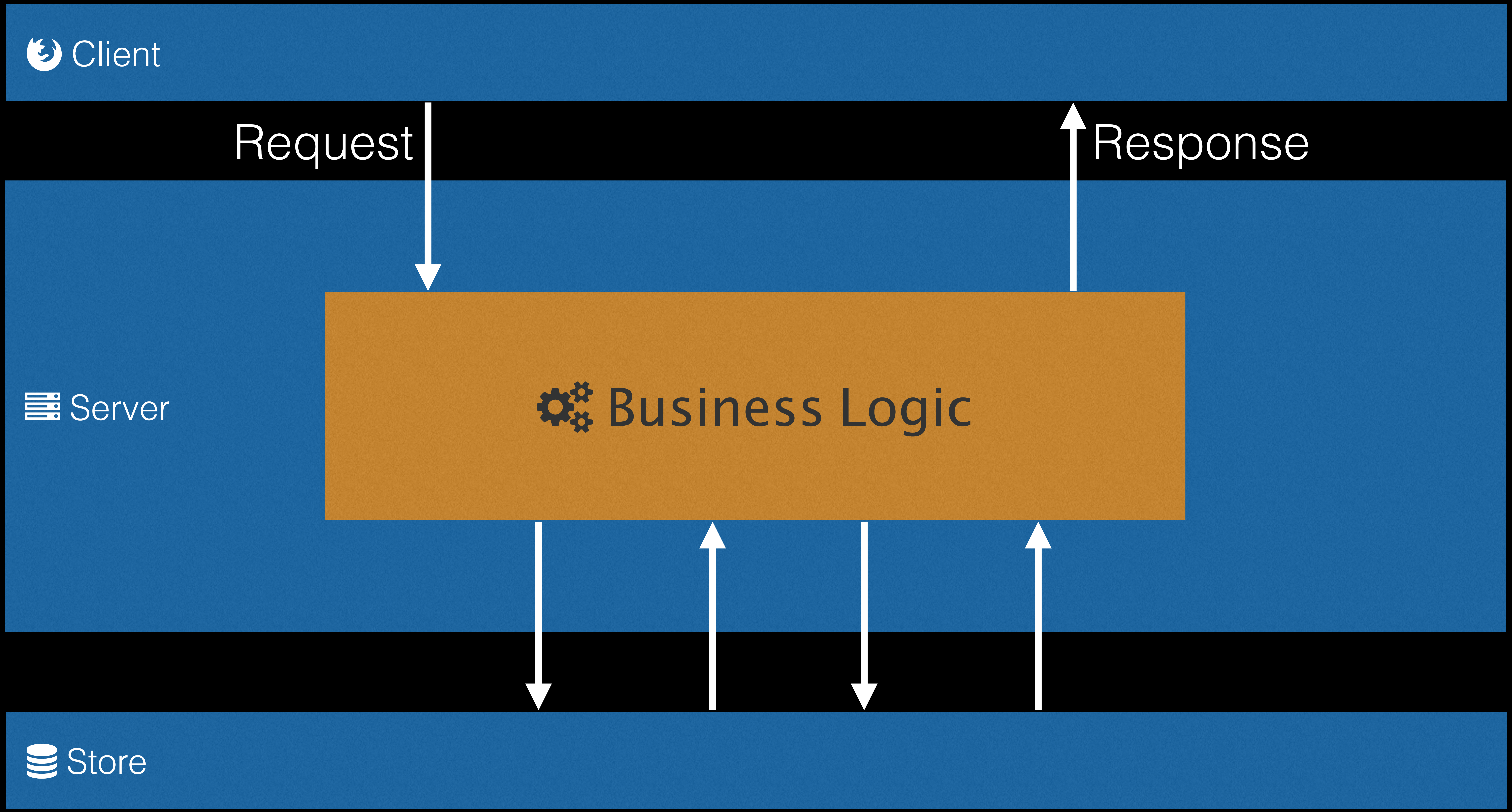
CQRS at Enterprise Scale

Graham Brooks
Coding Architect

@grahamcbrooks

graham@grahambrooks.com

grahambrooks.com/talks

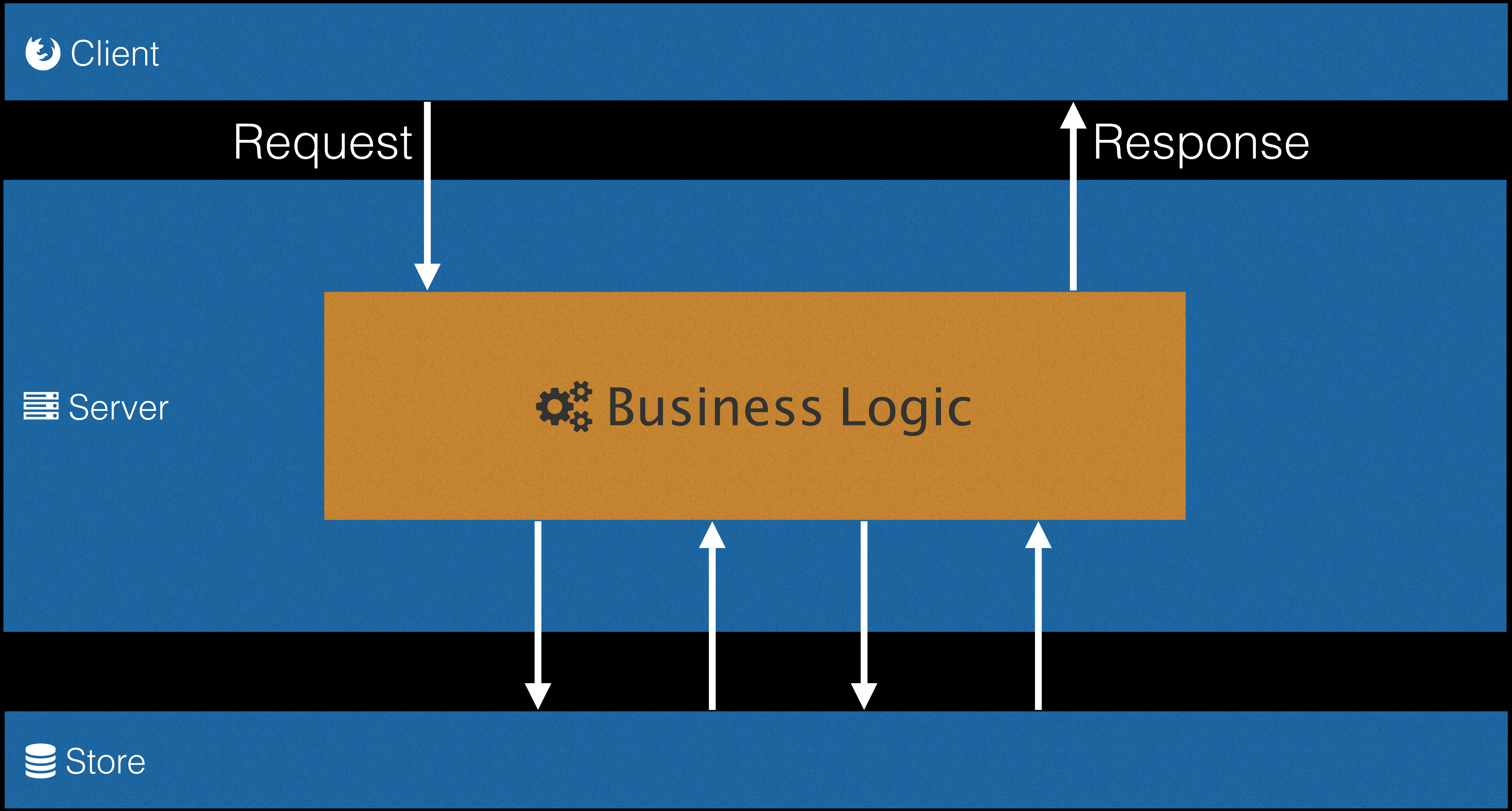

Client
Request
Response
Server
Business Logic
Store

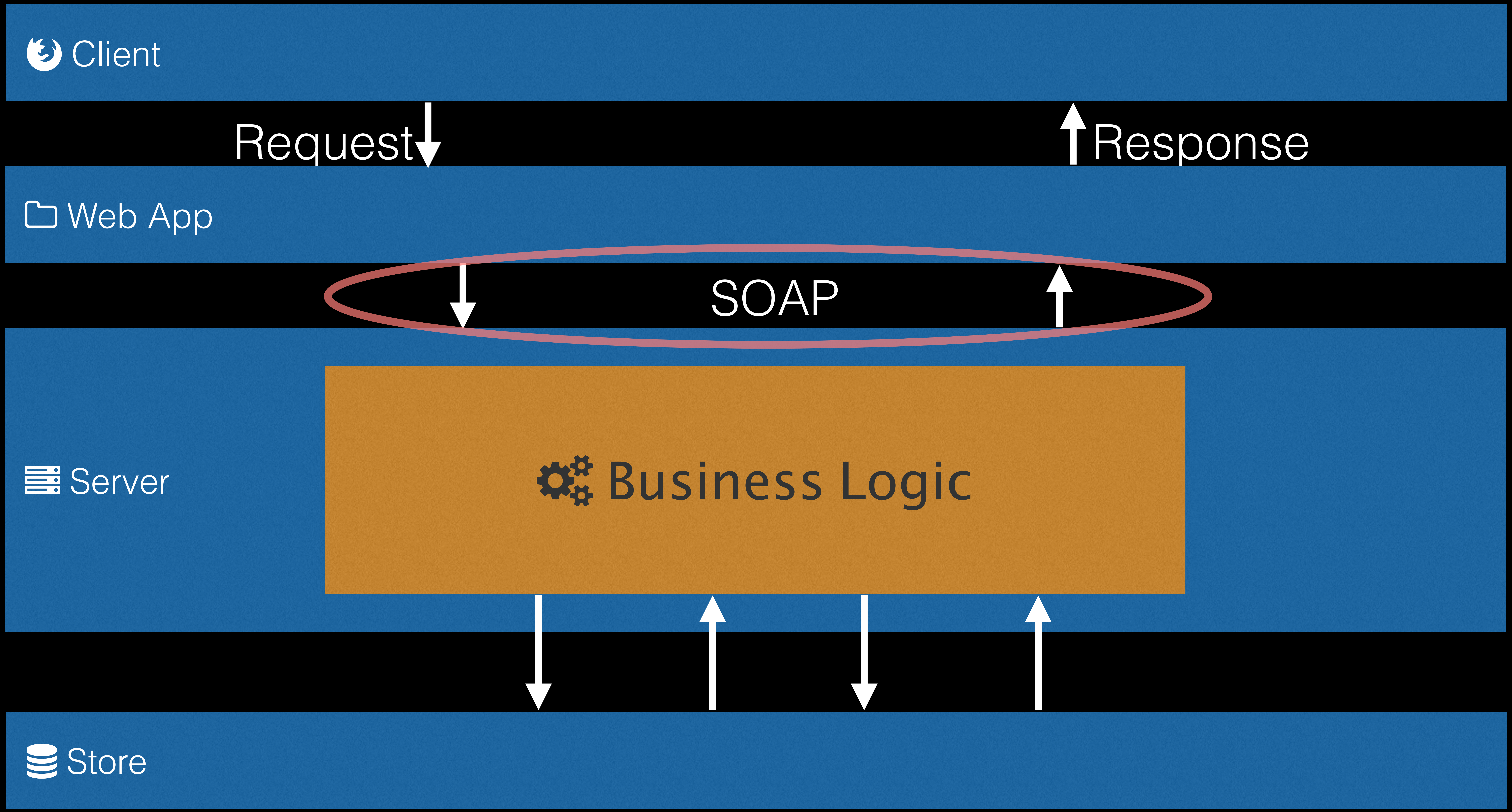
Simple

Object

Access

Protocol


Client
Request
Response
Server
Business Logic
Store


Client
Request
Response
Web App
SOAP
Server
Business Logic
Store

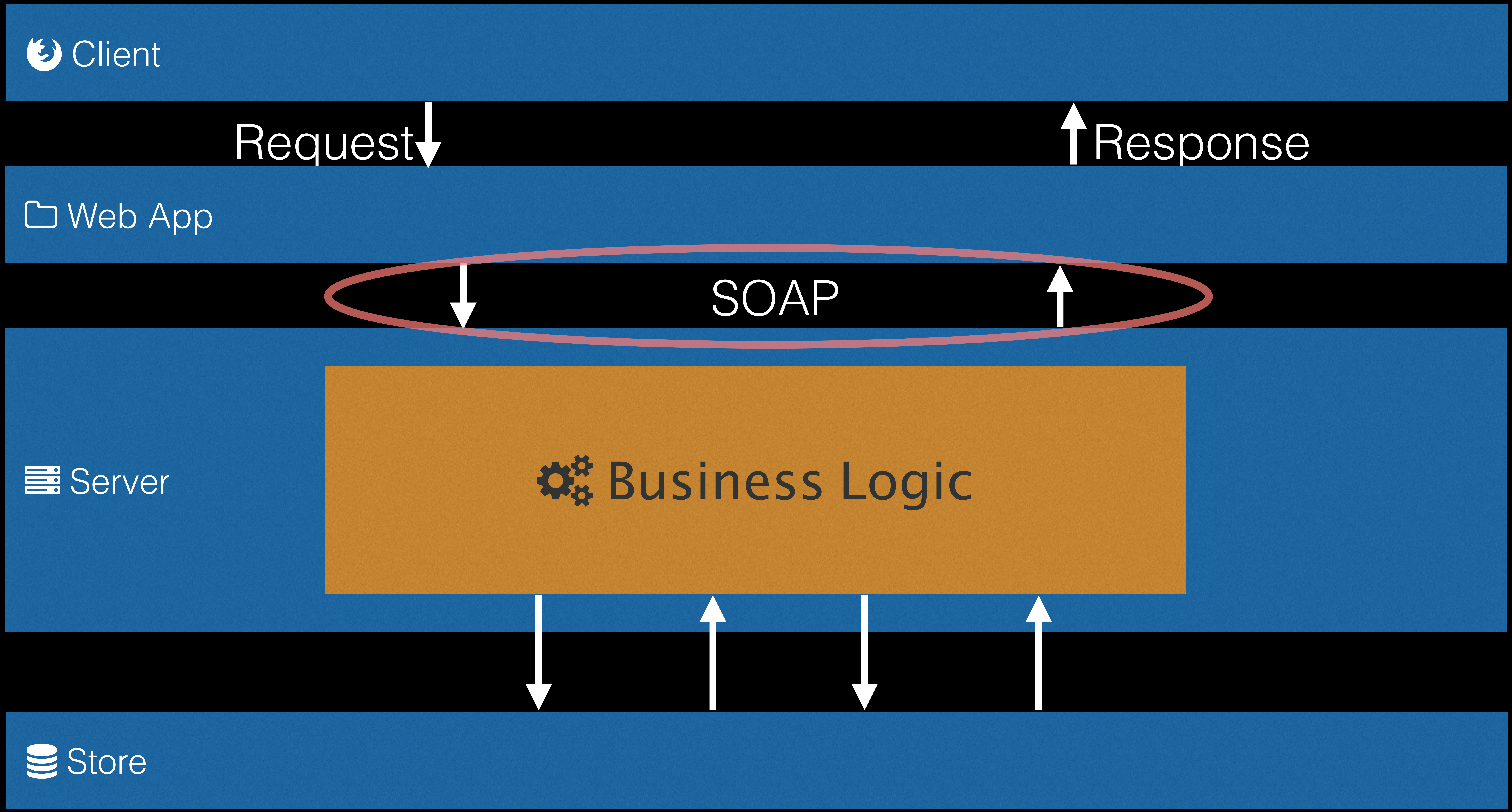
```
<definitions name="EndorsementSearch"
  targetNamespace="http://namespaces.snowboard-info.com" xmlns:es="http://www.snowboard-info.com/EndorsementSearch.wsdl"
  xmlns:esxsd="http://schemas.snowboard-info.com/EndorsementSearch.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
>

  <!-- omitted types section with content model schema info -->

  <message name="GetEndorsingBoarderRequest">
    <part name="body" element="esxsd:GetEndorsingBoarder"/>
  </message>

  <message name="GetEndorsingBoarderResponse">
    <part name="body" element="esxsd:GetEndorsingBoarderResponse"/>
  </message>

  <portType name="GetEndorsingBoarderPortType">
    <operation name="GetEndorsingBoarder">
      <input message="es:GetEndorsingBoarderRequest"/>
      <output message="es:GetEndorsingBoarderResponse"/>
      <fault message="es:GetEndorsingBoarderFault"/>
    </operation>
  </portType>

  <binding name="EndorsementSearchSoapBinding"
           type="es:GetEndorsingBoarderPortType">
    <soap:binding style="document"
```

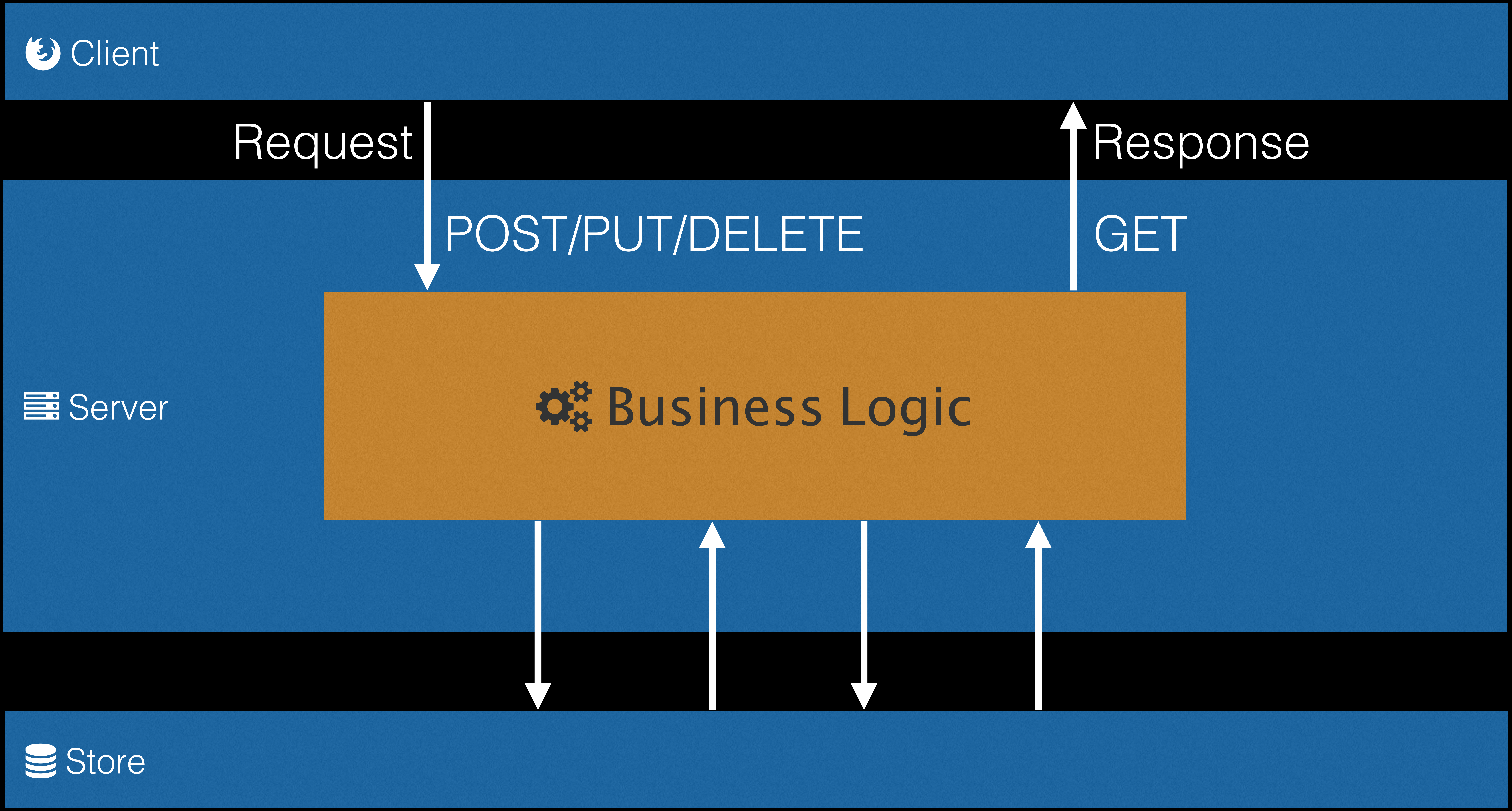
source w3c.org

SOAP

- XML based
- Intolerant to change versioning is particularly difficult
- Middleware not (web) client facing
- Its XML based

REpresentational
State
Transfer


Client
Request
Response
Web App
SOAP
Server
Business Logic
Store

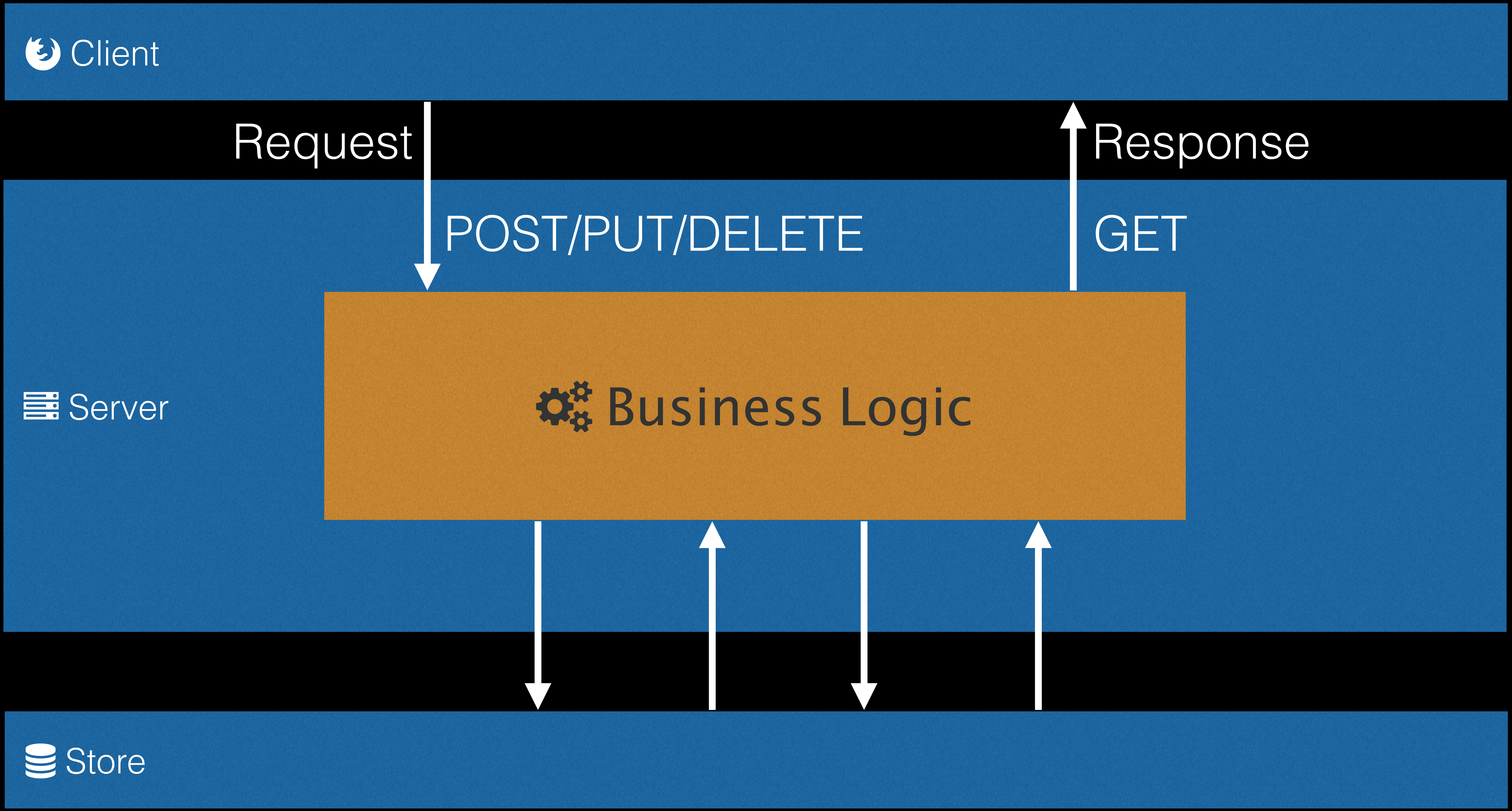

Client
Request
Response
POST/PUT/DELETE
GET
Server
Business Logic
Store

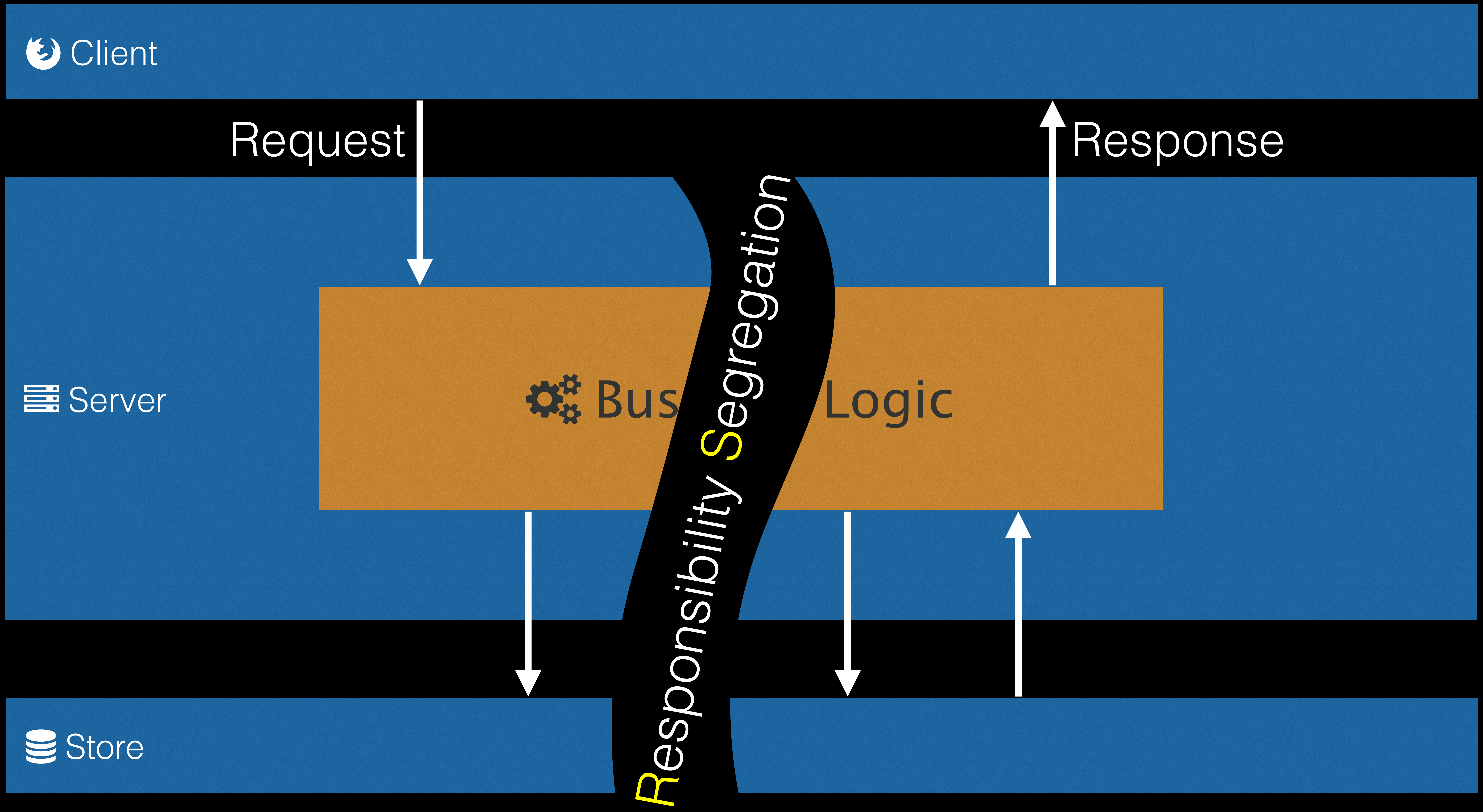
REST

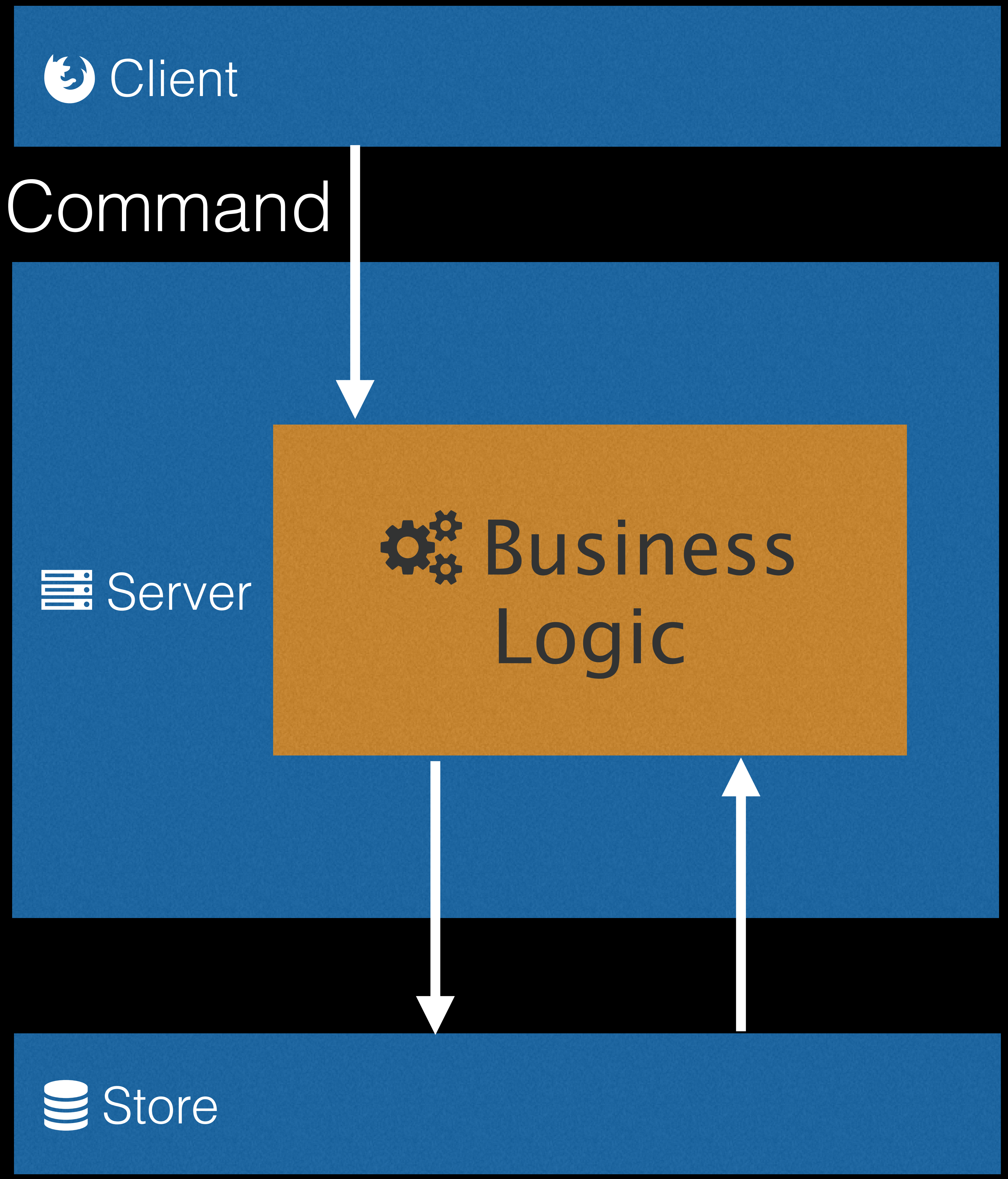
- Uniform Interface Client–server
- Stateless
- Cacheable
- Layered system
- Identification of resources
- Manipulation of resources through these representations
- Self-descriptive messages
- Hypermedia as the engine of application state (HATEOAS)

Command
Query
Responsibility
Segregation

Responsibility
Segregation


Client
Request
Response
POST/PUT/DELETE
GET
Server
Business Logic
Store

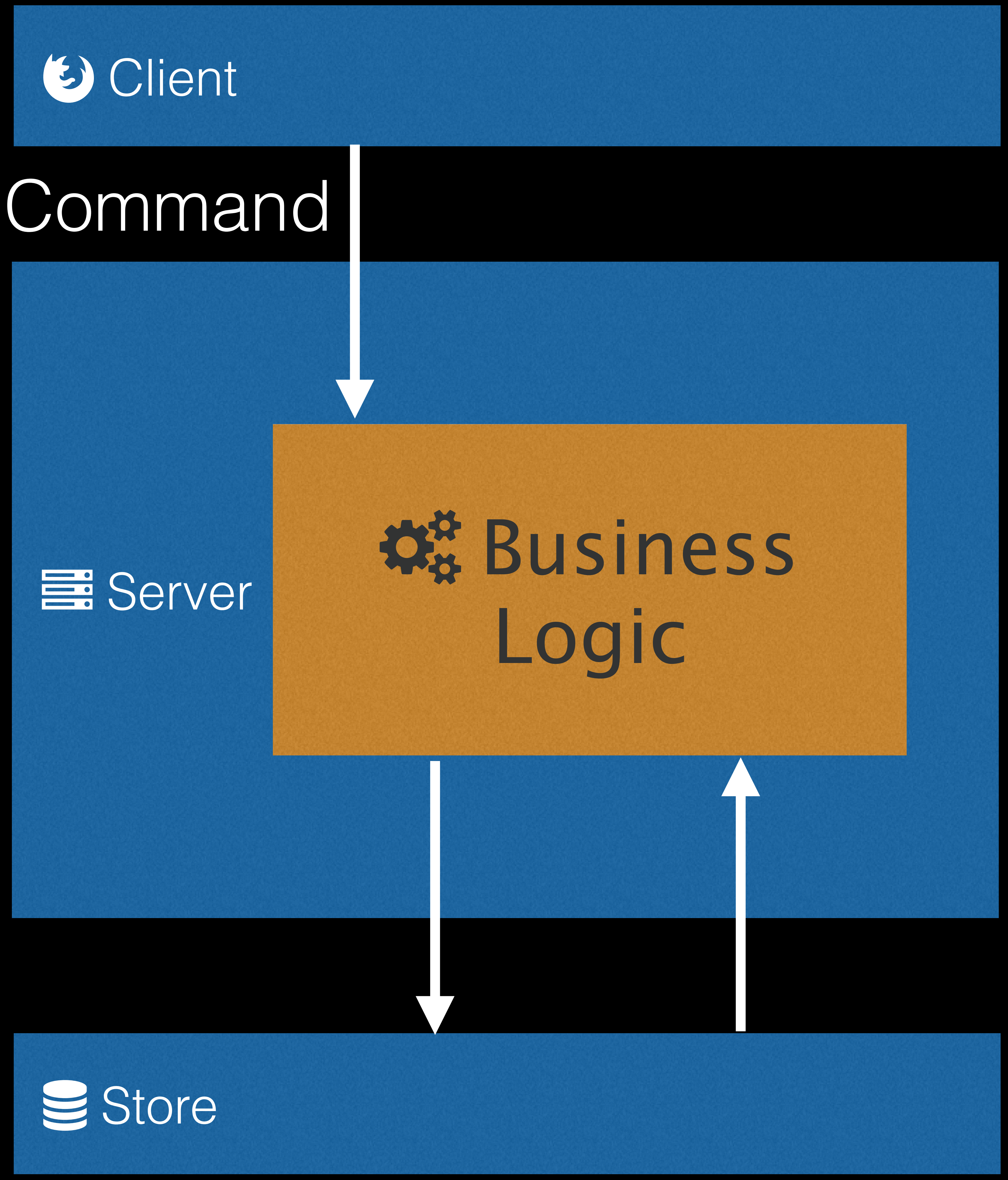

Client
Request
Response
Server
Bus
Logic
Responsibility Segregation
Store

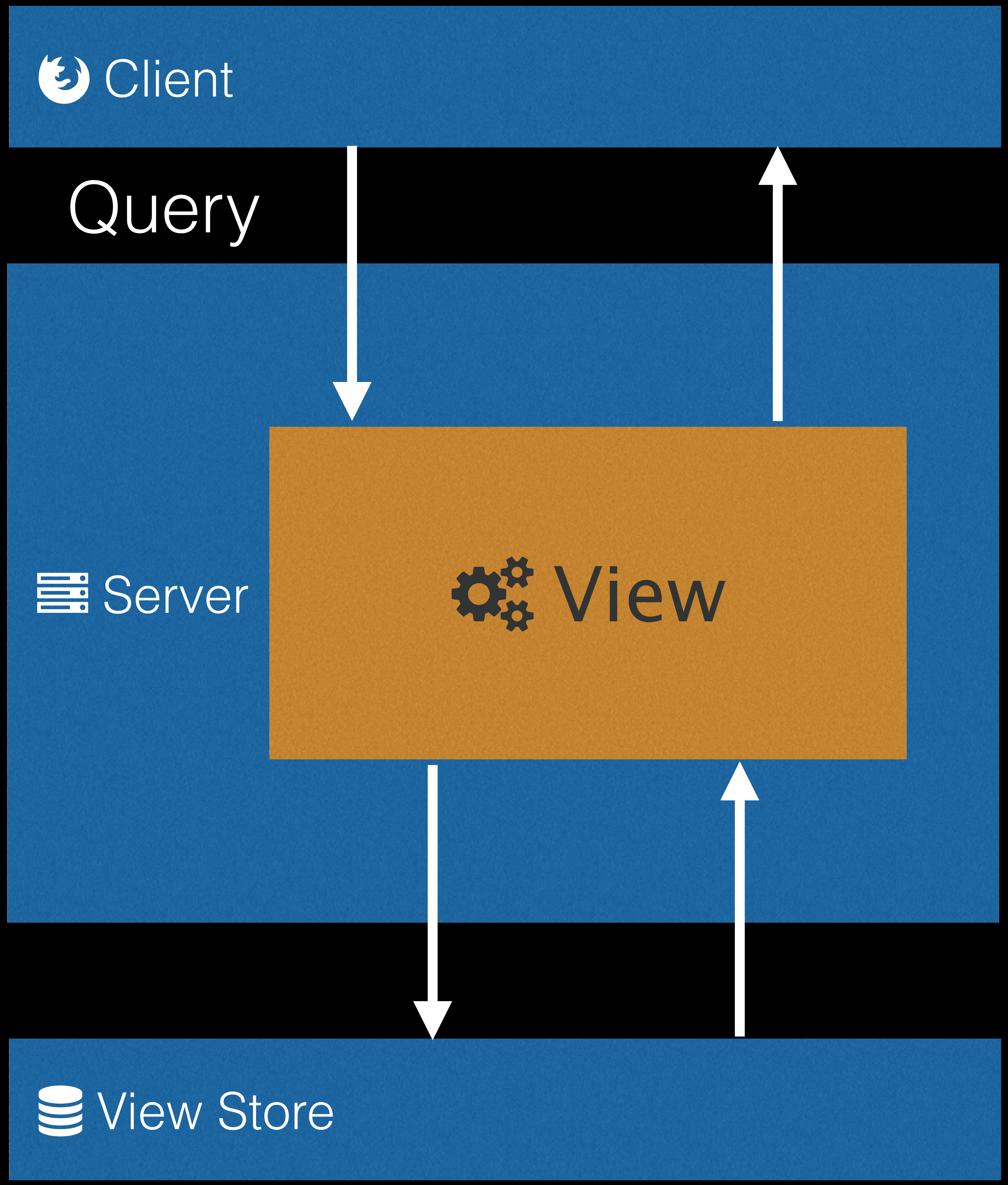

Client
Command
Server
Business Logic
Store

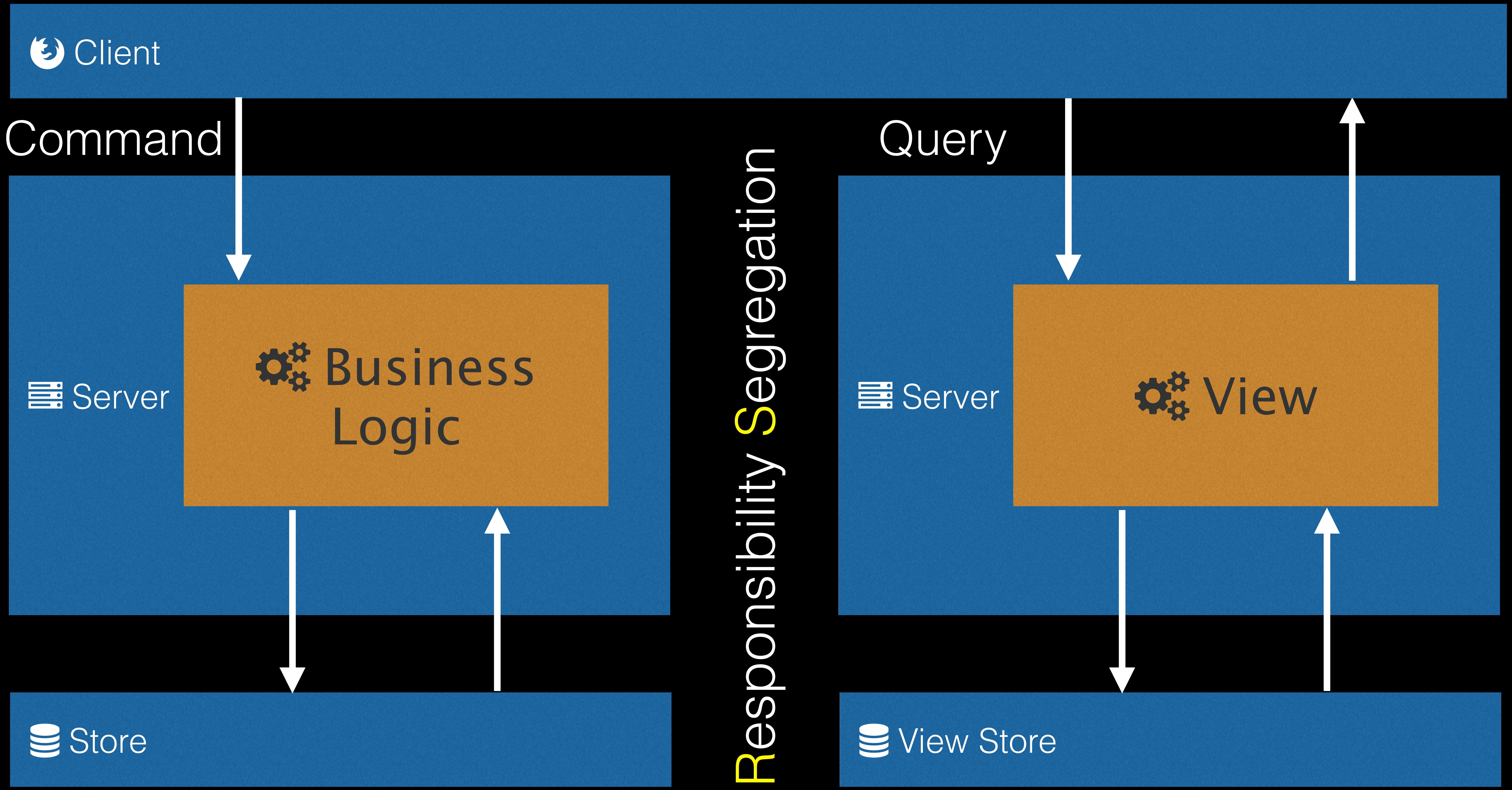
Commands

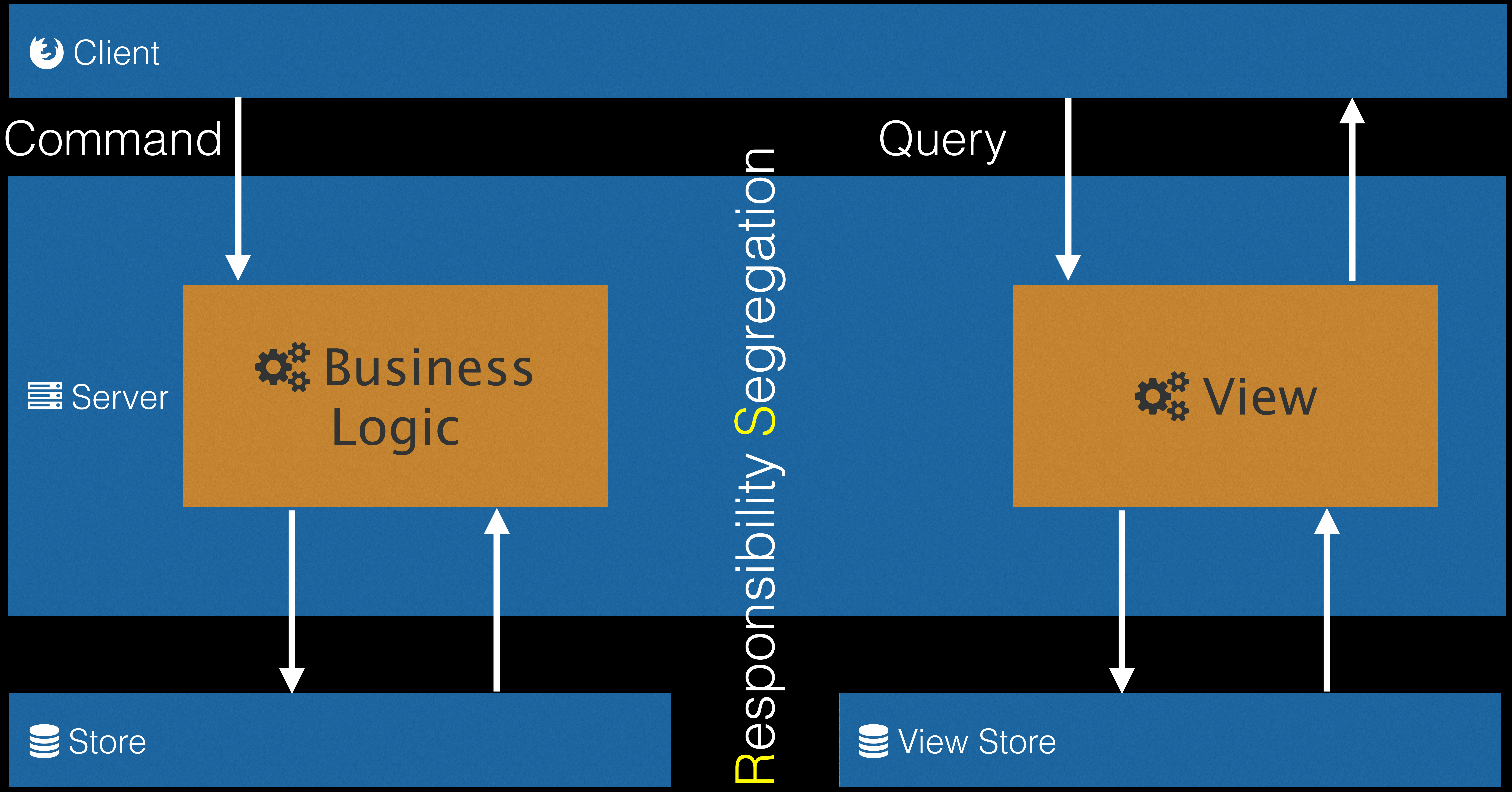
- What makes a good command?
- RESTful? - POST/PUT/DELETE
- SOAP - set()
- Something else
    - POST /customers/123/addresses

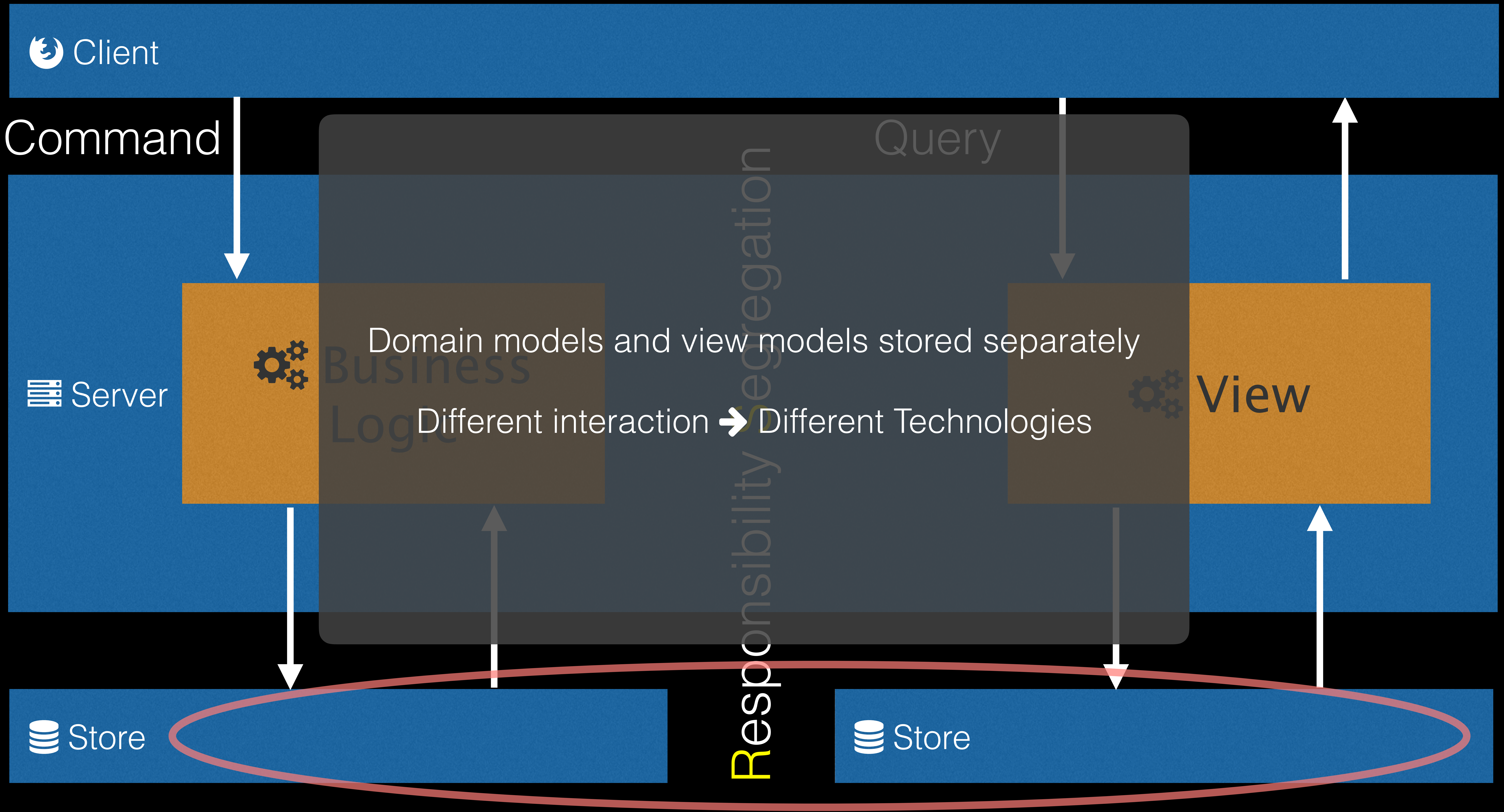
```
{
    "command": "change-address",
    "reason": "moved",
    "address": {...}
}
```




Client
Query
Server
View
View Store

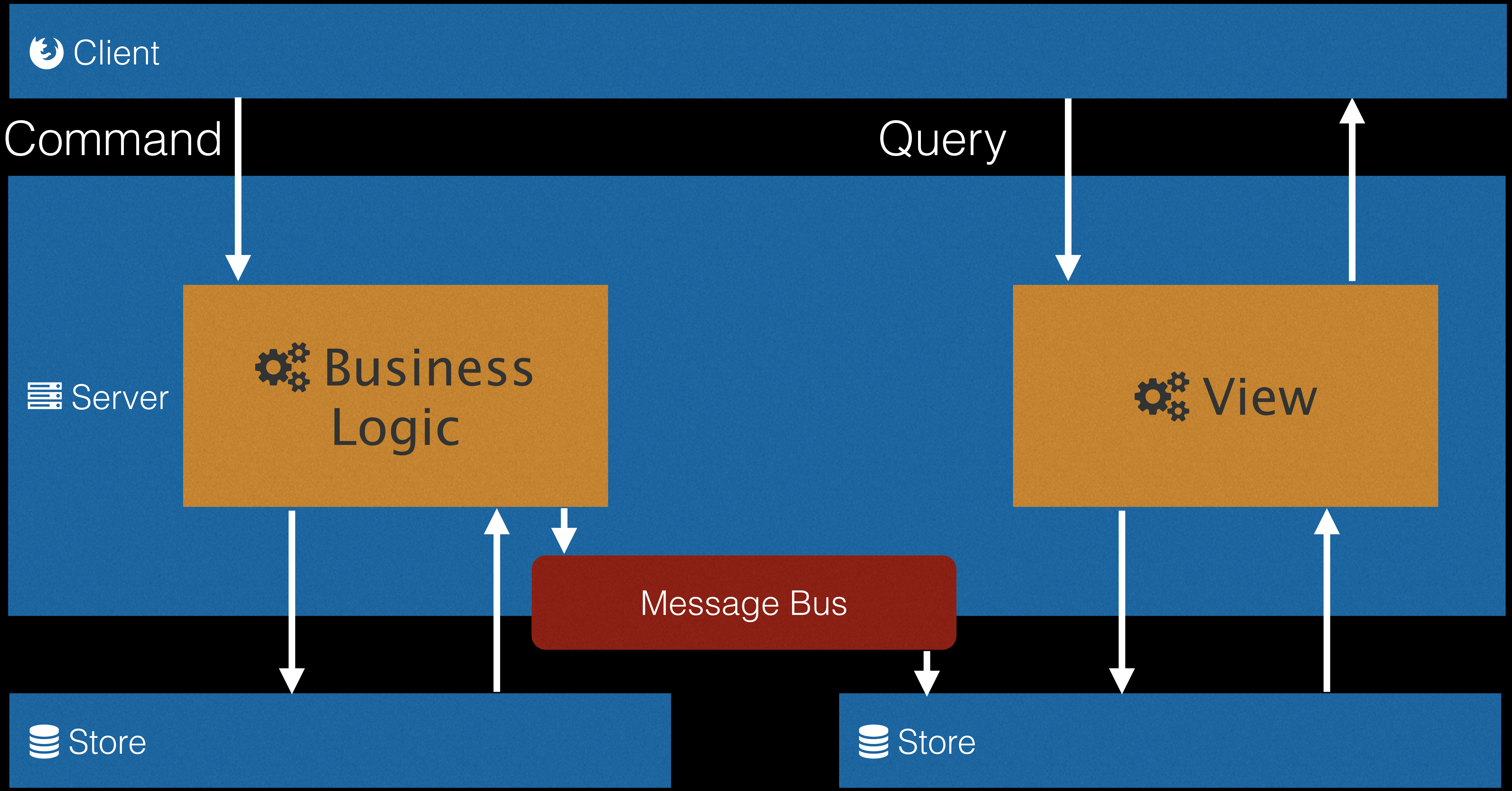

Client
Command
Query
Responsibility Segregation
Server
Business Logic
Server
View
Store
View Store

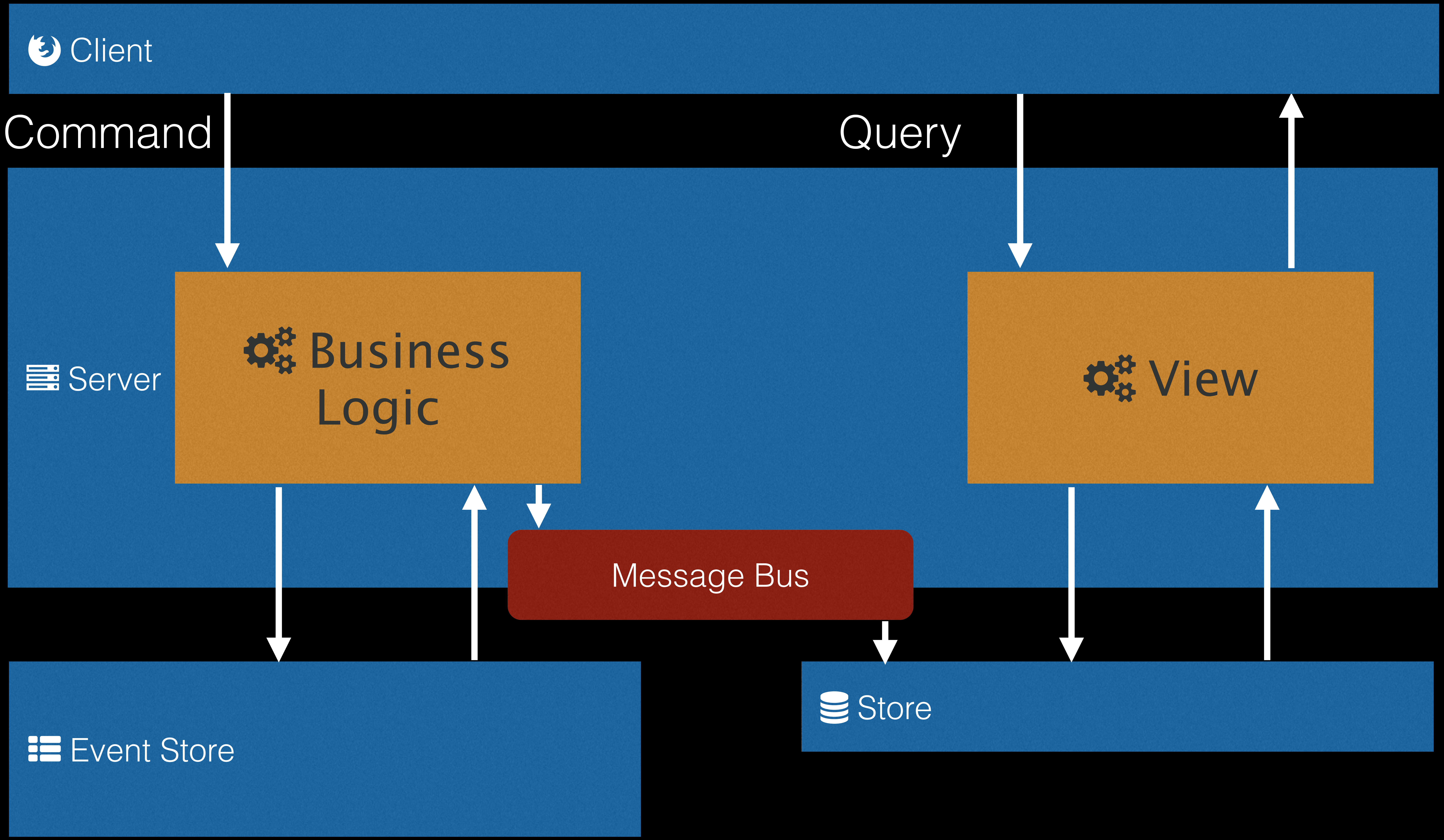

Client
Command
Query
Server
Business Logic
View
Responsibility Segregation
Store
View Store

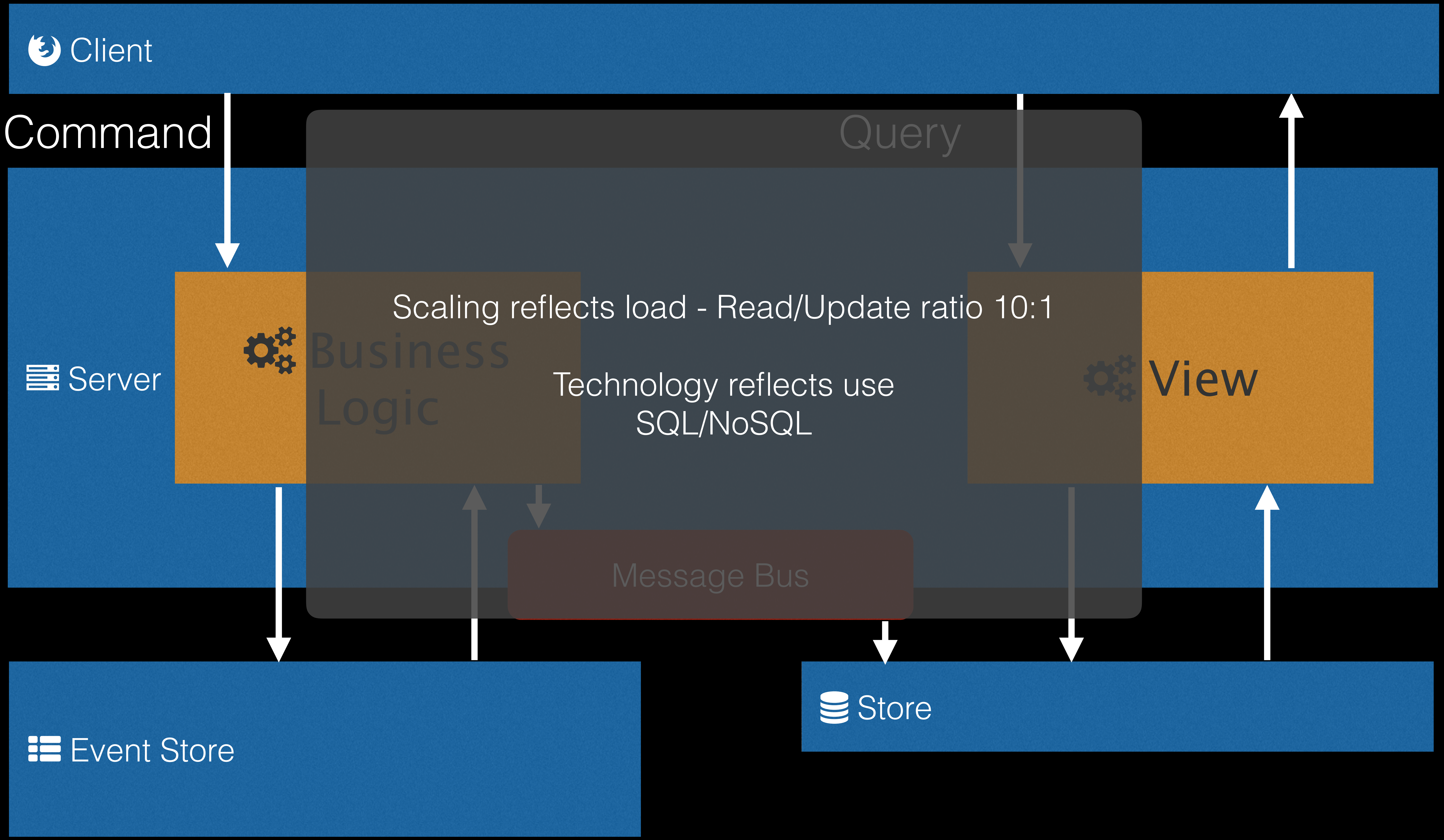


Domain models and view models stored separately

Different interaction ➔ Different Technologies

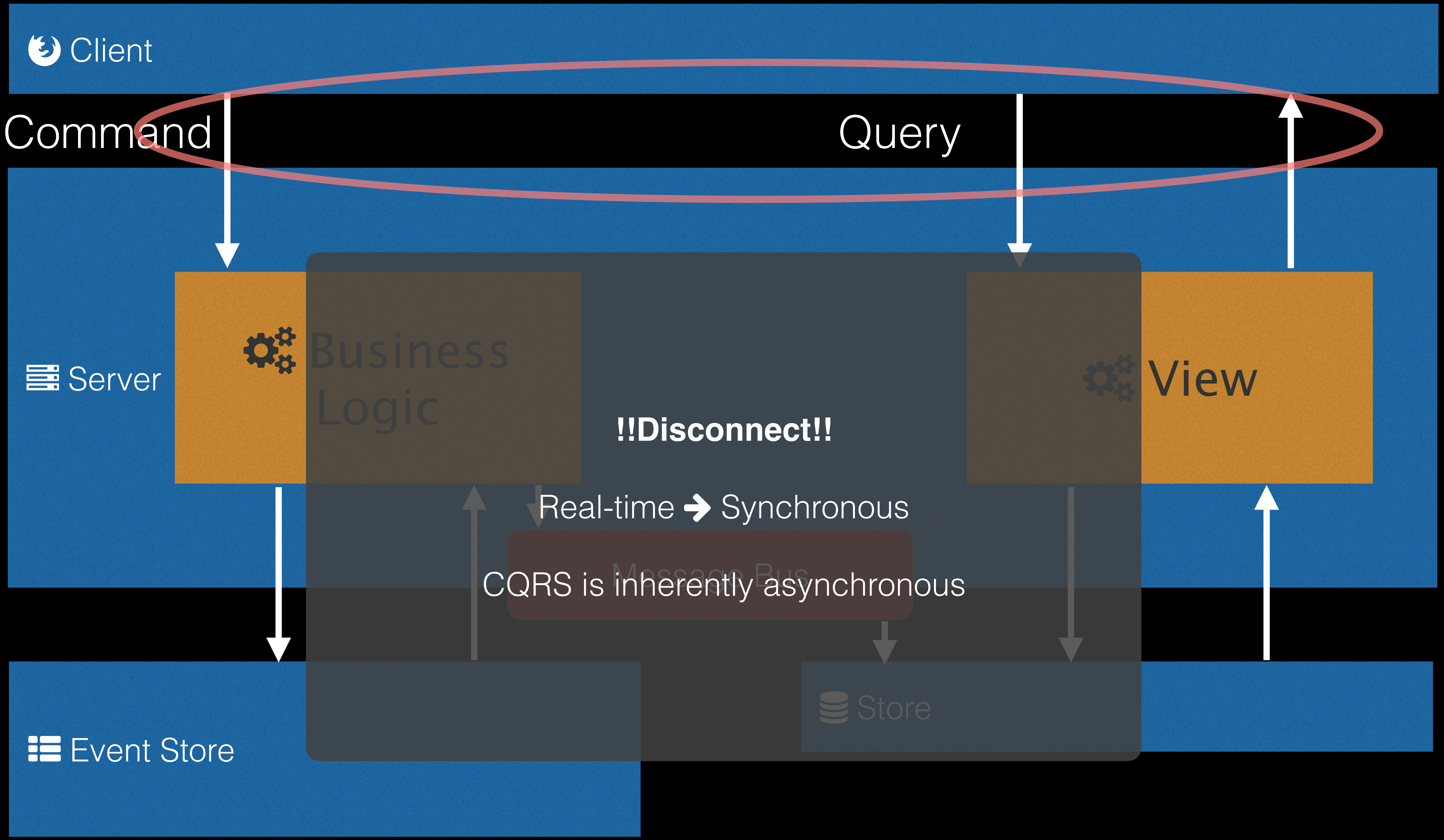

Client
Command
Query
Server
Business Logic
View
Message Bus
Store
Store


Client
Command
Query
Server
Business Logic
View
Message Bus
Event Store
Store



Scaling reflects load - Read/Update ratio 10:1

Technology reflects use
SQL/NoSQL



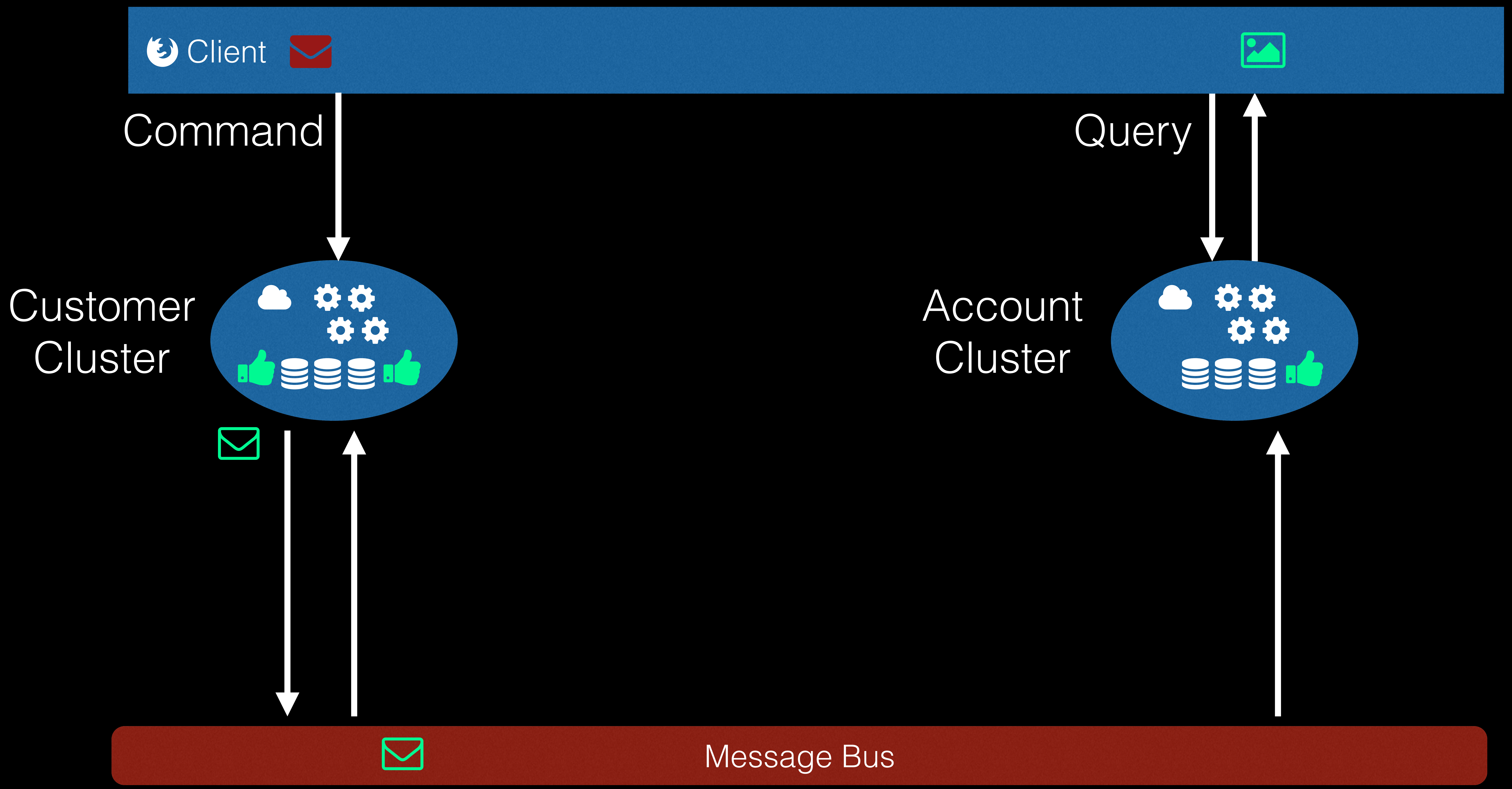
!!Disconnect!!

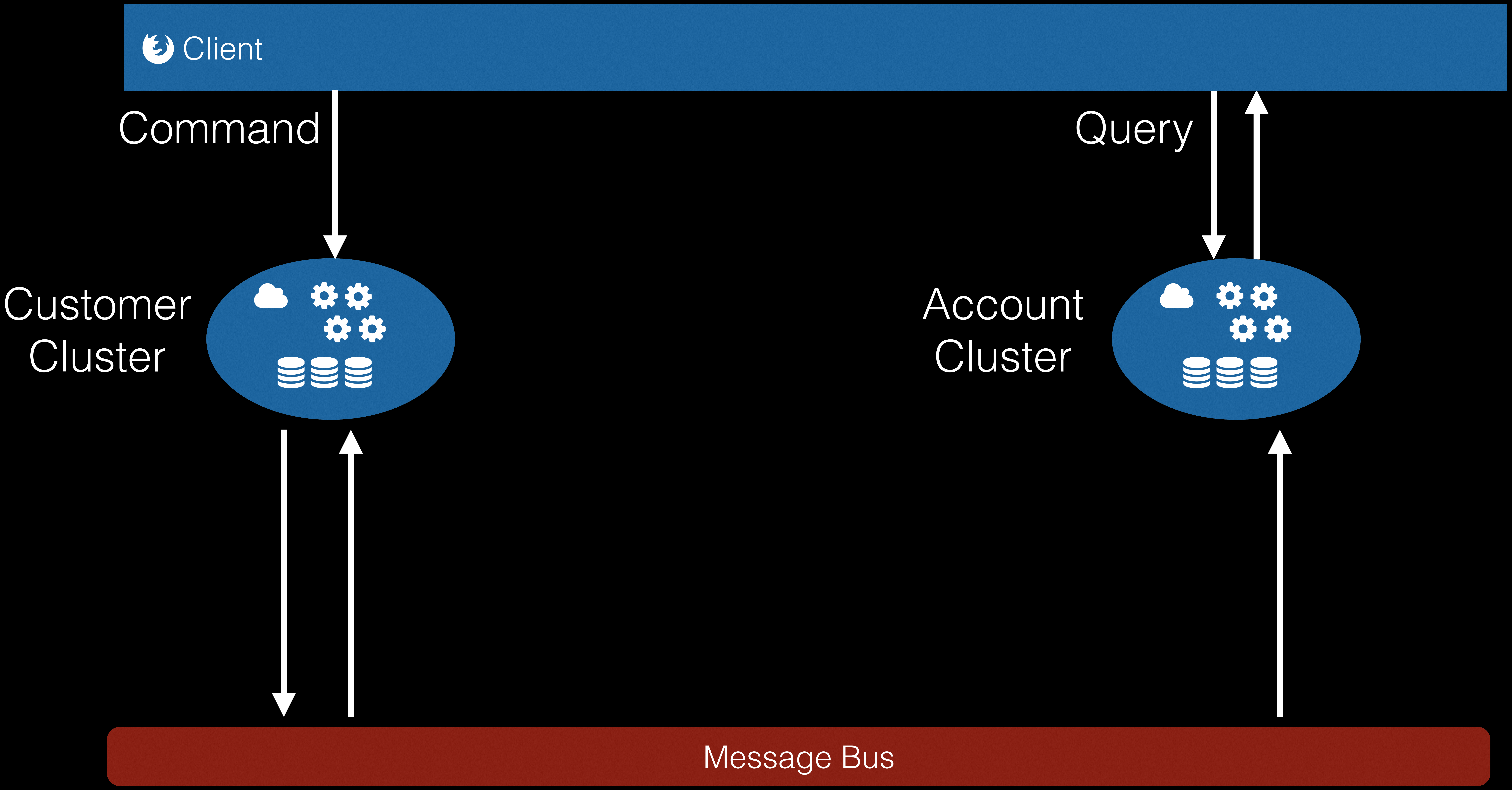
Real-time ➔ Synchronous

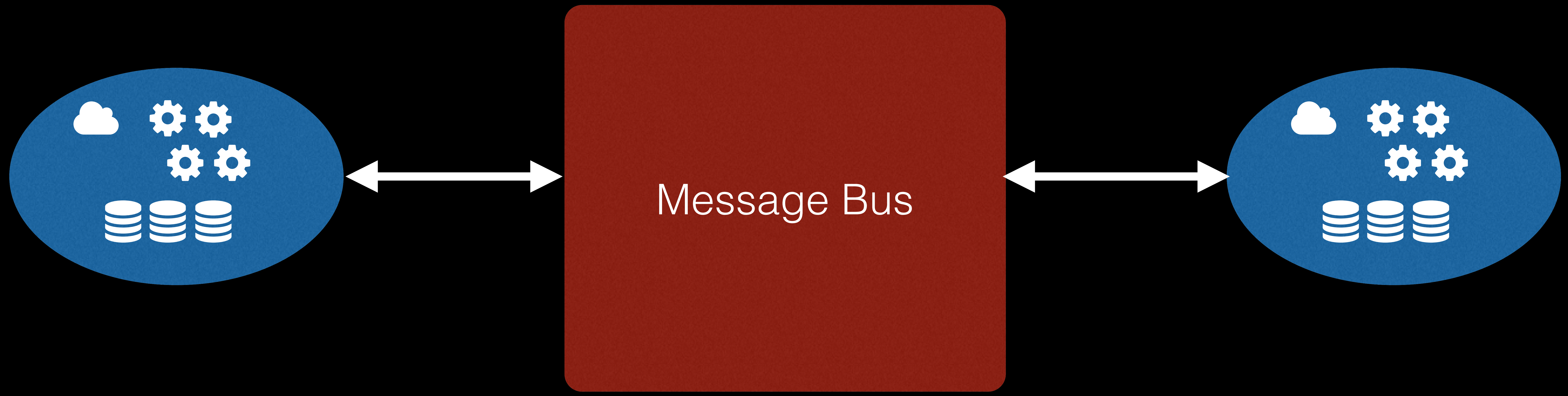
CQRS is inherently asynchronous

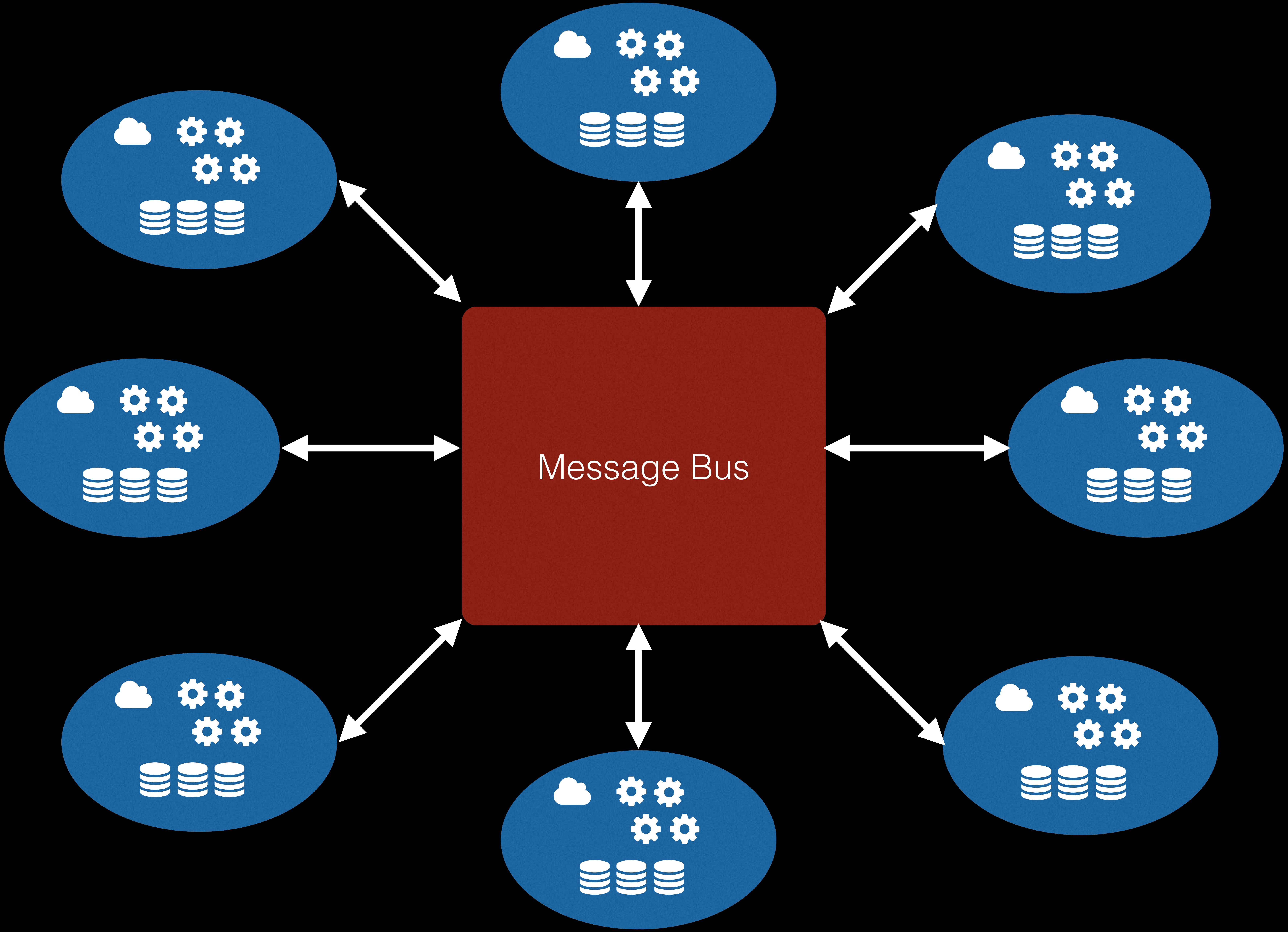
CQRS & the Enterprise

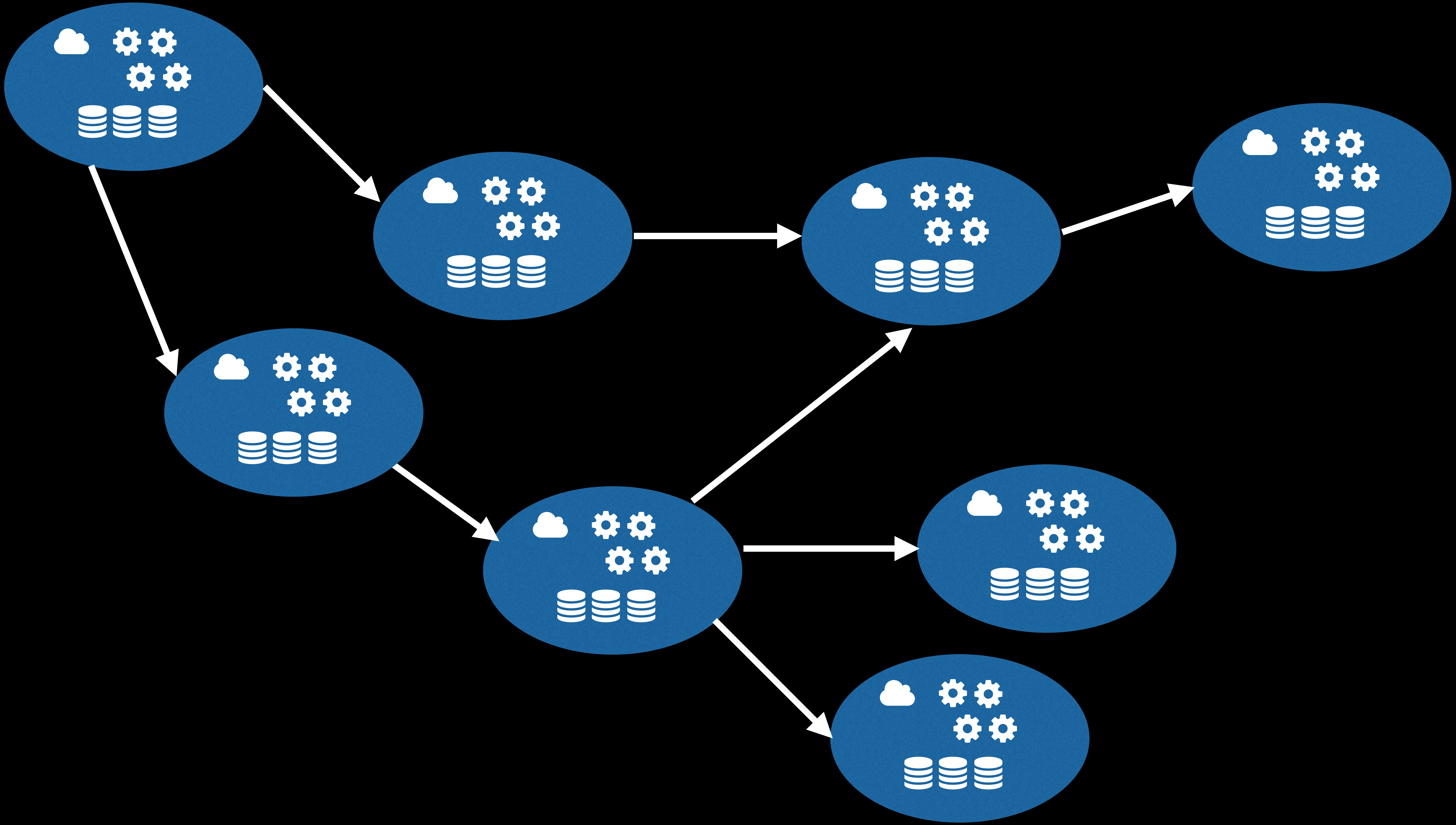
Scale

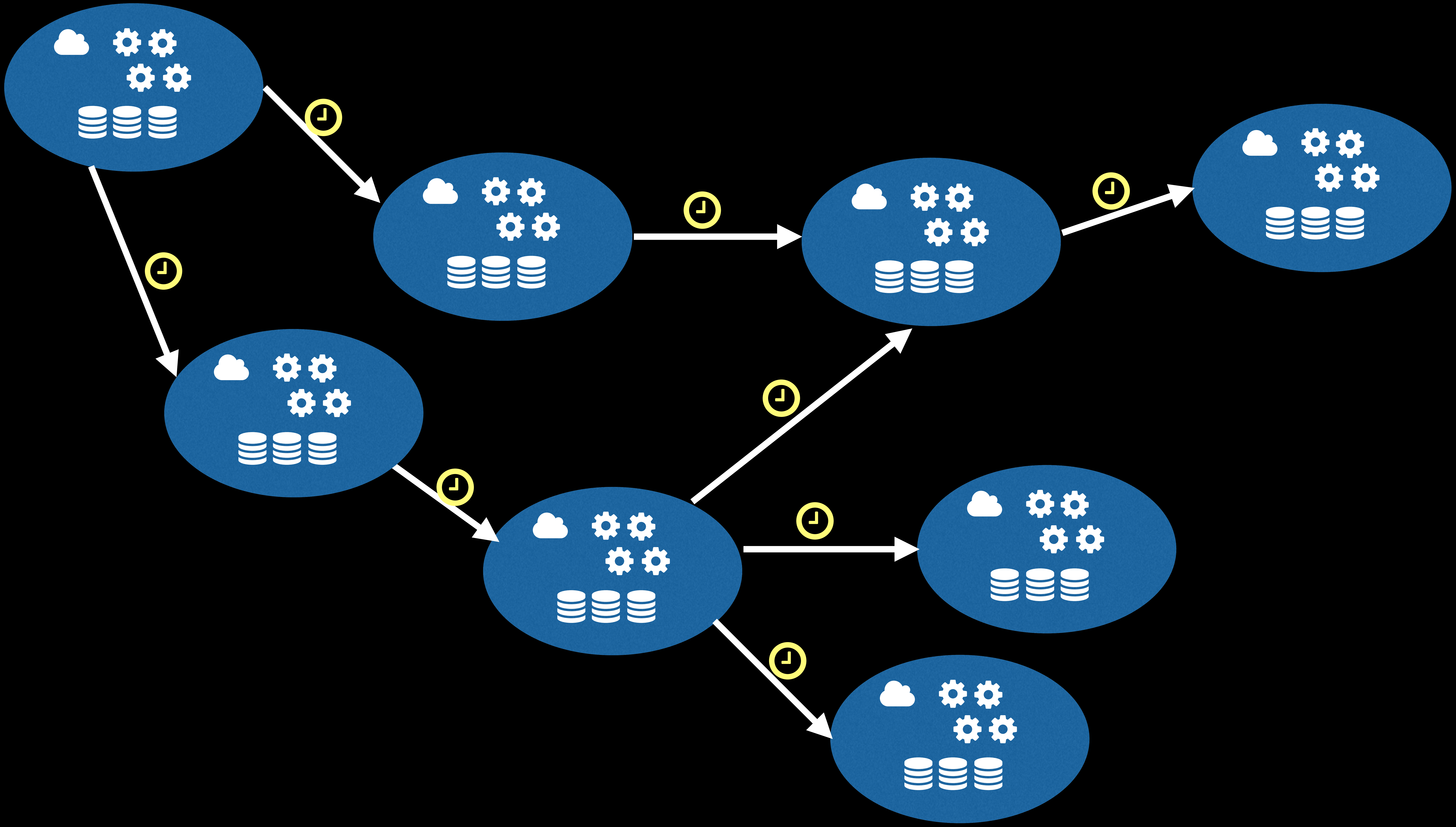

Client
Command
Query
Customer Cluster
Account Cluster
Message Bus


Client
Command
Query
Customer Cluster
Account Cluster
Message Bus


Message Bus


Message Bus





Consistency Latency 🕘+🕘+🕘

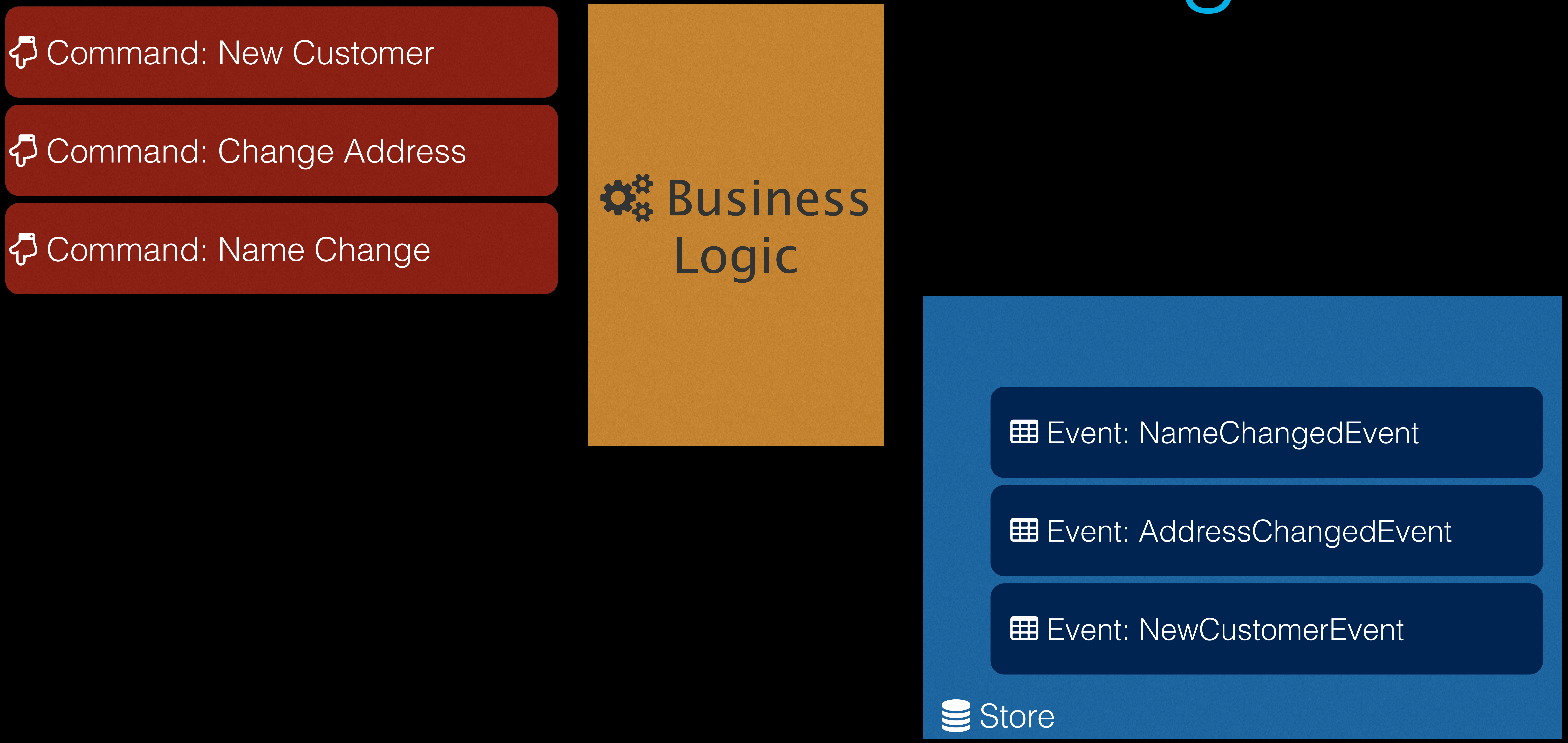
Resilience

Each Service is independent because it contains all the data needed to complete or accept a command from the user.

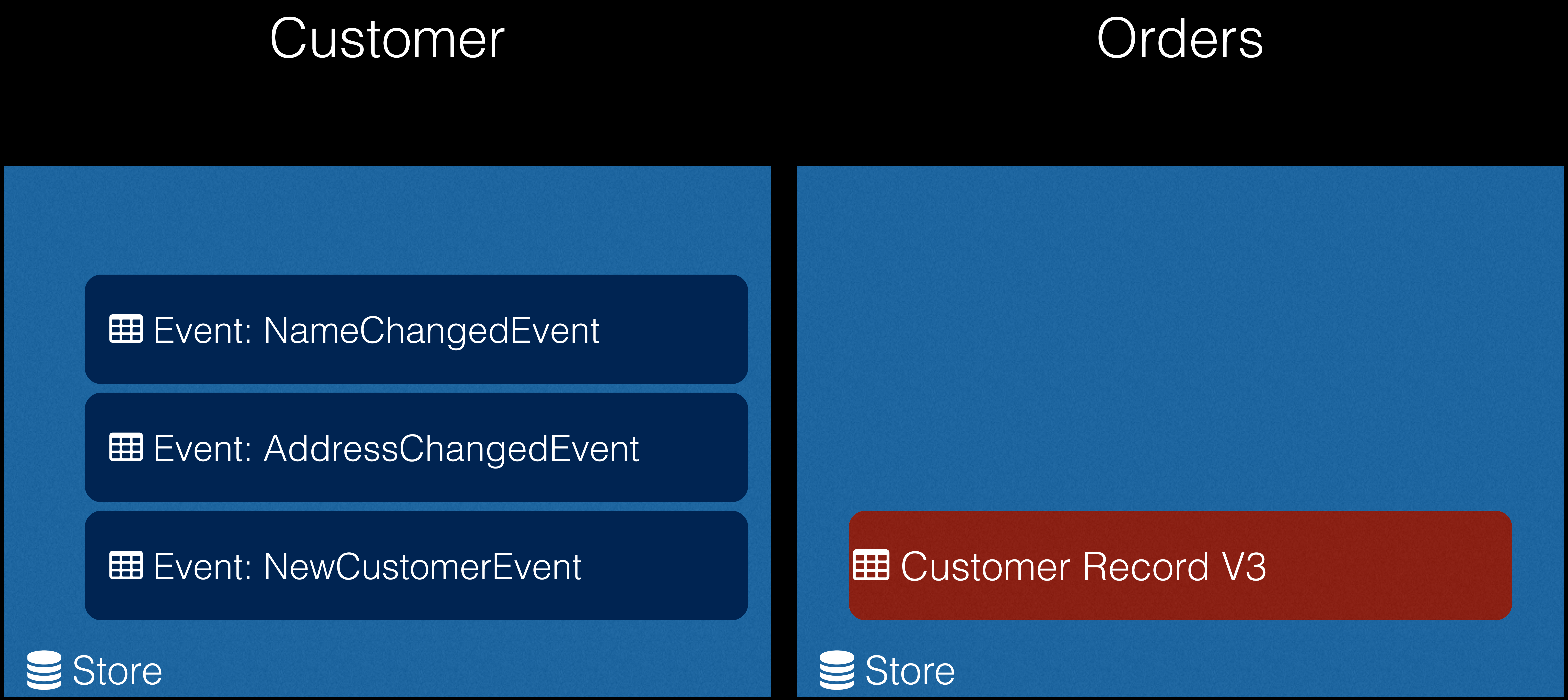
Coupling

Services communicate the results of an action. Services are only coupled by the data and not for processing.

60s Event Sourcing



60s Event Sourcing



Replay

Customer

Event: NameChangedEvent

Event: AddressChangedEvent

Event: NewCustomerEvent

re

Orders

Customer Record V3

Store

Complaints

Store

Replay - endstate

Customer
Orders
Complaints
Event: NameChangedEvent
Event: AddressChangedEvent
Event: NewCustomerEvent
re
Customer Record V3
Store
Complaints Customer V3
Store

Replay

Customer
Orders
Complaints
Event: NameChangedEvent
Event: AddressChangedEvent
Event: NewCustomerEvent
re
Customer Record V3
Store
Store

Replay

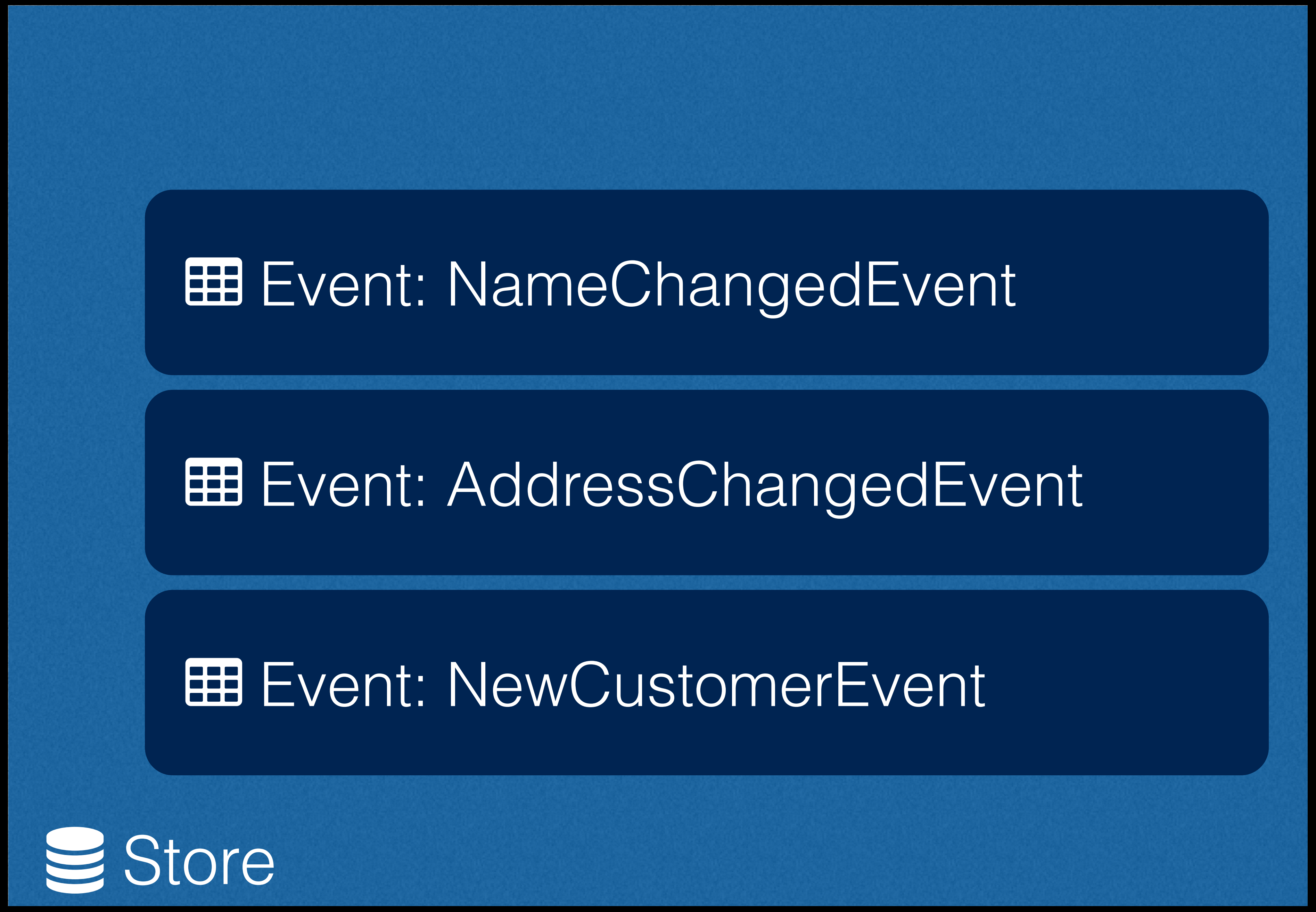
Customer
Orders
Complaints
Event: NameChangedEvent
Event: AddressChangedEvent
Event: NewCustomerEvent
re
Customer Record V3
Store
Customer Record V3
Store

Replay - endstate

Customer
Orders
Complaints
Event: NameChangedEvent
Event: AddressChangedEvent
Event: NewCustomerEvent
re
Customer Record V3
Store
Complaints Customer V3
Store

Replay Challenges

- Scale: How do you replay billions of events
  - Snapshotting can help but event volume >= aggregate count
  - Event processing must be idempotent.
  - Compound problem with fan in - service requires events from many others.



Highlights

- Captures Intent - Customer Moved - change address
- Encourages DDD and Event Sourcing
- Handles complexity well
- Distinct Command and View model(s)
- Becoming Popular
- Extremely scaleable!
- Very decoupled

Lowlights

Complex

Deceptively complex

Relatively new

Immature framework support

Not good for simple domains

Axon

AxonFramework
CQRS FRAMEWORK FOR JAVA

http://www.axonframework.org

Thank You

Graham Brooks

@grahamcbrooks

graham@grahambrooks.com

grahambrooks.com/talks