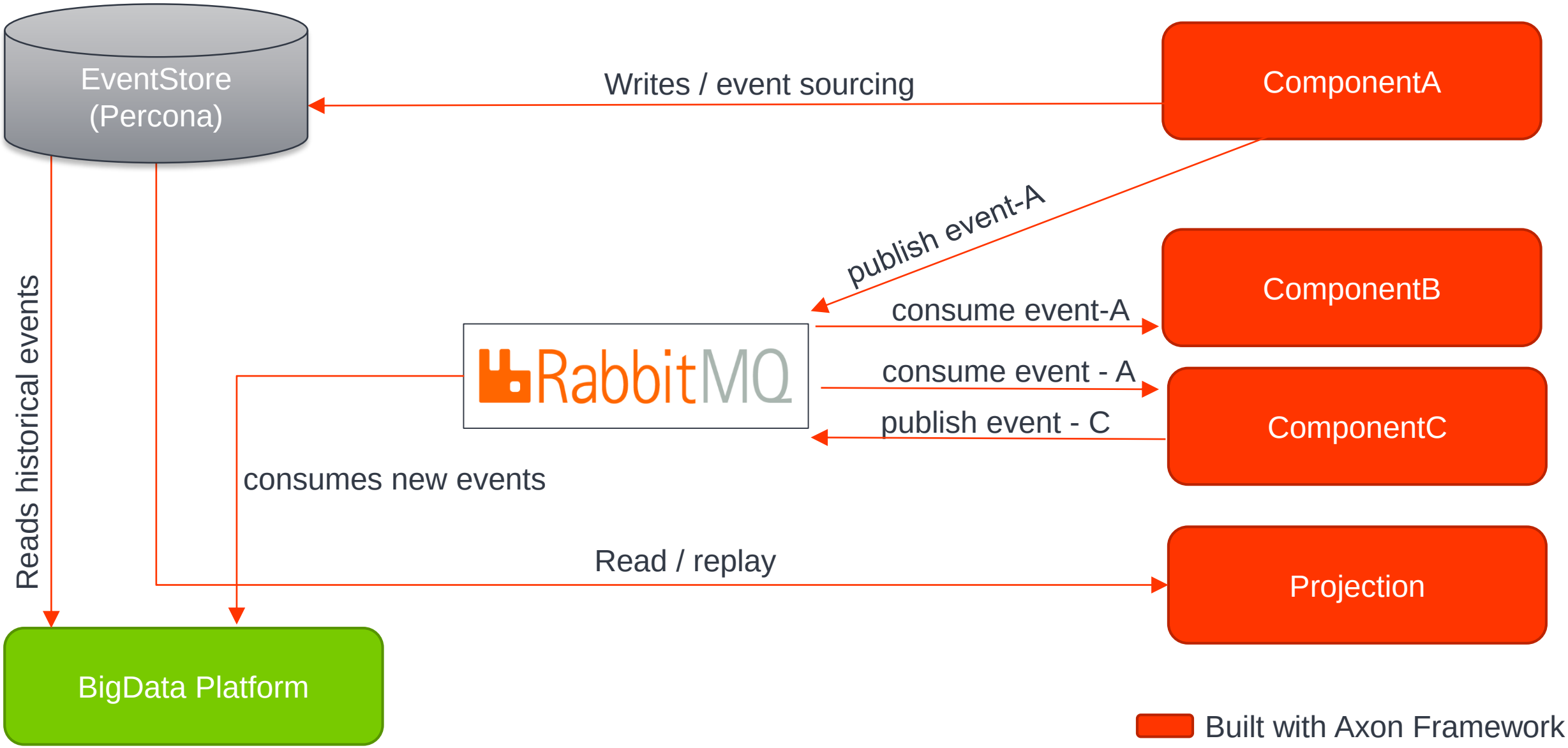
Location transparency starts with good API design

(but doesn't end there)

Microservices Messaging



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


EventStore
(Percona)
Writes / event sourcing
ComponentA
publish event-A
ComponentB
consume event-A
RabbitMQ
consume event - A
publish event - C
ComponentC
Reads historical events
consumes new events
Read / replay
Projection
BigData Platform
Built with Axon Framework

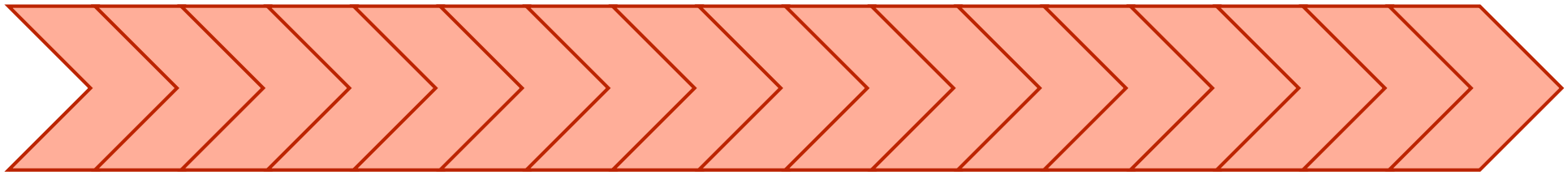
Casumo

- Event store with >30 billion events
- Hundreds of millions of events, every day
- Every event is EQUALLY important

“I was thinking I might win 50 pounds but when it went all the way to the jackpot I was shocked.” - Mega Fortune £2,700,000 jackpot won on the 3rd spin.

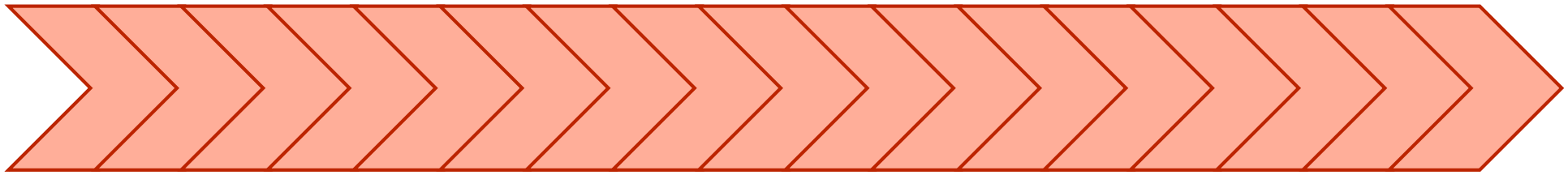
Event Store operations

- Append
- Validate ‘sequence’



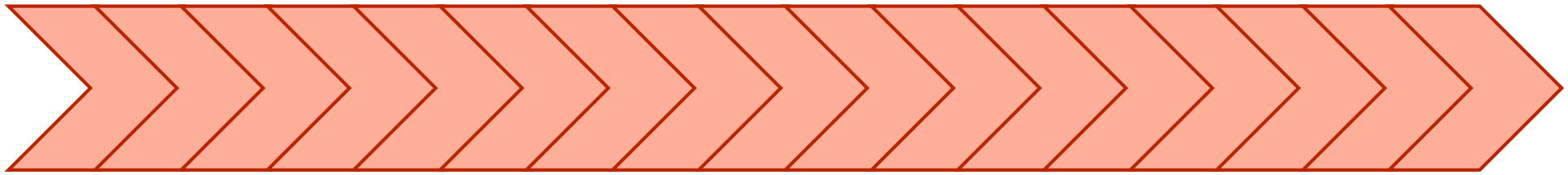
Event Store operations

- Full sequential read



Event Store operations

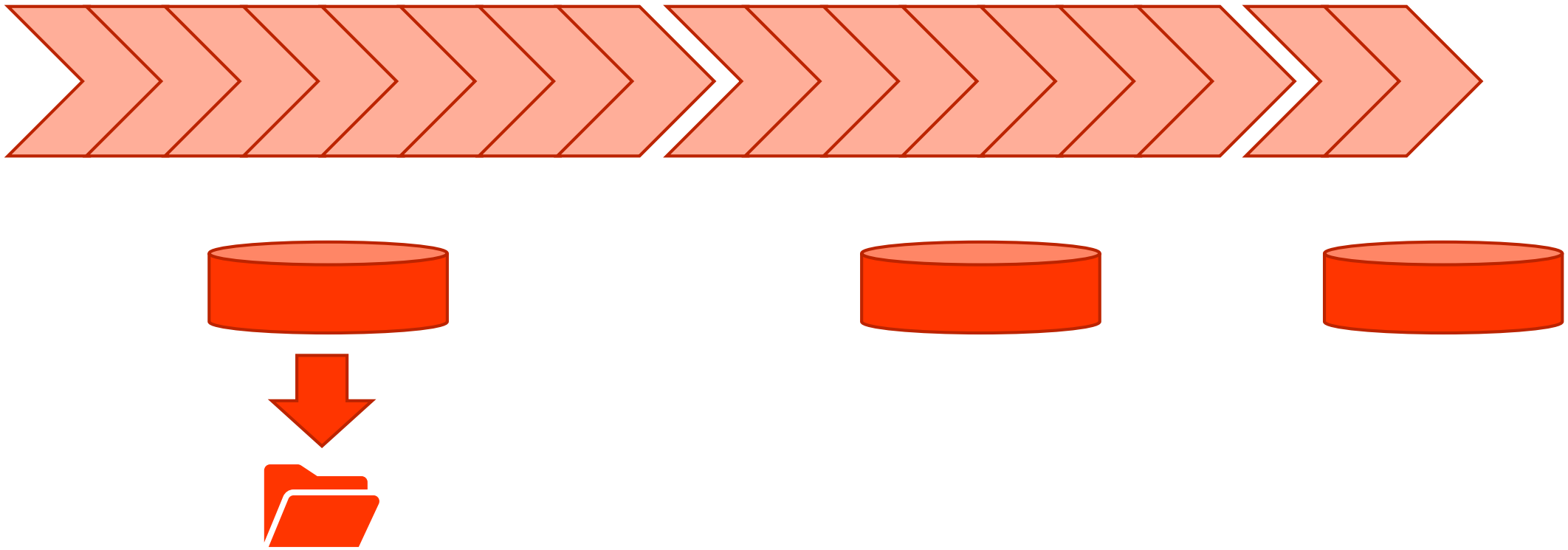
- Read aggregate's events



Serialized form

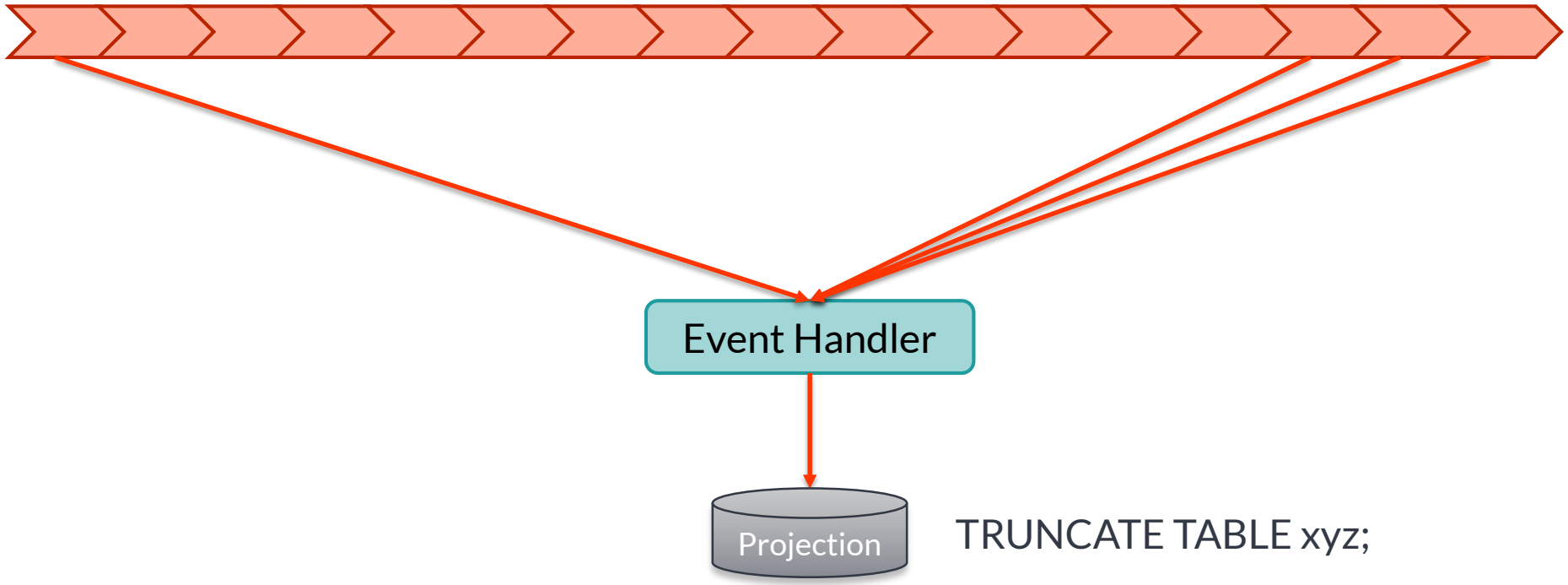
- Keep it small
- Use aliases & ~~abbreviations~~
- Carefully select serializer

Partitioning and archiving

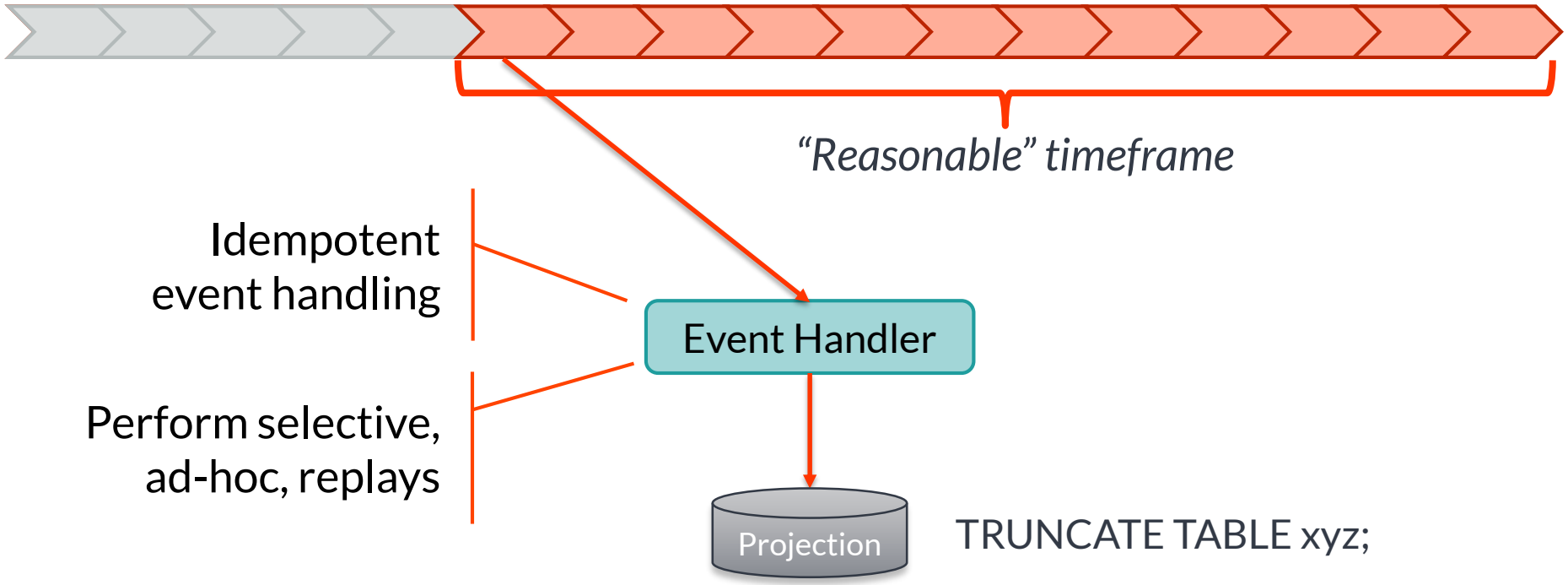


Loading aggregates - snapshots

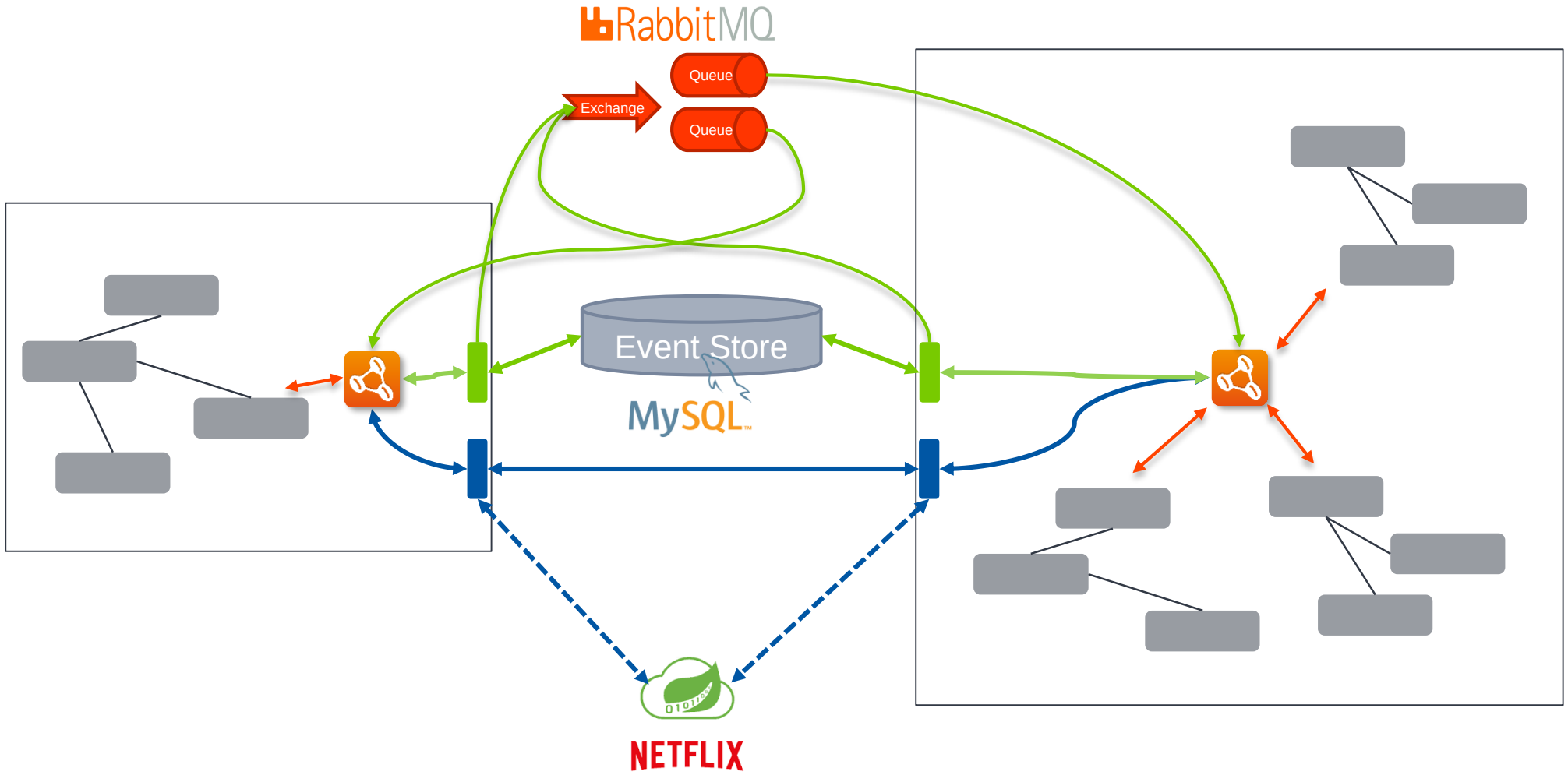
Replays

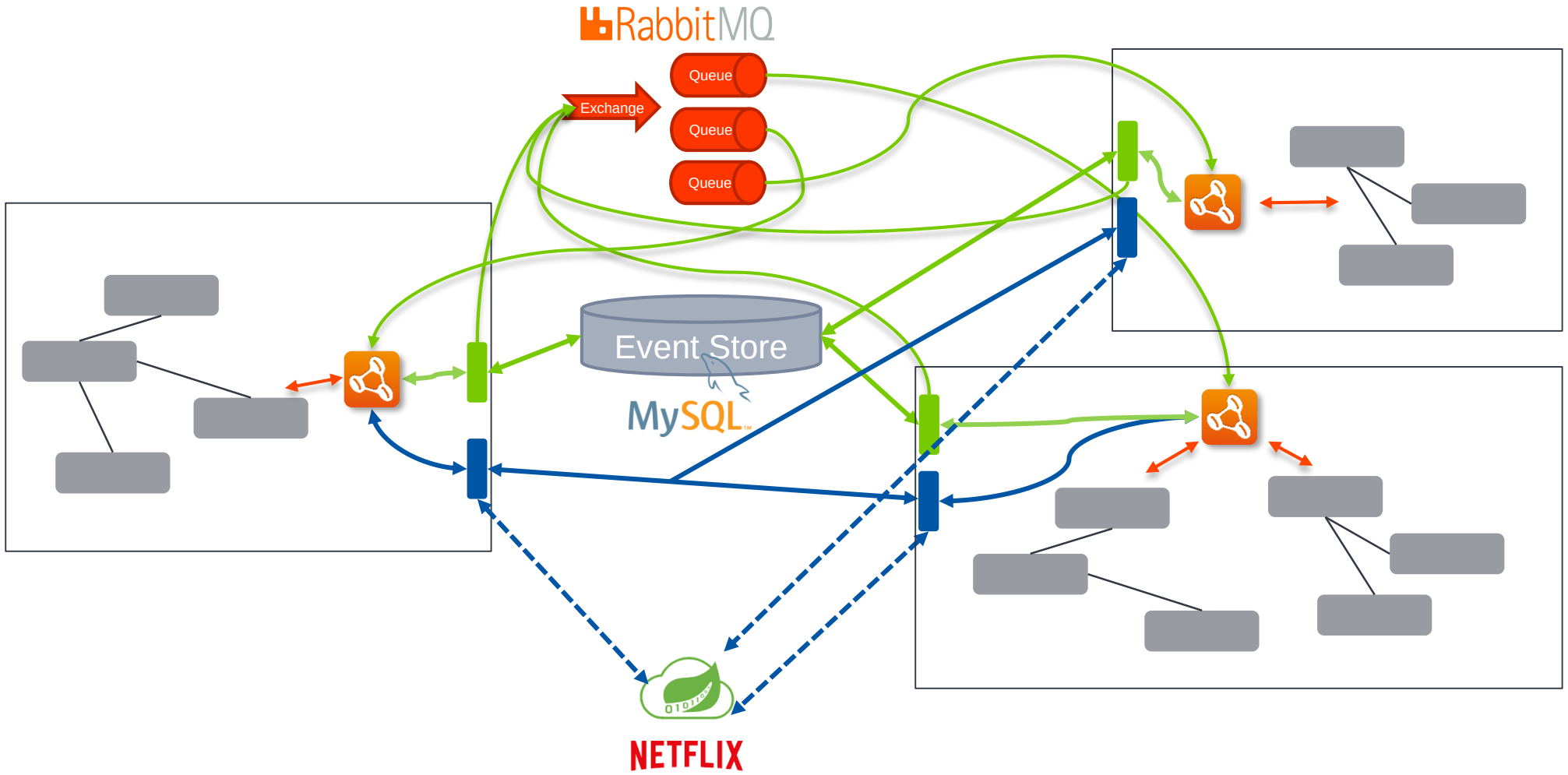


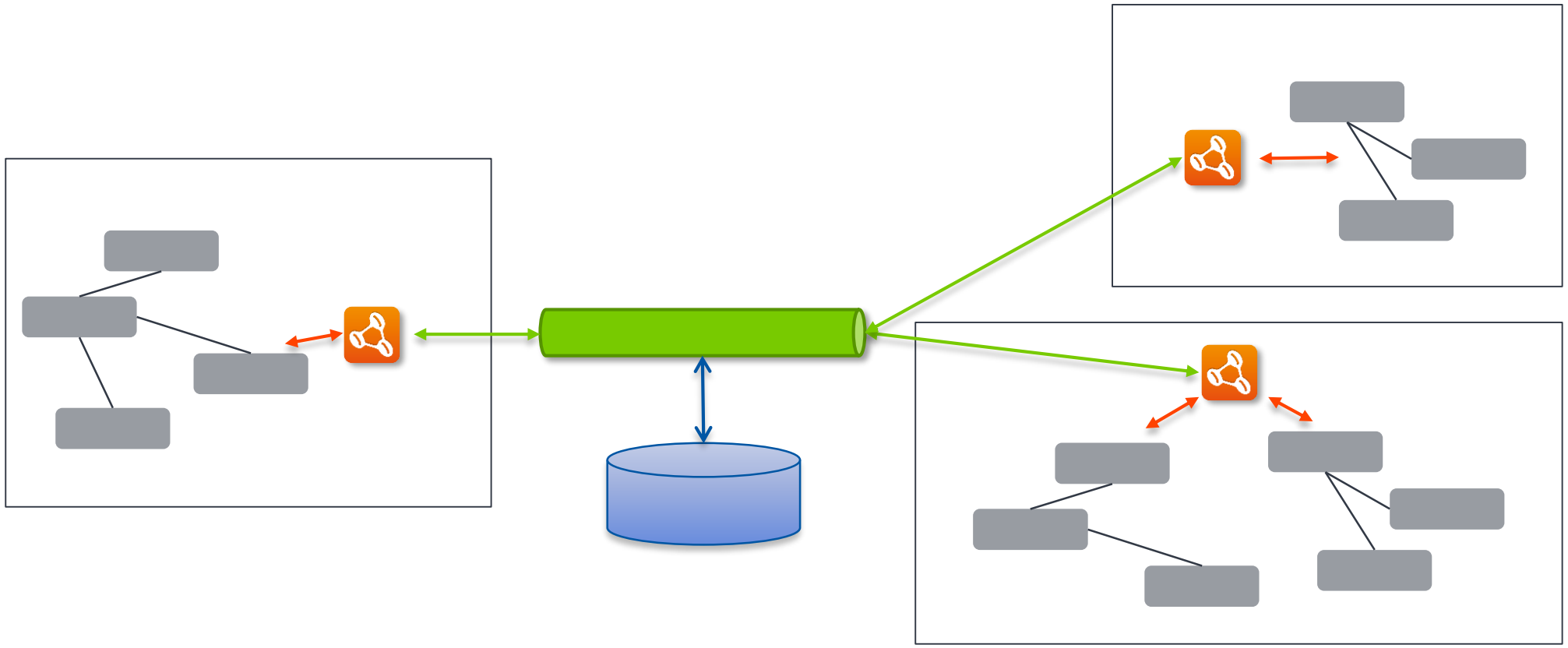
Partial replays



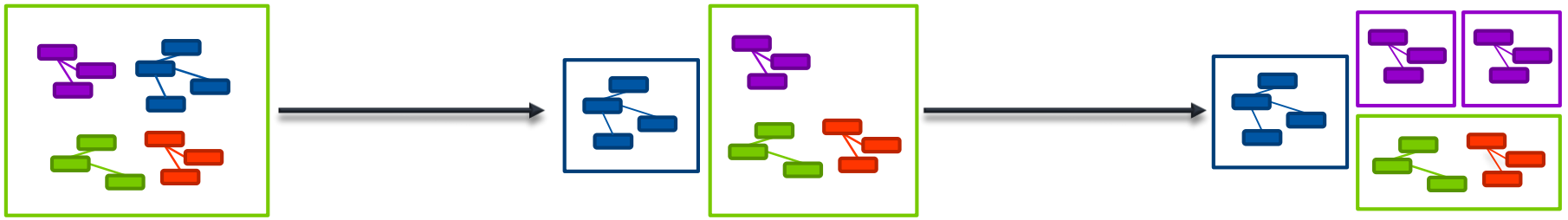

كبدة فهمة
الاصل اسكندرانى
B.TECH

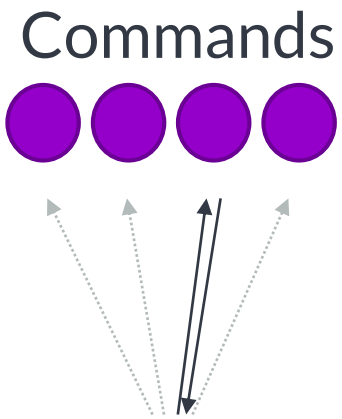

RabbitMQ
Queue
Exchange
Queue
Event Store
MySQL
NETFLIX

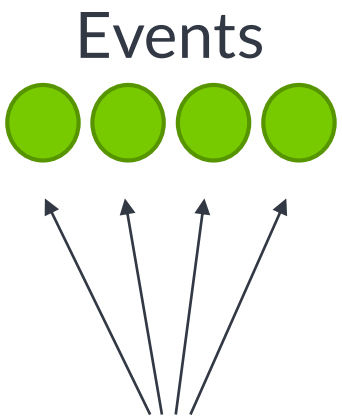

RabbitMQ
Queue
Exchange
Queue
Queue
Event Store
MySQL
NETFLIX

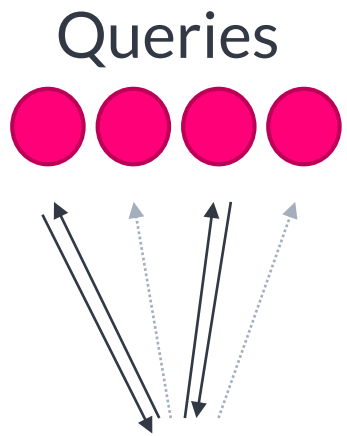


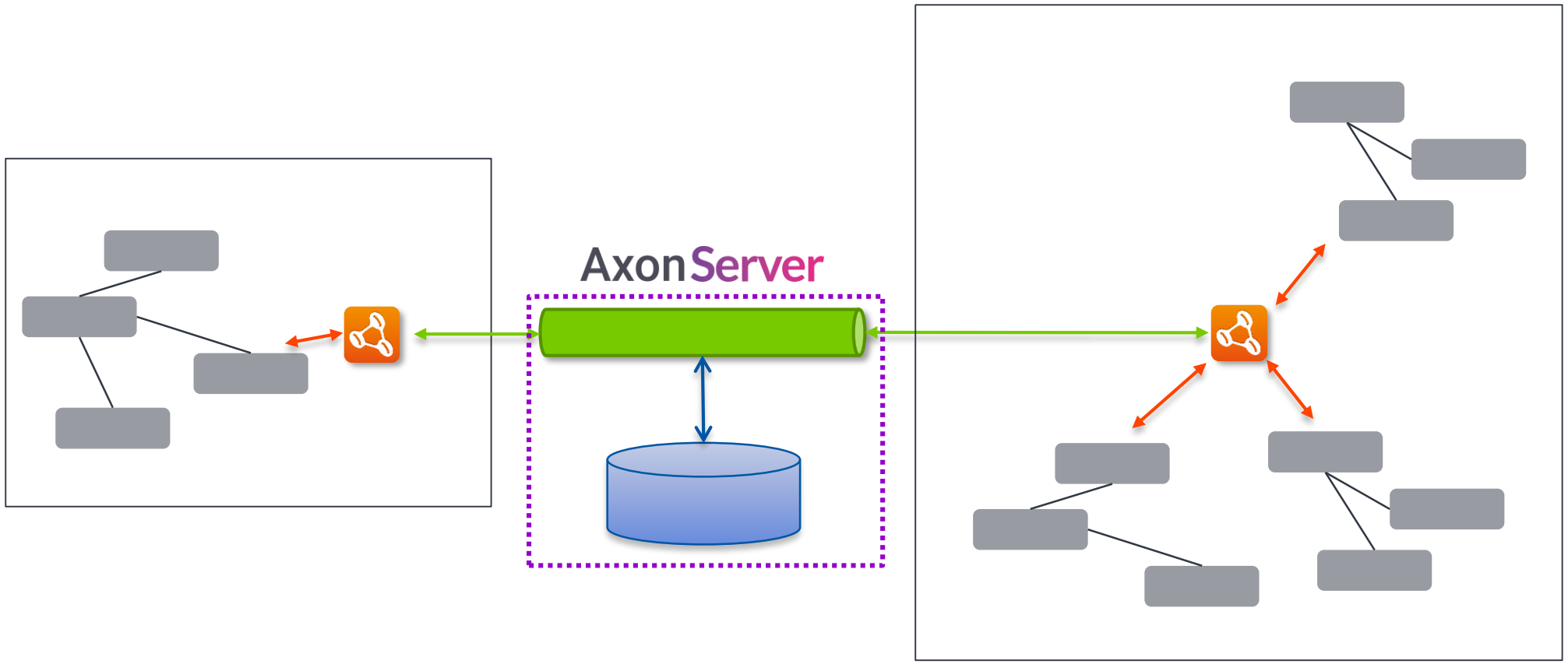
Evolutionary microservices

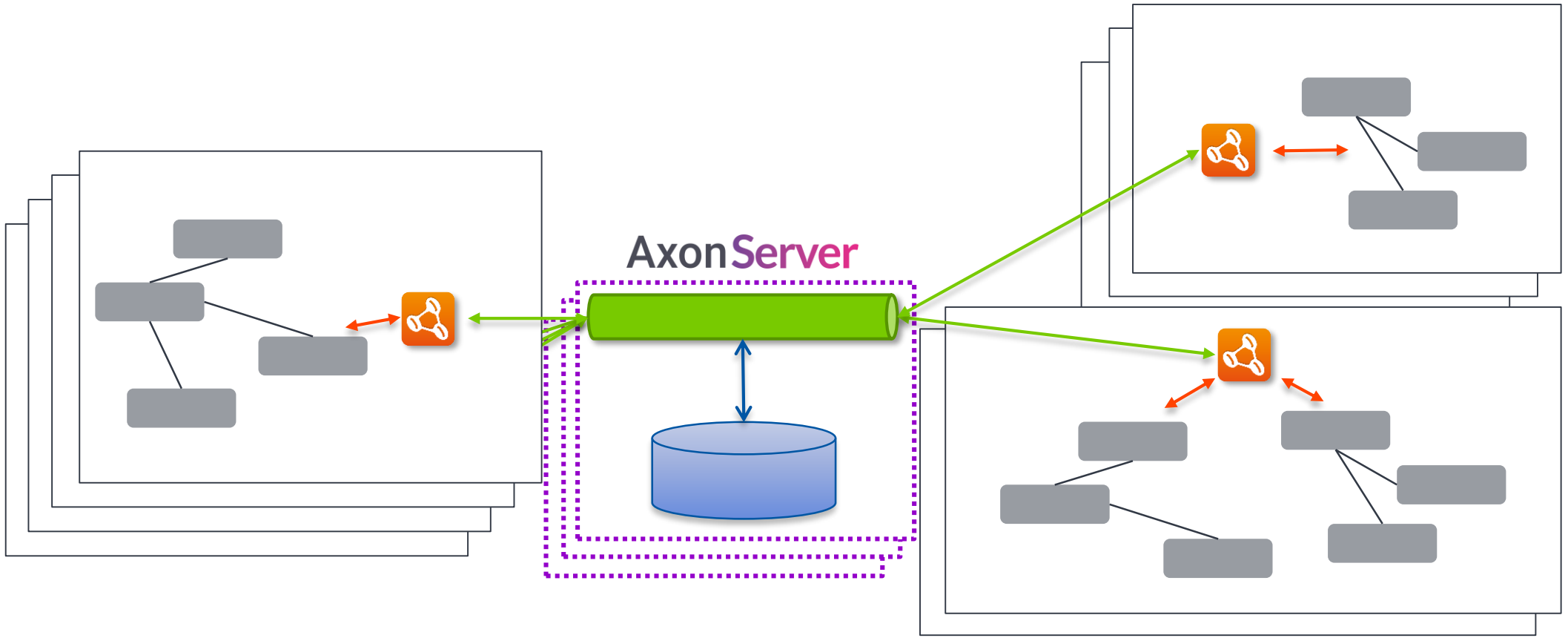






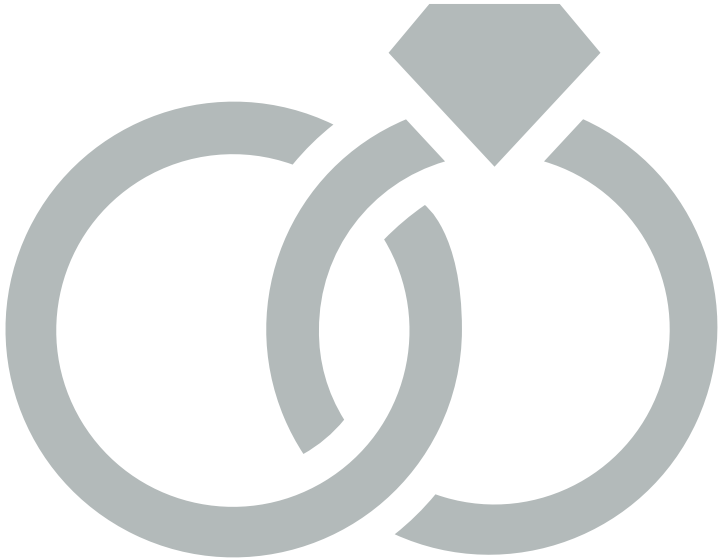



AxonServer


AxonServer

Unmanageable mess

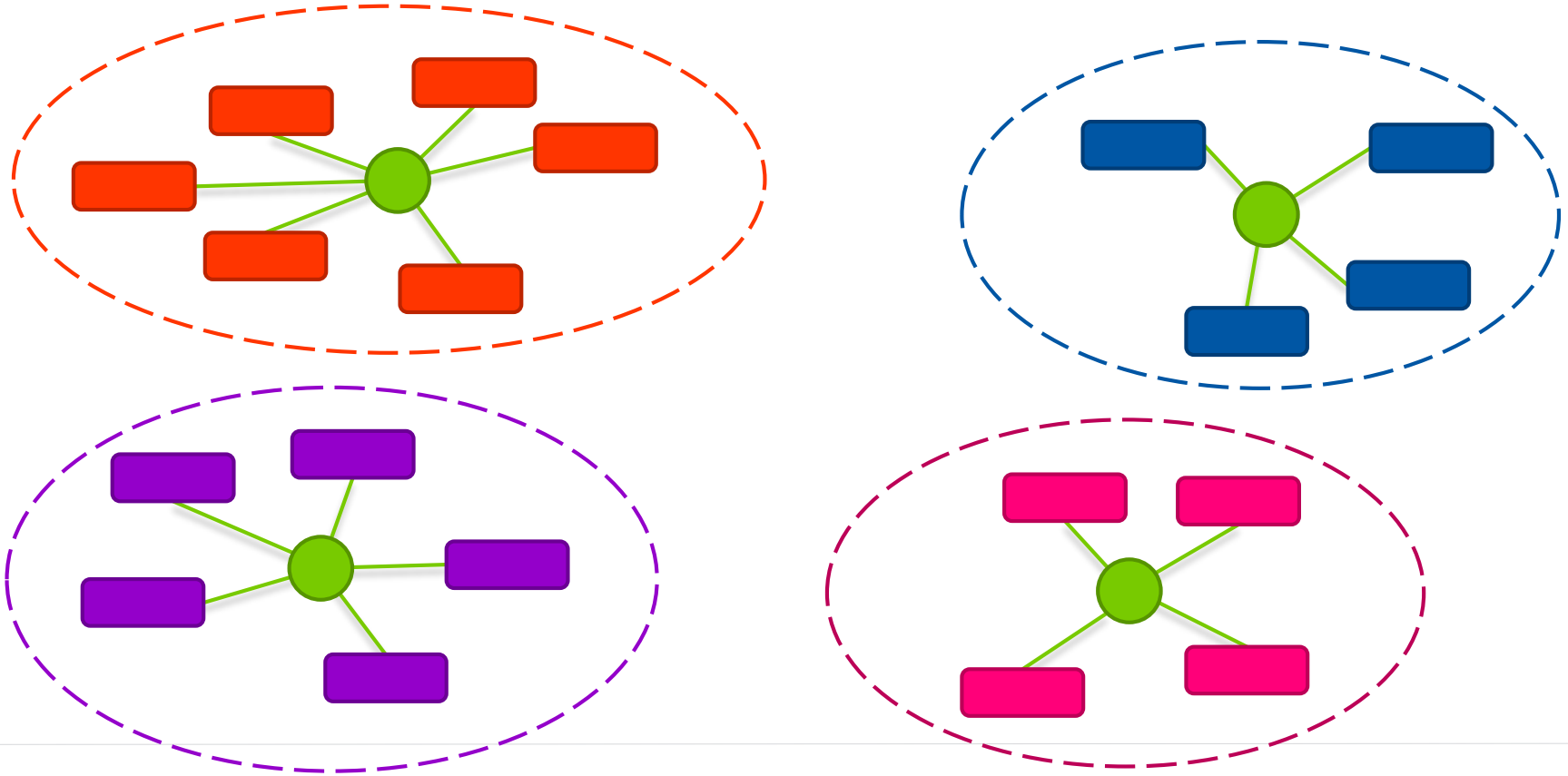
Communication = Contract



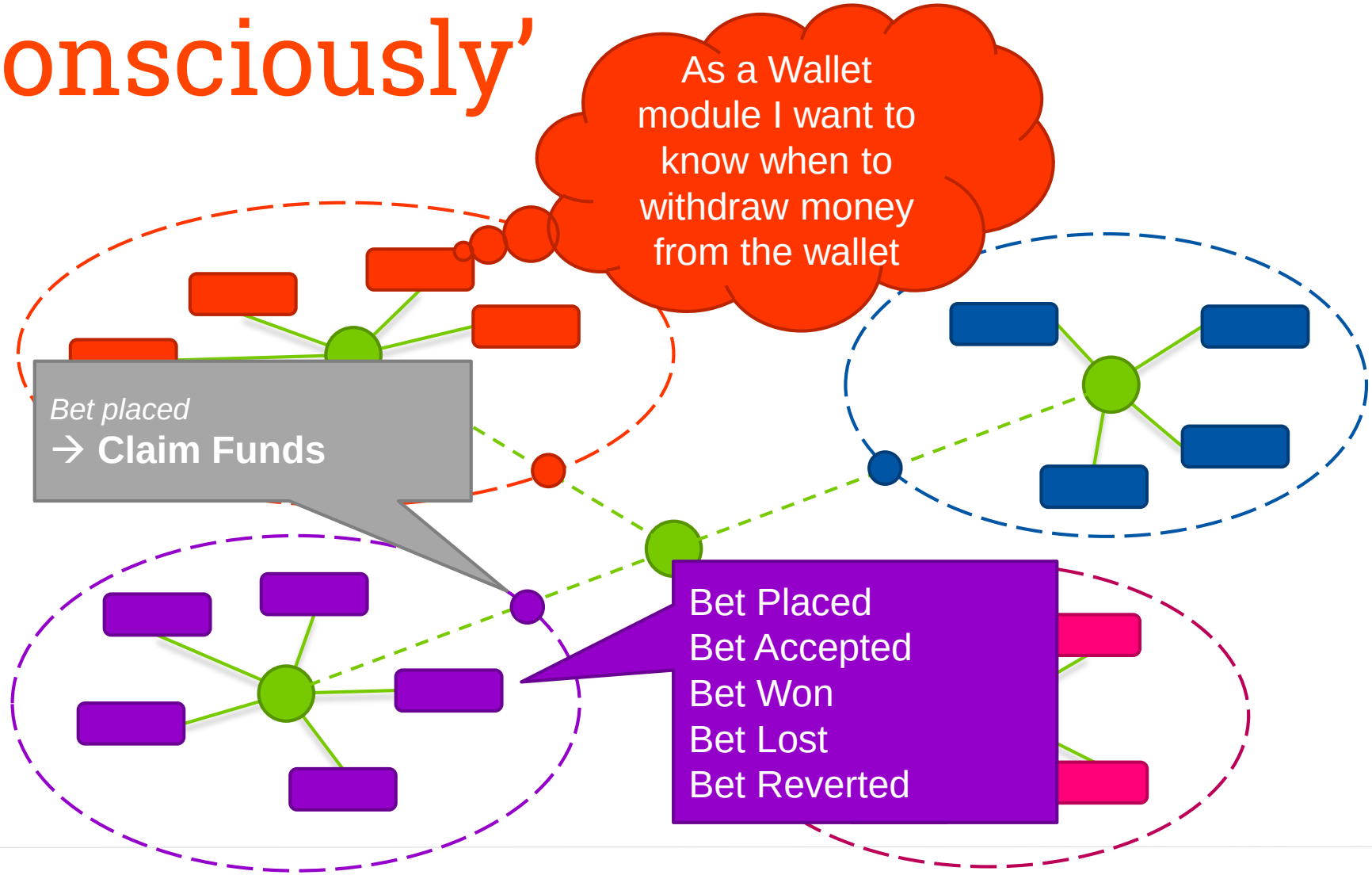
Bounded context

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’



Lessons learned

References

- Axon
  - axoniq.io
  - github.com/axonframework
  - @axonframework

- Casumo
  - casumo.com
  - @casumotech