upvest

The Unreasonable Effectiveness of Events

Mastodon @lutzhuehnken@mastodon.social

LinkedIn https://linkedin.com/in/lutzh

Blog https://www.reactivesystems.eu/

I wrote this
without
ChatGPT

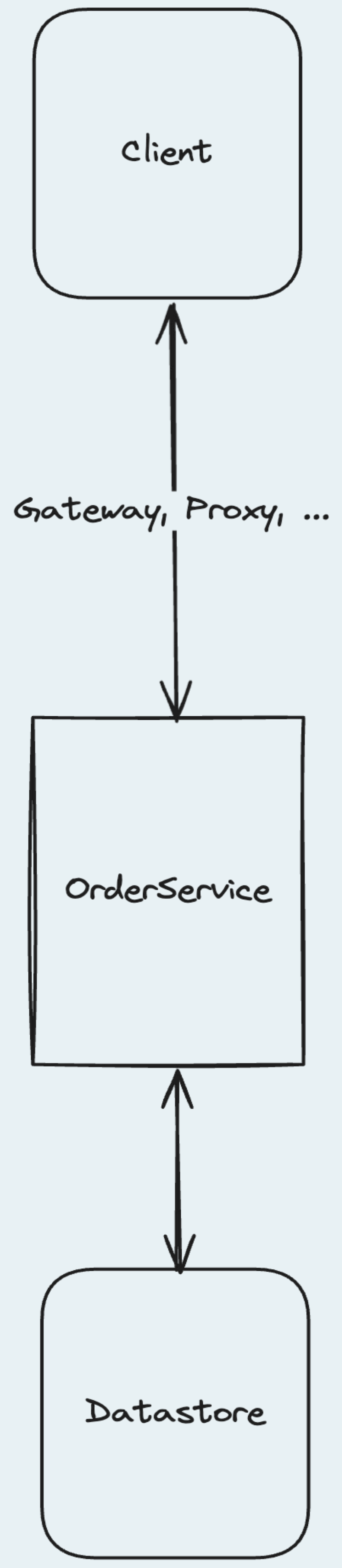
upvest

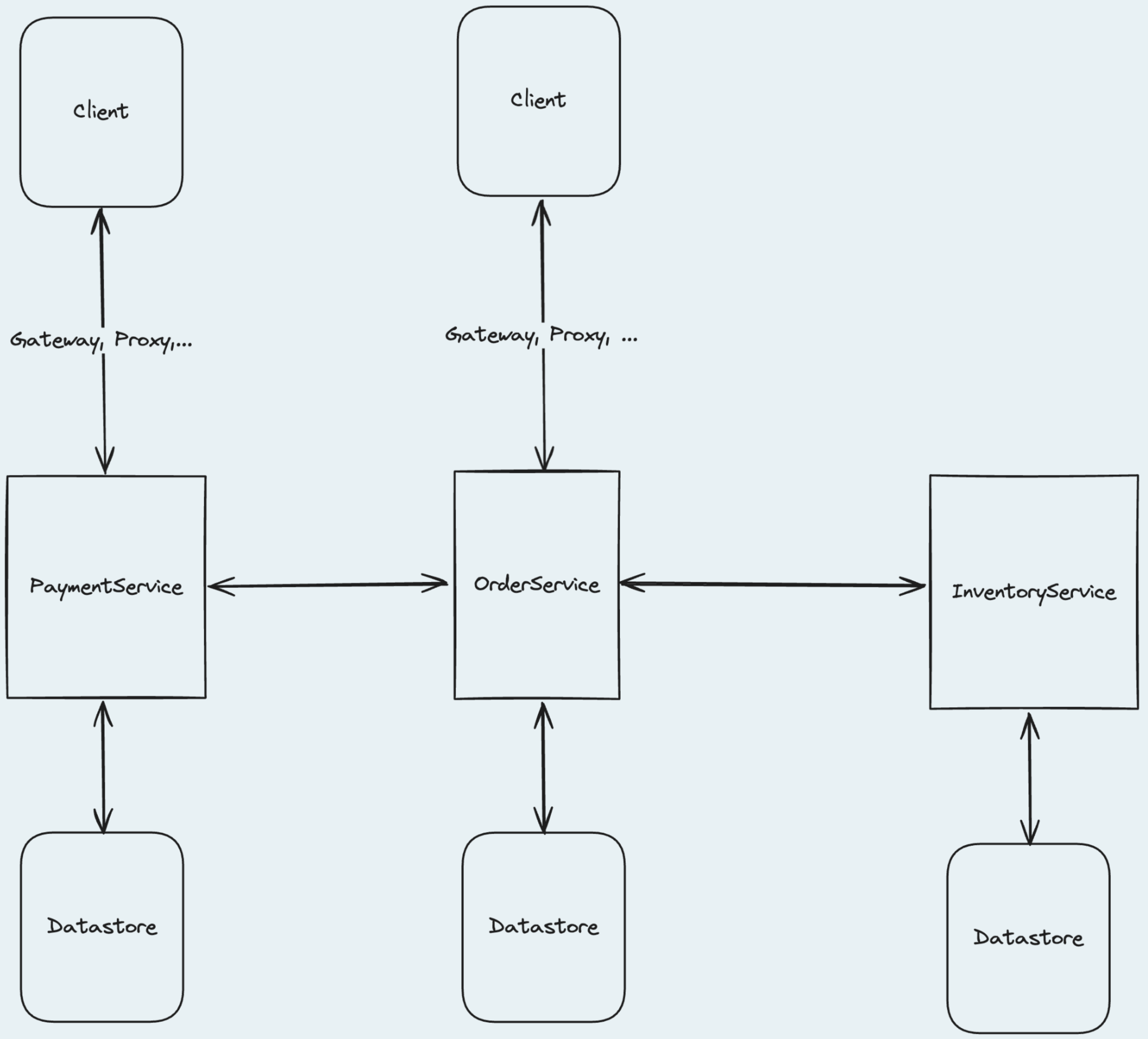
Why me?

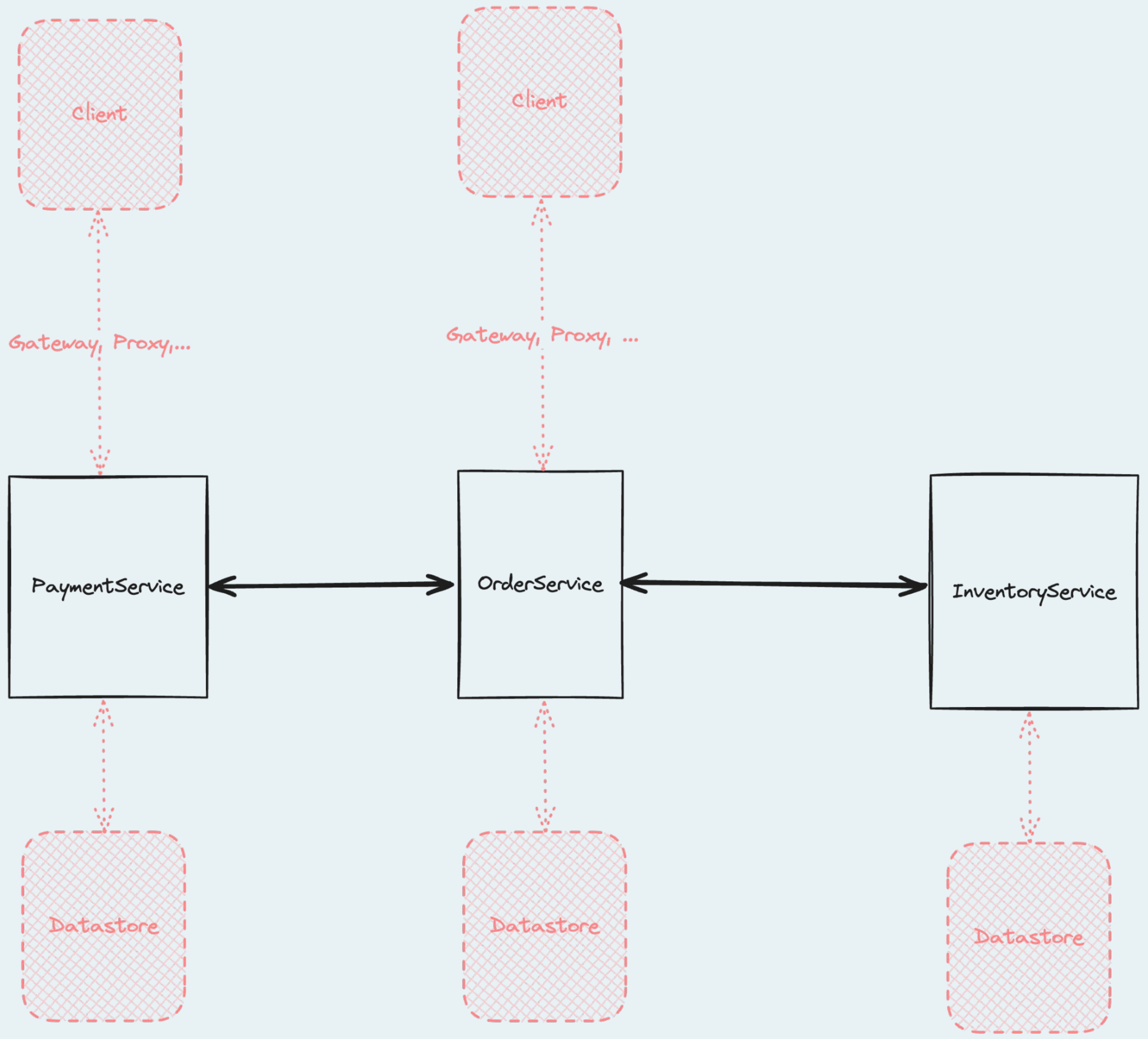
- Built my first message-driven, asynchronous system for the Bundesbank in 2002

- Worked on various Event-Driven systems, e.g. for Zalando, ING, ista, Maersk

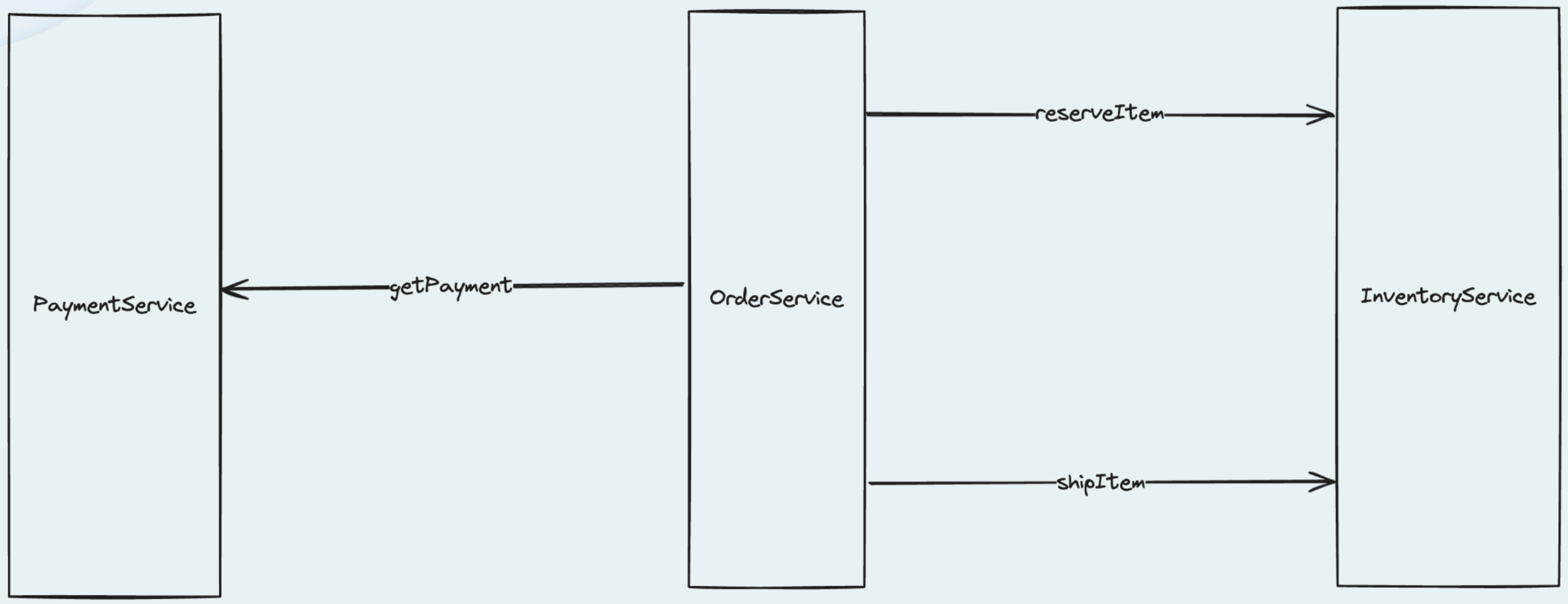
- Currently building a bank (!) with Event-Driven Architecture, Microservices, DDD


Client
Gateway, Proxy, ...
OrderService
Datastore

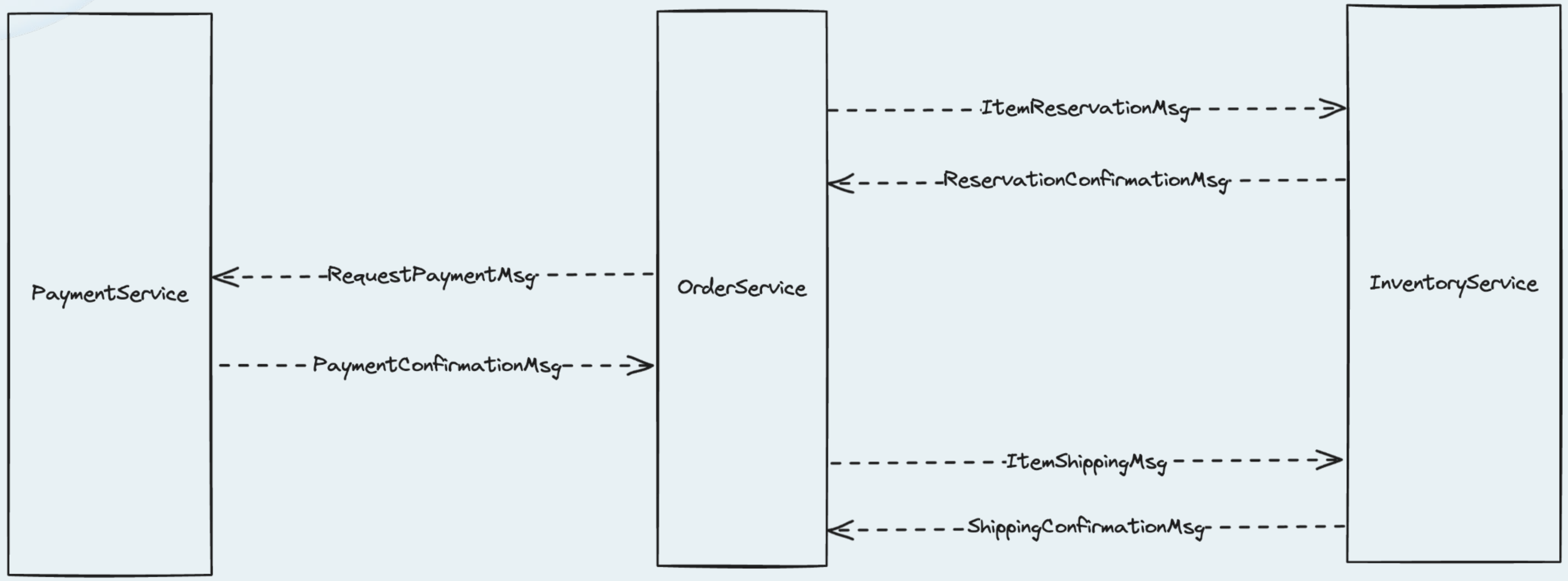

Client
Client
Gateway, Proxy,...
Gateway, Proxy, ...
PaymentService
OrderService
InventoryService
Datastore
Datastore
Datastore


Client
Client
Gateway, Proxy,...
Gateway, Proxy, ...
PaymentService
OrderService
InventoryService
Datastore
Datastore
Datastore

Synchronous calls (e.g. HTTP, gRPC)



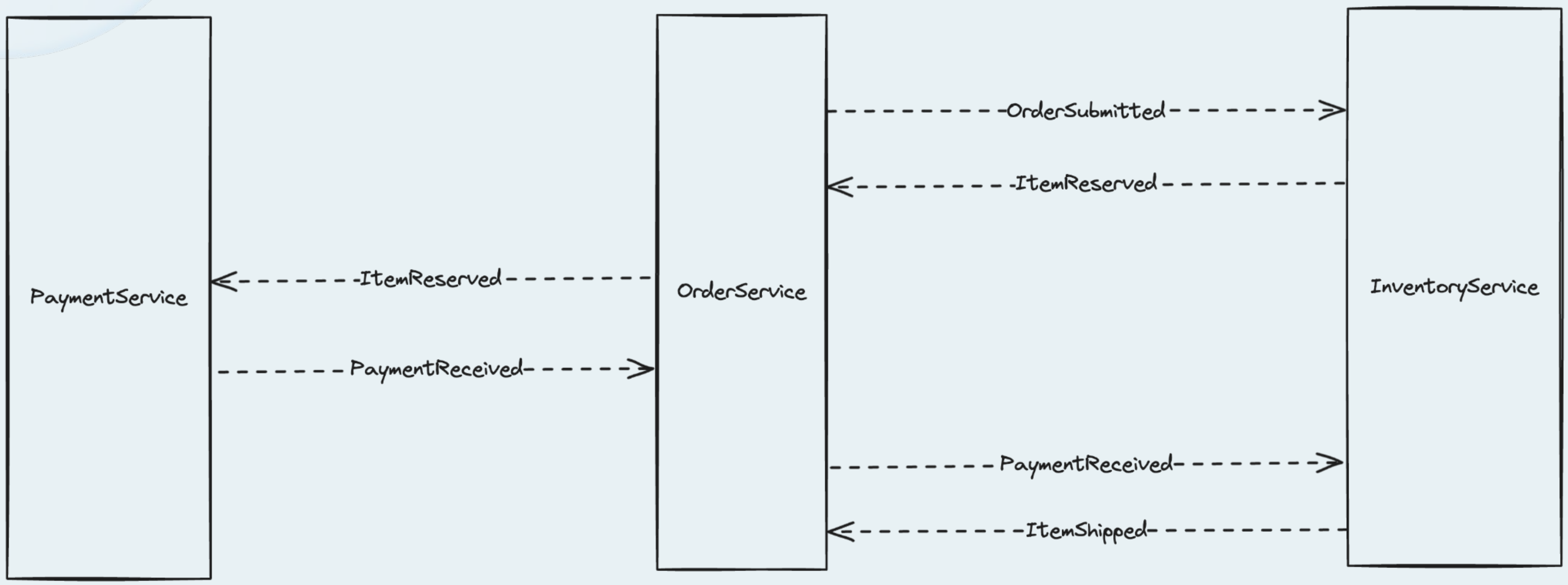
Asynchronous messaging



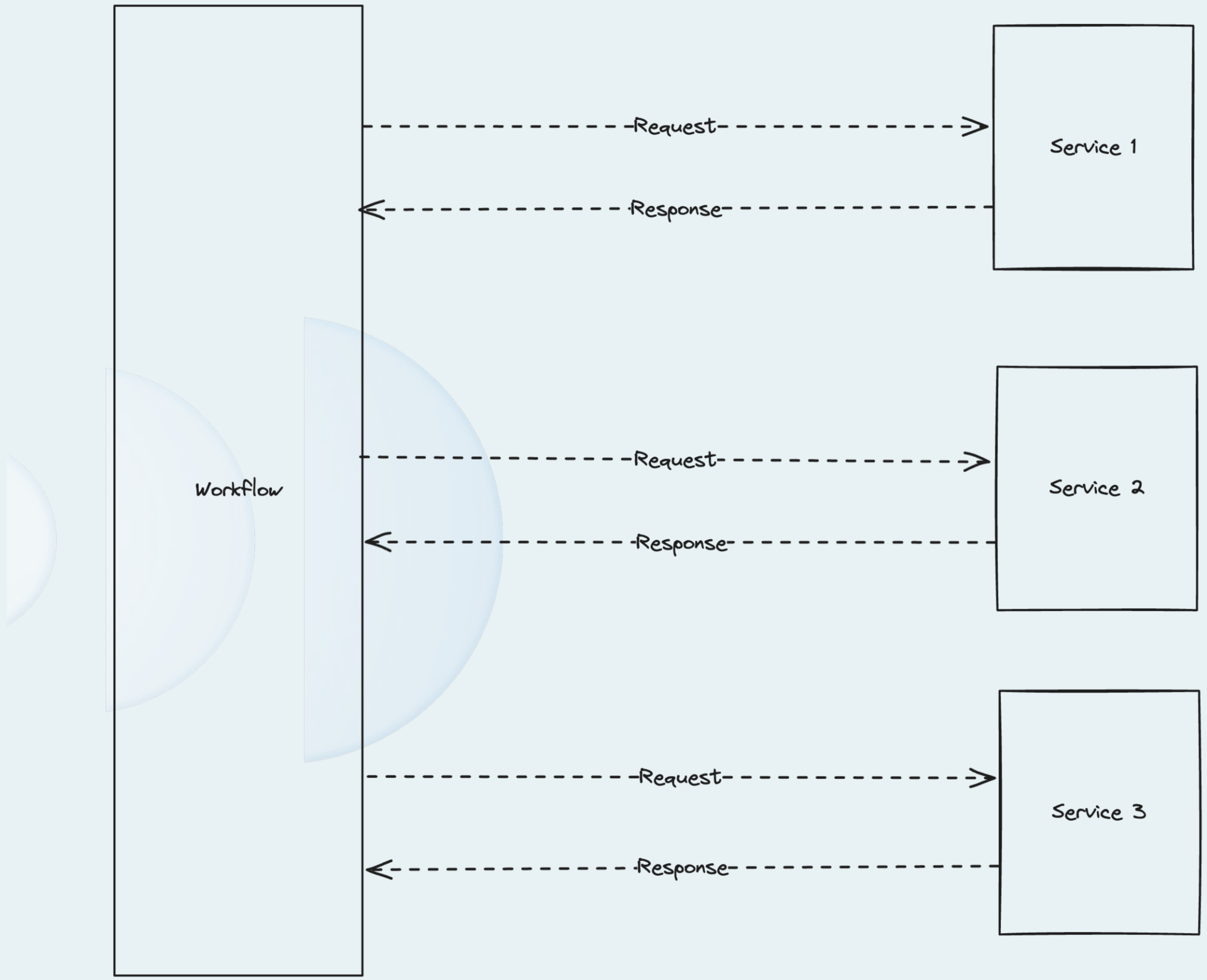
Reminder: Queries, Commands, Events

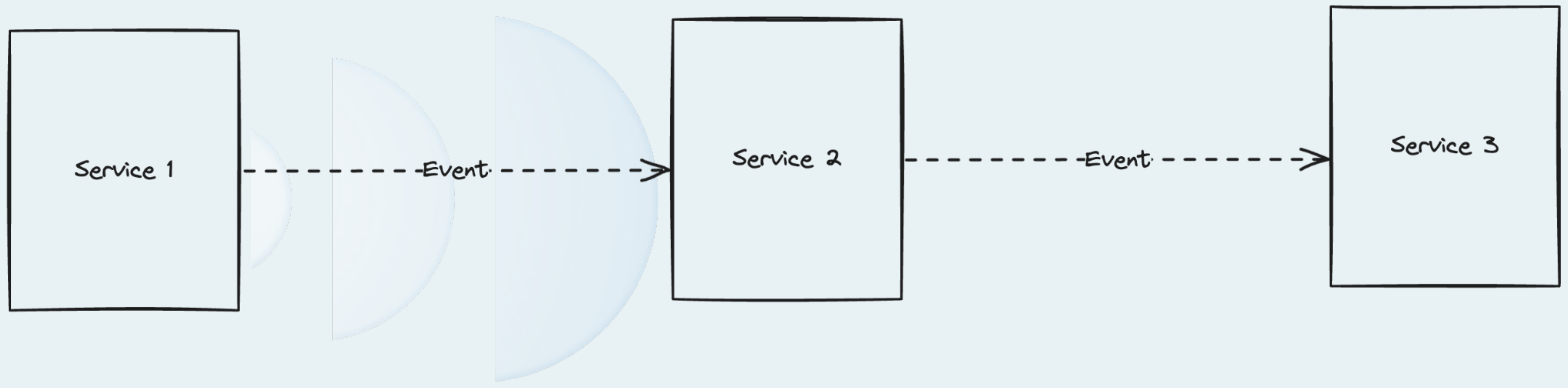
	Is..		
Query	A request for information about the current state of one or many objects		
Command	An intention to perform an operation or change a state		
Event	A fact, something that undisputedly happened in the past		

Events



PaymentService
OrderService
InventoryService
OrderSubmitted
ItemReserved
ItemReserved
PaymentReceived
PaymentReceived
ItemShipped


Workflow
Request
Response
Service 1
Request
Response
Service 2
Request
Response
Service 3


Service 1
Event
Service 2
Event
Service 3

upvest

Reminder: Queries, Commands, Events

	Is..	Expected Response	
Query	A request for information about the current state of one or many objects	The requested information	
Command	An intention to perform an operation or change a state	A confirmation that the command has been executed, or an error message if the command failed	
Event	A fact, something that undisputedly happened in the past	None (events are facts, they can't "fail")	

upvest

Reminder: Queries, Commands, Events

	Is..	Expected Response	Communication Pattern
Query	A request for information about the current state of one or many objects	The requested information	Request-Response
Command	An intention to perform an operation or change a state	A confirmation that the command has been executed, or an error message if the command failed	Request-Response
Event	A fact, something that undisputedly happened in the past	None (events are facts, they can't "fail")	Fire-and-Forget

upvest

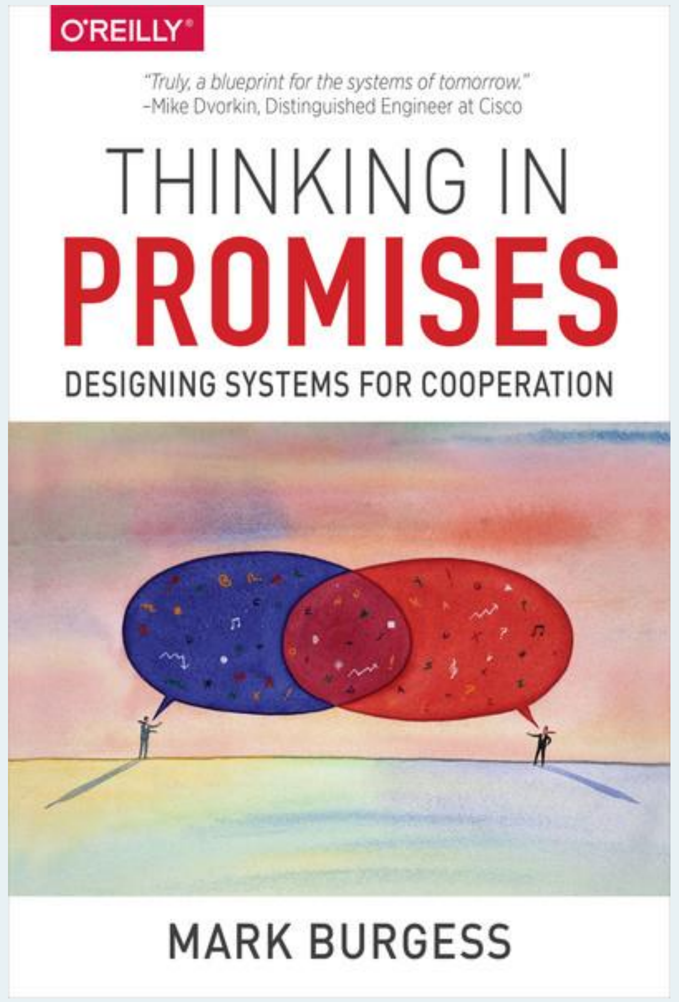
Reminder: Queries, Commands, Events

	Is..	Expected Response	Communication Pattern
Query	A request for information about the current state of one or many objects	The requested information	Request-Response
Command	An intention to perform an operation or change a state	A confirmation that the command has bee executed, o mes ommand	Response
Event	A fact, something th undisputed d in the pas	None (events are facts, they can't "fail")	Fire-and-Forget

But how can I know it'll be picked up and processed?

@lutzhuehnken@mastodon.social

Mental model 1/2
Thinking in Promises


O'REILLY
"Truly, a blueprint for the systems of tomorrow."
–Mike Dvorkin, Distinguished Engineer at Cisco
THINKING IN
PROMISES
DESIGNING SYSTEMS FOR COOPERATION
MARK BURGESS

$$\begin{tikzcd}A1 \arrow[r, "(-b)"] & A2\end{tikzcd}$$

$$\begin{tikzcd}\text{PaymentService} \arrow[r, "(\text{-OrderCreated})"] & \text{OrderService}\end{tikzcd}$$

upvest

Agents
Agents in Promise Theory are said to be autonomous, meaning that they are causally independent of one another. This independence implies that they cannot be controlled from without, they originate their own behaviours entirely from within, yet they can rely on one another's services through the making of promises to signal cooperation.

Promises
Promises arise when an agent shares one of its intentions with another agent voluntarily, e.g. by publishing its intent.

https://en.wikipedia.org/wiki/Promise_theory

upvest

Synchronous Command	Asynchronous Command	Asynchronous Event
📞	💬	❓

upvest

Events are “fire-and-forget”…

… but based on previous agreements (promises)

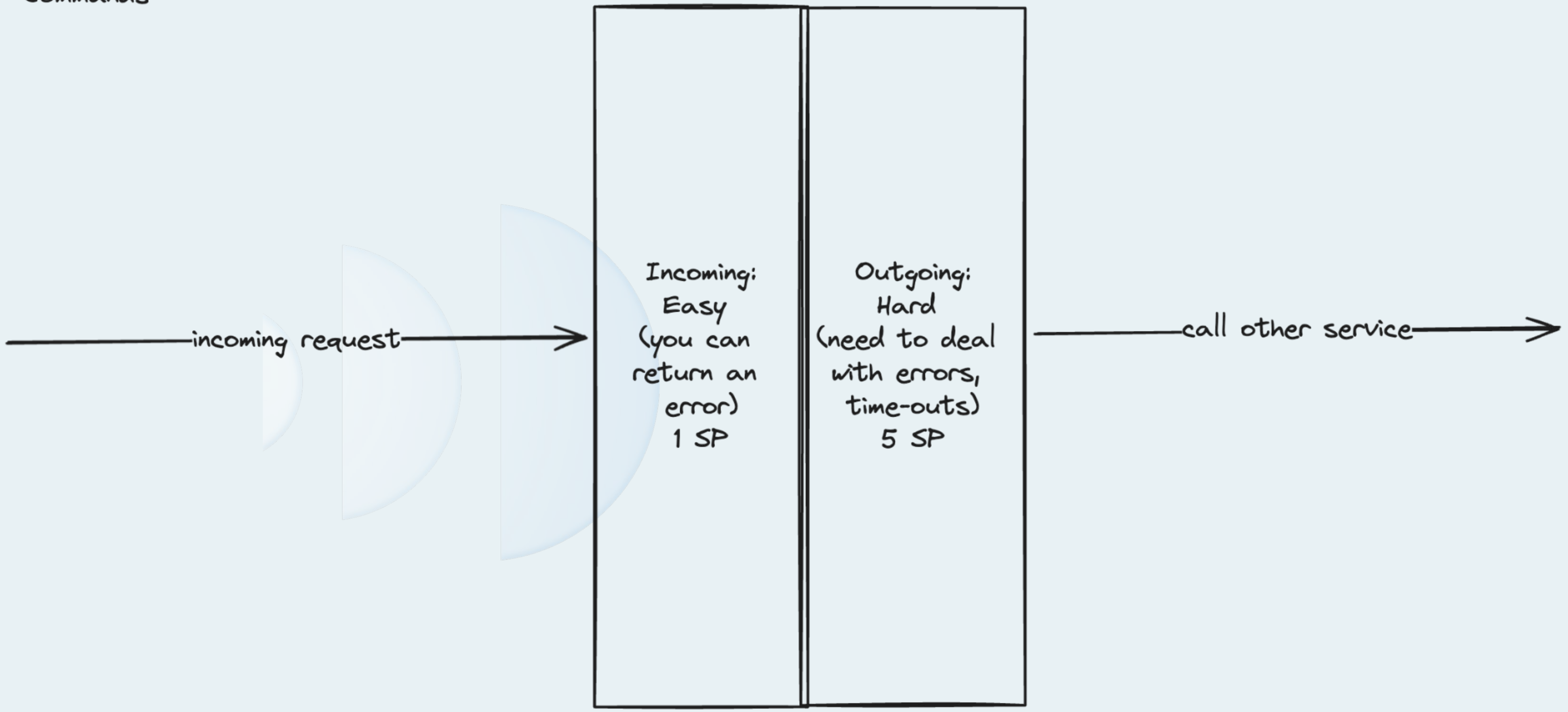
upvest

Reminder: Queries, Commands, Events

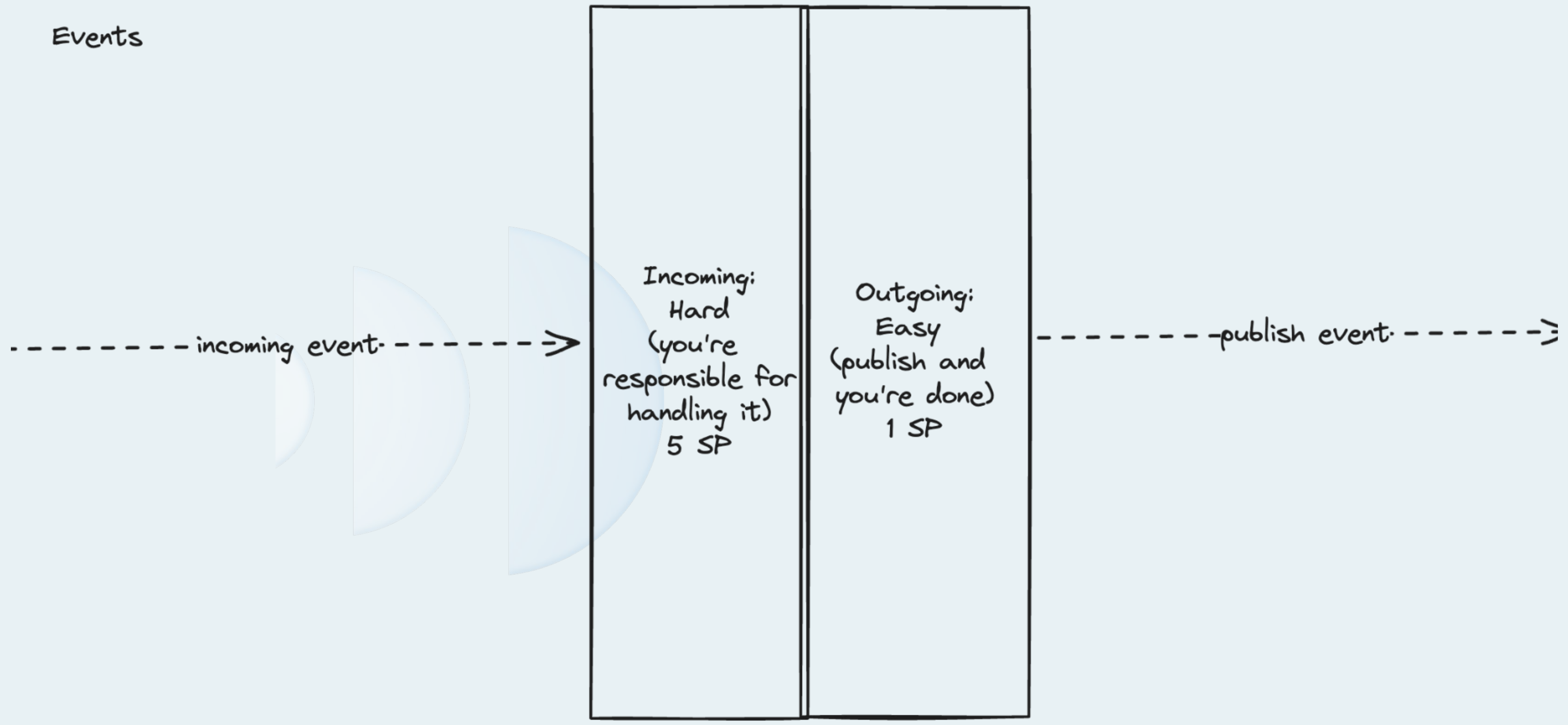
	Is..	**Expected Response**	**Communication Pattern**
Query	A request for information about the current state of one or many objects	The requested information	Reques[illegible]e
Command	An intention to perform an operation or change a state	A confirm[illegible]e co[illegible]been [illegible]d, or an error [illegible]essage if the command failed	Request-Response
Event	A f[illegible]ing that [illegible]atedly happened in the past	None (events are facts, they can't "fail")	Fire-and-Forget **(rely on promises)**

But surely there'll still be errors we need to handle?

Commands


incoming request
Incoming: Easy (you can return an error) 1 SP
Outgoing: Hard (need to deal with errors, time-outs) 5 SP
call other service

upvest

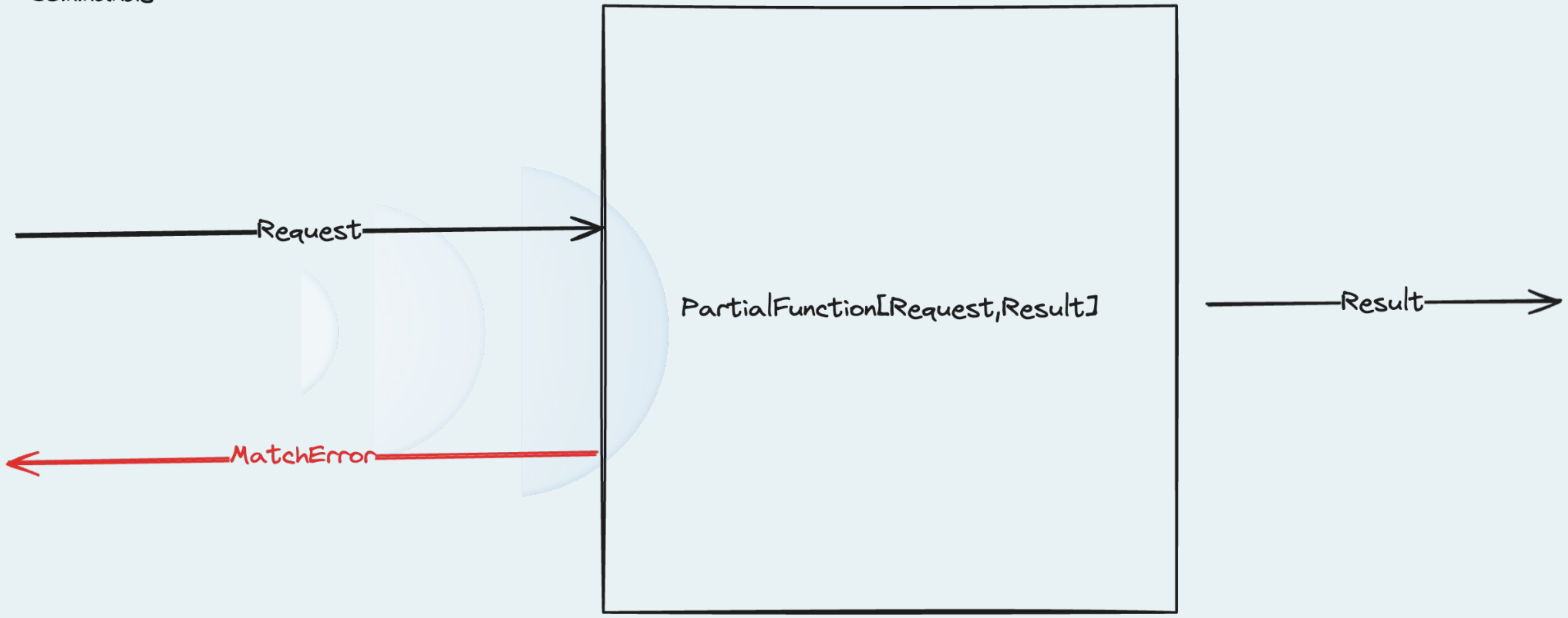

Events
incoming event
Incoming: Hard (you're responsible for handling it) 5 SP
Outgoing: Easy (publish and you're done) 1 SP
publish event

upvest

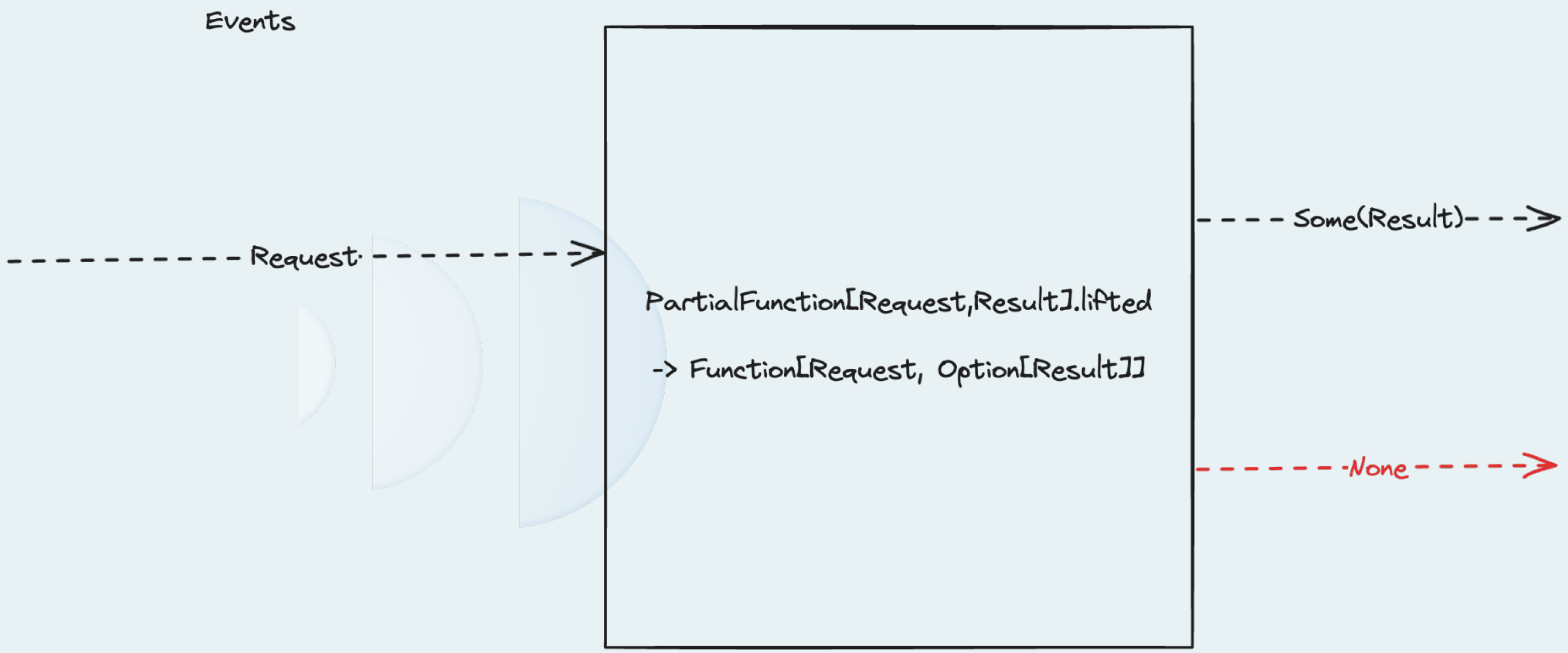
Mental model 2/2
Functional Programming

upvest

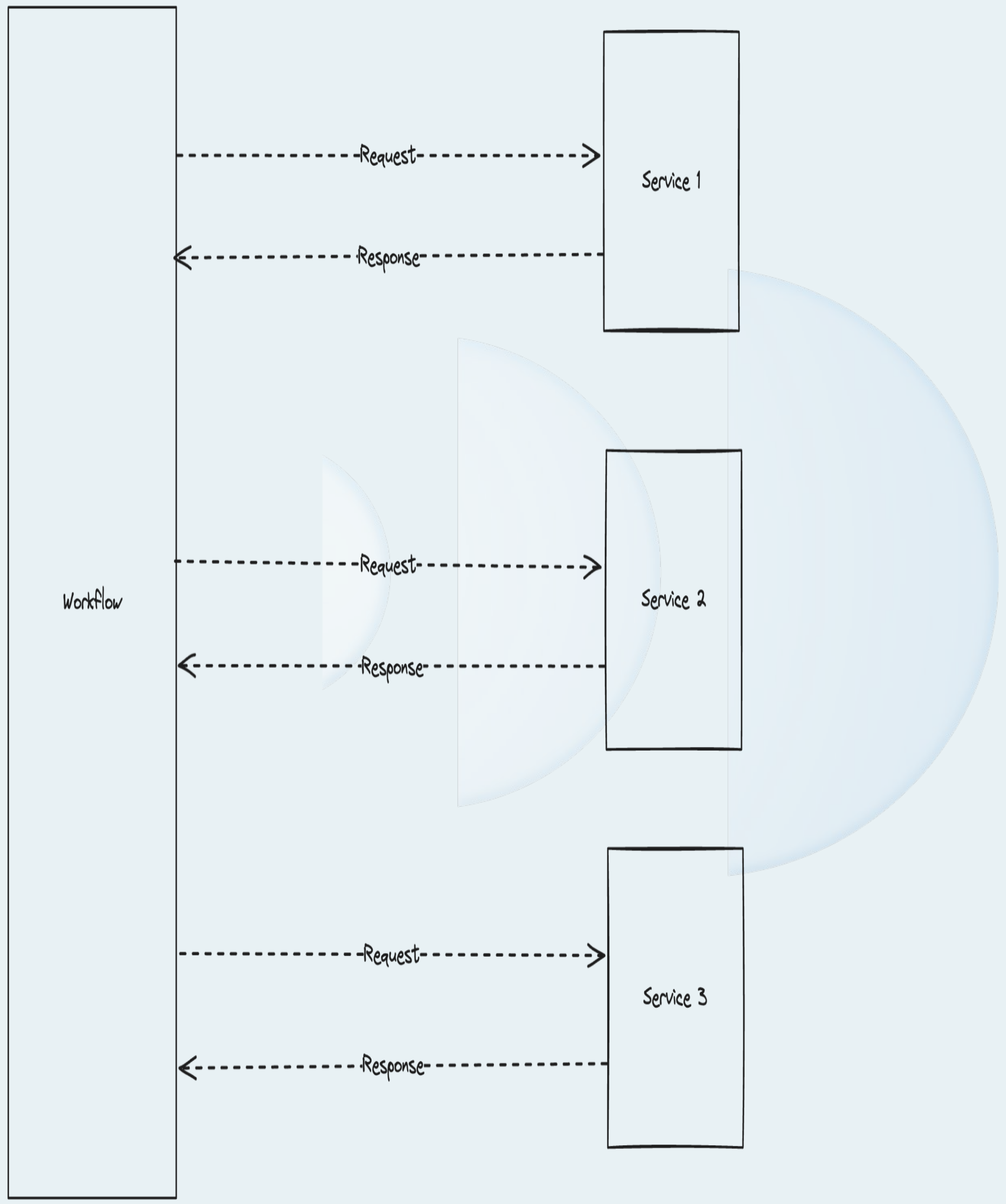
Commands



upvest


Events
Request
PartialFunction[Request,Result].lifted
-> Function[Request, Option[Result]]
Some(Result)
None

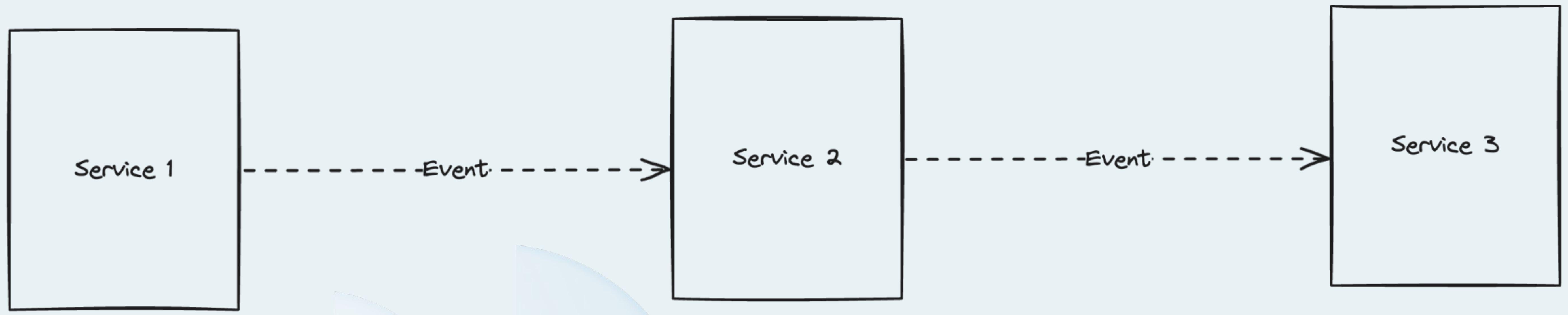
upvest



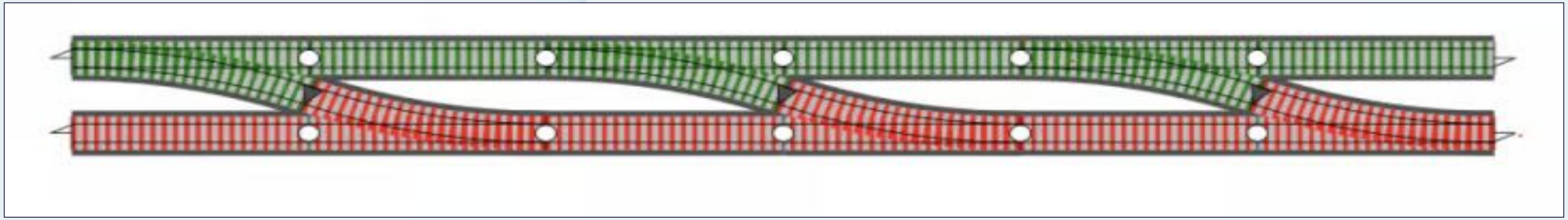
```go
item, err := reserveItem(itemNumber)
if err != nil {
    return nil, errors.New("out of stock")
}

confirmation, err := getPayment(itemNumber)
if err != nil {
    return nil, errors.New("insufficient funds")
}

result, err := shipItem(itemNumber)
```



```
reserveItem(itemNumber).flatMap(getPayment).flatMap(shipItem)
```



upvest

The unreasonable effectiveness of events: Beyond asynchronous communication, being event-driven has a massive effect on the overall workflow and system design. Event-Driven Architecture (EDA) is at least as much about the flow as it is about the actual events.

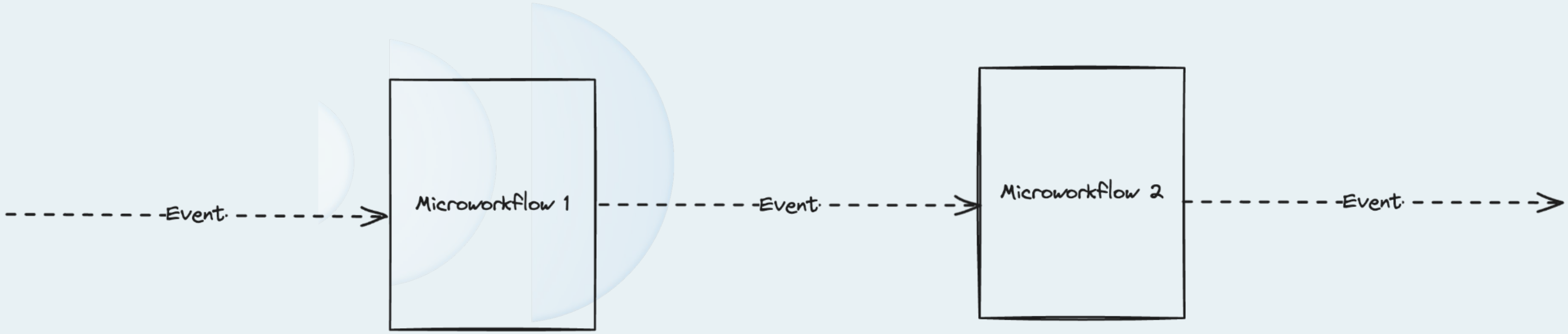
upvest

Mindbender:
We could treat a command like an event

	Is..	Expected Response	Communication Pattern
Command	An intention to perform an operation or change a state	None (receiver is not allowed to reject)	Fire-and-Forget
Event	A fact, something that undisputedly happened in the past	None (events are facts, they can't "fail")	Fire-and-Forget

upvest

Maybe, instead of Event-Driven Architecture, we should talk about microworkflows?



Coming soon: https://microworkflows.org

The unreasonable effectiveness of events:

- scalability and resilience
- testability
- organizational clarity / domain boundaries

upvest

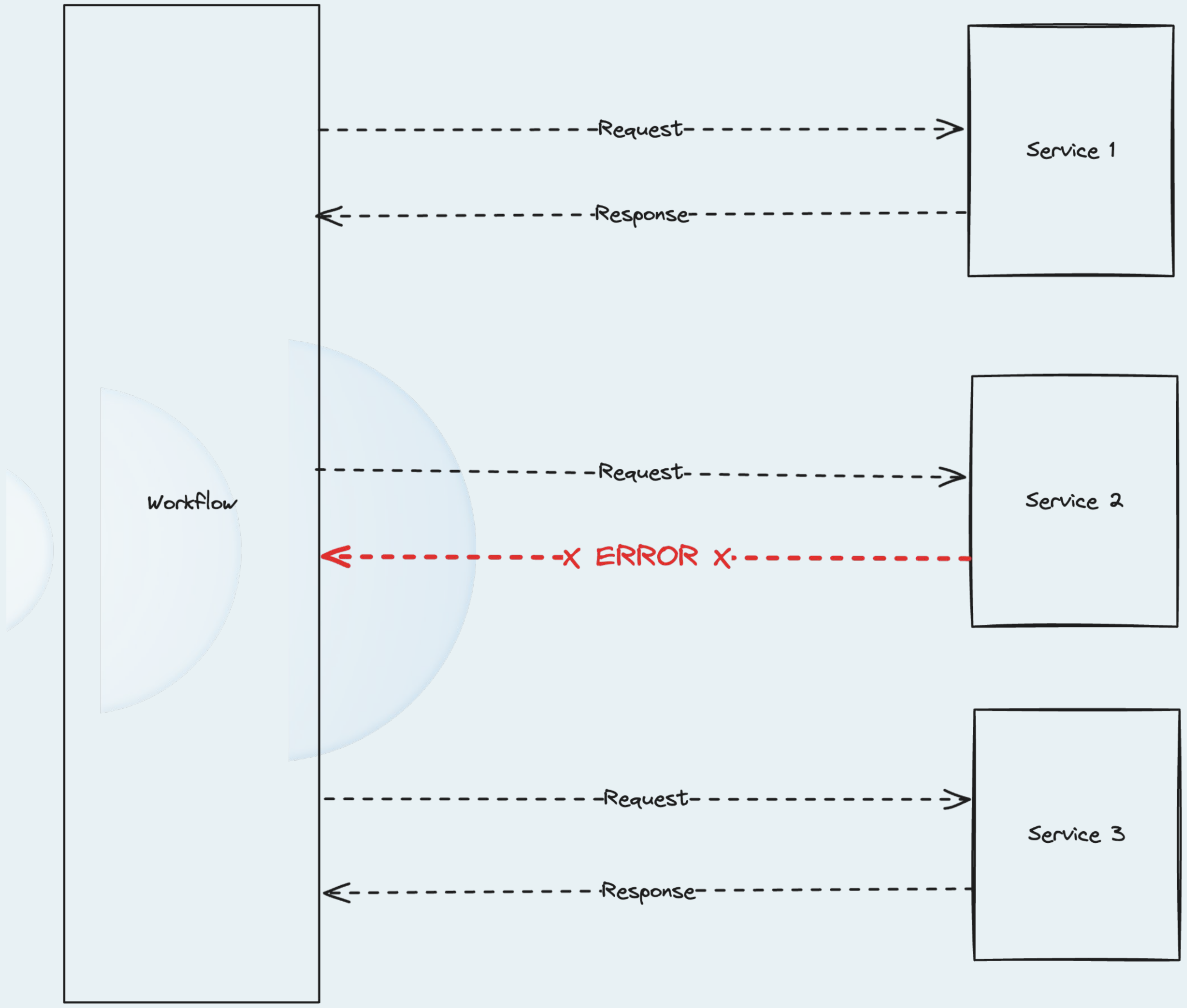
Enjoy the (load) testing

Fewer (no?) runtime dependencies to other services = simpler tests.

Testing an event-driven service (or a microworkflow) is still an integration test.

But it’s nice to be able to test e.g. throughput in a very isolated way.

upvest

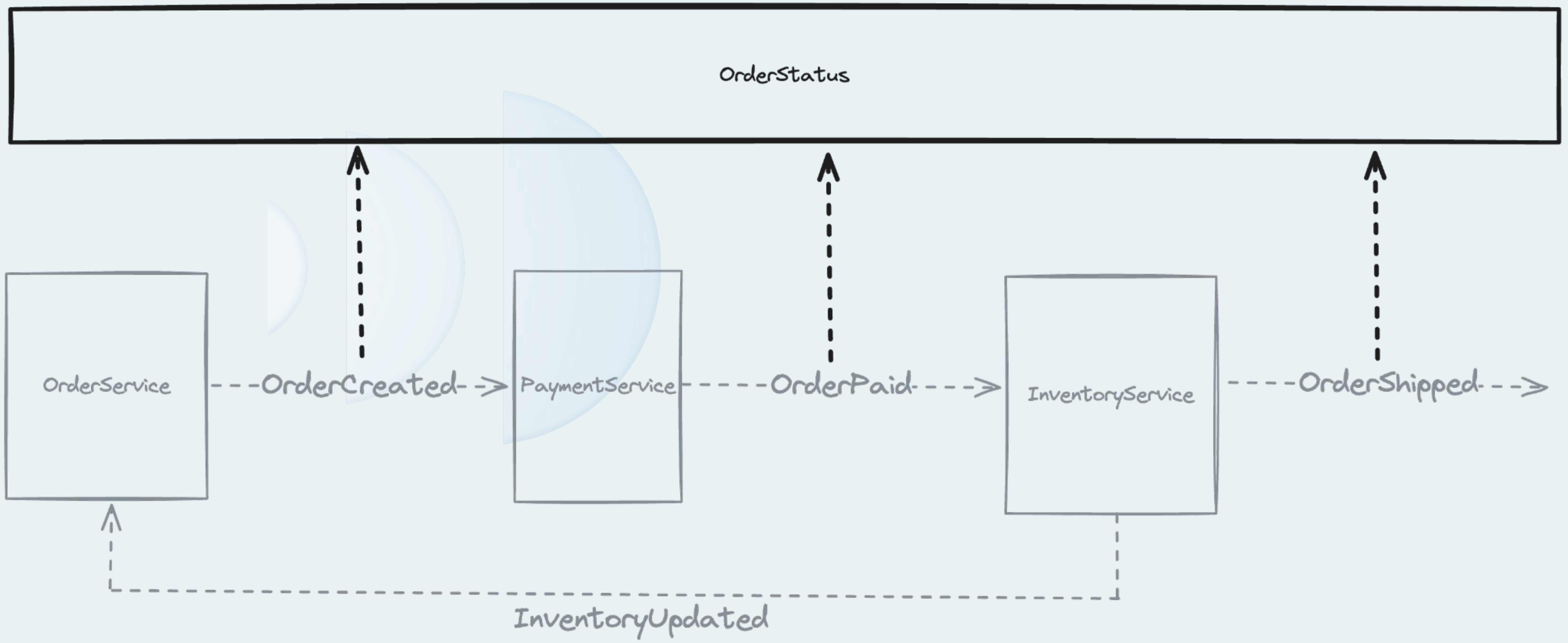

Workflow
Request
Response
Service 1
Request
X ERROR X
Service 2
Request
Response
Service 3

upvest

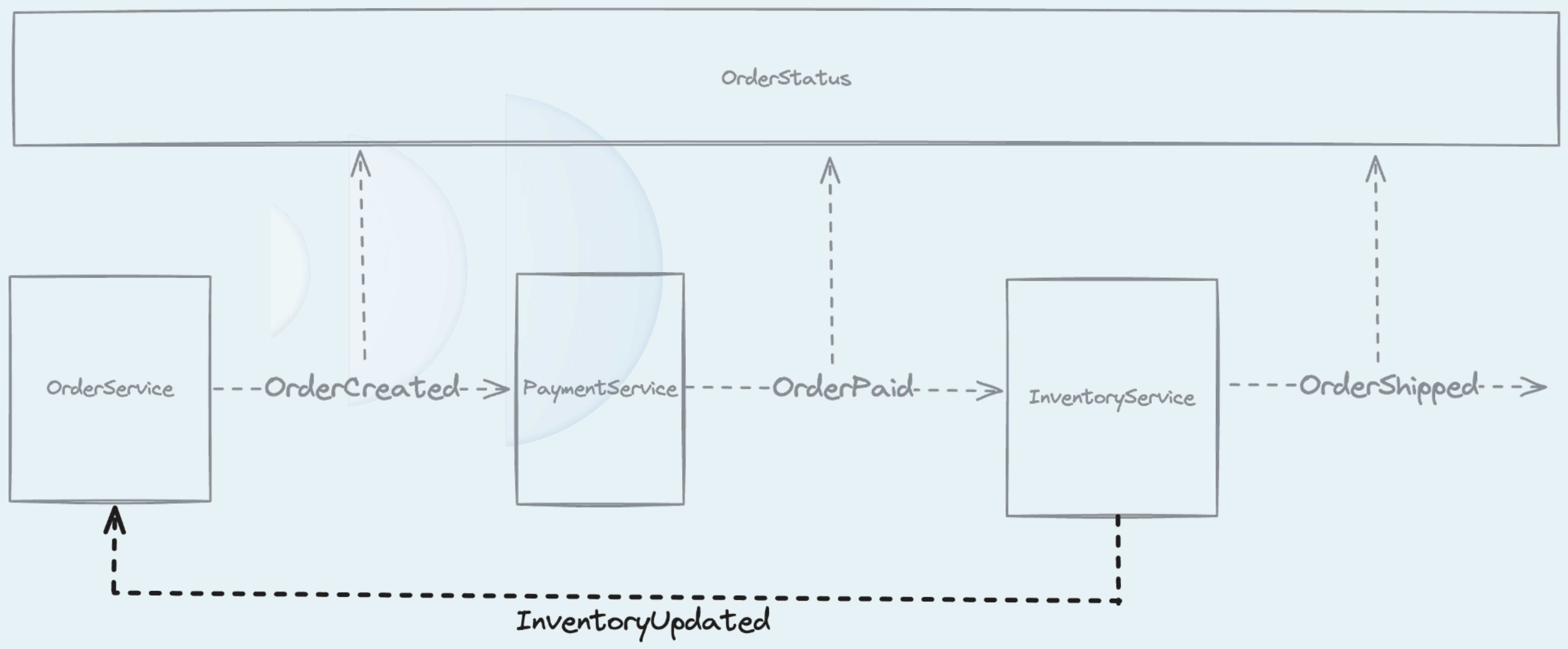
Important Patterns

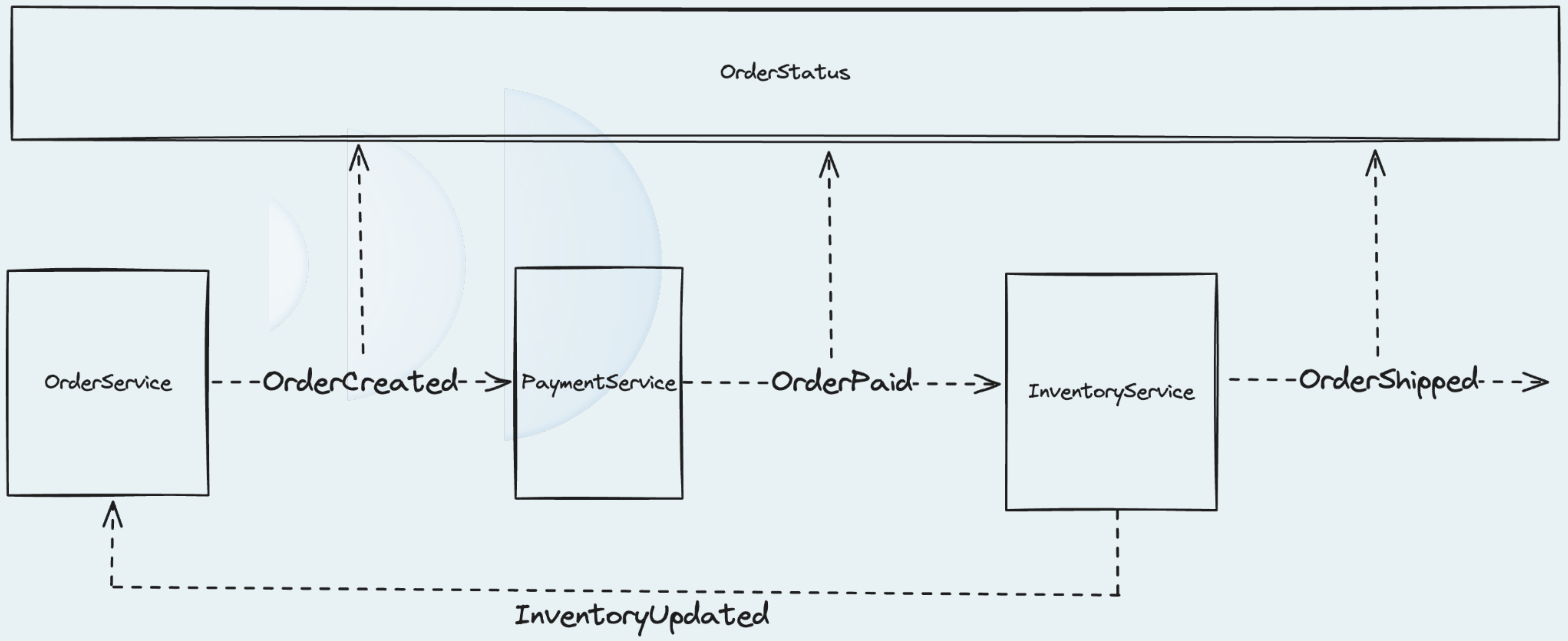
upvest

Separate Observing from Control



Bring the data to the process




OrderStatus
OrderService
OrderCreated
PaymentService
OrderPaid
InventoryService
OrderShipped
InventoryUpdated

upvest

Bonus models

upvest

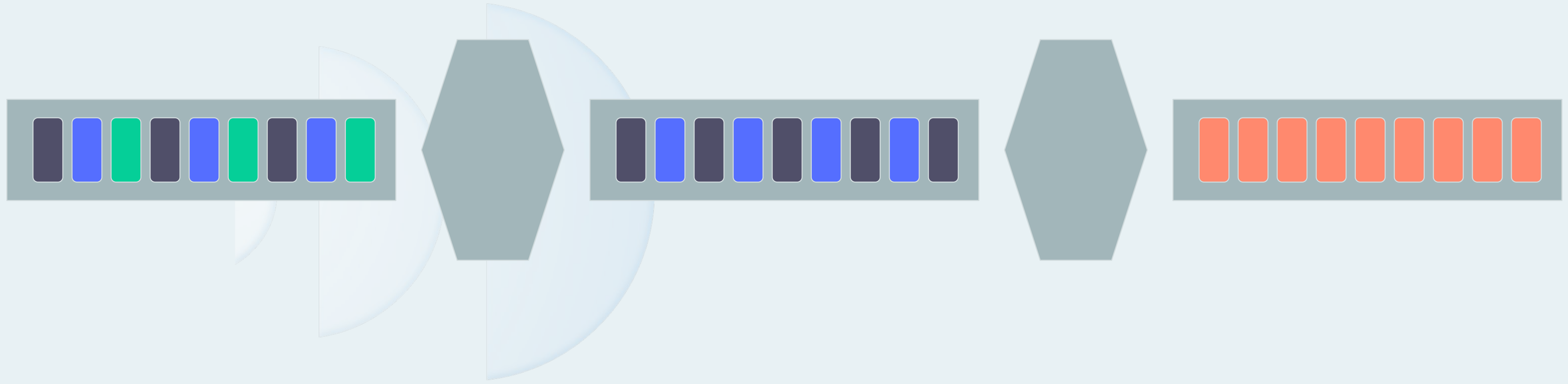
Think Unix Philosophy

1. Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features".
2. Expect the output of every program to become the input to another, as yet unknown, program. ...

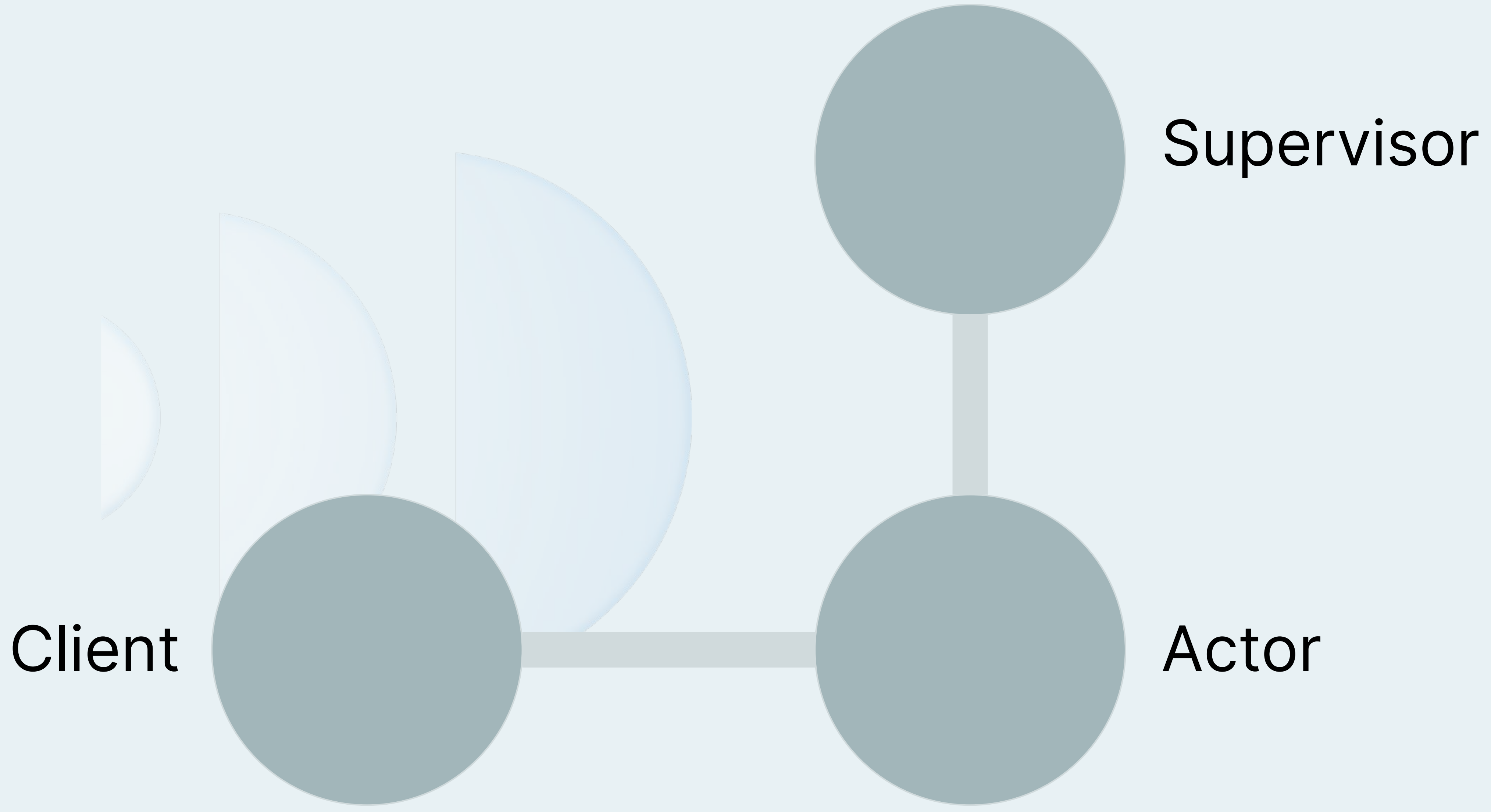
```
$ Cat file3.txt | grep “dwx” | tee file4.txt | wc -l
```

upvest

Think Stream Processing

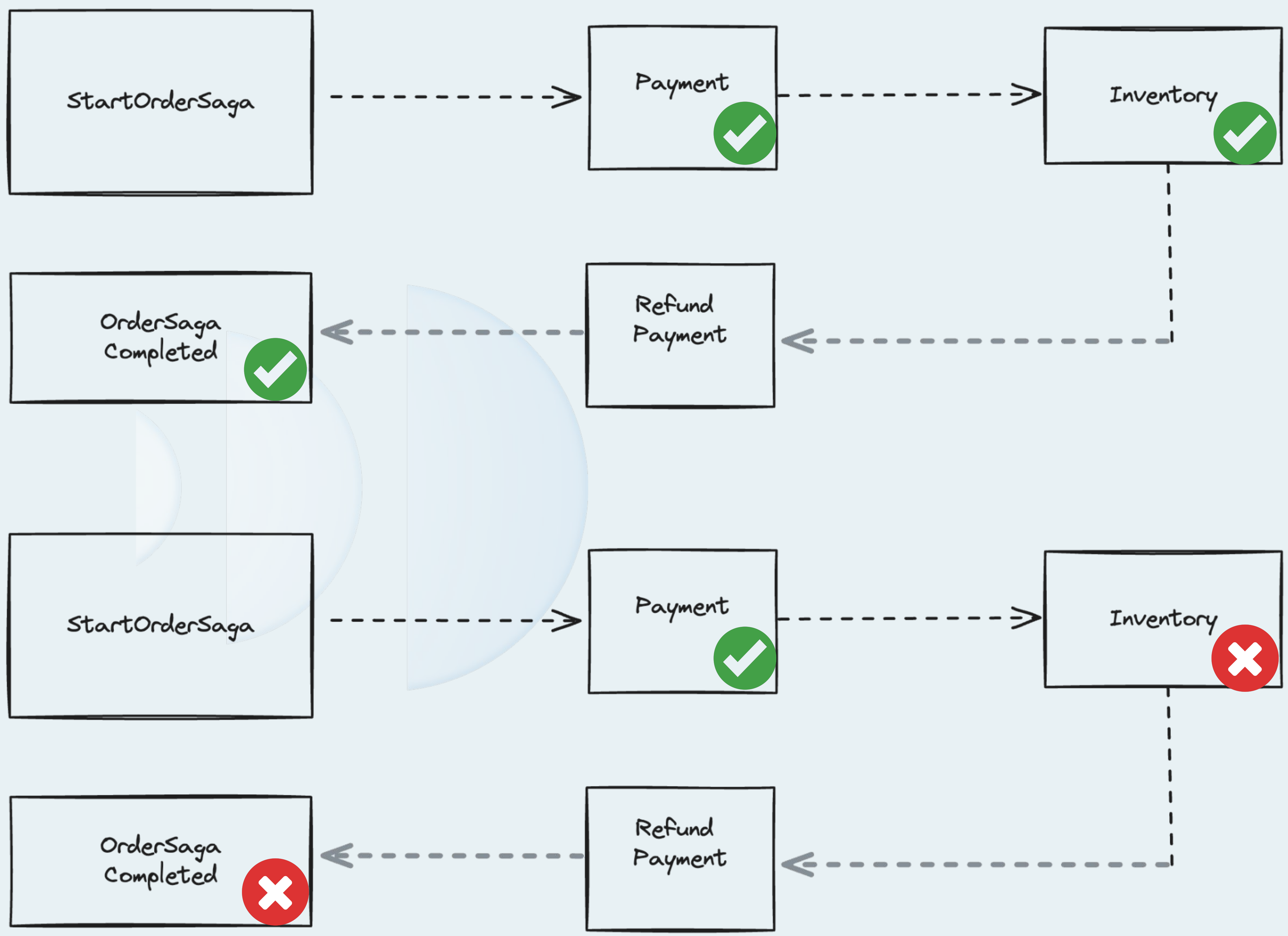


Think Actors



upvest

Bonus content:
The Saga Pattern Considered Harmful


StartOrderSaga
Payment
Inventory
OrderSaga Completed
Refund Payment
StartOrderSaga
Payment
Inventory
OrderSaga Completed
Refund Payment

upvest

Event-Driven Architecture

enables scalability, resilience
promotes clear responsibility boundaries

upvest

Event-Driven Architecture

Thinking in events is not enough - you'll also need to think in terms of promises and railways.

Make sure not to approach it with an imperative mindset.

upvest

What do you think?
Let's discuss!

Please rate your experience ⭐⭐⭐⭐⭐

Mastodon @lutzhuehnken@mastodon.social

LinkedIn https://linkedin.com/in/lutzh

Blog https://www.reactivesystems.eu/