

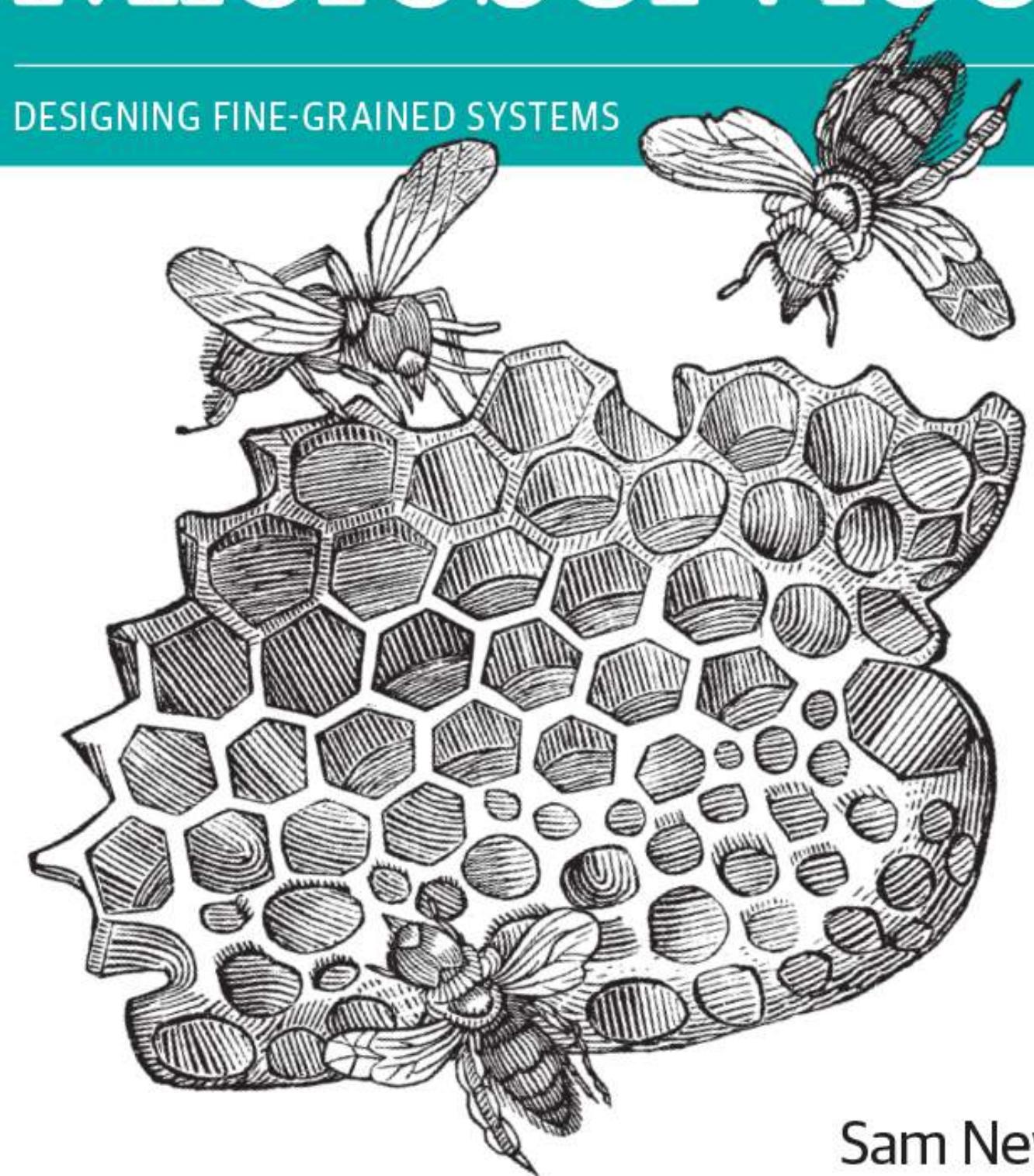
MICROSERVICE COLLABORATION

Sam Newman

O'REILLY®

Building Microservices

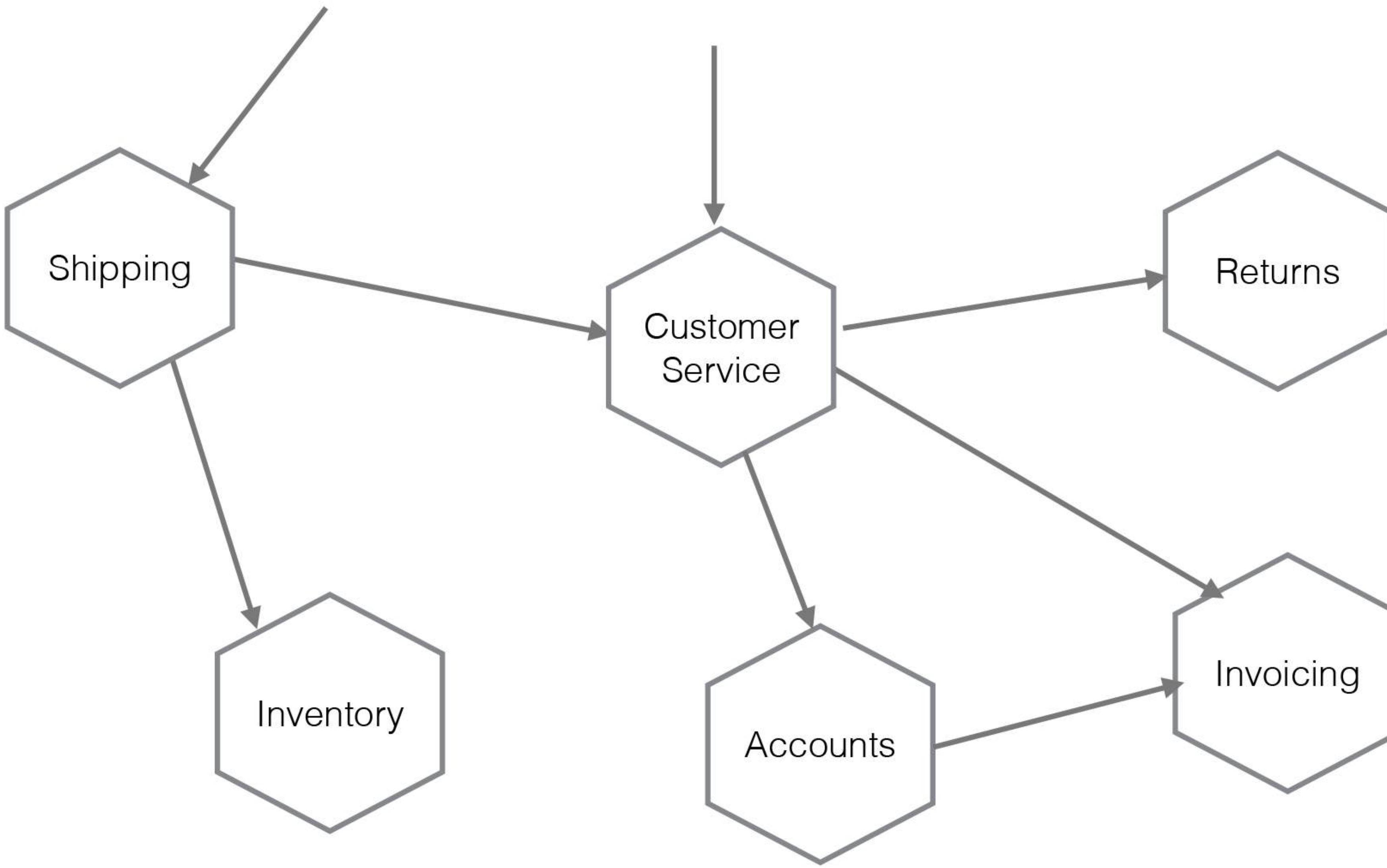
DESIGNING FINE-GRAINED SYSTEMS

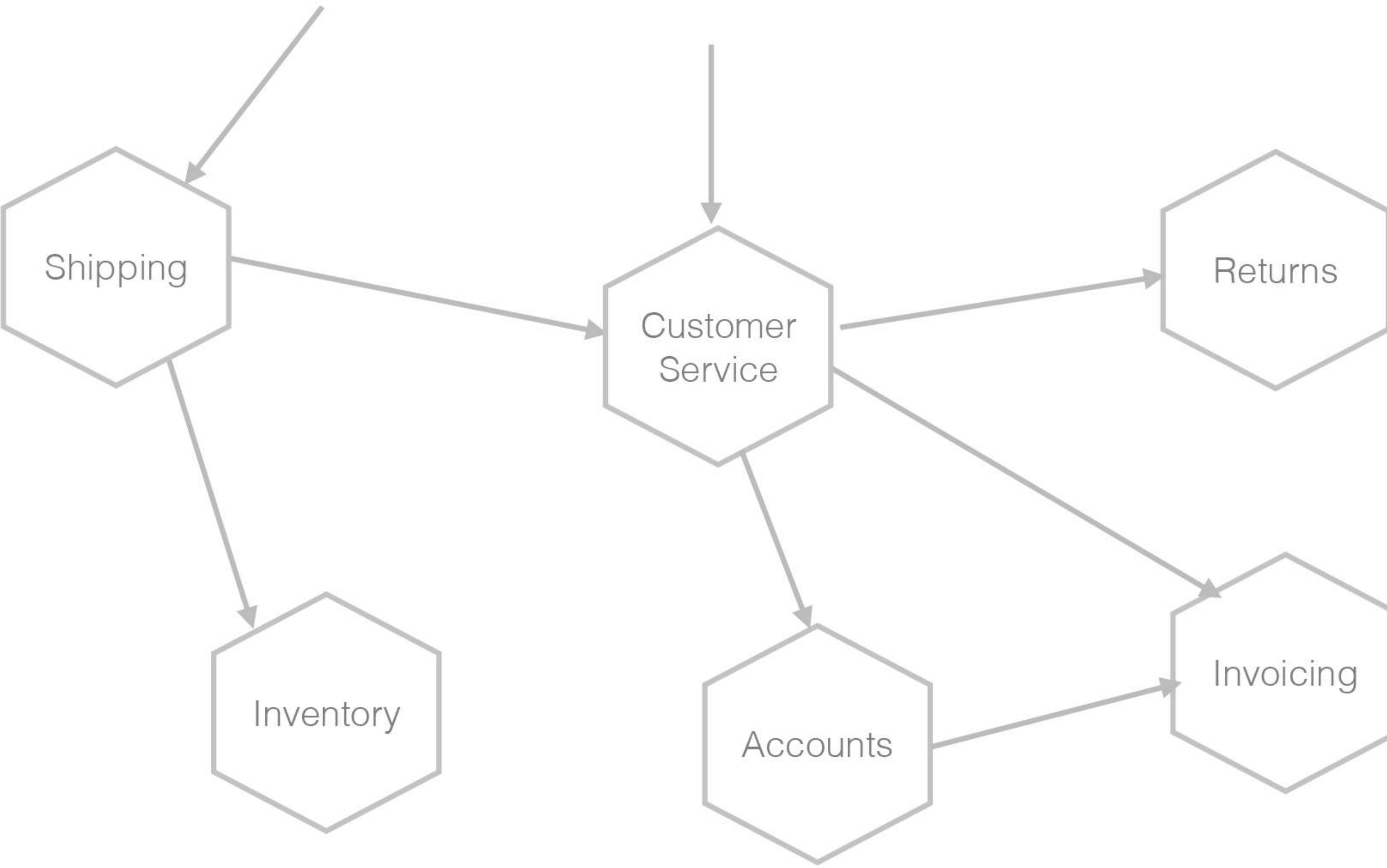


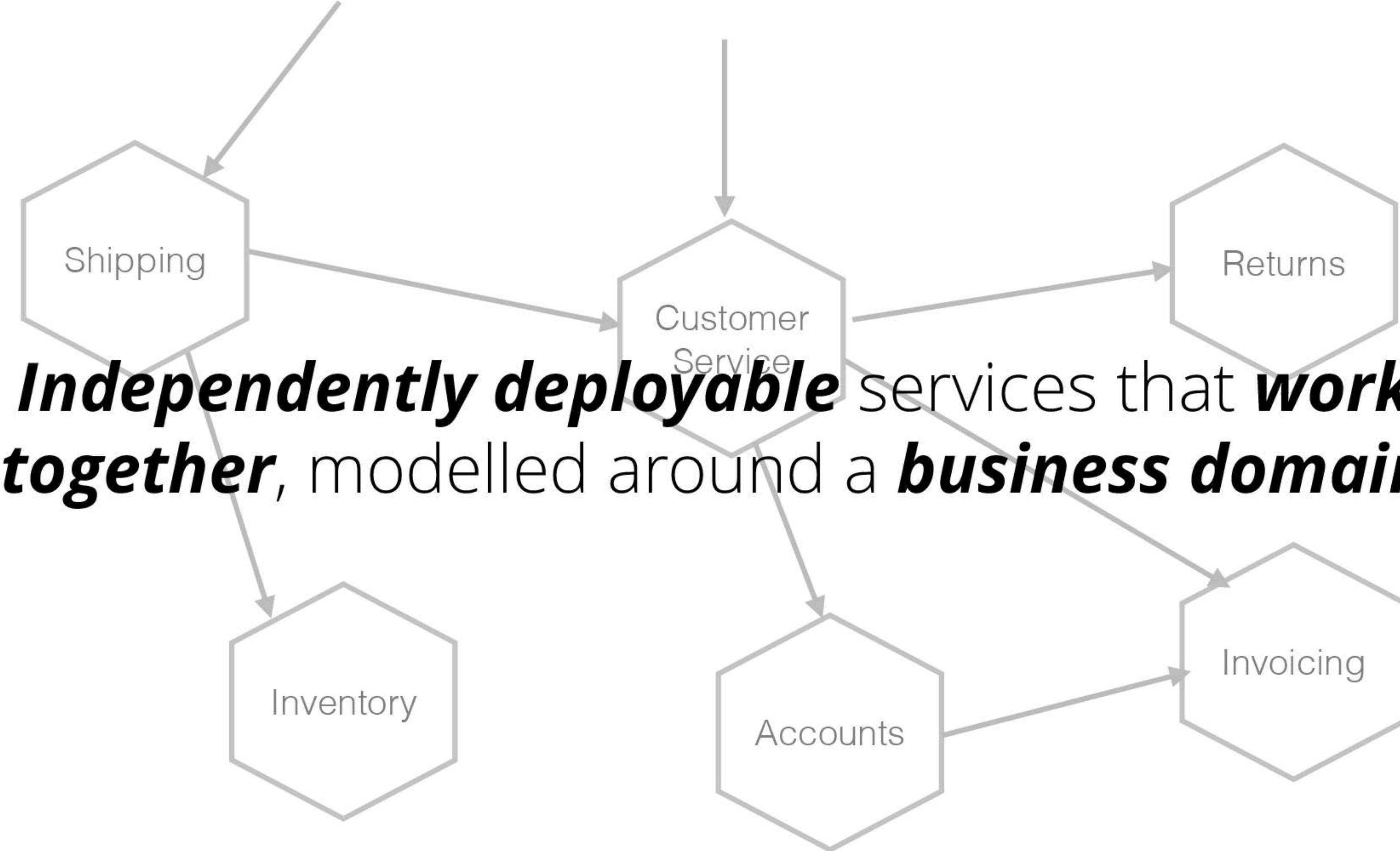
Sam Newman

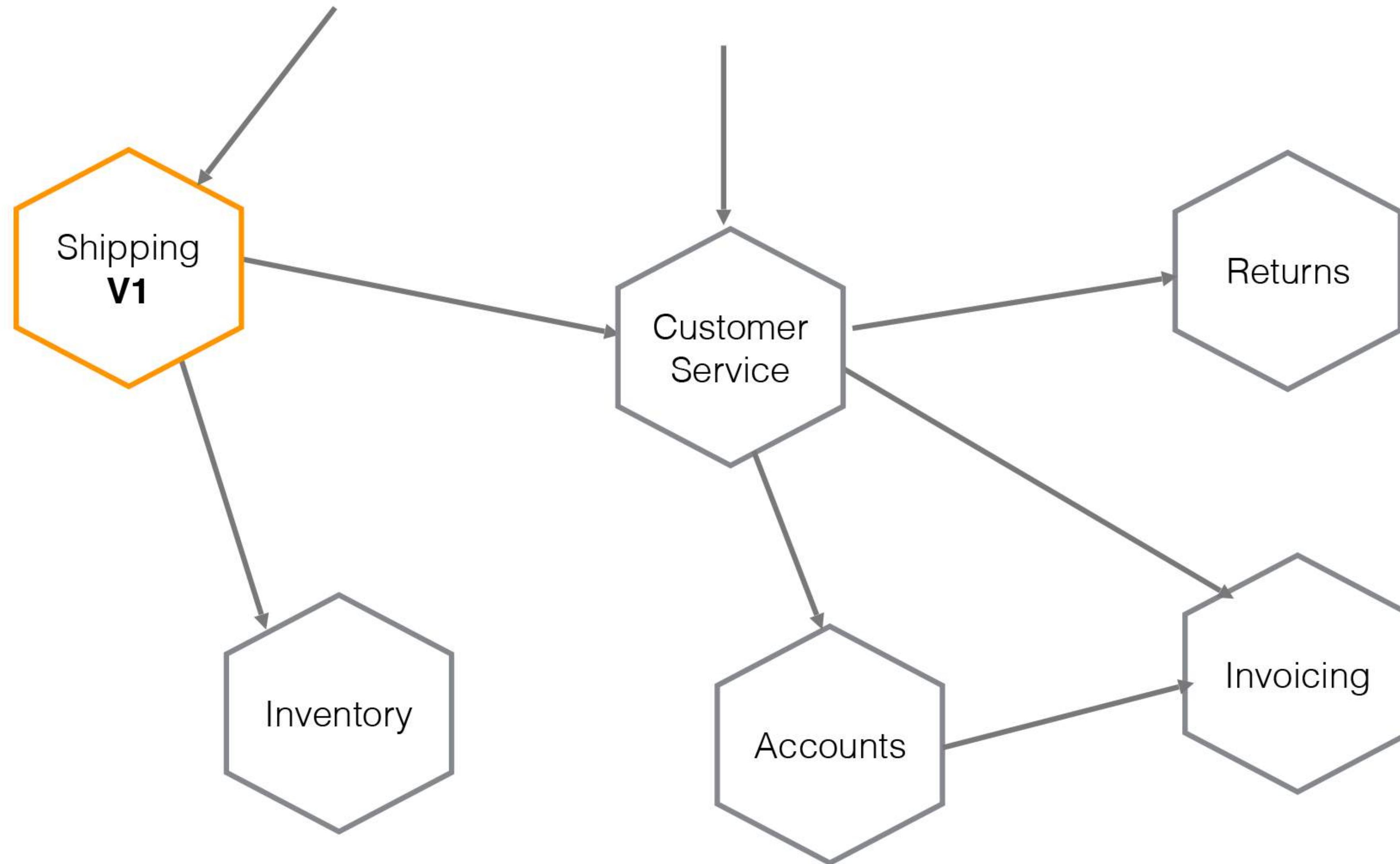
**Sam
Newman
& Associates**

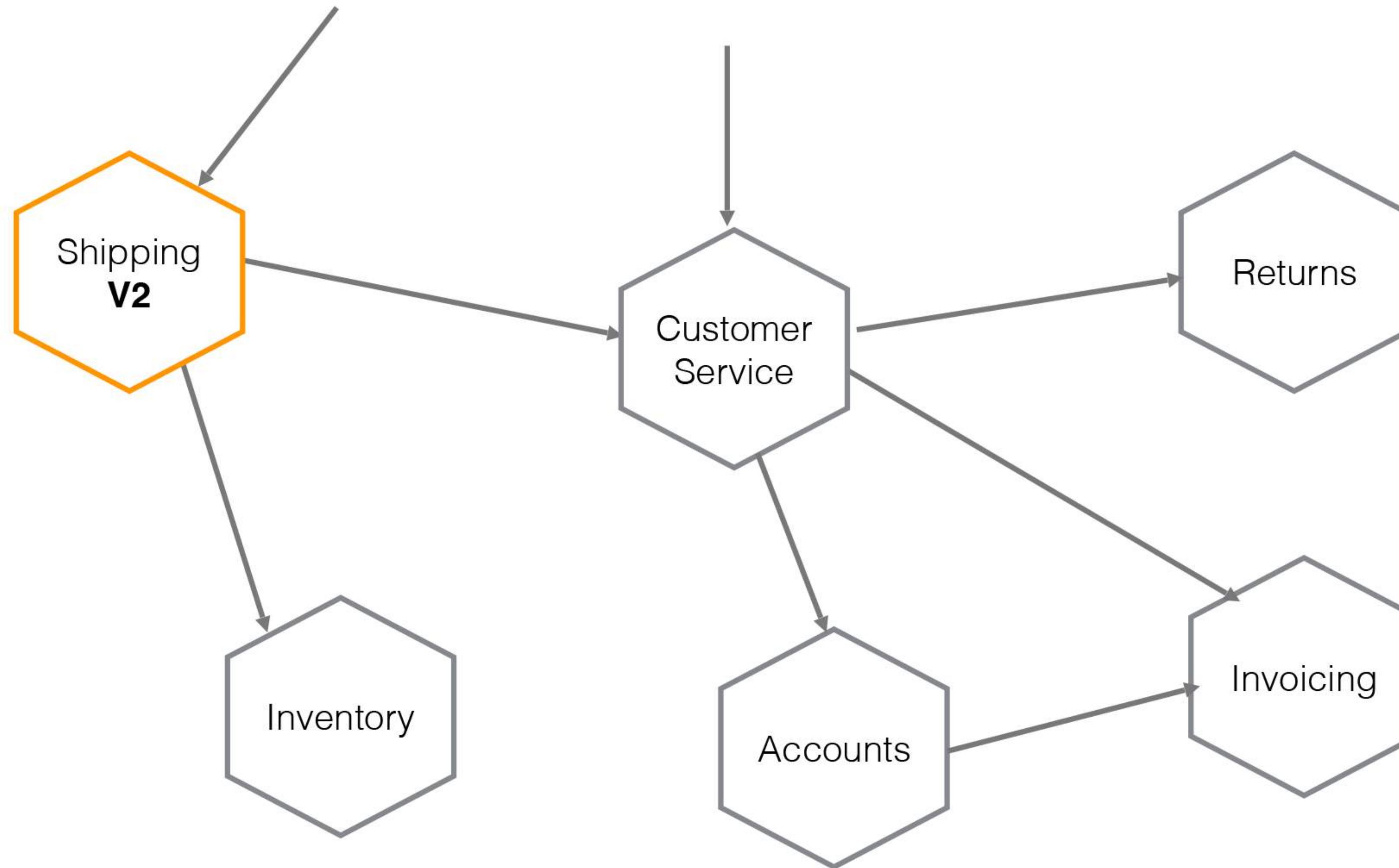












Autonomy

Autonomy?

Distributing Responsibility

Reducing Coordination

Poll

Where are you in your microservice journey?

Poll

Where are you in your microservice journey?

- a.) Thinking about if they're right for me

Poll

Where are you in your microservice journey?

- a.) Thinking about if they're right for me

Poll

Where are you in your microservice journey?

- a.) Thinking about if they're right for me
- b.) Just started using them

Poll

Where are you in your microservice journey?

- a.) Thinking about if they're right for me
- b.) Just started using them

Poll

Where are you in your microservice journey?

- a.) Thinking about if they're right for me
- b.) Just started using them
- c.) Been using them for a while!

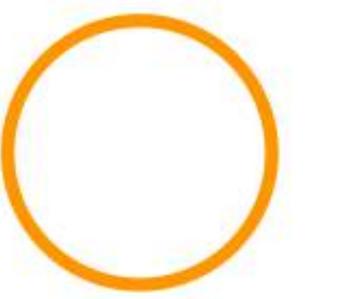
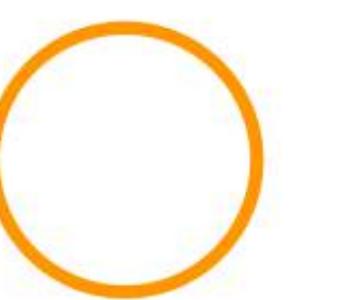
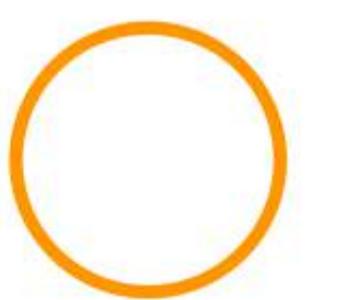
“No, communication is terrible!”

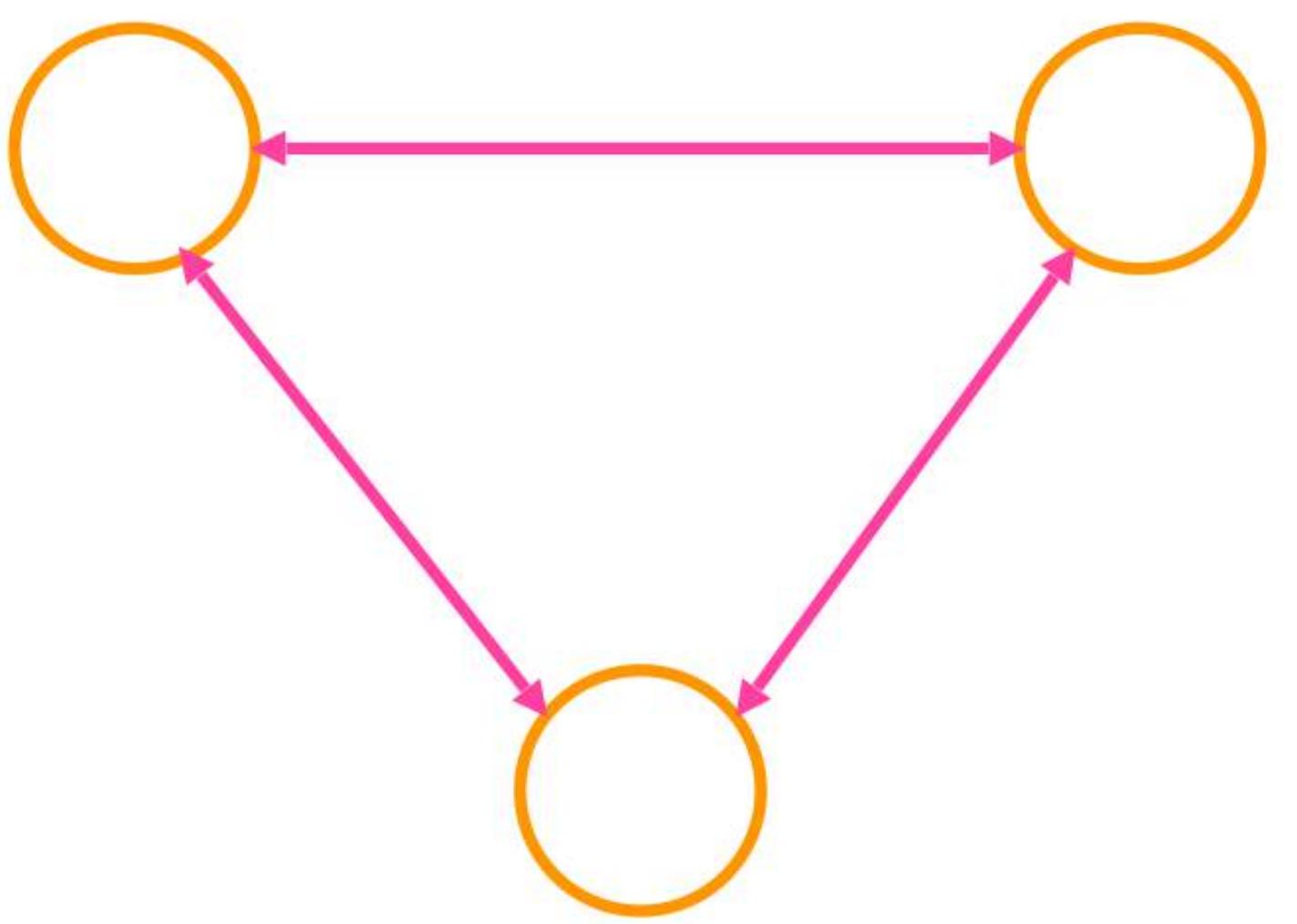
“No, communication is terrible!”

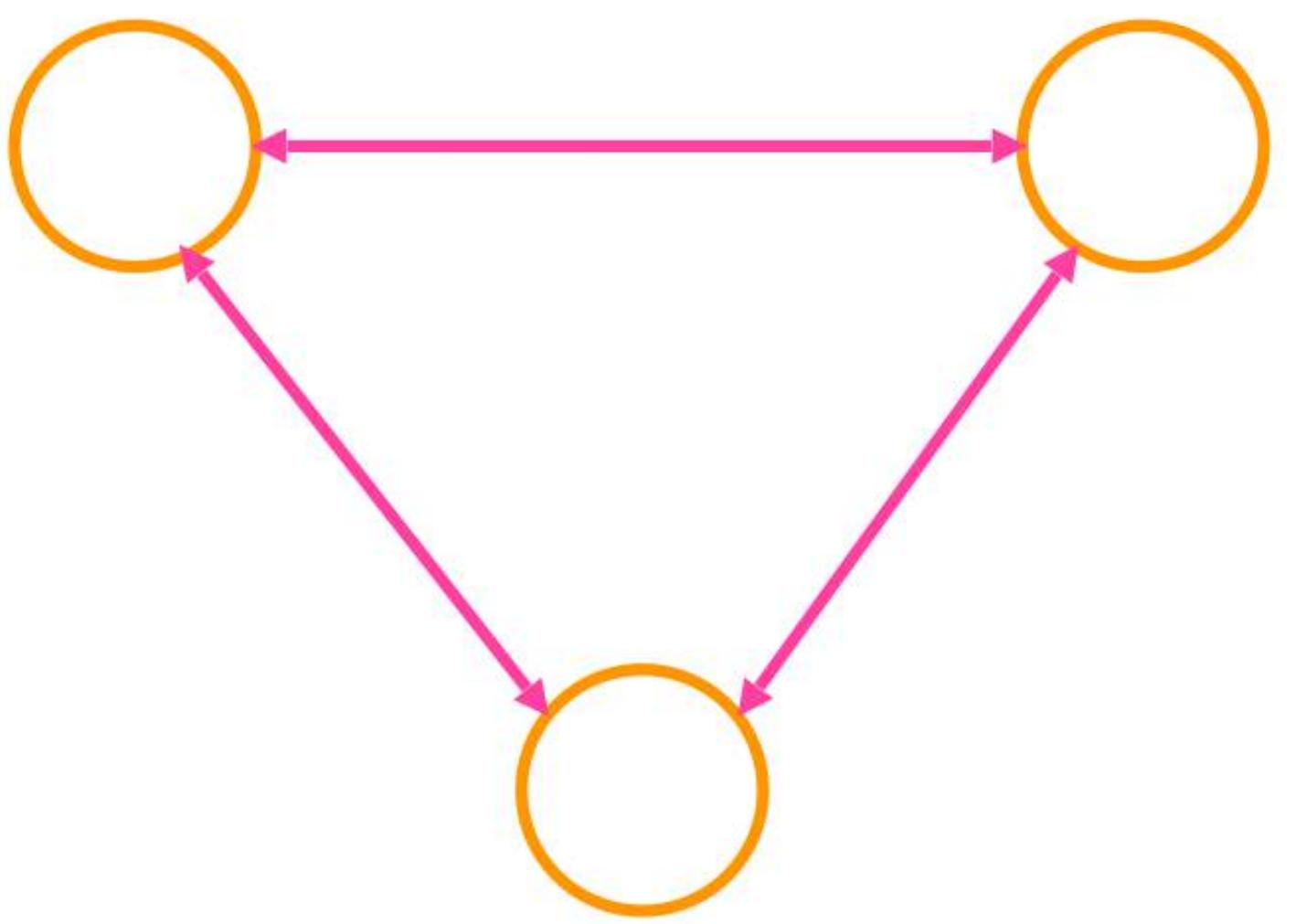
- Jeff Bezos

Citation: <http://www.businessinsider.com/the-strategies-jeff-bezos-used-to-build-the-amazon-empire-2014-3>

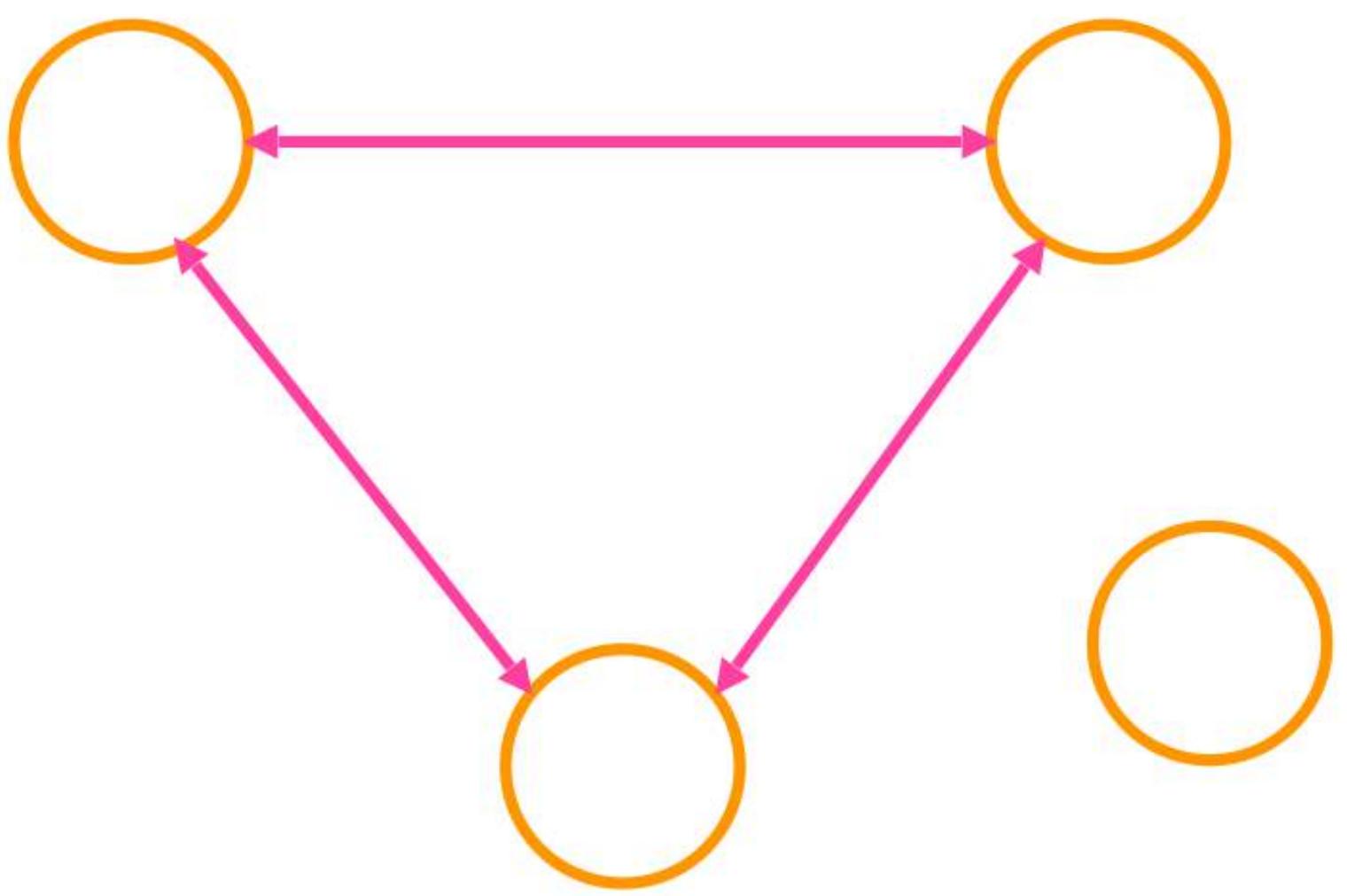
@samnewman



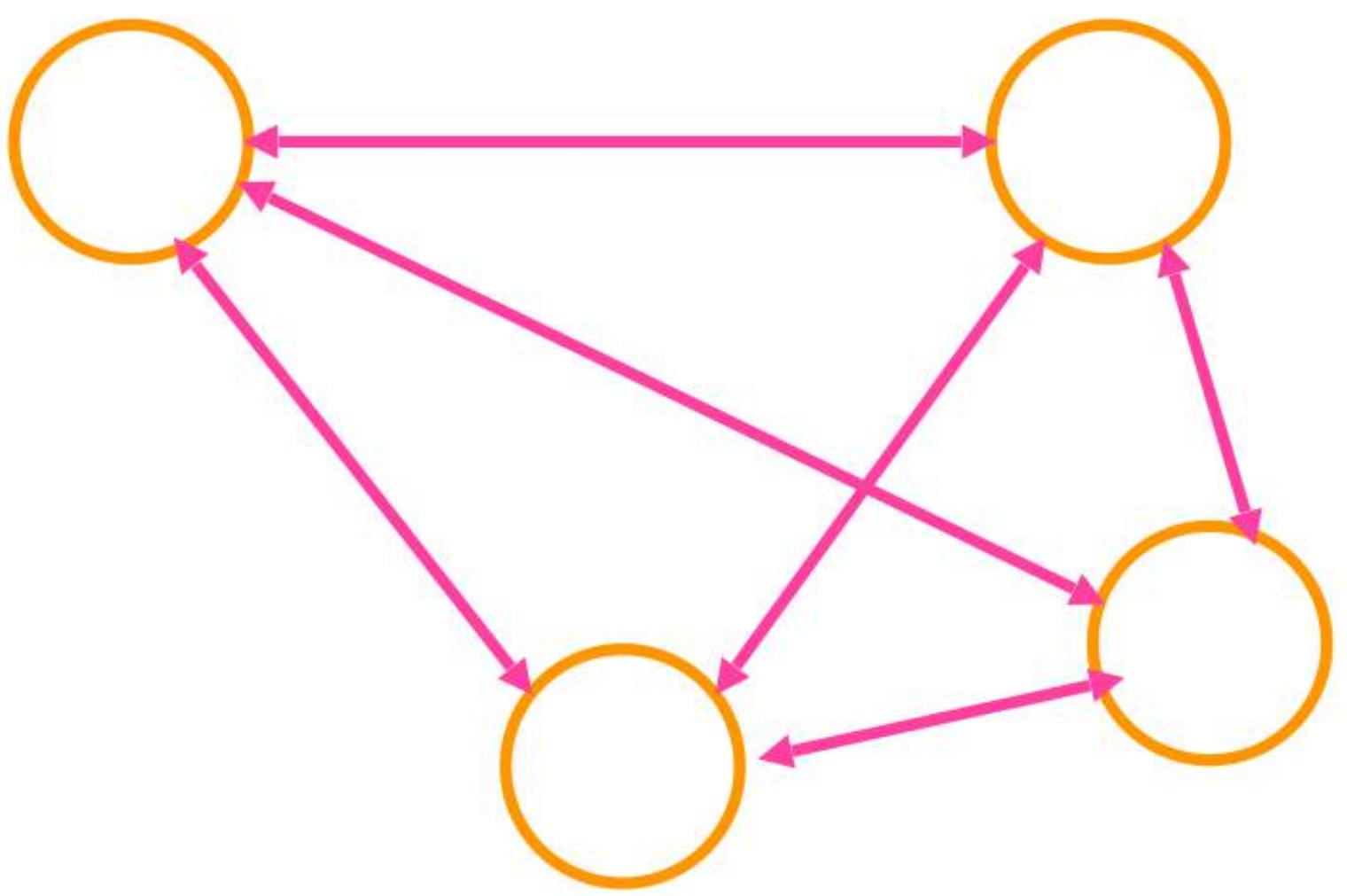




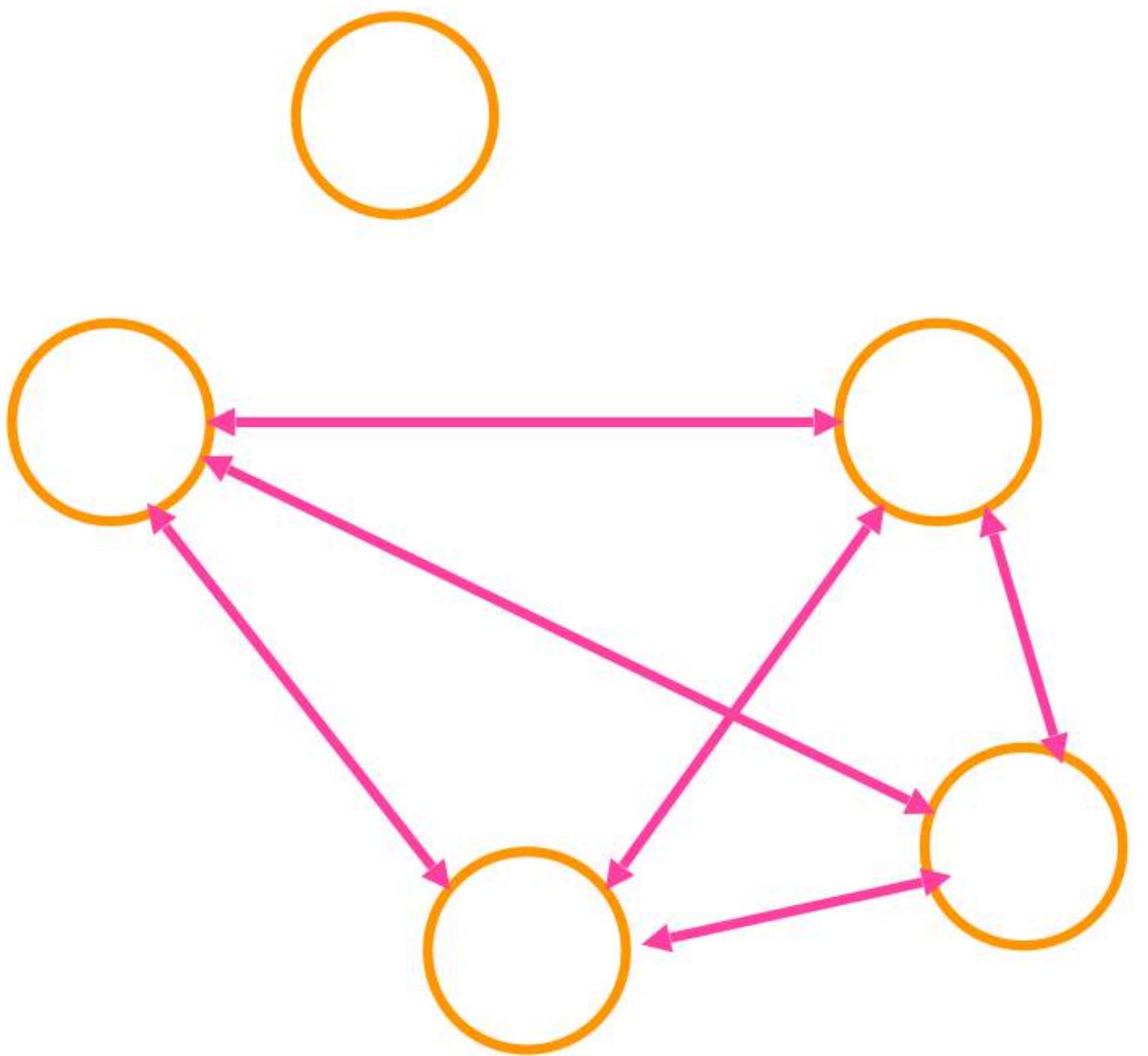
Communication pathways 3



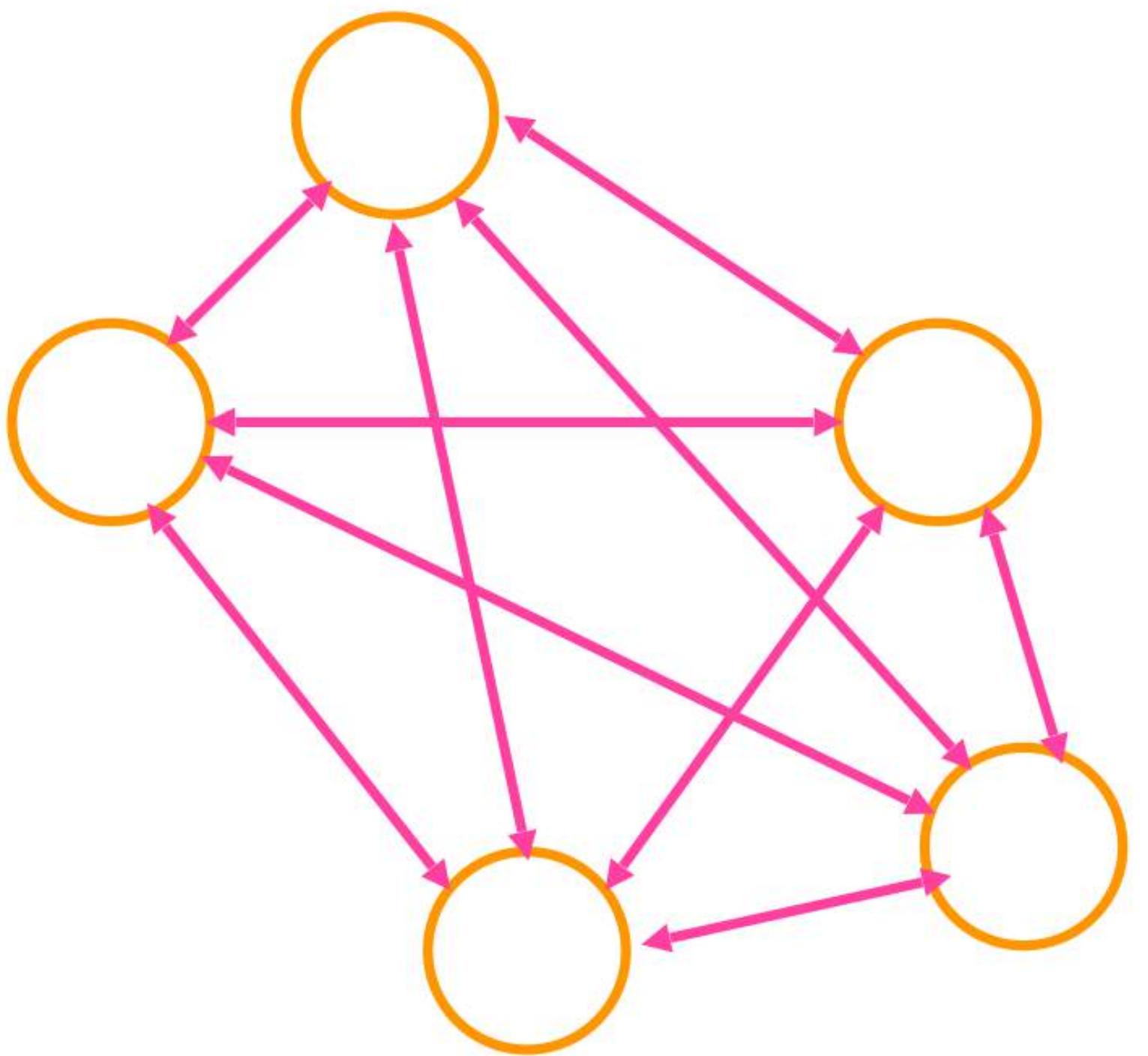
Communication pathways 3



Communication pathways 6



Communication pathways 6



Communication pathways 10

STEVE YEGGE'S GOOGLE PLATFORMS RANT

<https://plus.google.com/+RipRowan/posts/eVeouesvaVX>

STEVE YEGGE'S GOOGLE PLATFORMS RANT

“All teams will henceforth expose their data and functionality through service interfaces”

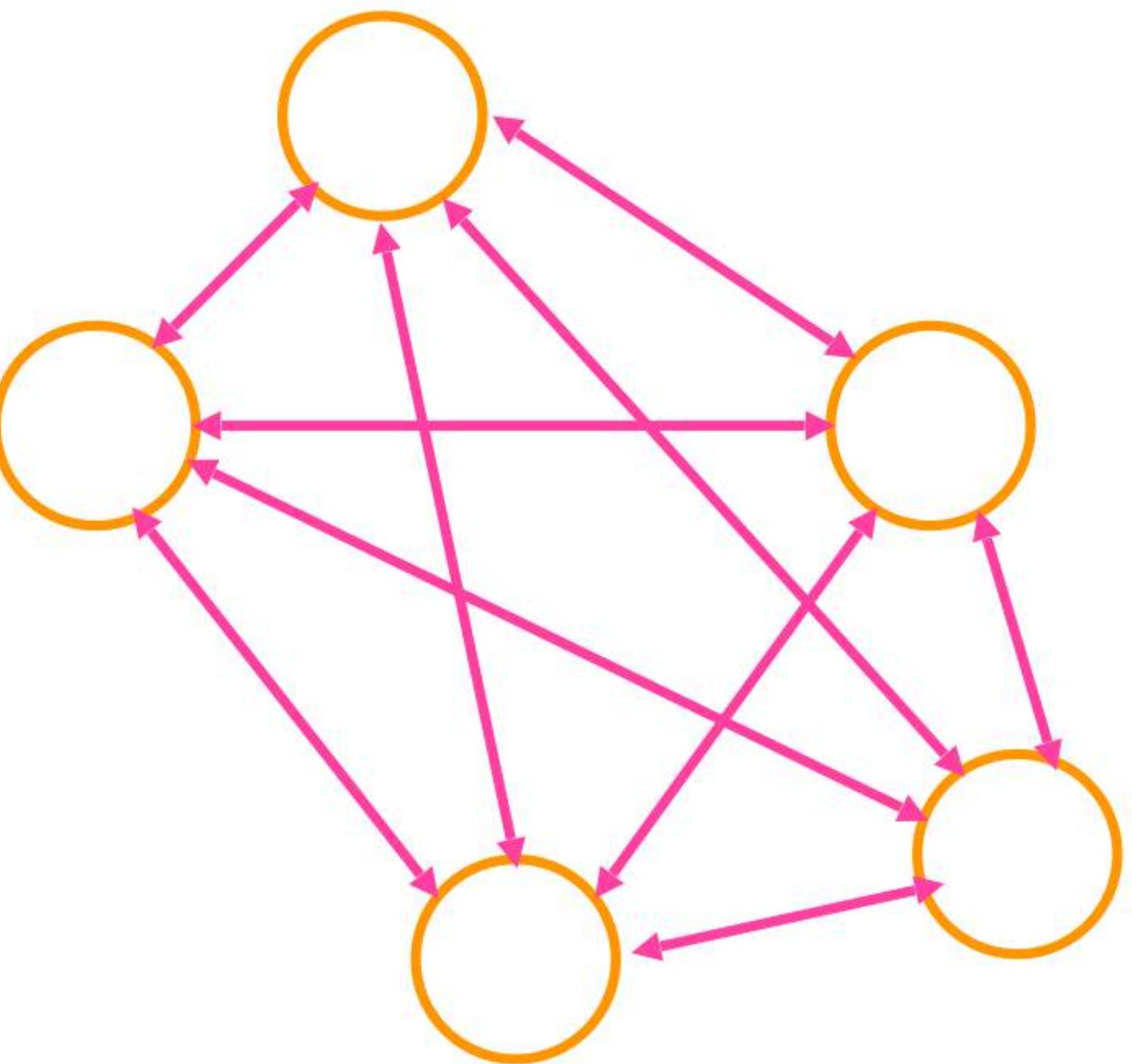
<https://plus.google.com/+RipRowan/posts/eVeouesvaVX>

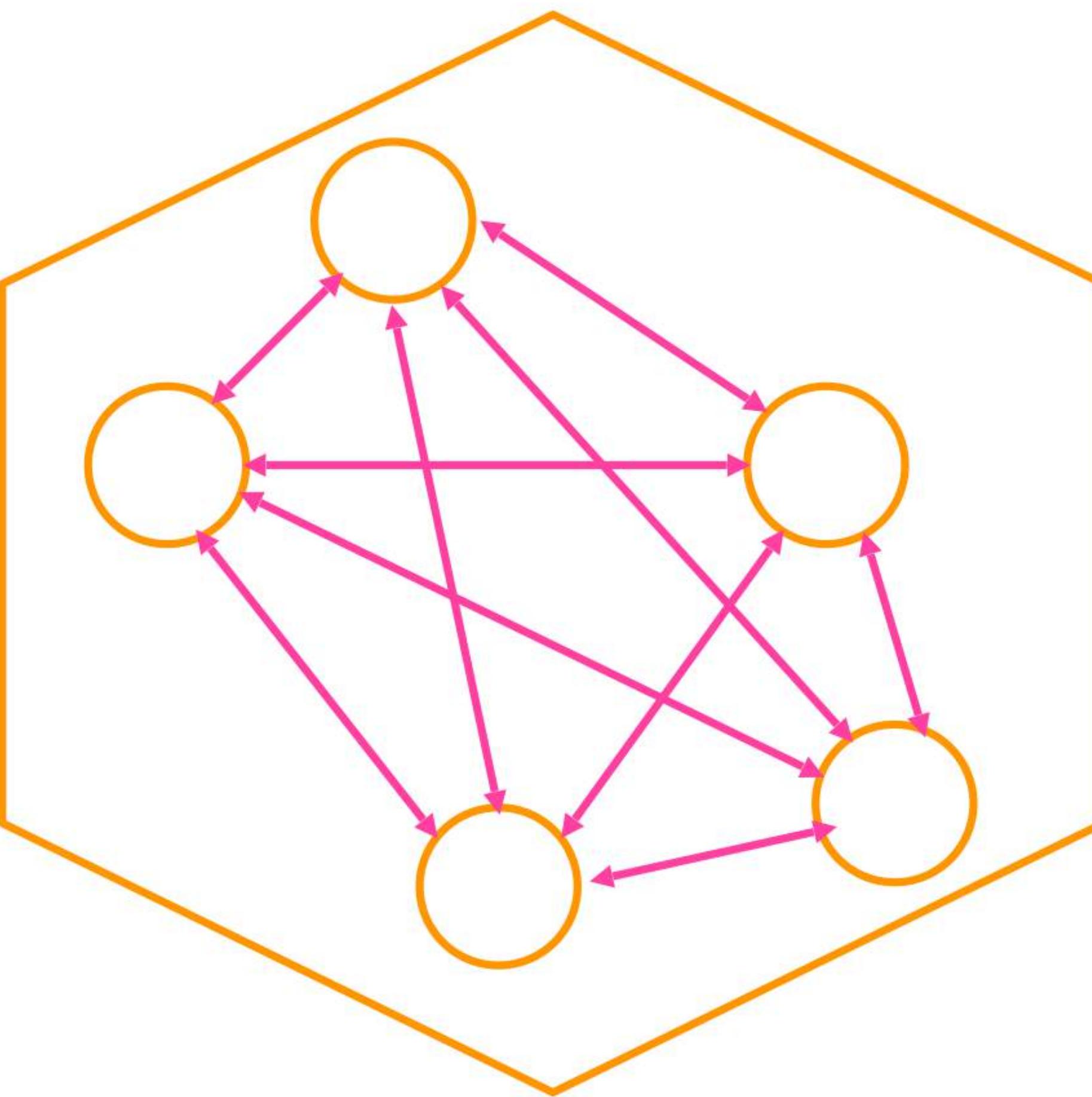
STEVE YEGGE'S GOOGLE PLATFORMS RANT

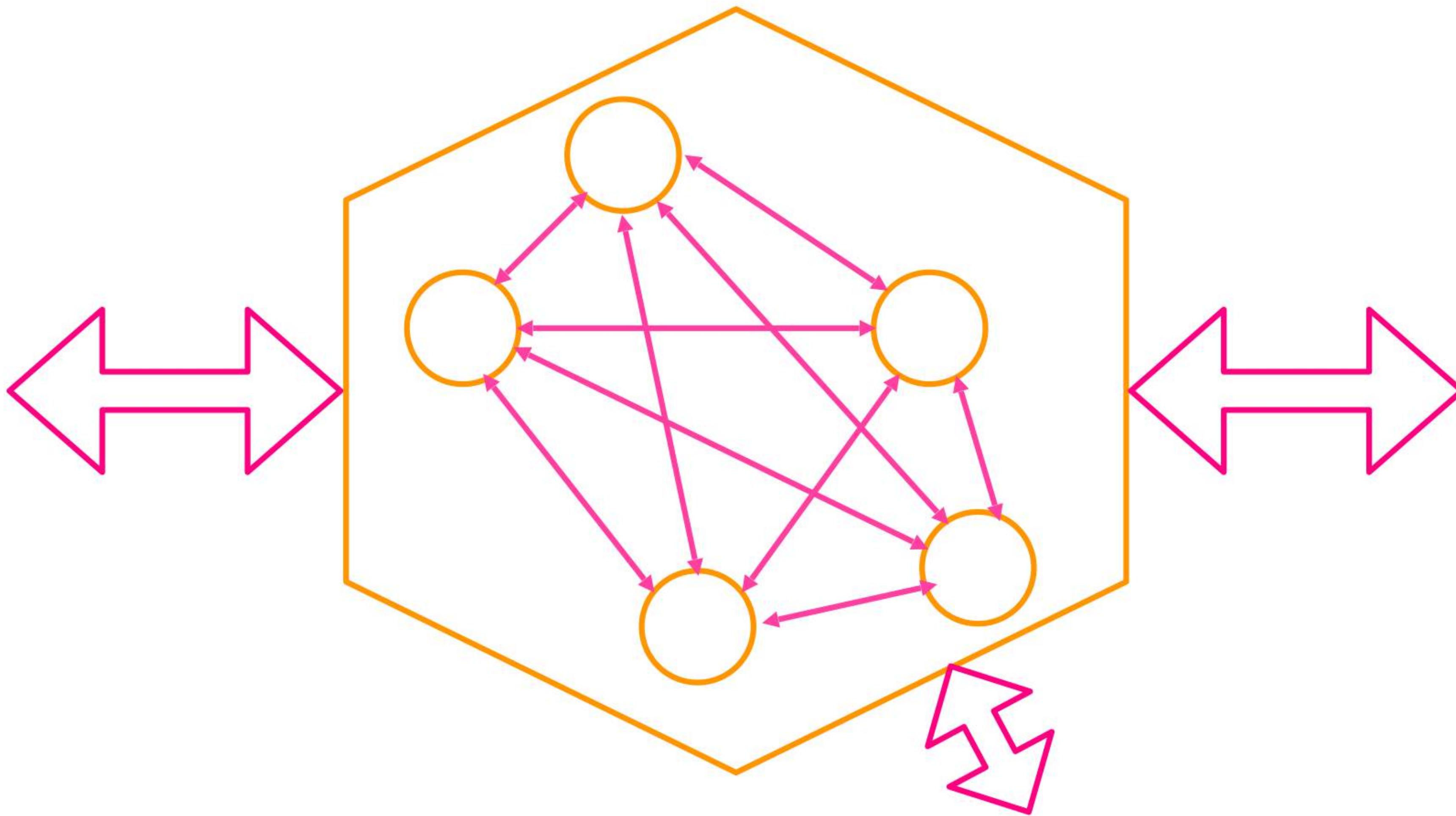
“All teams will henceforth expose their data and functionality through service interfaces”

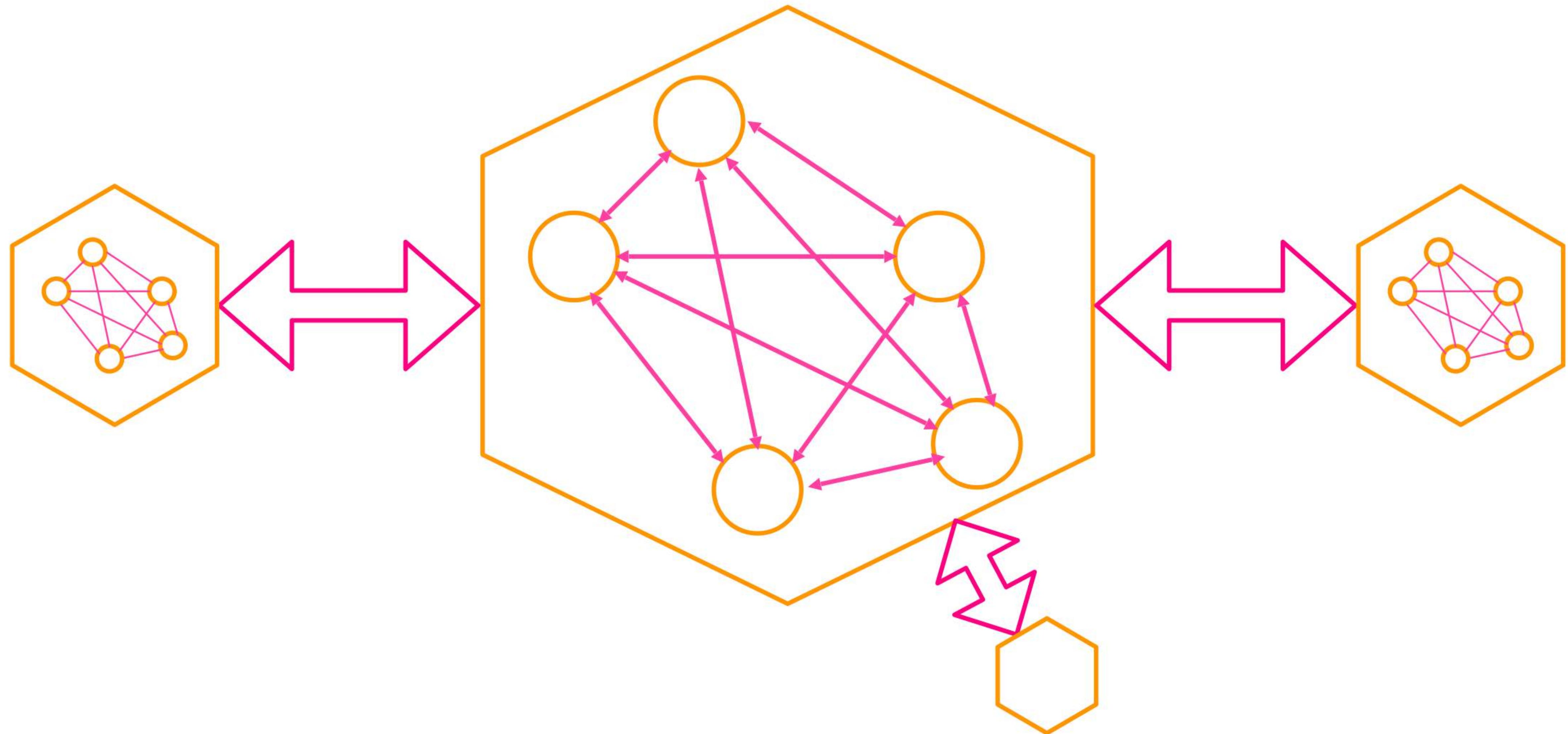
“Teams must communicate with each other through these interfaces”

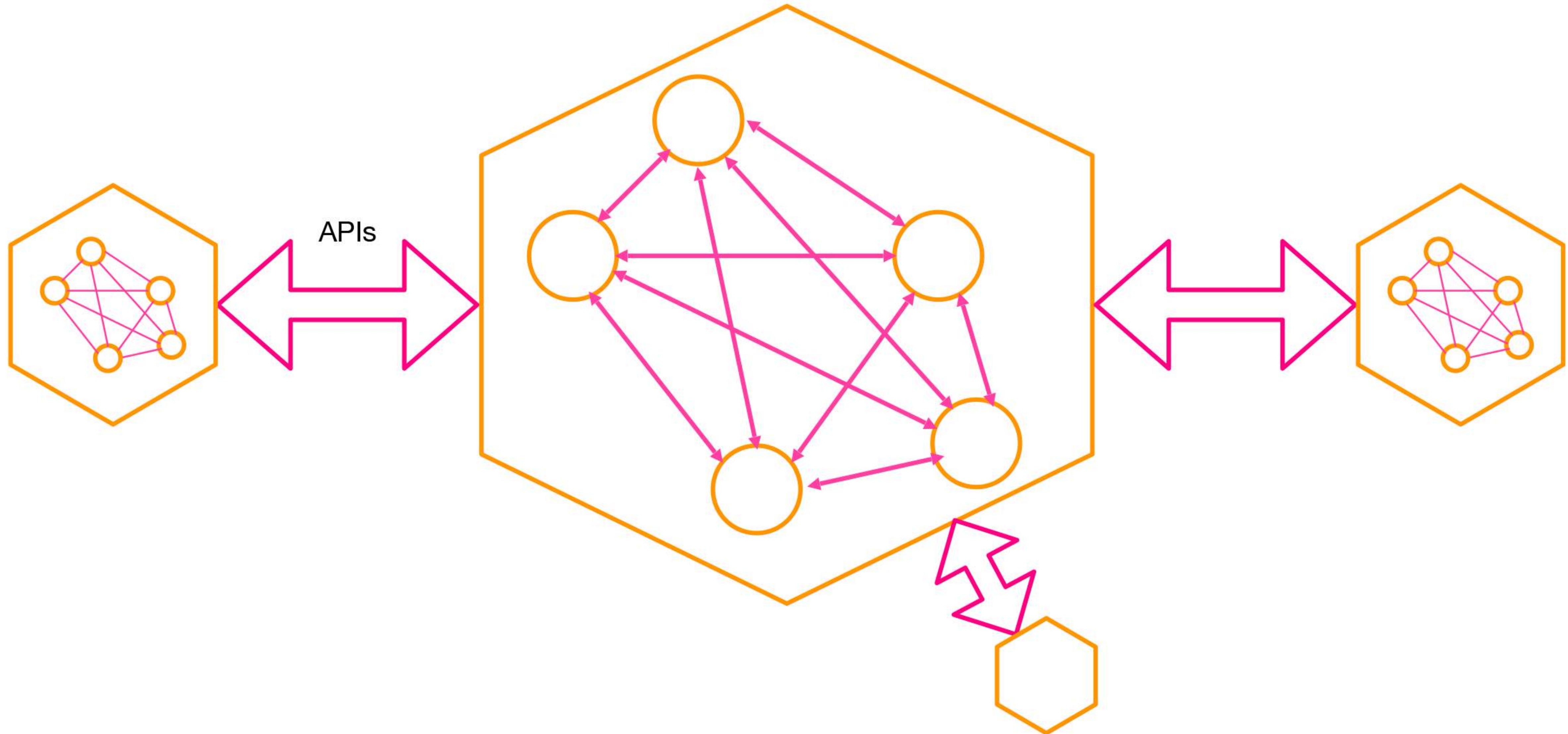
<https://plus.google.com/+RipRowan/posts/eVeouesvaVX>

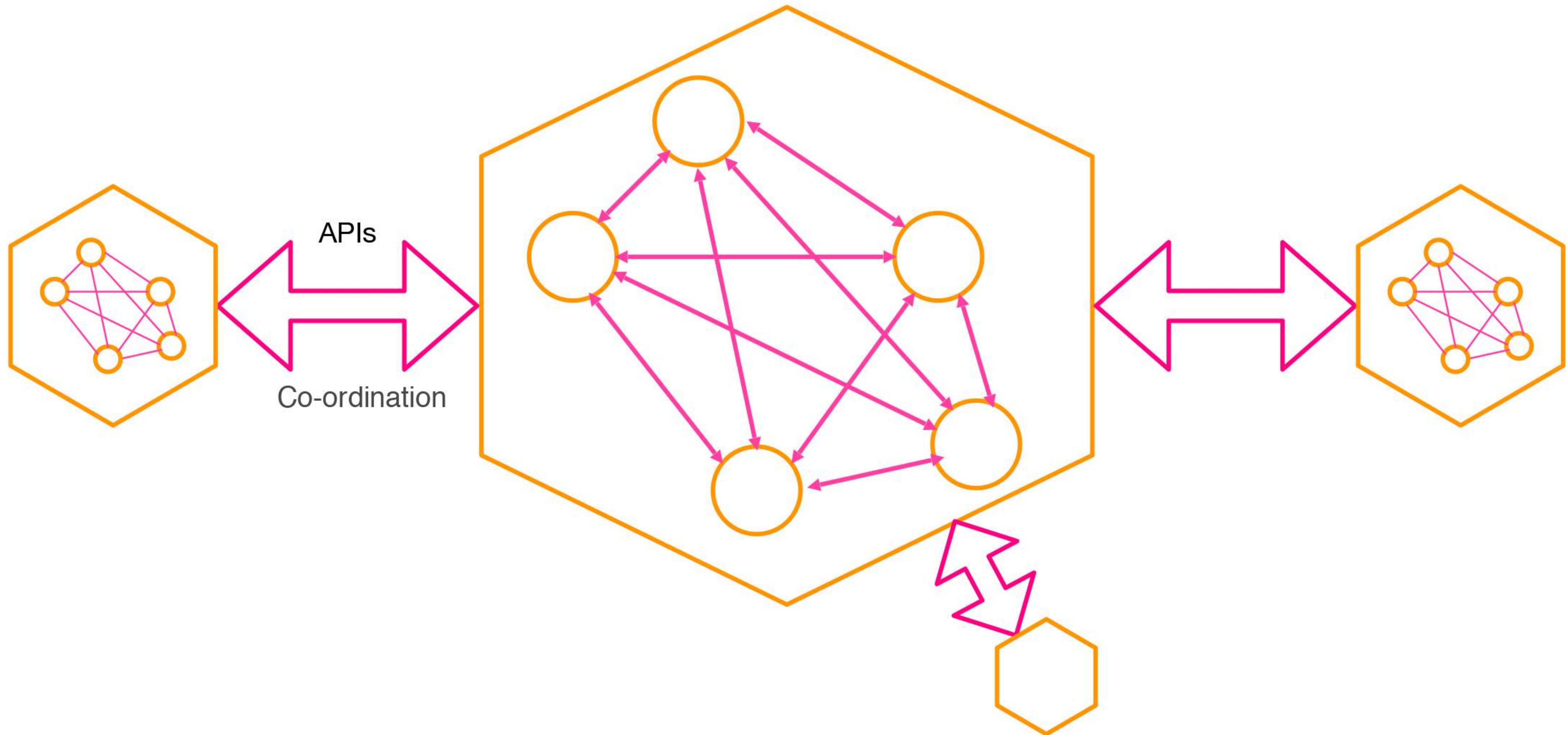


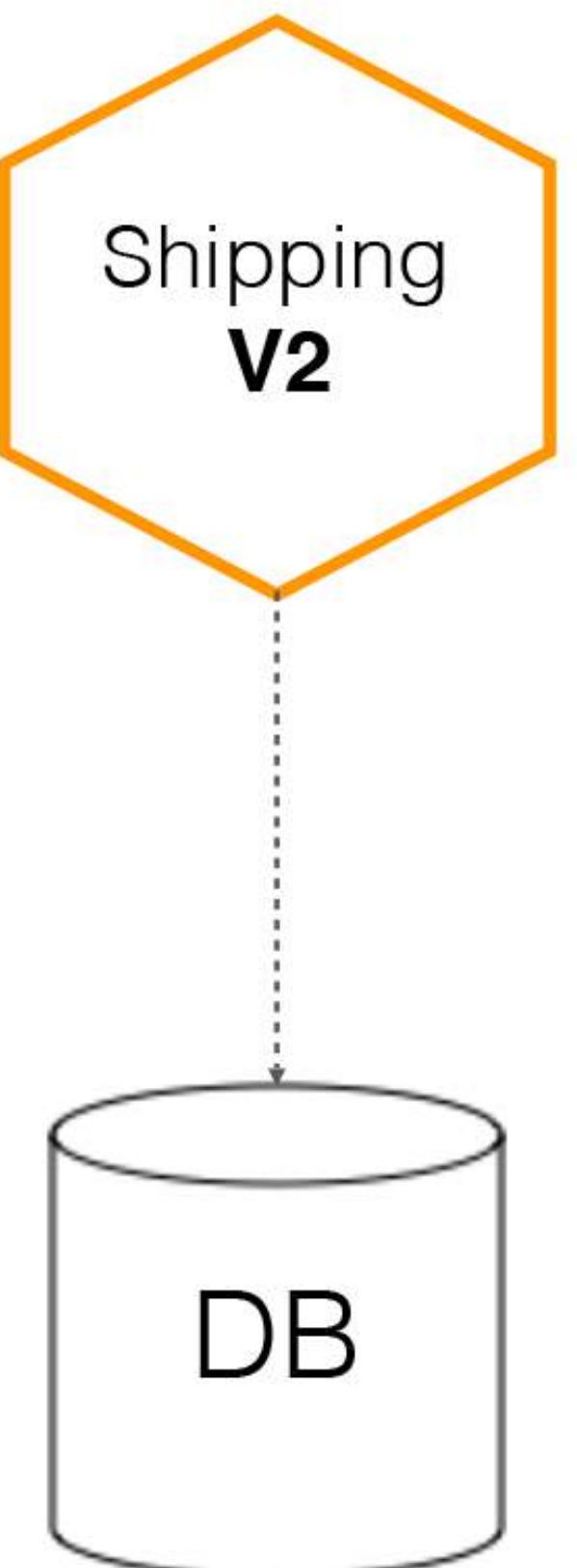


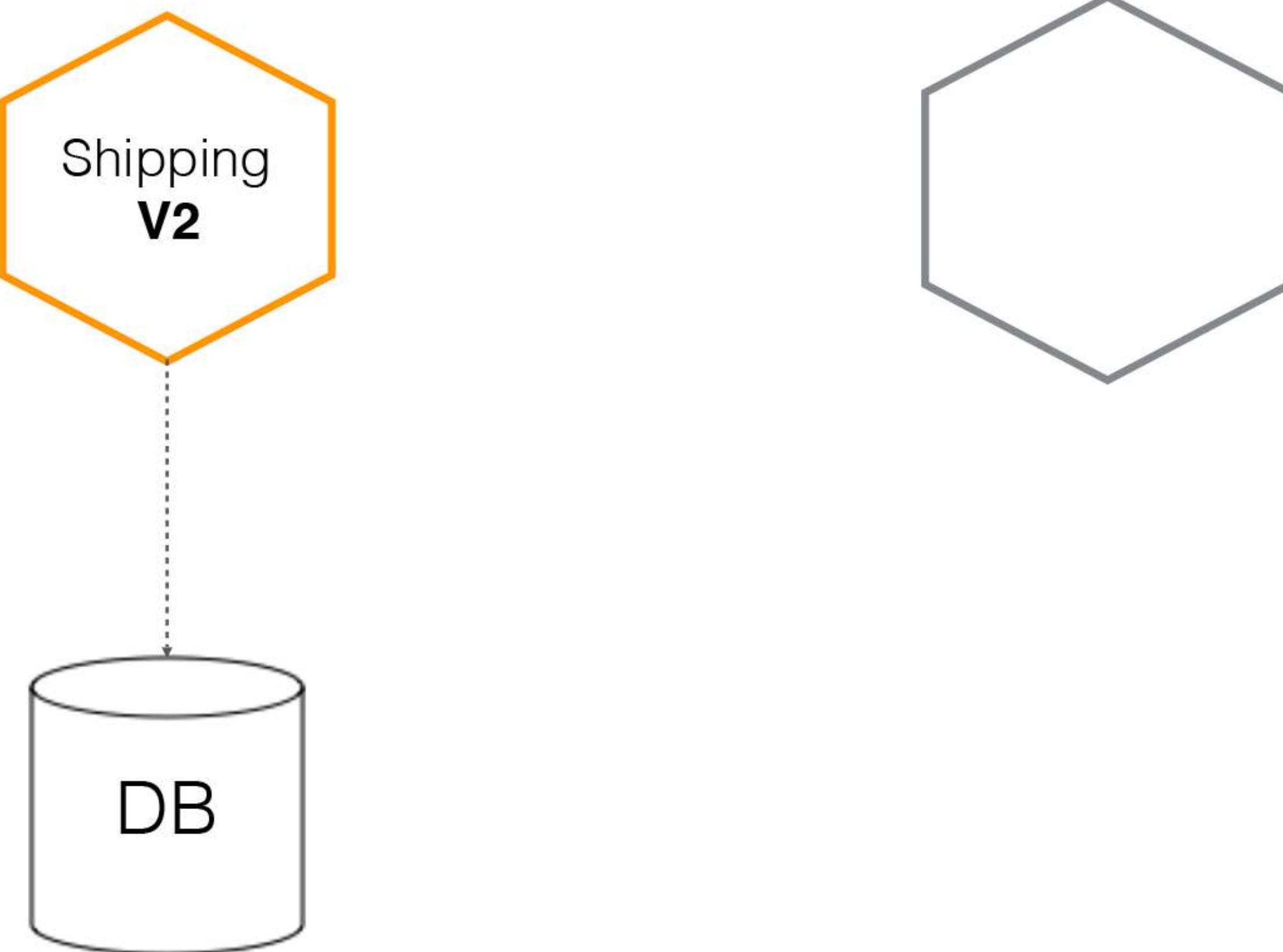




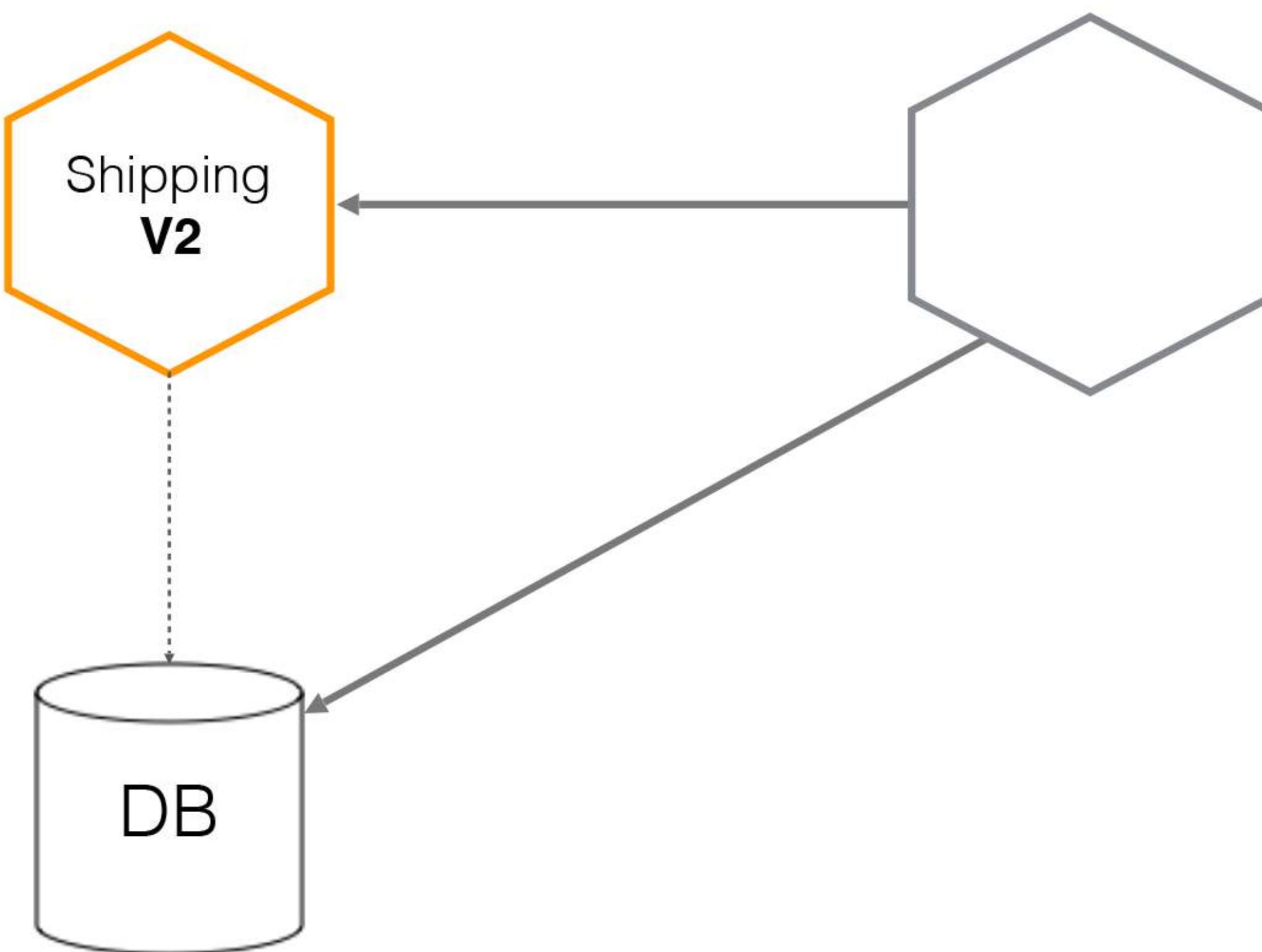


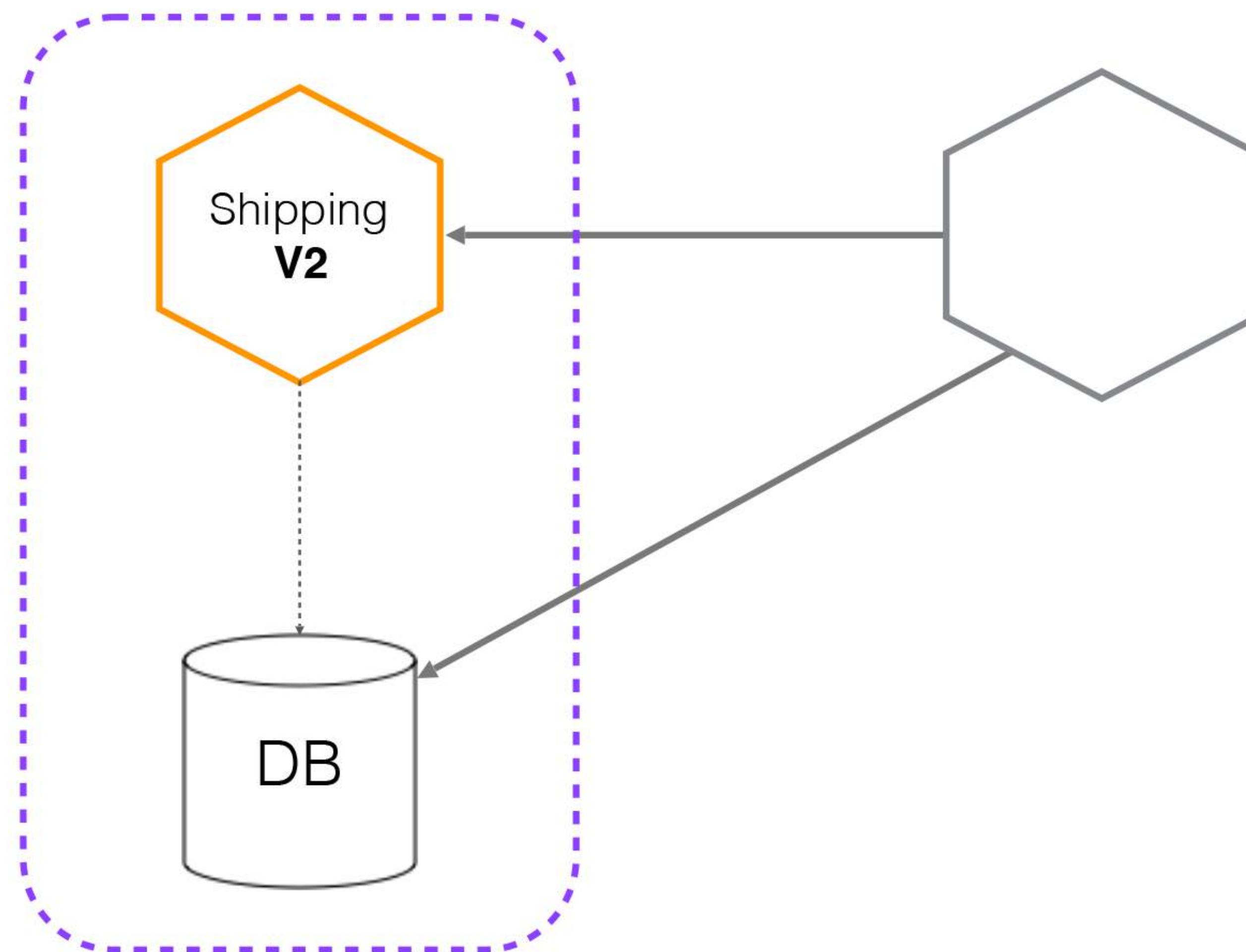


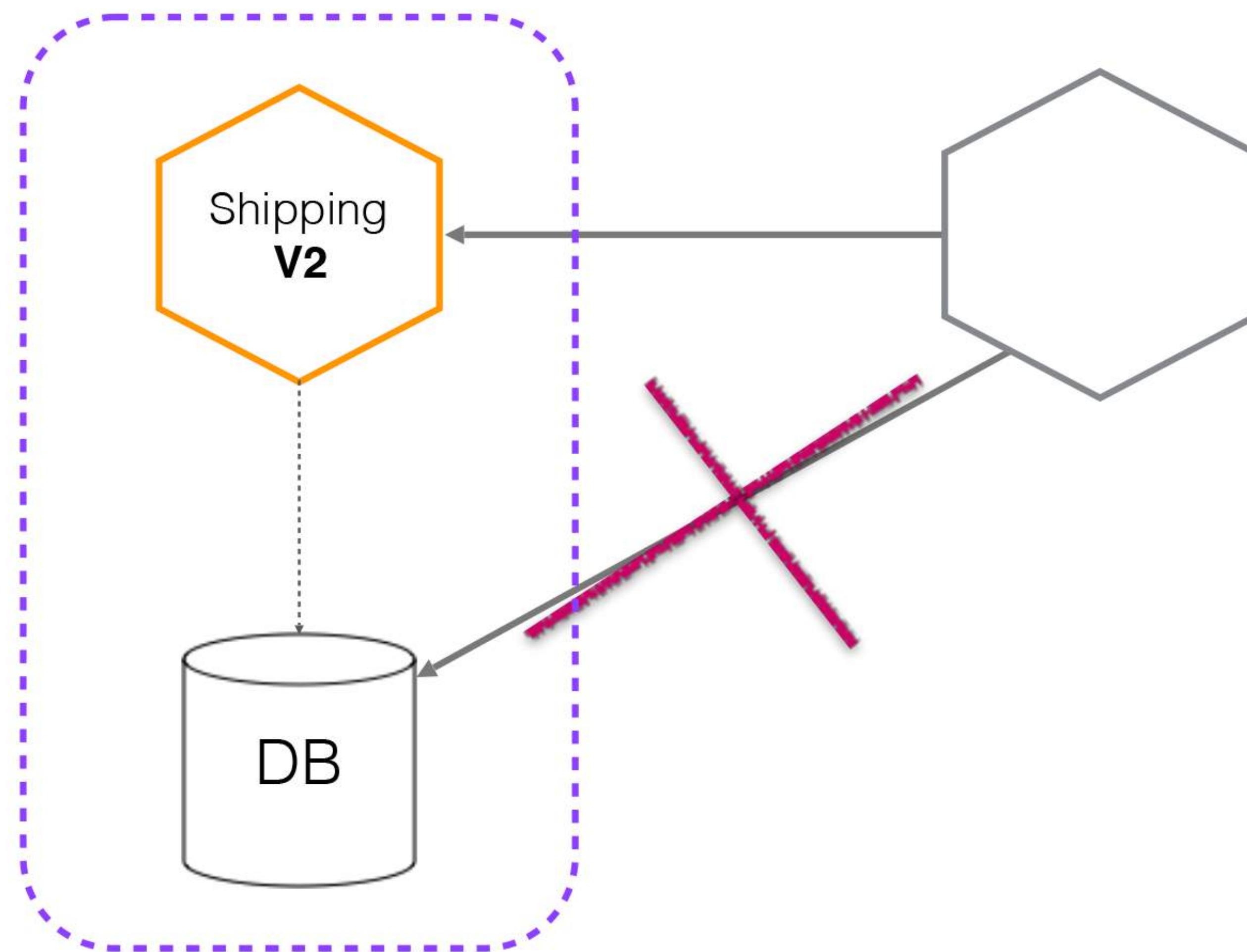




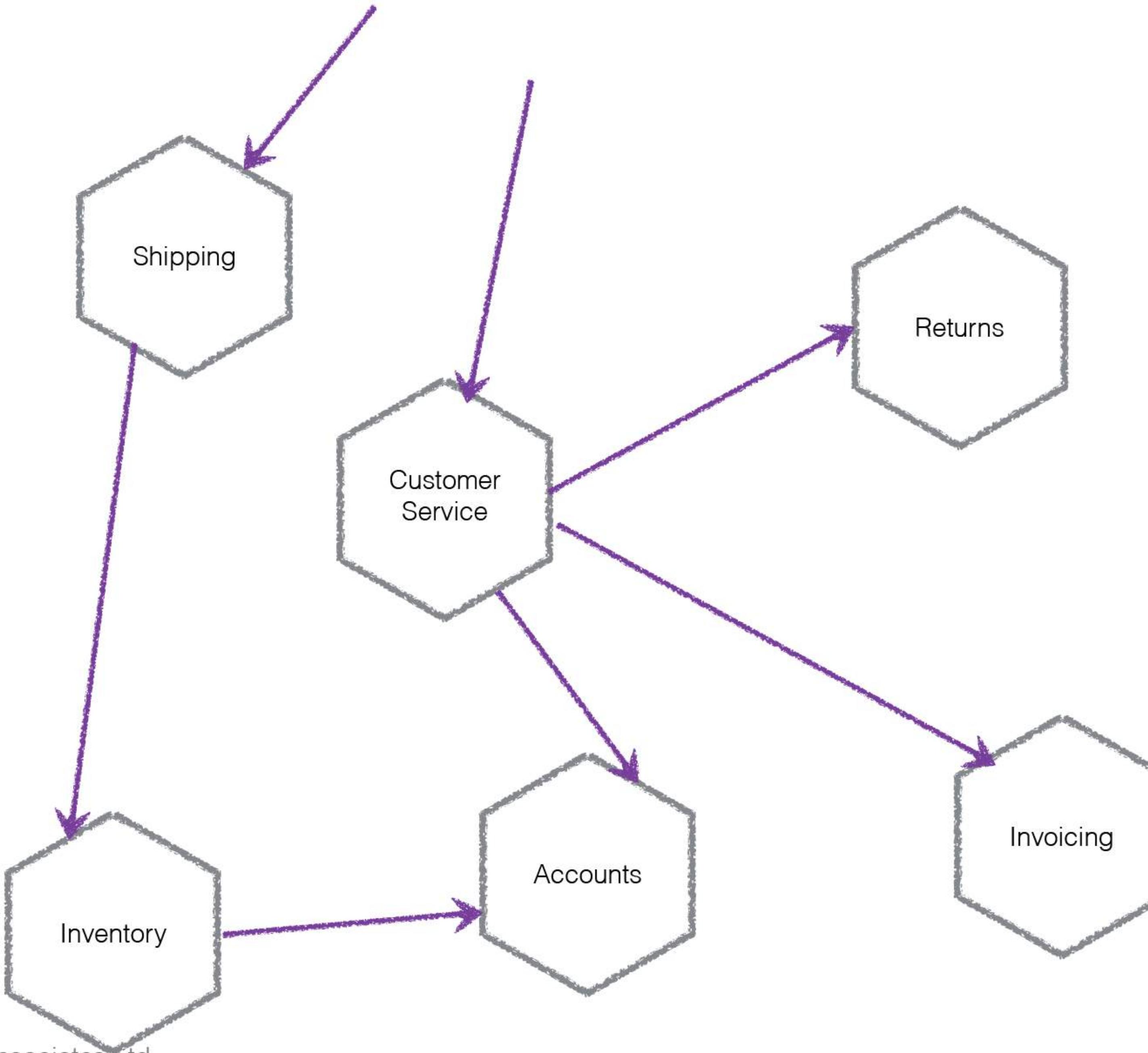


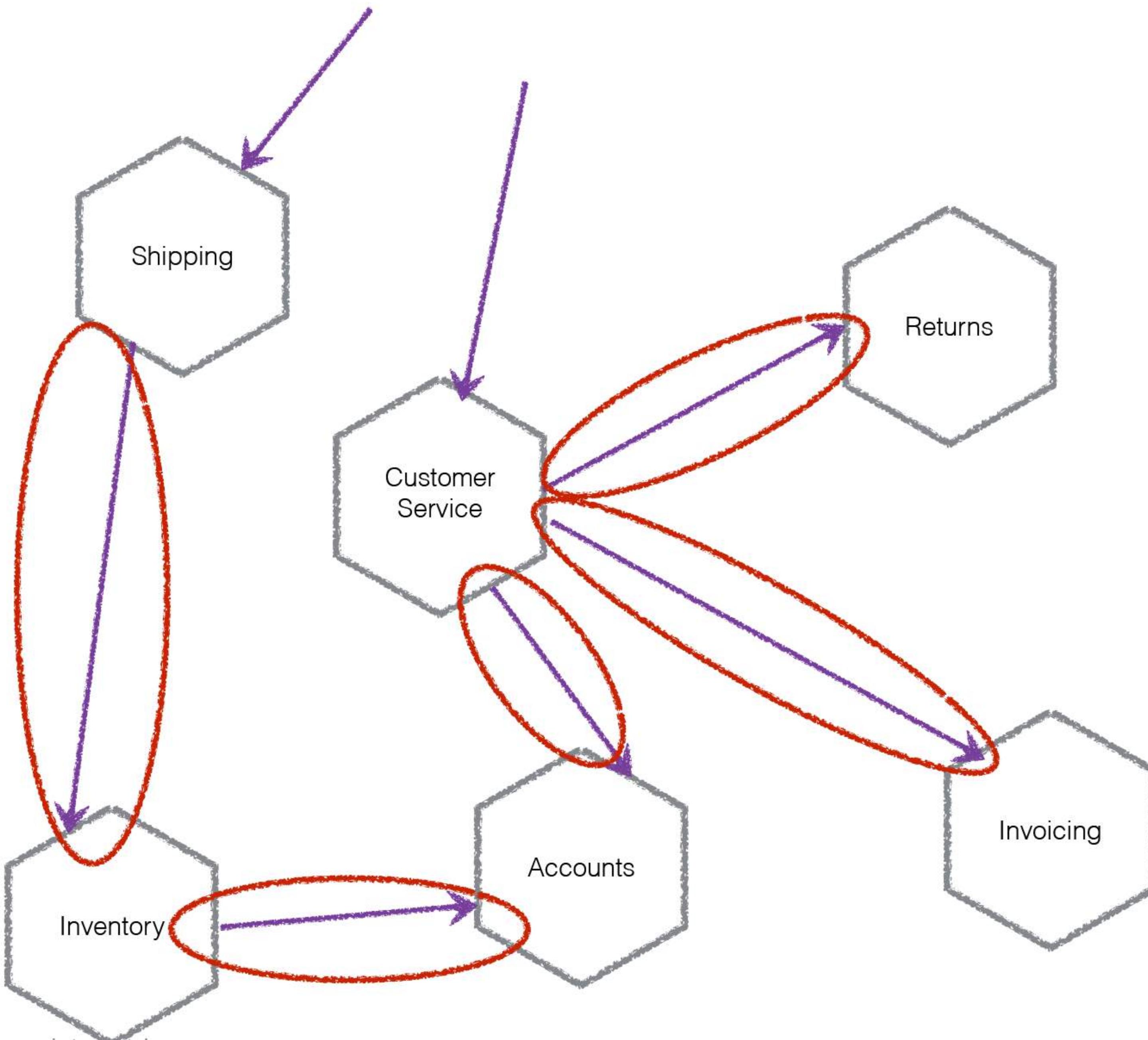




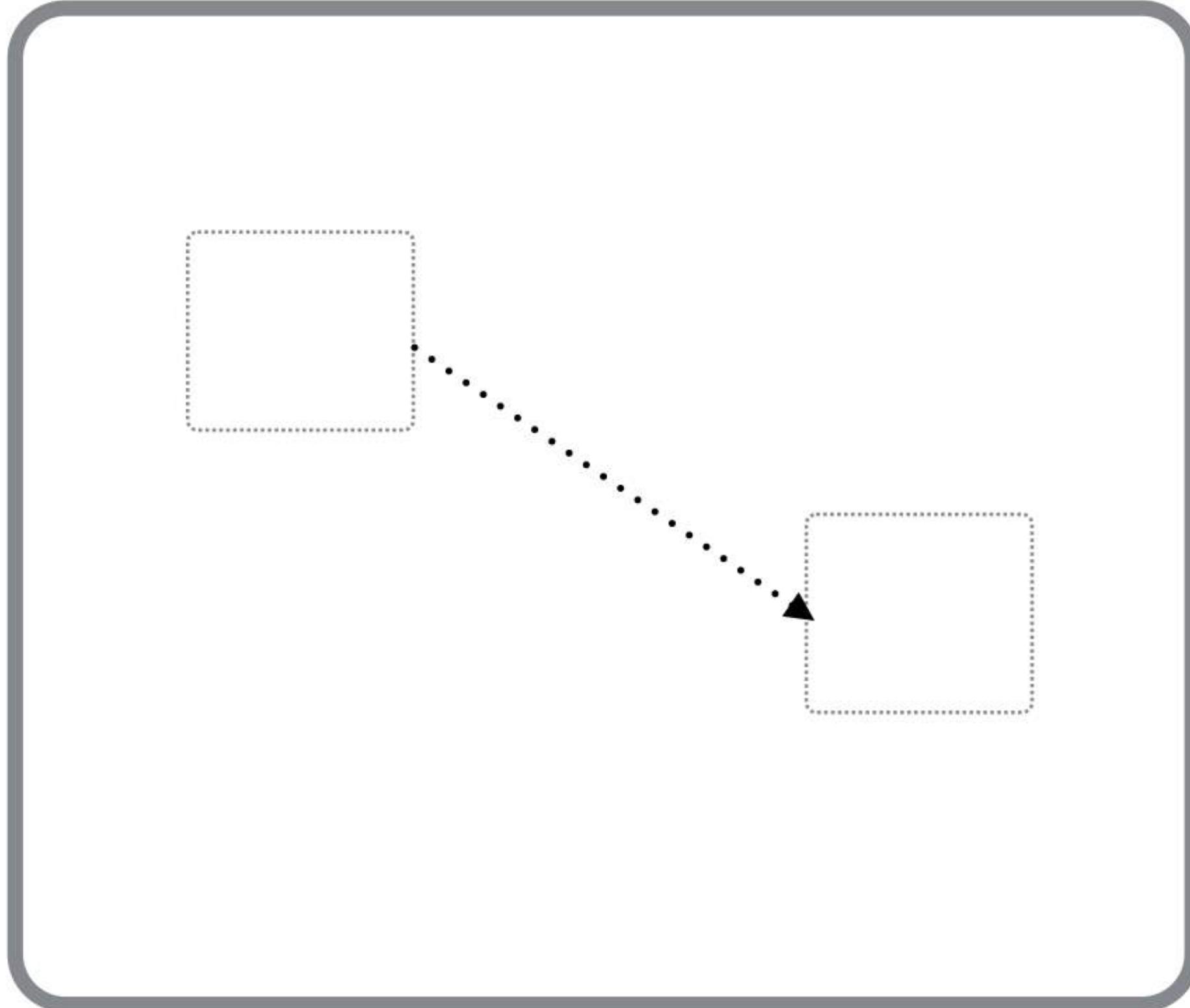






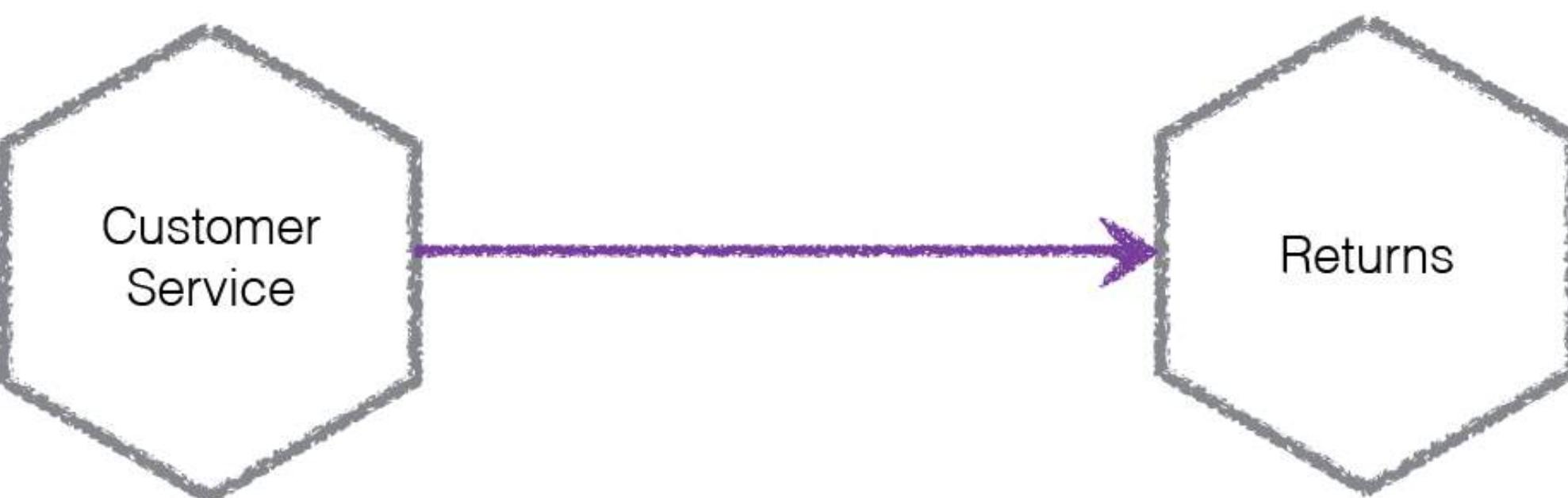


**Are calls between services like
calls inside a process boundary?**

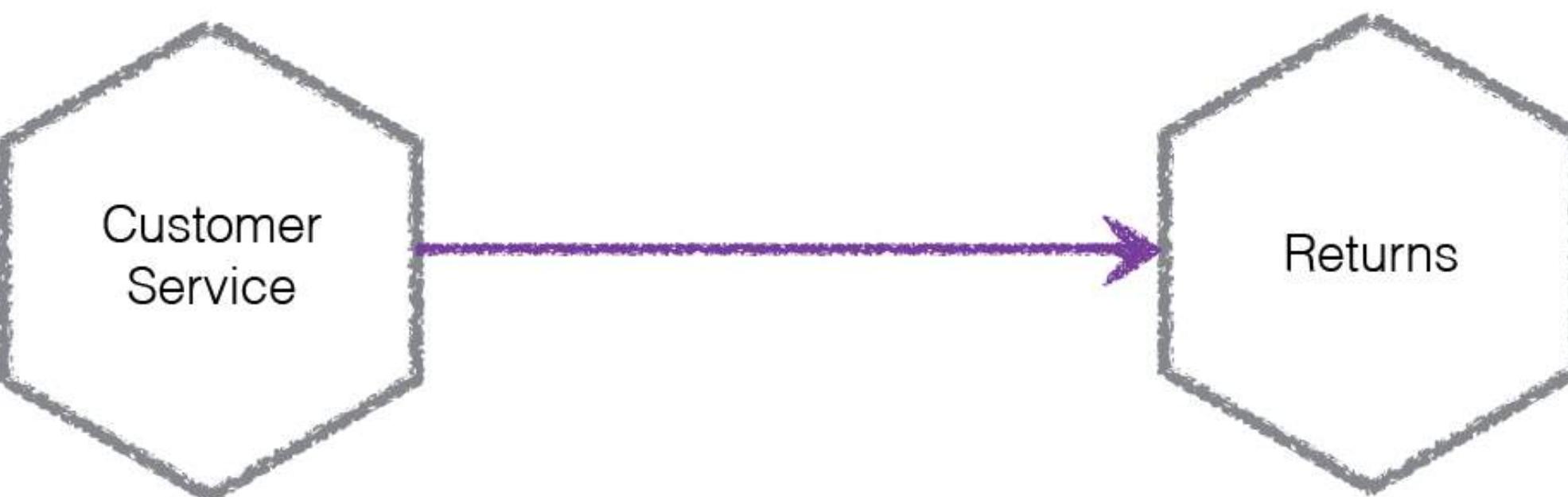


**Cost of change
is low**

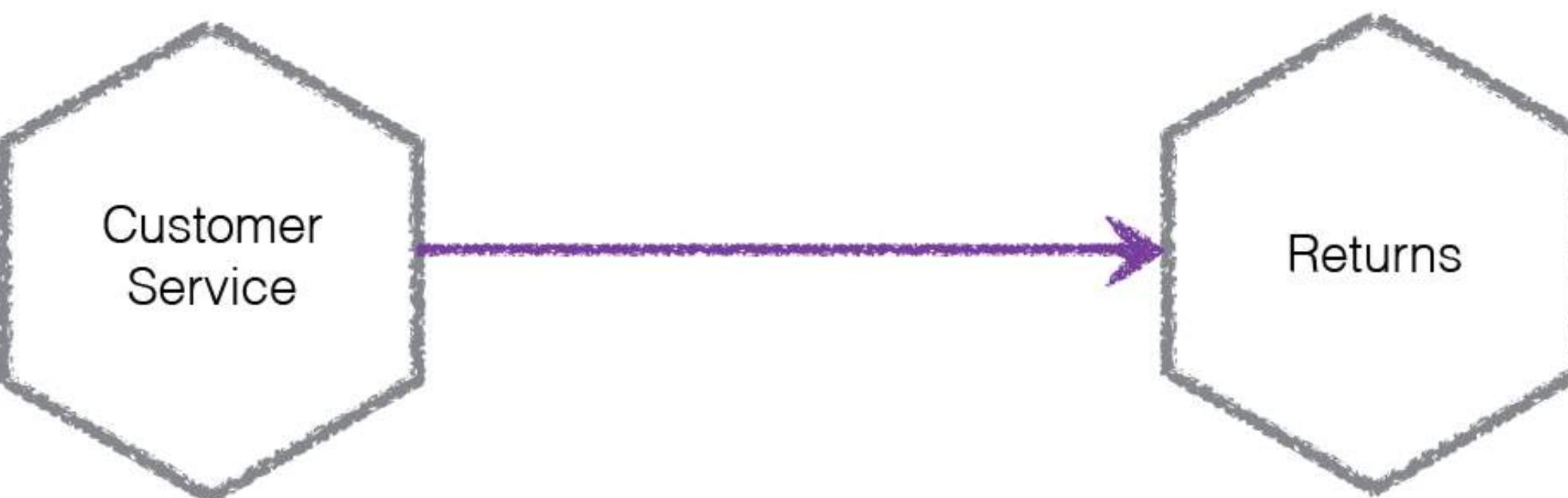
**Easy to reason
about**



Changing a call ?

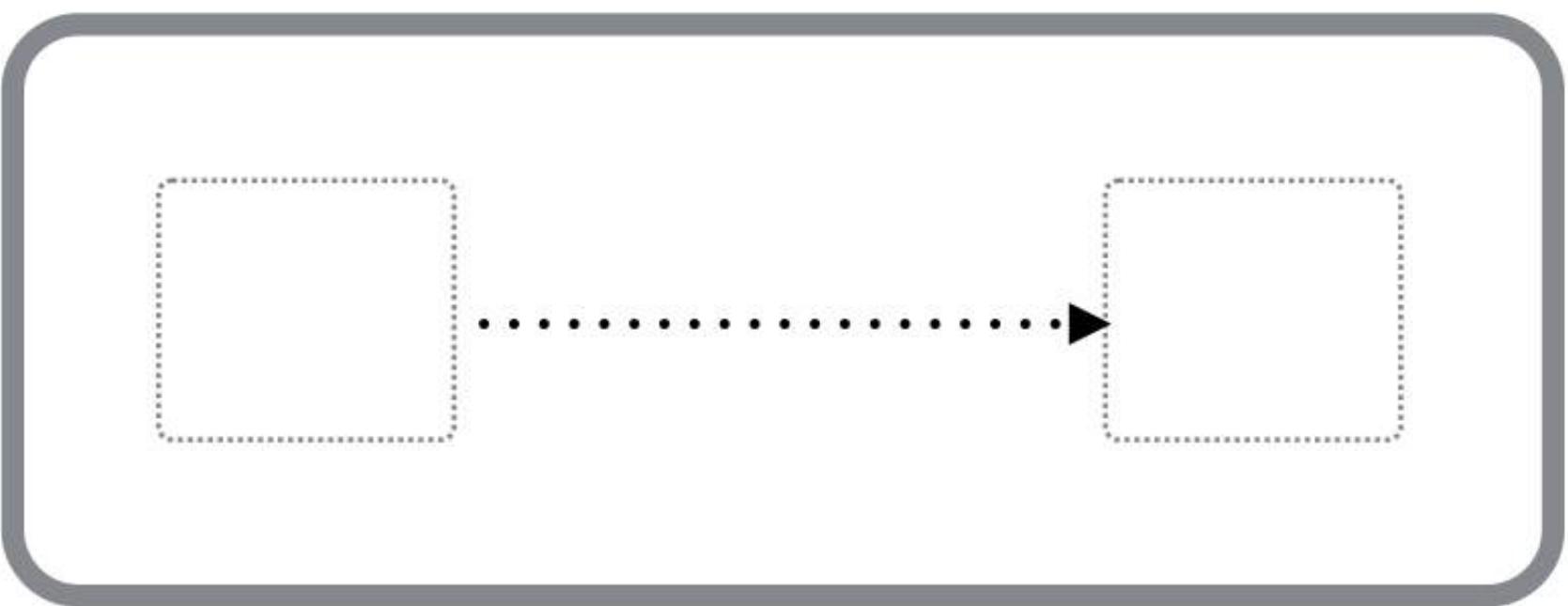


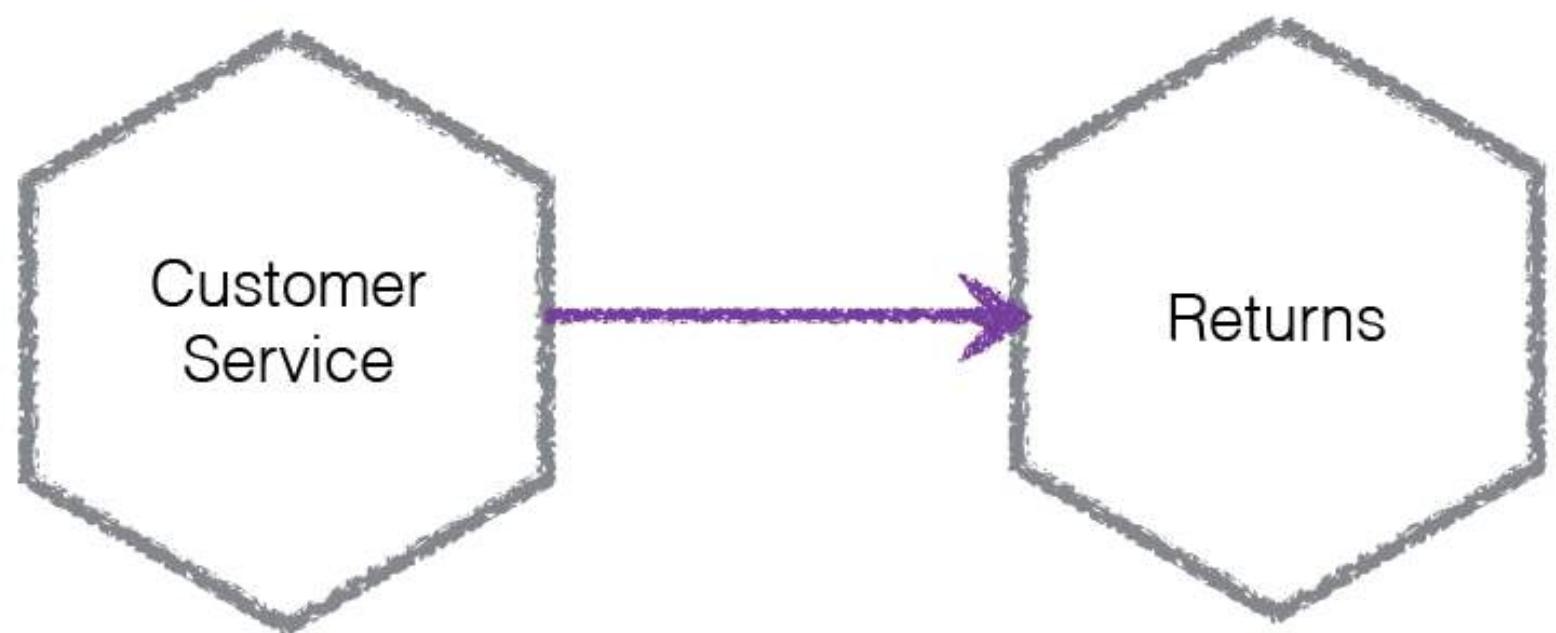
**Changing a call ?
potential API breakage**



**Changing a call ?
potential API breakage
two deployments to rollout a change**

PERFORMANCE IMPLICATIONS

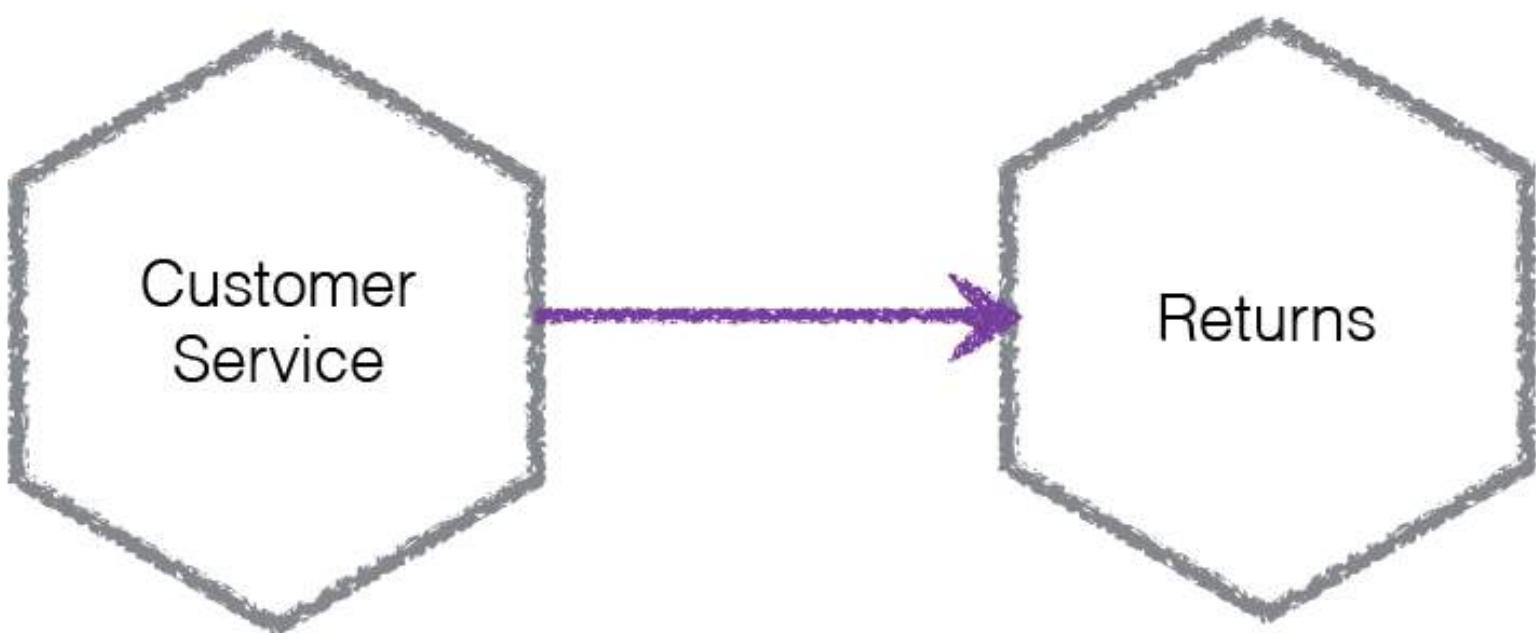




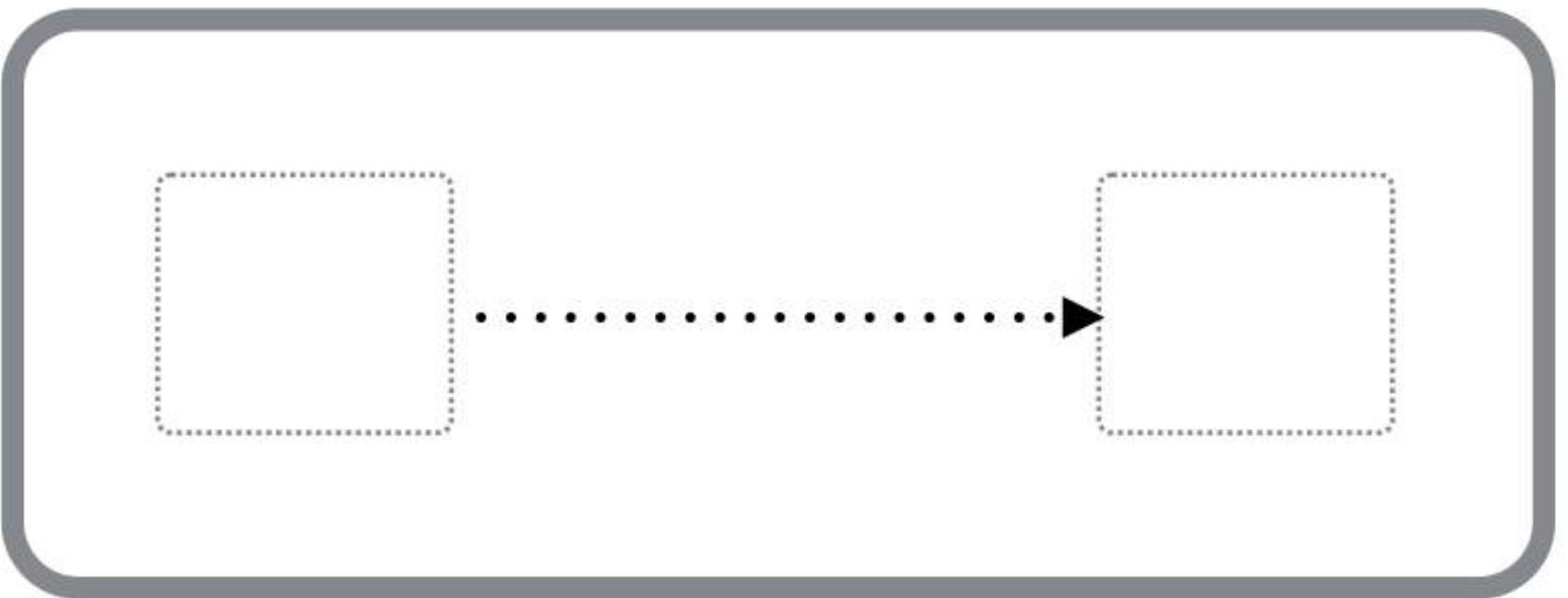
PERFORMANCE IMPLICATIONS



Per-call overhead is very low

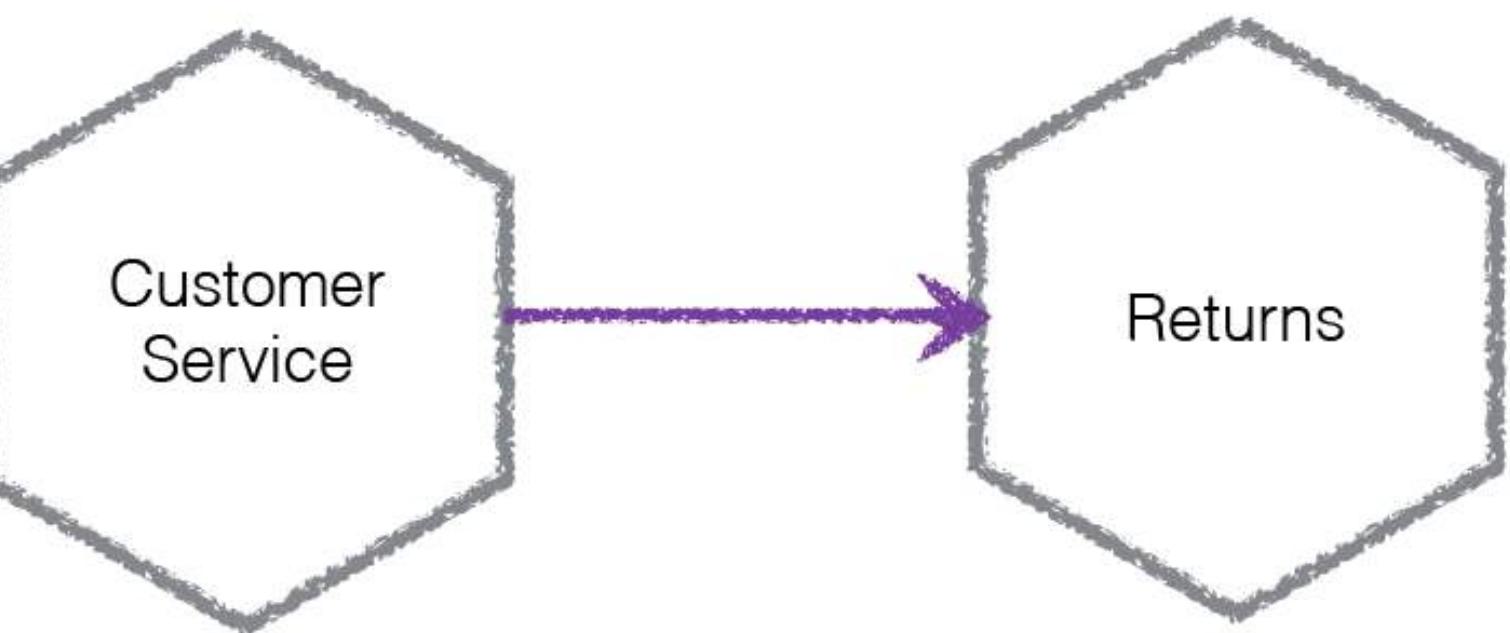


PERFORMANCE IMPLICATIONS



Per-call overhead is very low

Movement of data by reference

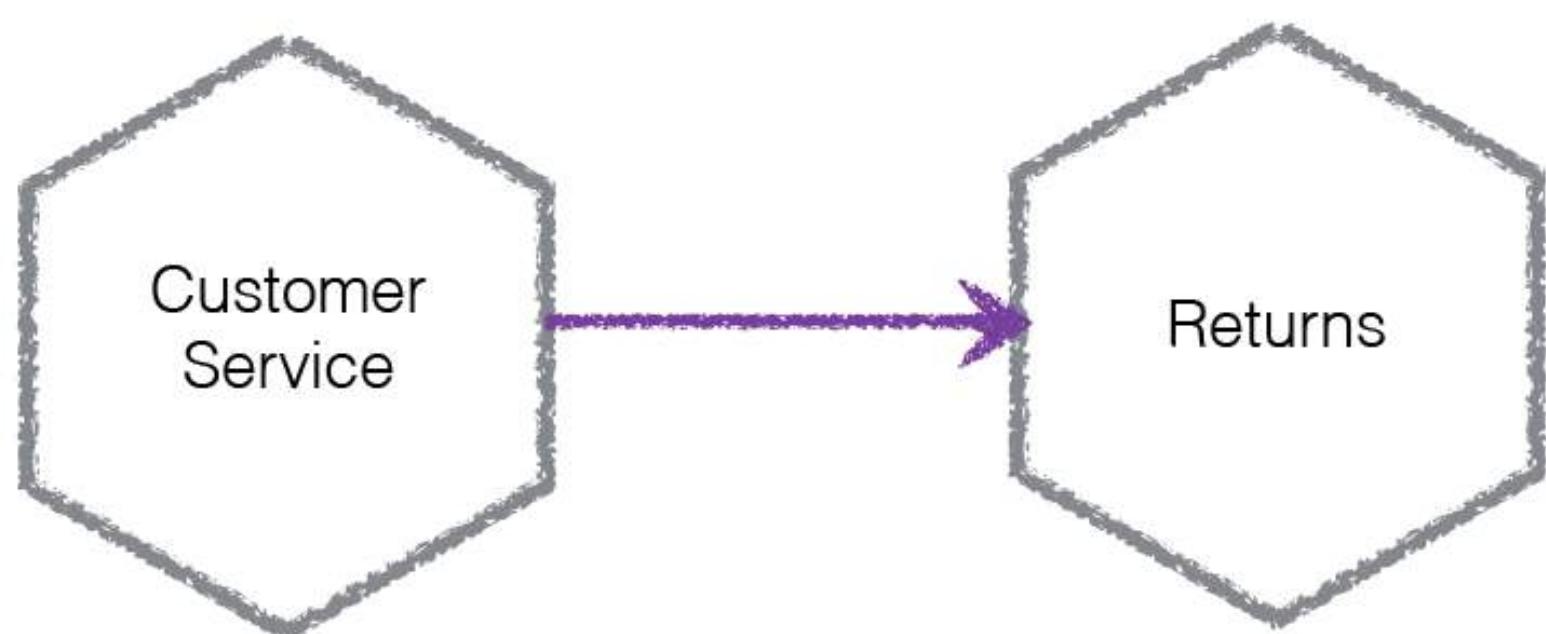


PERFORMANCE IMPLICATIONS



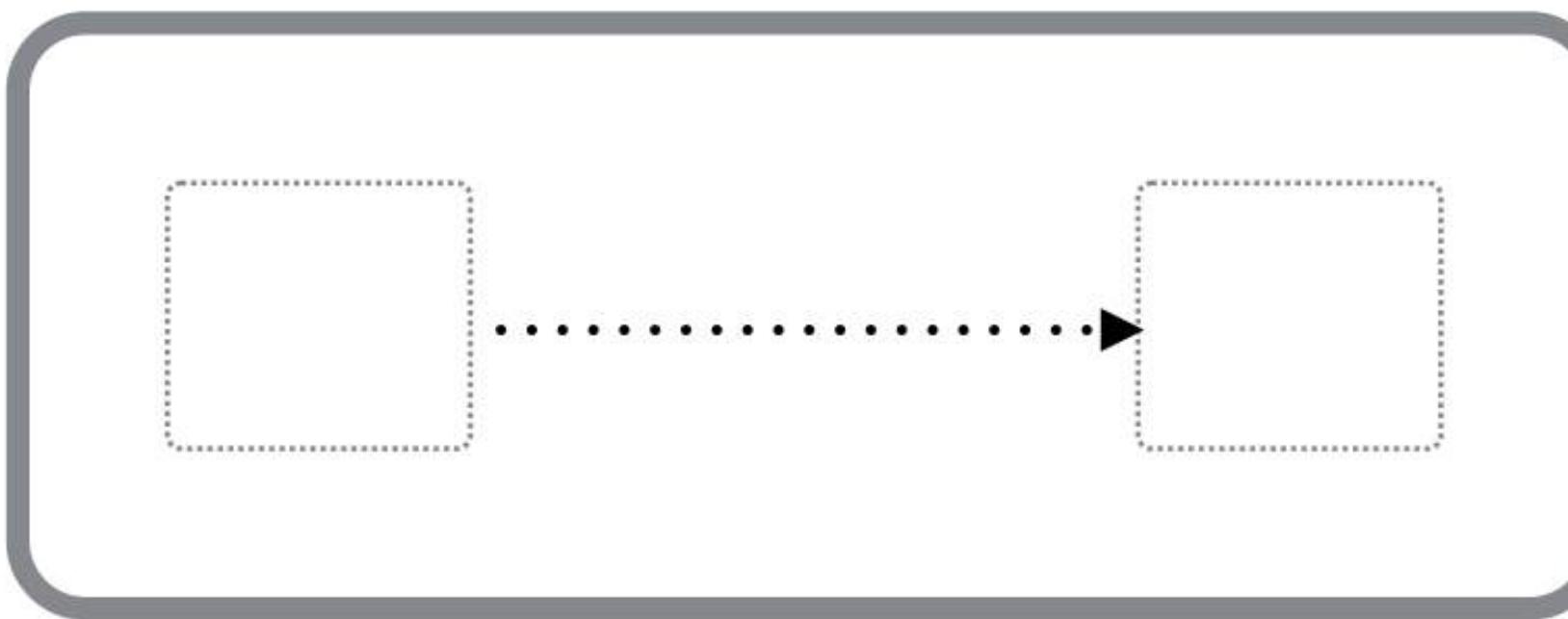
Per-call overhead is very low

Movement of data by reference



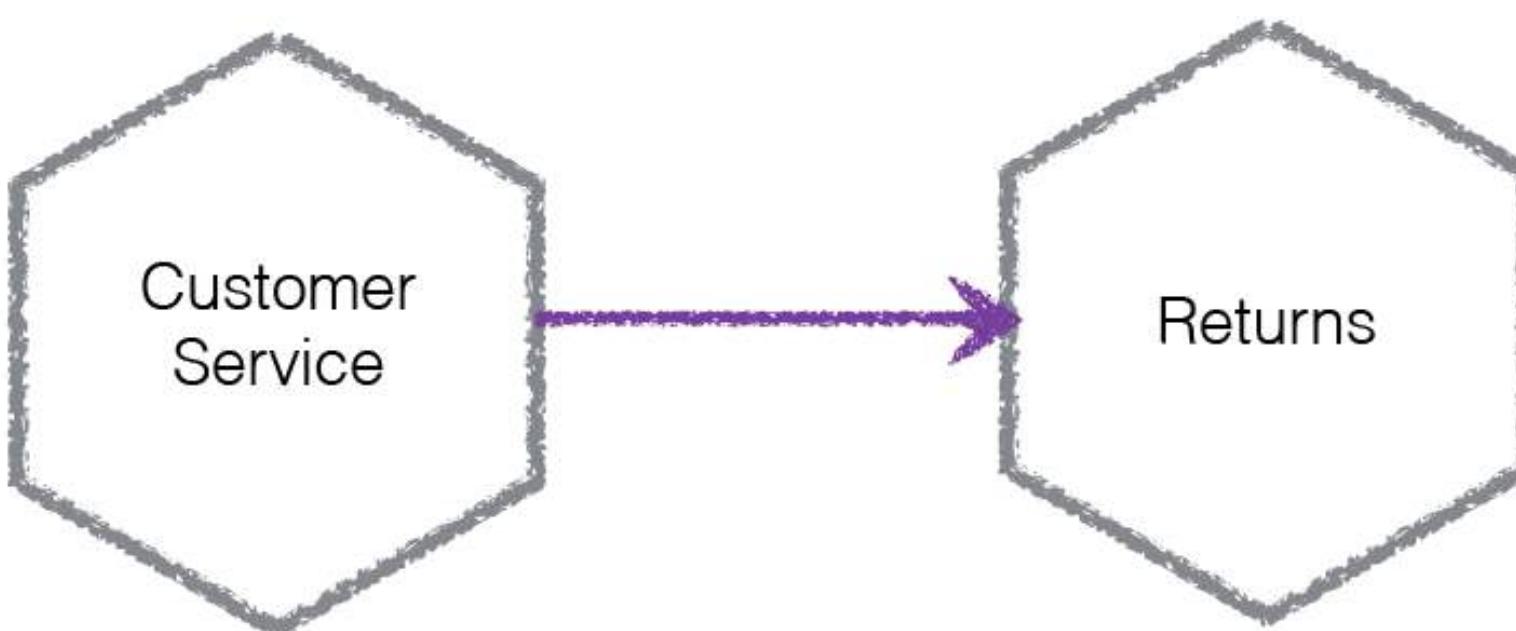
**Call overhead can be
very high**

PERFORMANCE IMPLICATIONS



Per-call overhead is very low

Movement of data by reference



**Call overhead can be
very high**

**Data moved by marshalling, or
handing off to an external store**

CHANGING APIs TO BALANCE PERFORMANCE

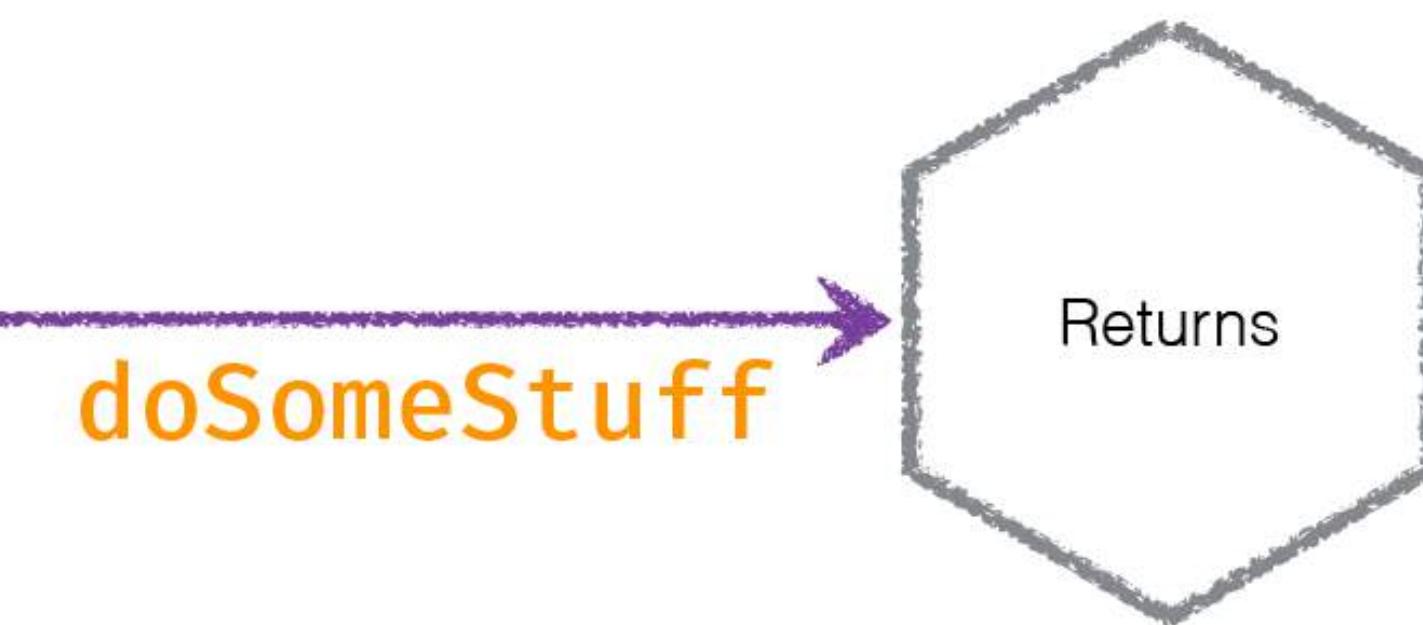
CLIENT

```
for (int i = 0; i < 1000; i++) {  
    doSomeStuff(myBigField);  
}
```

CHANGING APIs TO BALANCE PERFORMANCE

CLIENT

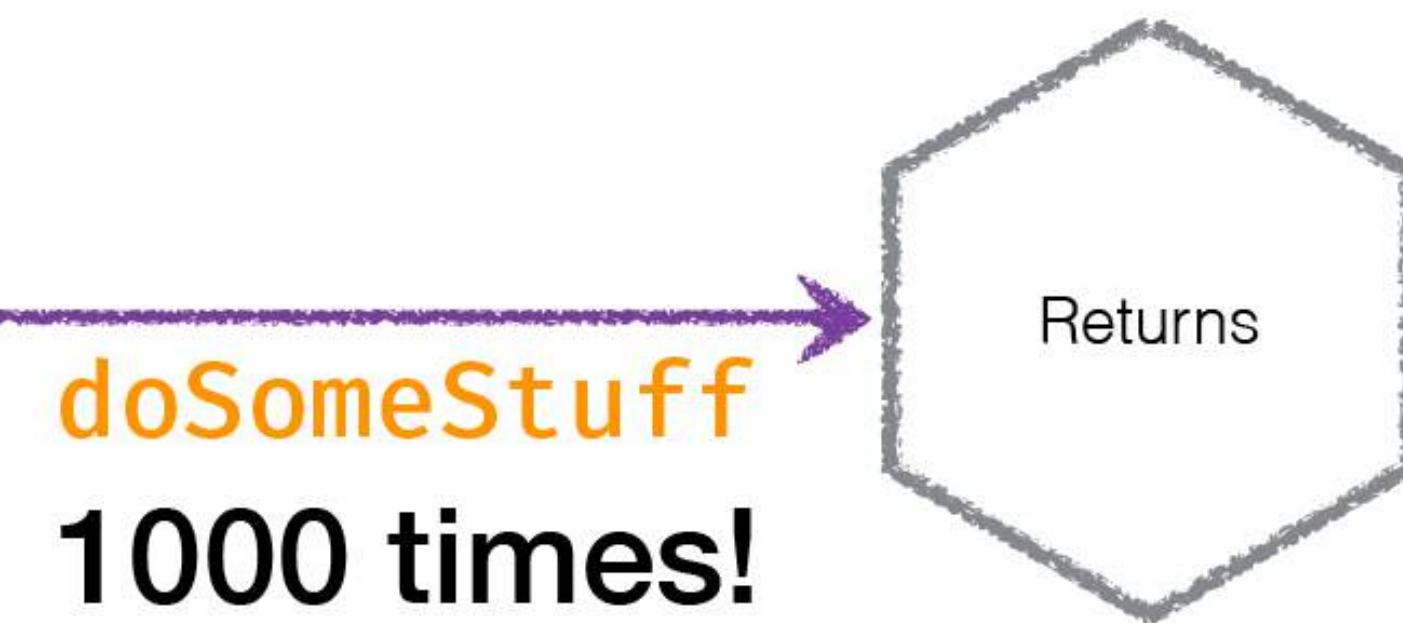
```
for (int i = 0; i < 1000; i++) {  
    doSomeStuff(myBigField);  
}
```



CHANGING APIs TO BALANCE PERFORMANCE

CLIENT

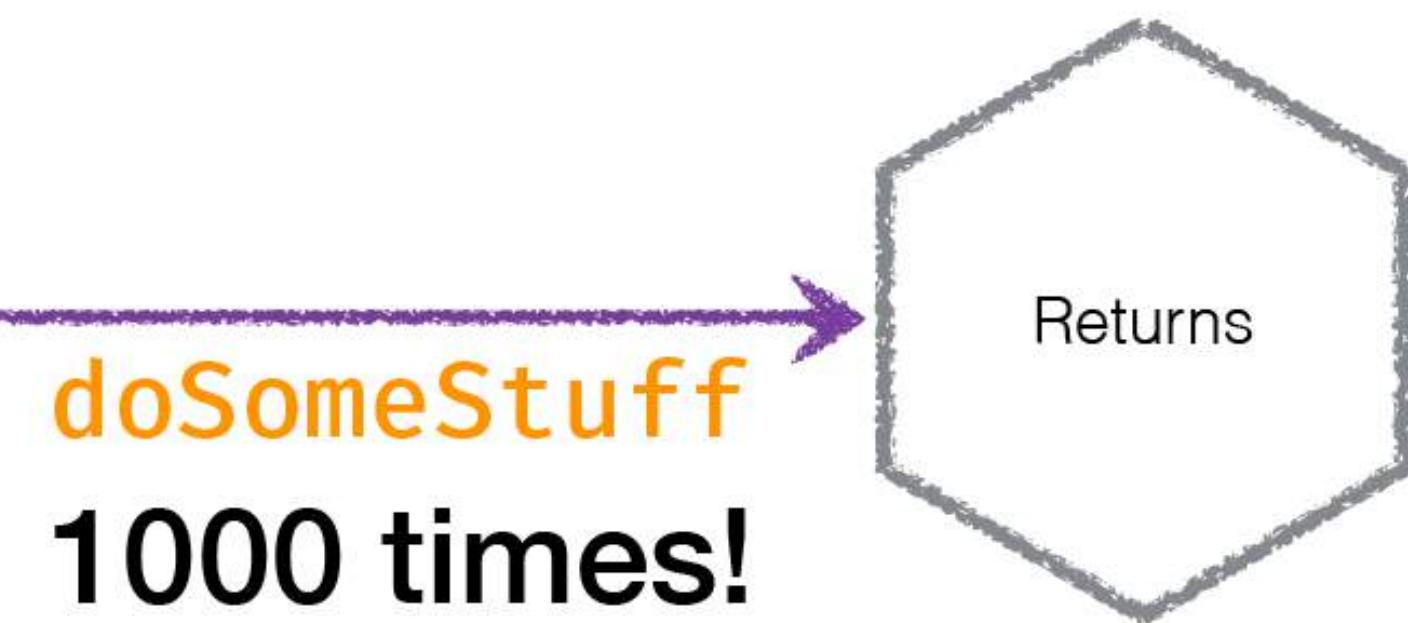
```
for (int i = 0; i < 1000; i++) {  
    doSomeStuff(myBigField);  
}
```



CHANGING APIs TO BALANCE PERFORMANCE

CLIENT

```
for (int i = 0; i < 1000; i++) {  
    doSomeStuff(myBigField);  
}
```



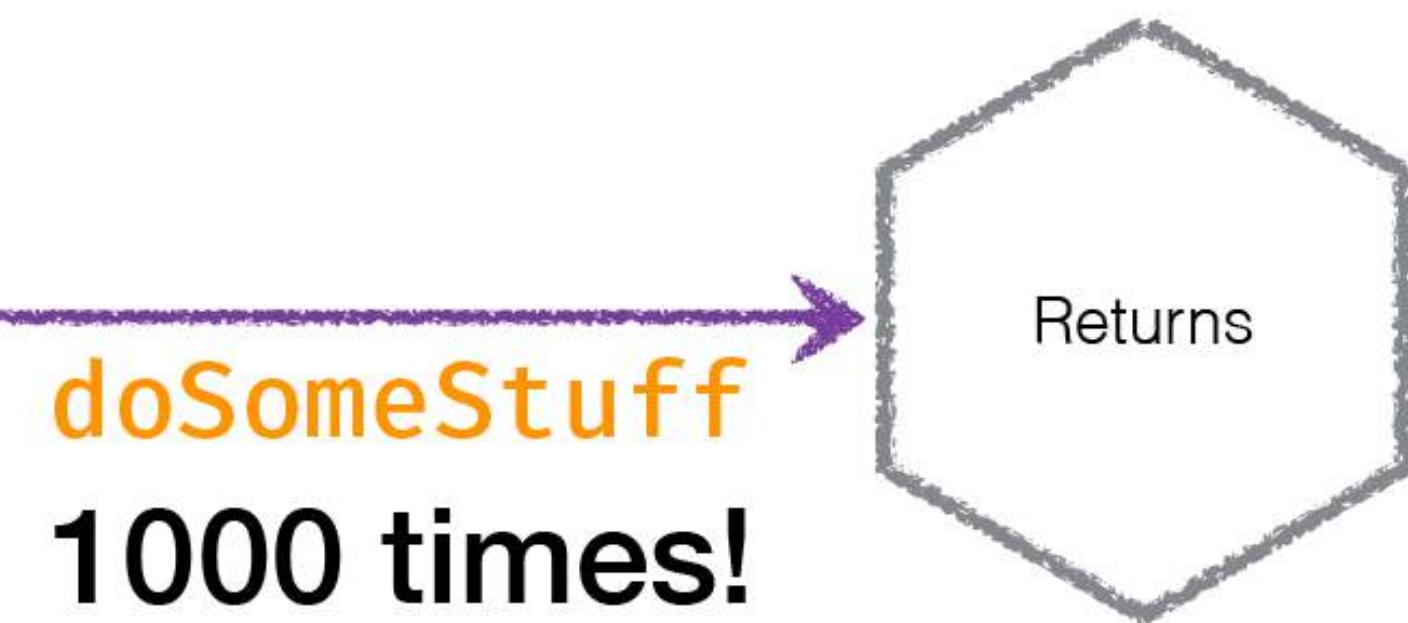
CLIENT

```
doSomeStuff(myBigField);
```

CHANGING APIs TO BALANCE PERFORMANCE

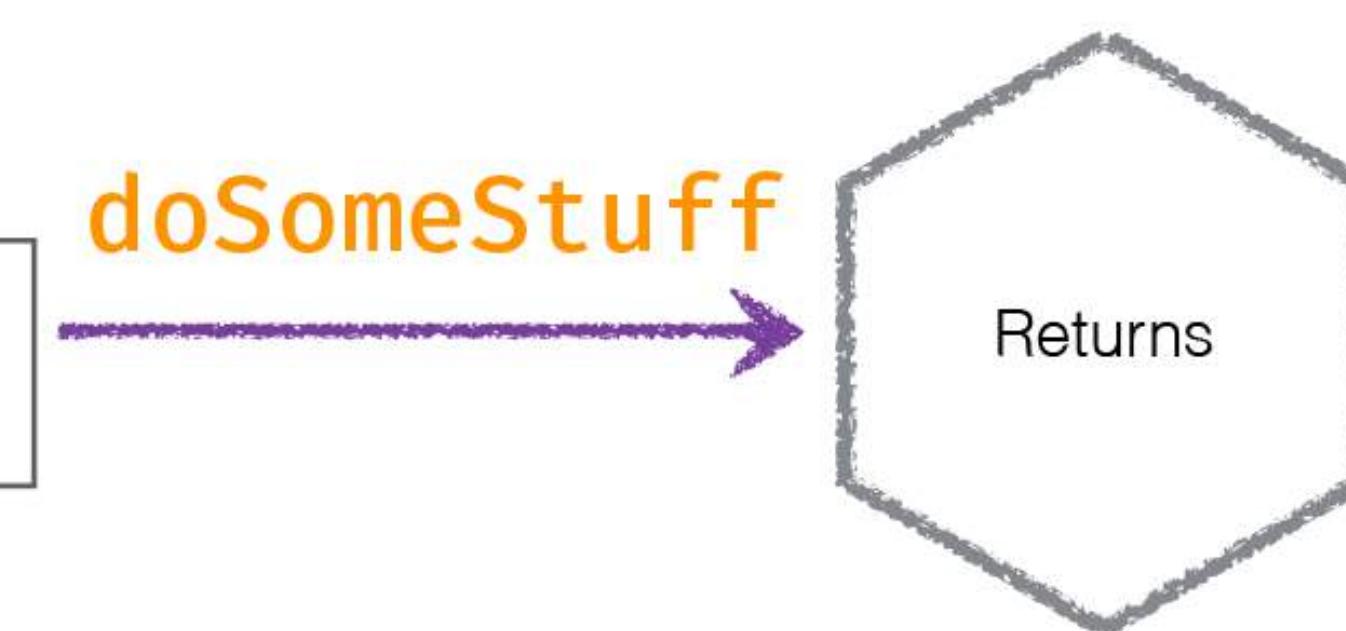
CLIENT

```
for (int i = 0; i < 1000; i++) {  
    doSomeStuff(myBigField);  
}
```



CLIENT

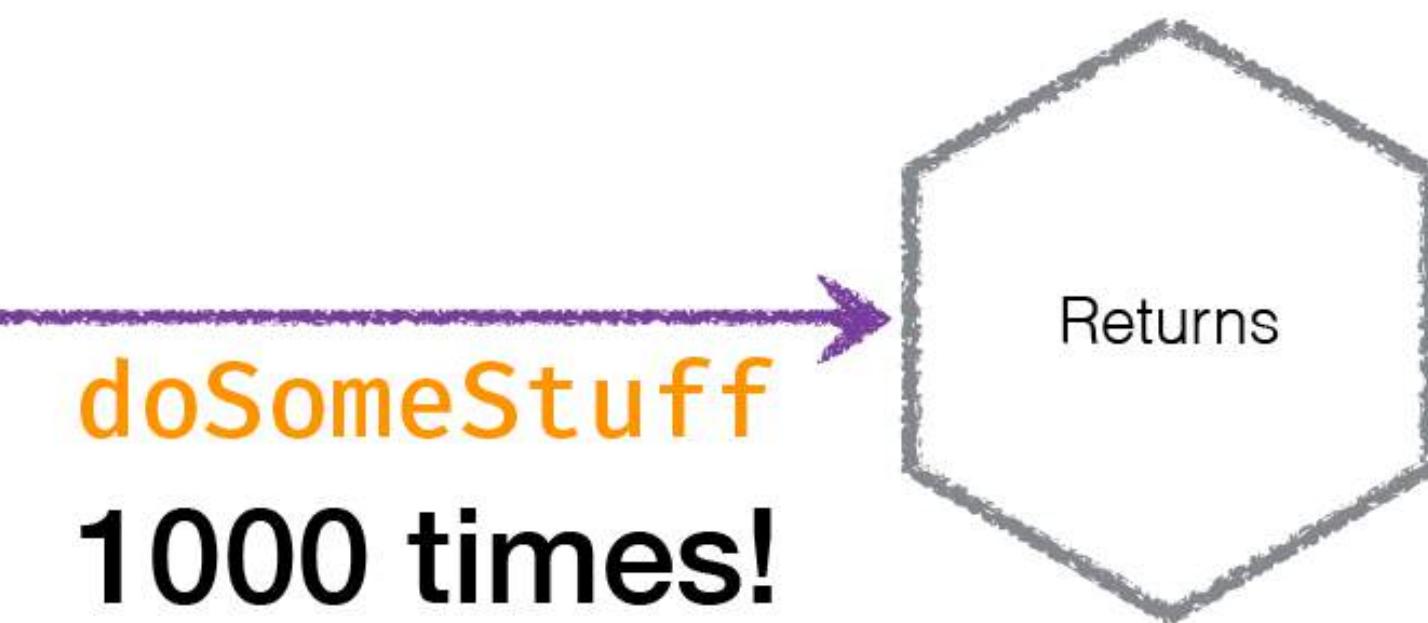
```
doSomeStuff(myBigField);
```



CHANGING APIs TO BALANCE PERFORMANCE

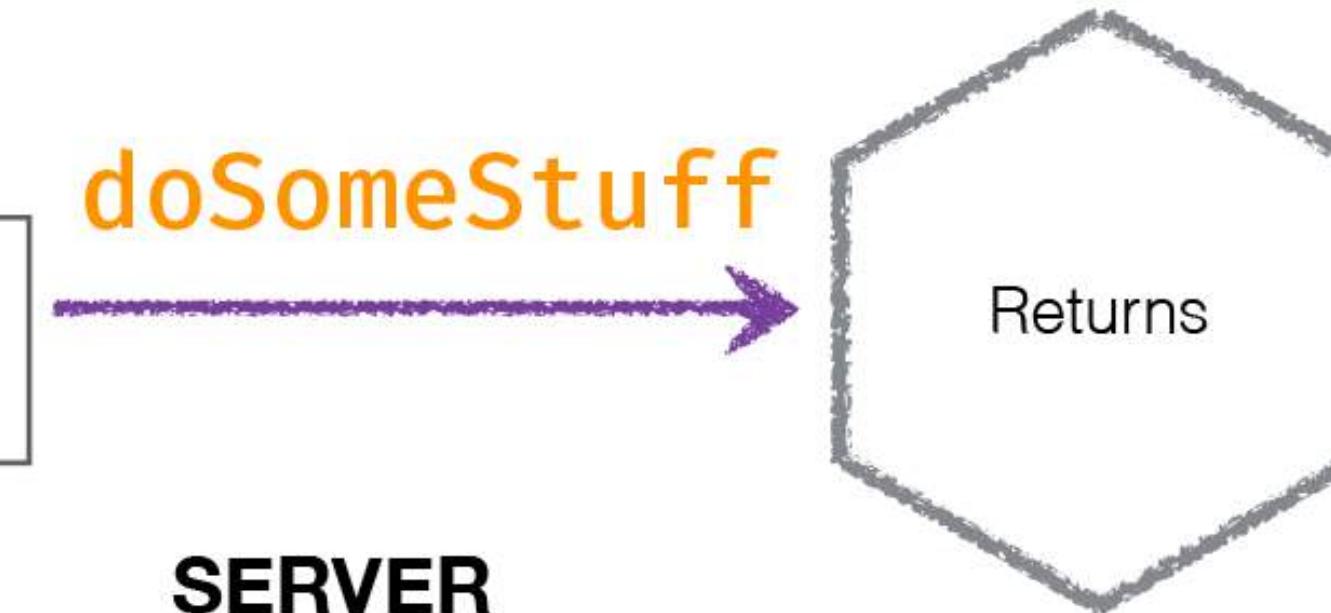
CLIENT

```
for (int i = 0; i < 1000; i++) {  
    doSomeStuff(myBigField);  
}
```



CLIENT

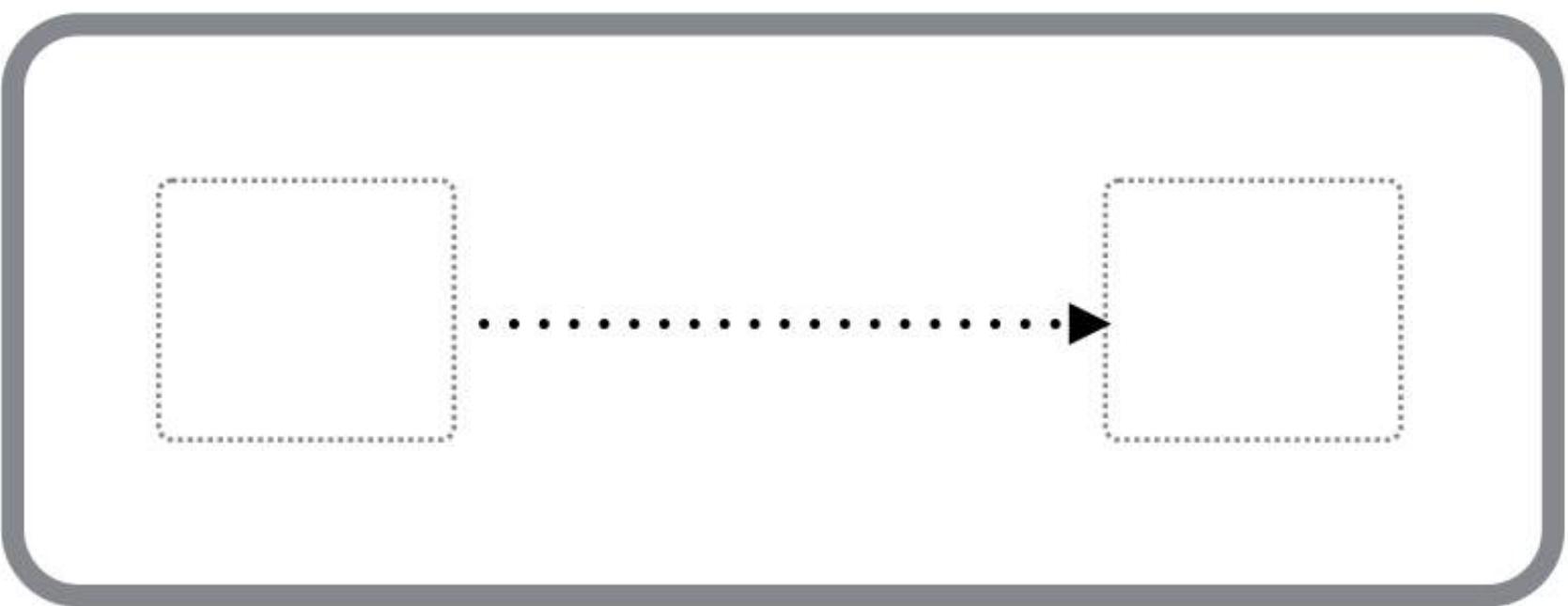
```
doSomeStuff(myBigField);
```

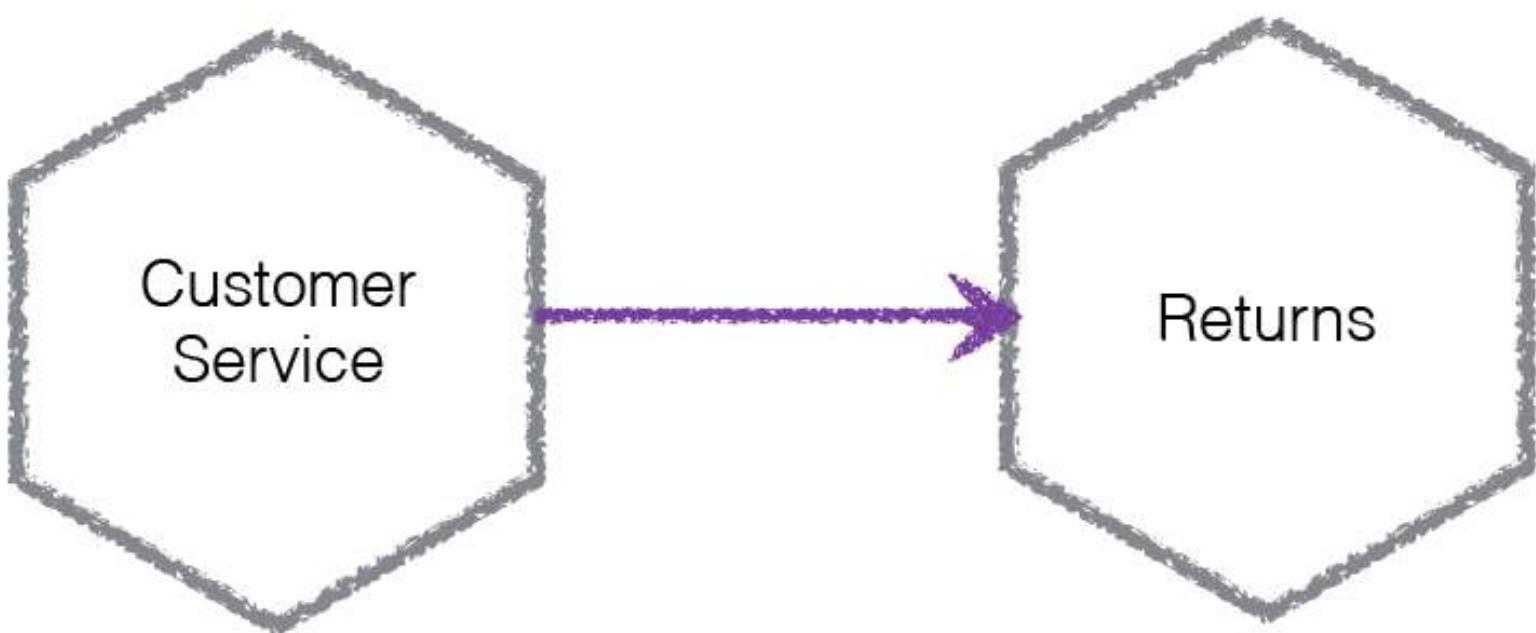


SERVER

```
for (int i = 0; i < 1000; i++) {  
    doSomeStuff(myBigField);  
}
```

HANDLING ERRORS

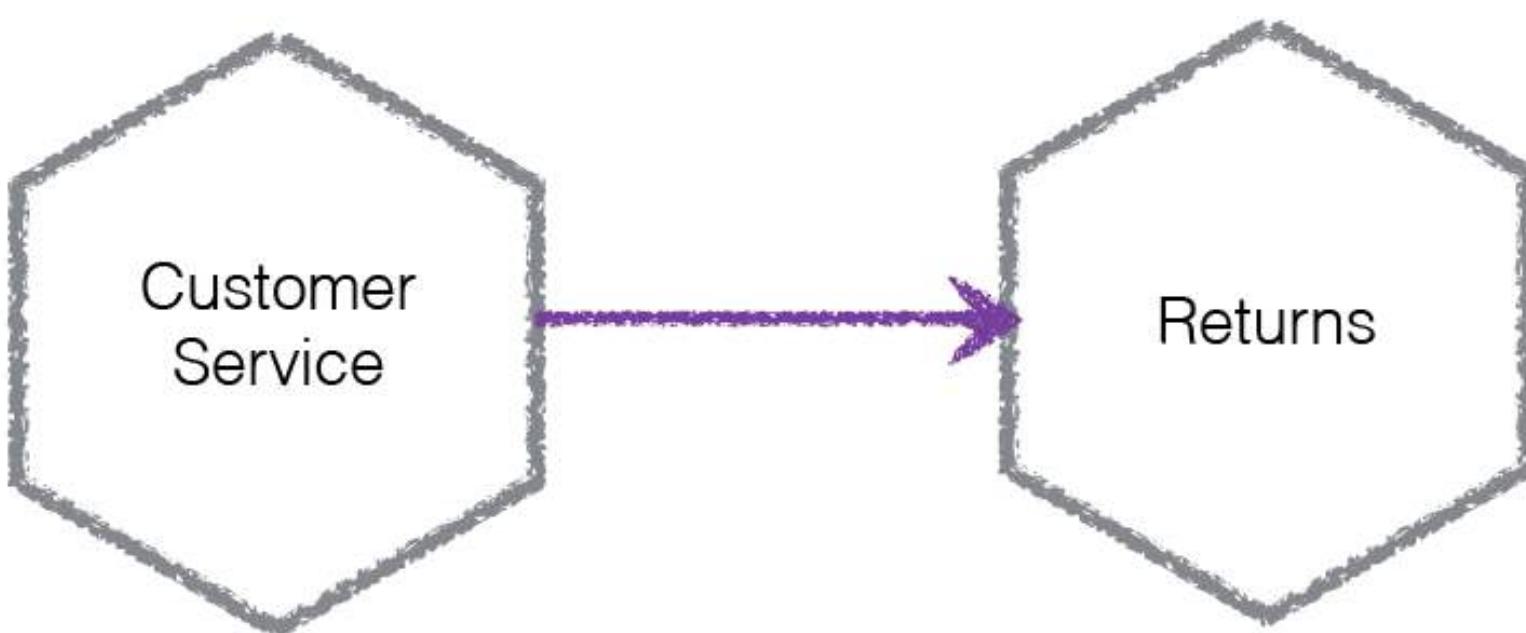




HANDLING ERRORS



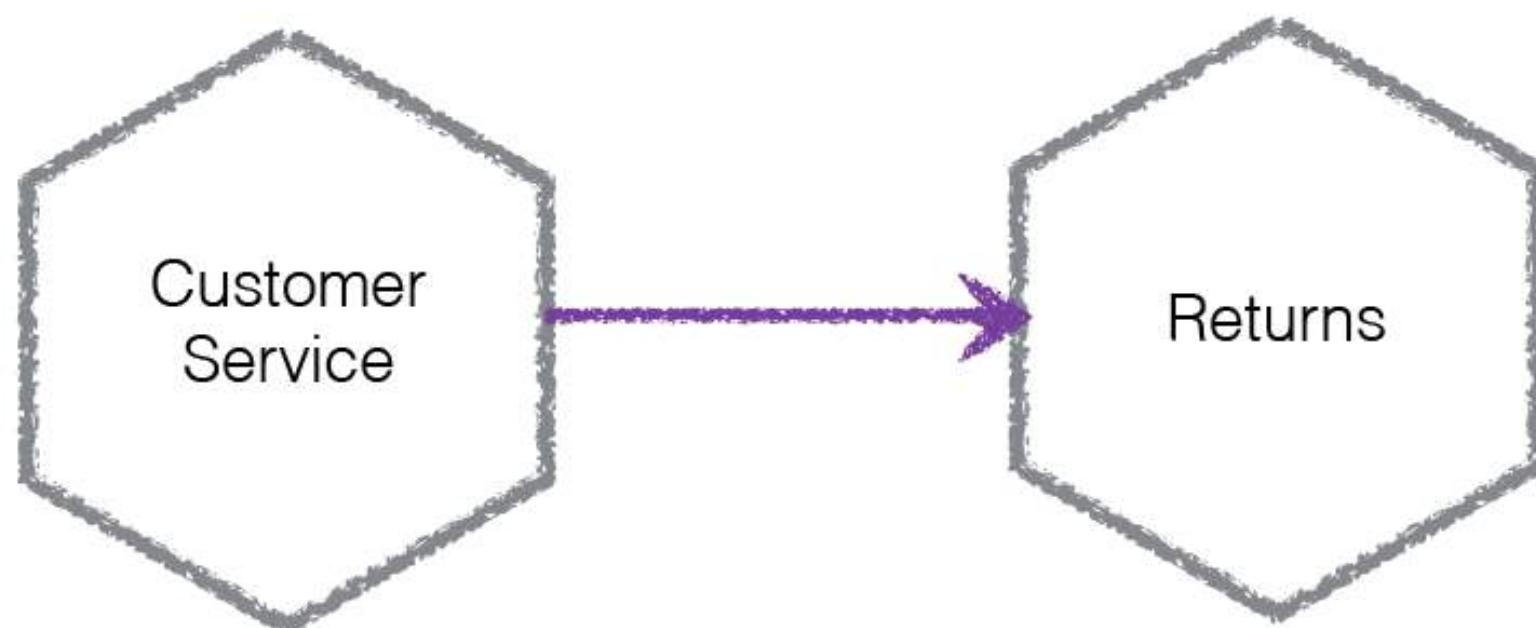
Errors straightforward



HANDLING ERRORS



Errors straightforward

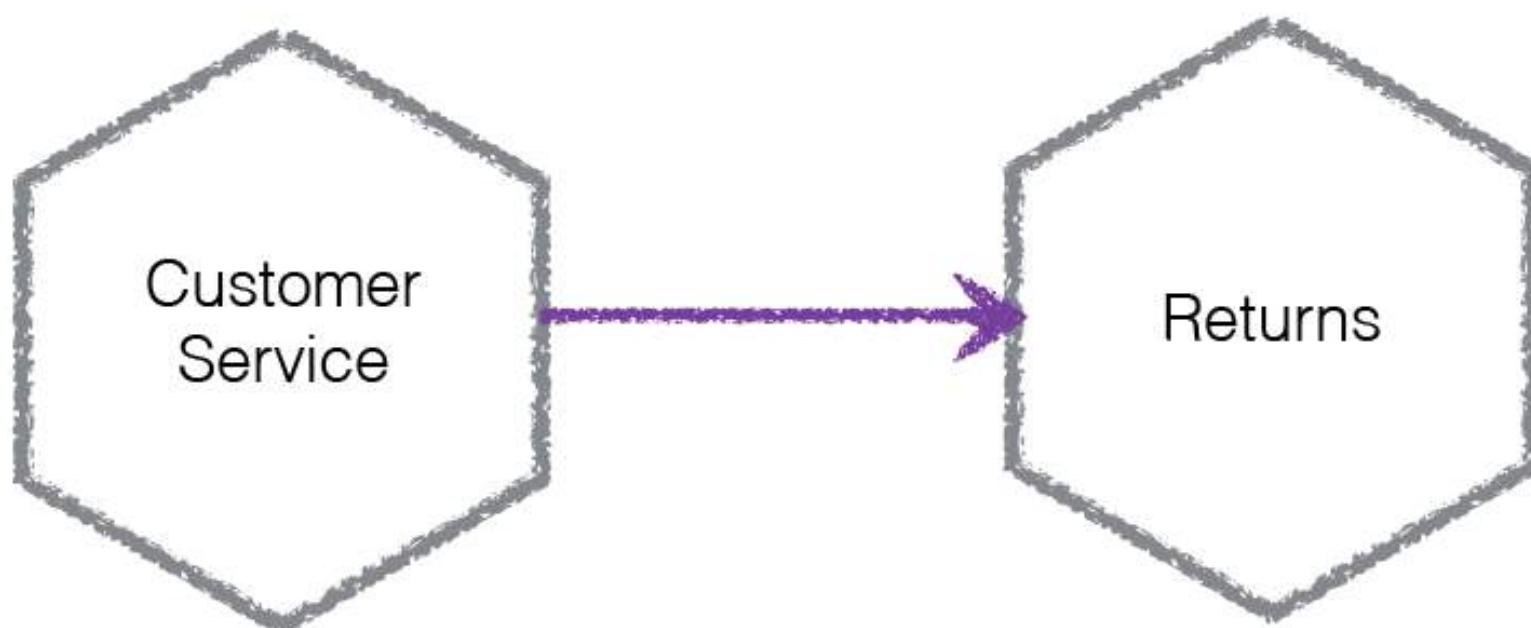


Timeouts

HANDLING ERRORS



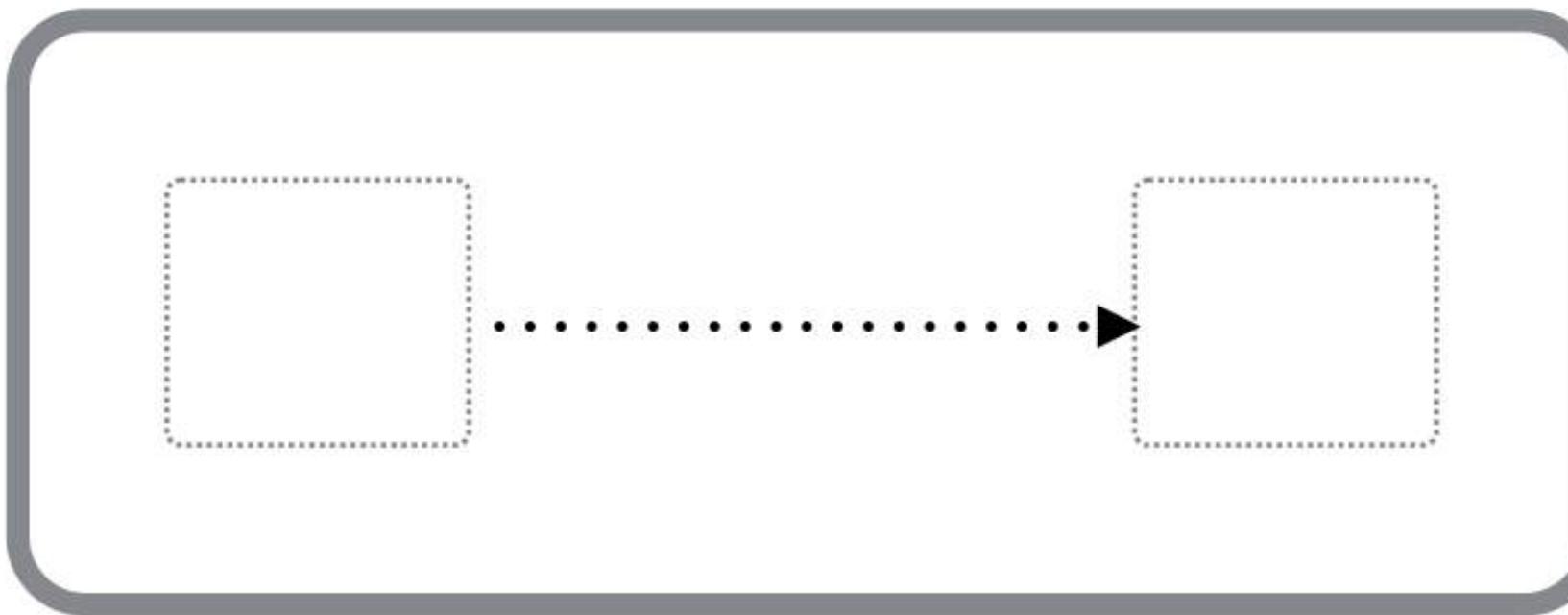
Errors straightforward



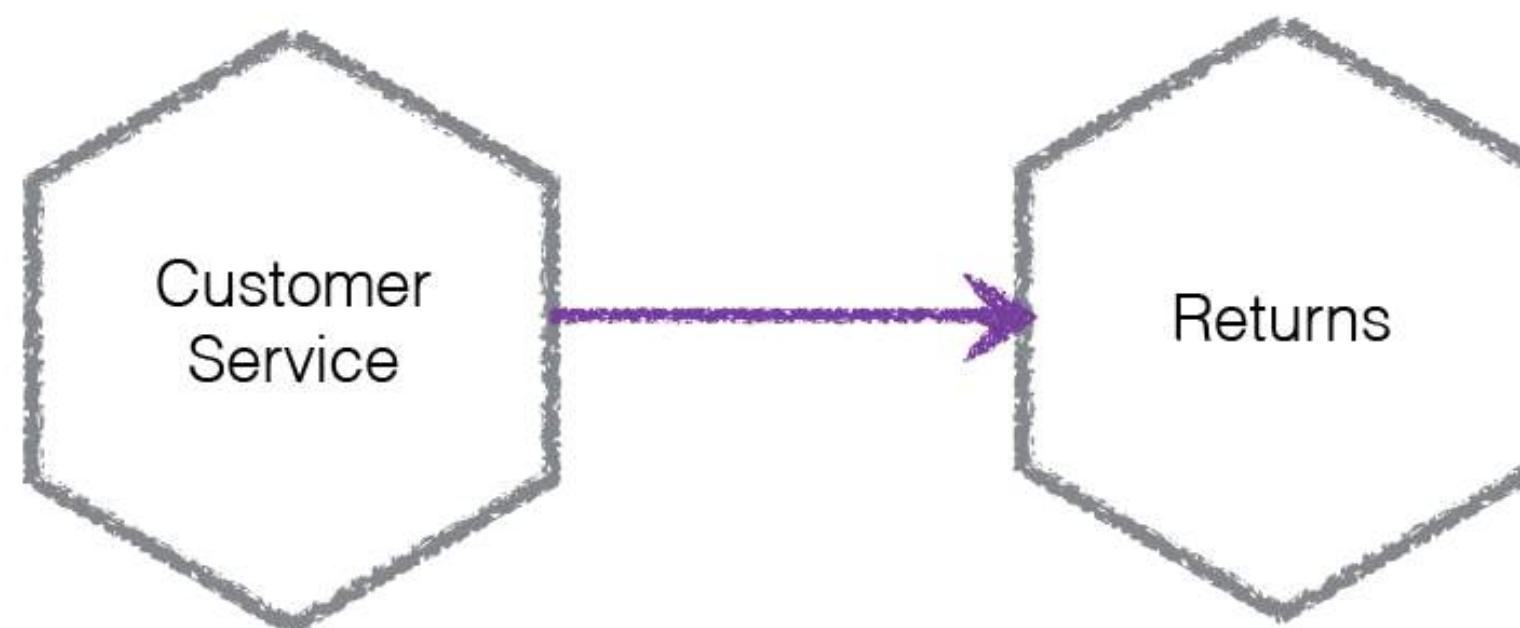
Timeouts

Downstream outage

HANDLING ERRORS



Errors straightforward

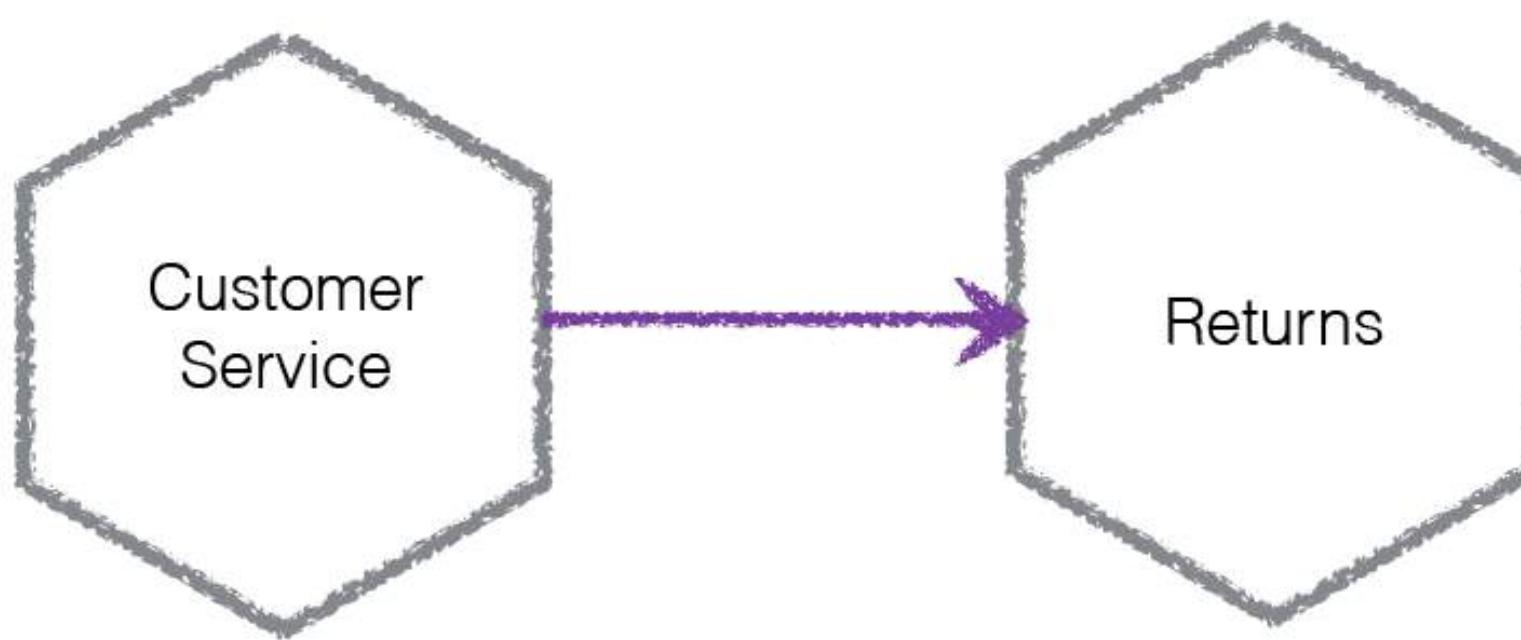


Timeouts

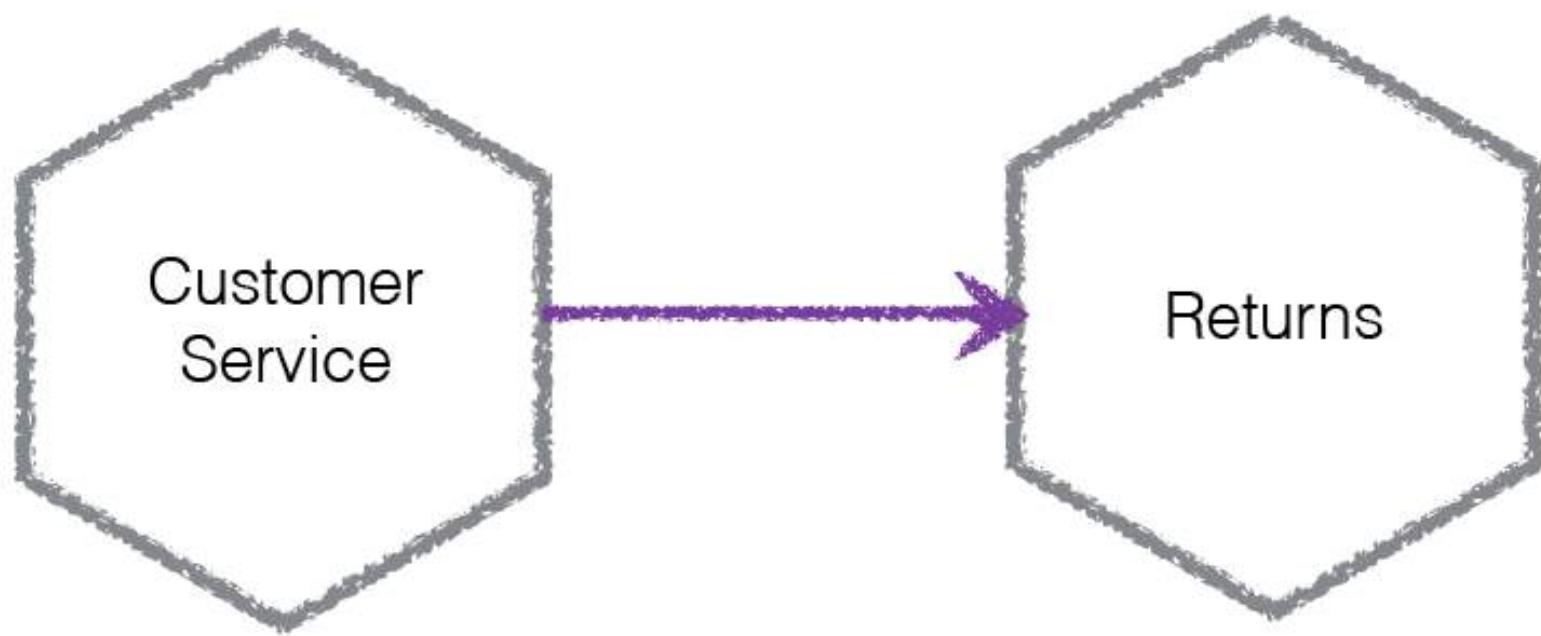
Downstream outage

**Difference between client
and server errors**

4XX vs 5XX status codes

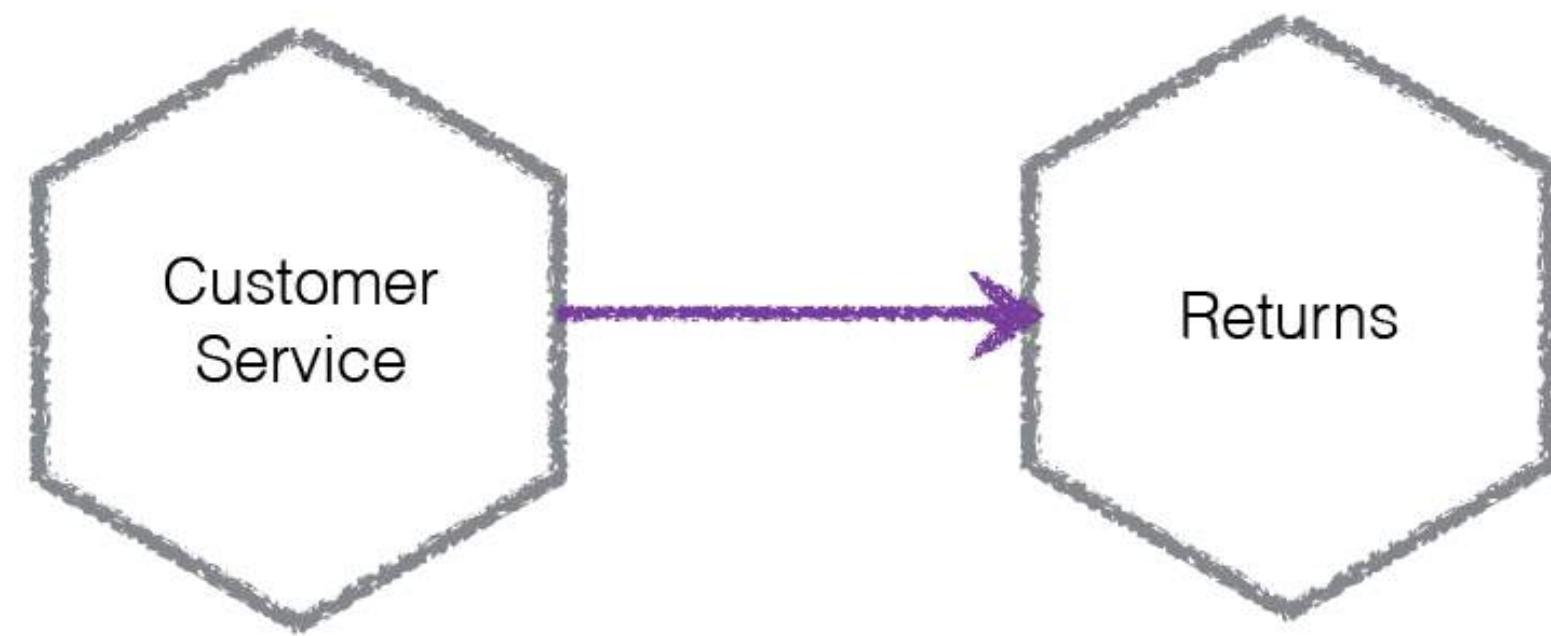


4XX vs 5XX status codes



4XX - you did something wrong!

4XX vs 5XX status codes



4XX - you did something wrong!

5XX - there is something wrong at my end...

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

4xx Client Error <small>[edit]</small>	
The 4xx class of status code is intended for situations in which the client seems to have erred. Except when responding to a HEAD request, the server should include a detailed explanation of the error in the response body.	
400 Bad Request	The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, too large size, invalid request message).
401 Unauthorized (RFC 7235-1)	Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include the necessary credentials.
402 Payment Required	Note: Some sites issue HTTP 401 when an IP address is banned from the website (usually the website domain) and that specific address is refused permission to access the site.
403 Forbidden	Reserved for future use. The original intention was that this code might be used as part of some form of digital cash or micropayment scheme, but that has not happened.
404 Not Found	The request was a valid request, but the server is refusing to respond to it. The user might be logged in but does not have the necessary permissions for the requested resource.
405 Method Not Allowed	The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.
406 Not Acceptable	A request method is not supported for the requested resource; for example, a GET request on a form which requires data to be presented via POST, or a PUT request on a file.
407 Proxy Authentication Required (RFC 7235-1)	The requested resource is capable of generating only content not acceptable according to the Accept headers sent in the request. See Content negotiation.
408 Request Timeout	The client must first authenticate itself with the proxy.
409 Conflict	The server timed out waiting for the request. According to HTTP specifications: "The client did not produce a request within the time that the server was prepared to accept it."
410 Gone	Indicates that the request could not be processed because of conflict in the request, such as an edit conflict between multiple simultaneous updates.
411 Length Required	Indicates that the resource requested is no longer available and will not be available again. This should be used when a resource has been intentionally removed and purged, and a "404 Not Found" may be used instead.
412 Precondition Failed (RFC 7232-1)	The request did not specify the length of its content, which is required by the requested resource.
413 Payload Too Large (RFC 7231-1)	The server does not meet one of the preconditions that the requester put on the request.
414 URI Too Long (RFC 7231-1)	The request is larger than the server is willing or able to process. Previously called "Request Entity Too Large".
415 Unsupported Media Type	The URI provided was too long for the server to process. Often the result of too much data being encoded as a query-string of a GET request, in which case the server is unable to parse the data.
416 Range Not Satisfiable (RFC 7232-1)	The request entity has a media type which the server or resource does not support. For example, the client uploads an image as image/svg+xml, but the server only understands image/png and image/jpeg.
417 Expectation Failed	The client has asked for a portion of the file (byte serving), but the server cannot supply that portion. For example, if the client asked for a part of the file that the server does not have.
418 I'm a teapot (RFC 2324)	The server cannot meet the requirements of the Expect request-header field.
421 Misdirected Request (RFC 7540-1)	This code was defined in 1998 as one of the traditional IETF April Fools' jokes, in RFC 2324's, Hyper Text Coffee Pot Control Protocol, and is not expected to be used.
422 Unprocessable Entity (WebDAV; RFC 4918)	The request was well-formed but was unable to be followed due to semantic errors.
423 Locked (WebDAV; RFC 4918)	The resource that is being accessed is locked.
424 Failed Dependency (WebDAV; RFC 4918)	The request failed due to failure of a previous request (e.g., a PROPPATCH).
426 Upgrade Required	The client should switch to a different protocol such as TLS/1.0, given in the Upgrade header field.
428 Precondition Required (RFC 6585-1)	The origin server requires the request to be conditional. Intended to prevent the 'lost update' problem, where a client GETs a resource's state, modifies it, and then overwrites it.
429 Too Many Requests (RFC 6585-1)	The user has sent too many requests in a given amount of time. Intended for use with rate-limiting schemes.
431 Request Header Fields Too Large (RFC 6585-1)	The server is unwilling to process the request because either an individual header field, or all the header fields collectively, are too large.
451 Unavailable For Legal Reasons	A server operator has received a legal demand to deny access to a resource or to a set of resources that includes the requested resource.

28 4XX Error Codes

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

4xx Client Error [edit]

The 4xx class of status code is intended for situations in which the client seems to have erred. Except when responding to a HEAD request, the server should include a detailed explanation of the error in the response body.

400 Bad Request
The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, too large size, invalid request message).
401 Unauthorized (RFC 7235)
Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include the necessary credentials.
Note: Some sites issue HTTP 401 when an IP address is banned from the website (usually the website domain) and that specific address is refused permission to access the site.
402 Payment Required
Reserved for future use. The original intention was that this code might be used as part of some form of digital cash or micropayment scheme, but that has not happened.
403 Forbidden
The request was a valid request, but the server is refusing to respond to it. The user might be logged in but does not have the necessary permissions for the requested action.
404 Not Found
The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.
405 Method Not Allowed
A request method is not supported for the requested resource; for example, a GET request on a form which requires data to be presented via POST, or a PUT request on a resource which only supports GET and POST.
406 Not Acceptable
The requested resource is capable of generating only content not acceptable according to the Accept headers sent in the request.^[38] See Content negotiation.
407 Proxy Authentication Required (RFC 7235)
The client must first authenticate itself with the proxy.^[39]

408 Request Timeout
The server timed out waiting for the request. According to HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait." This is different from 404 Not Found.
409 Conflict
Indicates that the request could not be processed because of conflict in the request, such as an edit conflict between multiple simultaneous updates.
410 Gone
Indicates that the resource requested is no longer available and will not be available again. This should be used when a resource has been intentionally removed and purged. The client should not purge the resource, and a "404 Not Found" may be used instead.
411 Length Required
The request did not specify the length of its content, which is required by the requested resource.^[40]
412 Precondition Failed (RFC 7232)
The server does not meet one of the preconditions that the requester put on the request.^[41]

413 Payload Too Large (RFC 7231)
The request is larger than the server is willing or able to process. Previously called "Request Entity Too Large".^[42]
414 URI Too Long (RFC 7231)
The URI provided was too long for the server to process. Often the result of too much data being encoded as a query-string of a GET request, in which case the server can return a 414 error.
415 Unsupported Media Type
The request entity has a media type which the server or resource does not support. For example, the client uploads an image as image/svg+xml, but the server only understands image/jpeg.
416 Range Not Satisfiable (RFC 7233)
The client has asked for a portion of the file (byte serving), but the server cannot supply that portion. For example, if the client asked for a part of the file that did not exist.
417 Expectation Failed
The server cannot meet the requirements of the Expect request-header field.^[43]
418 I'm a teapot (RFC 2324)
This code was defined in 1968 as one of the traditional IETF April Fools' jokes, in RFC 2324, *Hyper Text Coffee Pot Control Protocol*, and is not expected to be used in earnest.
421 Misdirected Request (RFC 7540)
The request was directed at a server that is not able to produce a response (for example because a connection reuse).^[52]

422 Unprocessable Entity (WebDAV; RFC 4918)
The request was well-formed but was unable to be followed due to semantic errors.^[45]
423 Locked (WebDAV; RFC 4918)
The resource that is being accessed is locked.^[46]
424 Failed Dependency (WebDAV; RFC 4918)
The request failed due to failure of a previous request (e.g., a PROPPATCH).^[47]

426 Upgrade Required
The client should switch to a different protocol such as TLS/1.0, given in the Upgrade header field.^[53]
428 Precondition Required (RFC 6585)
The origin server requires the request to be conditional. Intended to prevent the 'lost update' problem, where a client GETs a resource's state, modifies it, and then sends a PUT with the same state, overwriting the changes made by another client.
429 Too Many Requests (RFC 6585)
The user has sent too many requests in a given amount of time. Intended for use with rate-limiting schemes.^[54]

431 Request Header Fields Too Large (RFC 6585)
The server is unwilling to process the request because either an individual header field, or all the header fields collectively, are too large.^[55]

451 Unavailable For Legal Reasons
A server operator has received a legal demand to deny access to a resource or to a set of resources that includes the requested resource.^[56] The code /

5xx Server Error [edit]

The server failed to fulfill an apparently valid request.^[56]

Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has encountered a problem with the user. These response codes are applicable to any request method.^[57]

500 Internal Server Error
A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.
501 Not Implemented
The server either does not recognize the request method, or it lacks the ability to fulfill the request. Usually this implies that the server does not support the requested function.
502 Bad Gateway
The server was acting as a gateway or proxy and received an invalid response from the upstream server.^[59]
503 Service Unavailable
The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary condition.
504 Gateway Timeout
The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.^[60]
505 HTTP Version Not Supported
The server does not support the HTTP protocol version used in the request.^[62]
506 Variant Also Negotiates (RFC 2295)
Transparent content negotiation for the request results in a circular reference.^[63]
507 Insufficient Storage (WebDAV; RFC 4918)
The server is unable to store the representation needed to complete the request.^[15]
508 Loop Detected (WebDAV; RFC 5842)
The server detected an infinite loop while processing the request (sent in lieu of 208 Already Reported).
510 Not Extended (RFC 2774)
Further extensions to the request are required for the server to fulfill it.^[64]
511 Network Authentication Required (RFC 6585)
The client needs to authenticate to gain network access. Intended for use by intercepting proxies used to control access to networks.
512 Resource In Use (WebDAV; RFC 4918)
The resource is in use.
513 Resource Moved Temporarily (WebDAV; RFC 4918)
The resource has moved temporarily.
514 Range Not Satisfiable (WebDAV; RFC 4918)
The range specified by the client is not satisfiable.
515 Too Many Hops (WebDAV; RFC 4918)
The number of hops in the request header field is greater than the maximum allowed.
516 Too Many Requests (WebDAV; RFC 4918)
The user has sent too many requests in a given amount of time. Intended for use with rate-limiting schemes.^[54]

28 4XX Error Codes

11 5XX Error Codes

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





418 - I'm A Teapot



@samnewman



410 - Gone

DOGS!

HTTP STATUS DOGS

Hypertext Transfer Protocol Response status codes. And dogs.

Inspired by the [HTTP Status Cats](#) from [@girlie_mac](#) :)



100

Continue



200

OK



201

Created



202

Accepted



203

Non-Authoritative Information



204

No Content



206

Partial Content



207

Multi-Status



<https://httpstatusdogs.com>

Keep it simple

**Simple = Synchronous, Request
Response Communication**

Synchronous

Synchronous

Asynchronous

Synchronous

Block and wait

Asynchronous

Synchronous

Block and wait

Asynchronous

Fire and (maybe) forget

Synchronous

Block and wait

Asynchronous

Fire and (maybe) forget

Simple to reason about

Synchronous

Block and wait

Asynchronous

Fire and (maybe) forget

Simple to reason about

Technology more
straight forward

Synchronous

Block and wait

Simple to reason about

Technology more
straight forward

Asynchronous

Fire and (maybe) forget

Great for long-
running jobs

Synchronous

Block and wait

Simple to reason about

Technology more
straight forward

Asynchronous

Fire and (maybe) forget

Great for long-
running jobs

And low latency too!

Synchronous

Block and wait

Simple to reason about

Technology more
straight forward

Asynchronous

Fire and (maybe) forget

Great for long-
running jobs

And low latency too!

More complex

Poll

Do you use sync or async communication?

Poll

Do you use sync or async communication?

a.) Synchronous

Poll

Do you use sync or async communication?

a.) Synchronous

Poll

Do you use sync or async communication?

a.) Synchronous

b.) Asynchronous

Poll

Do you use sync or async communication?

a.) Synchronous

b.) Asynchronous

Poll

Do you use sync or async communication?

- a.) Synchronous**
- b.) Asynchronous**
- c.) Both!**

Collaboration Styles

Collaboration Styles

Request/Response

Collaboration Styles

Request/Response

Event-based

Collaboration Styles

Request/Response

Initiate a request, expect
a response

Event-based

Collaboration Styles

Request/Response

Initiate a request, expect
a response

Event-based

Things happen,
things react

Request/Response

Request/Response

Event-based

Request/Response

Event-based

Synchronous

Request/Response

Event-based

Synchronous

Asynchronous

Request/Response

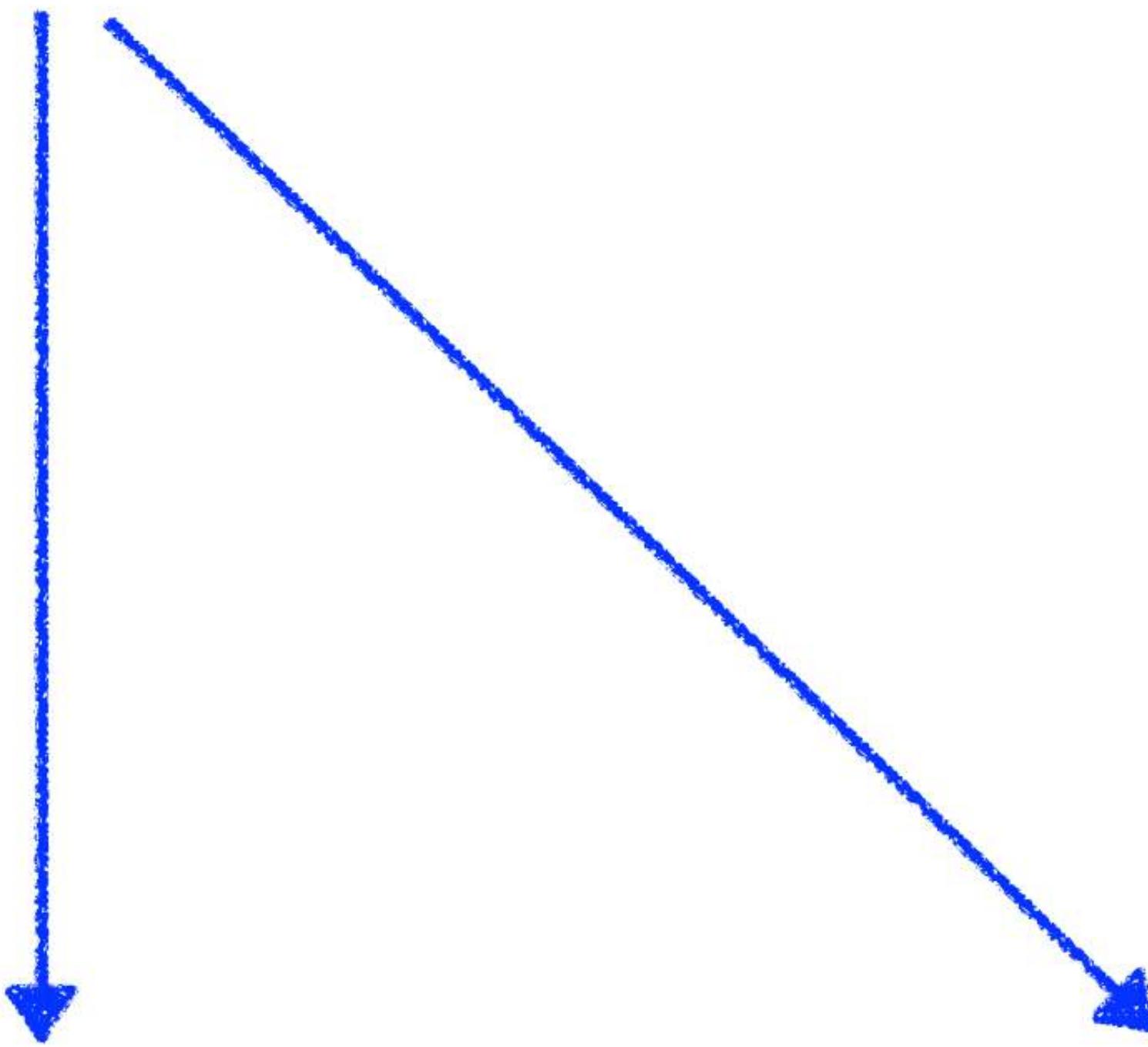


Synchronous

Event-based

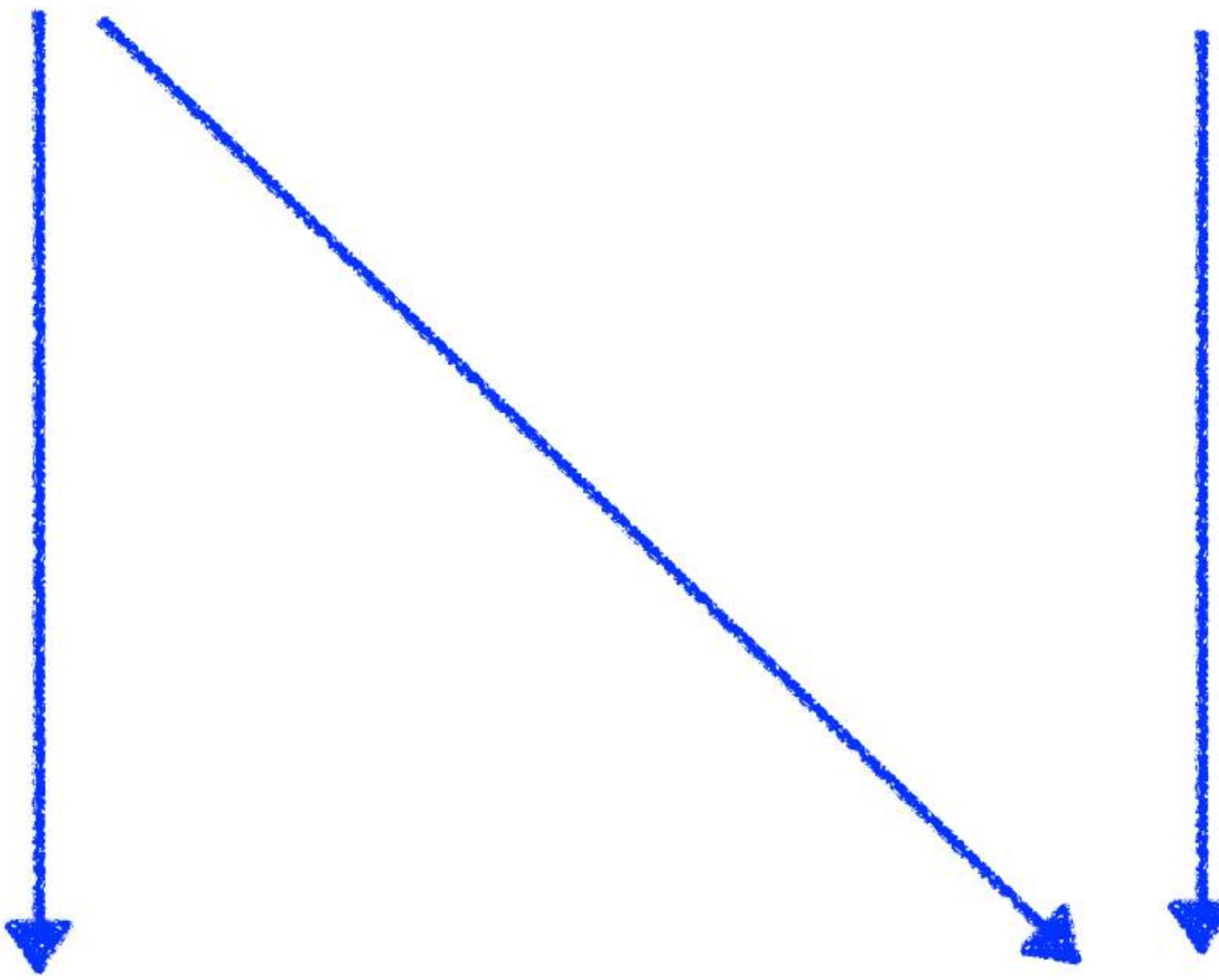
Asynchronous

Request/Response

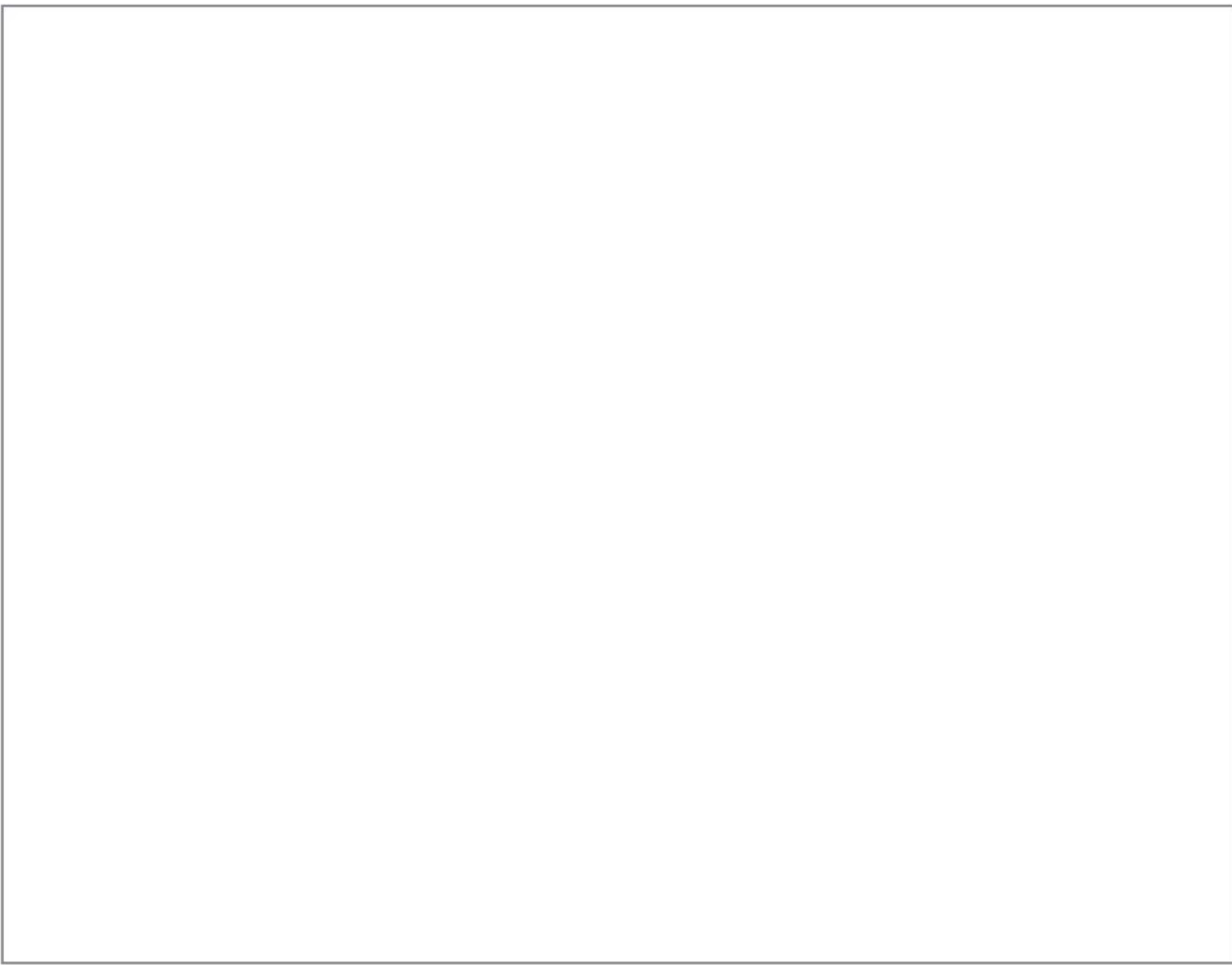


Event-based

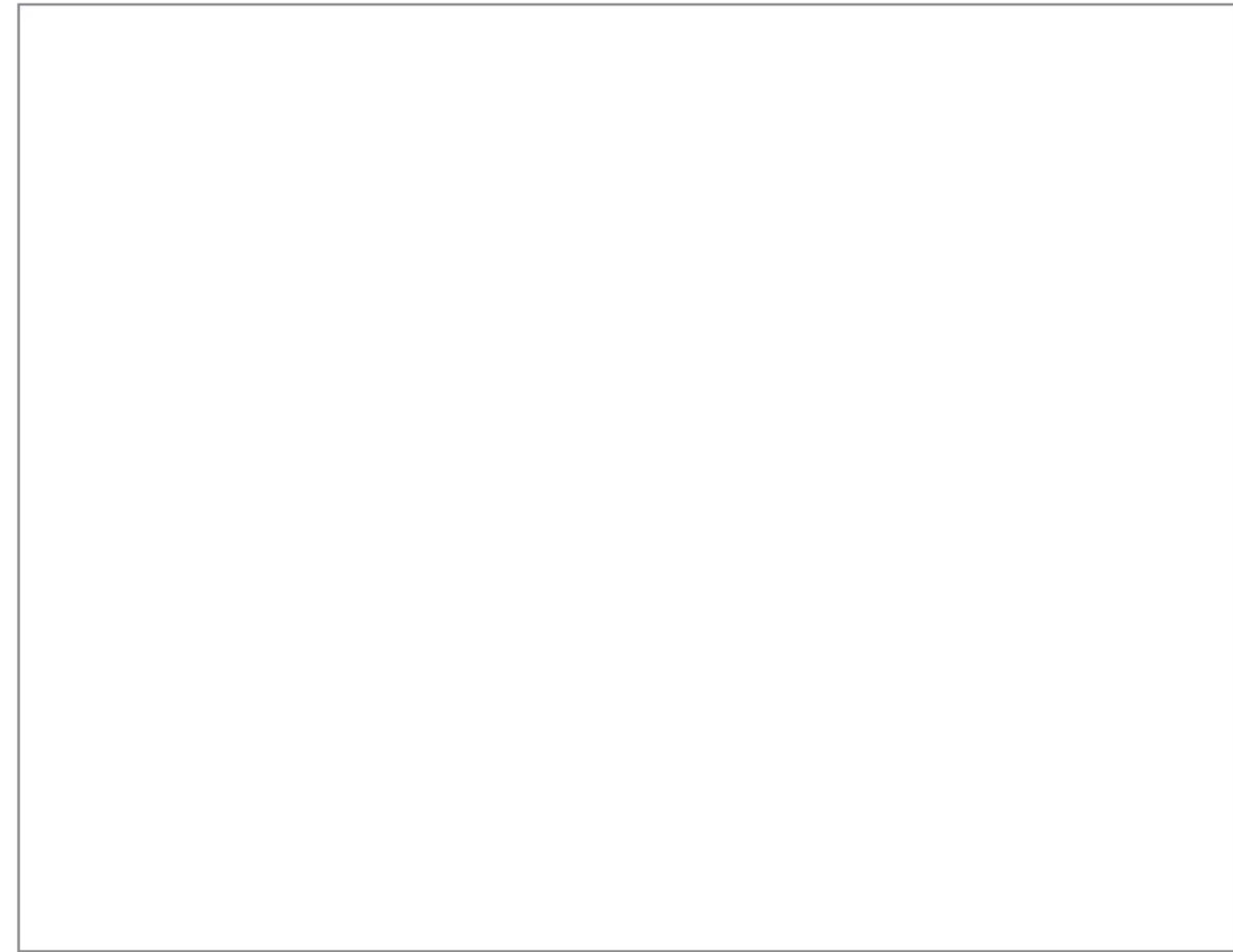
Request/Response



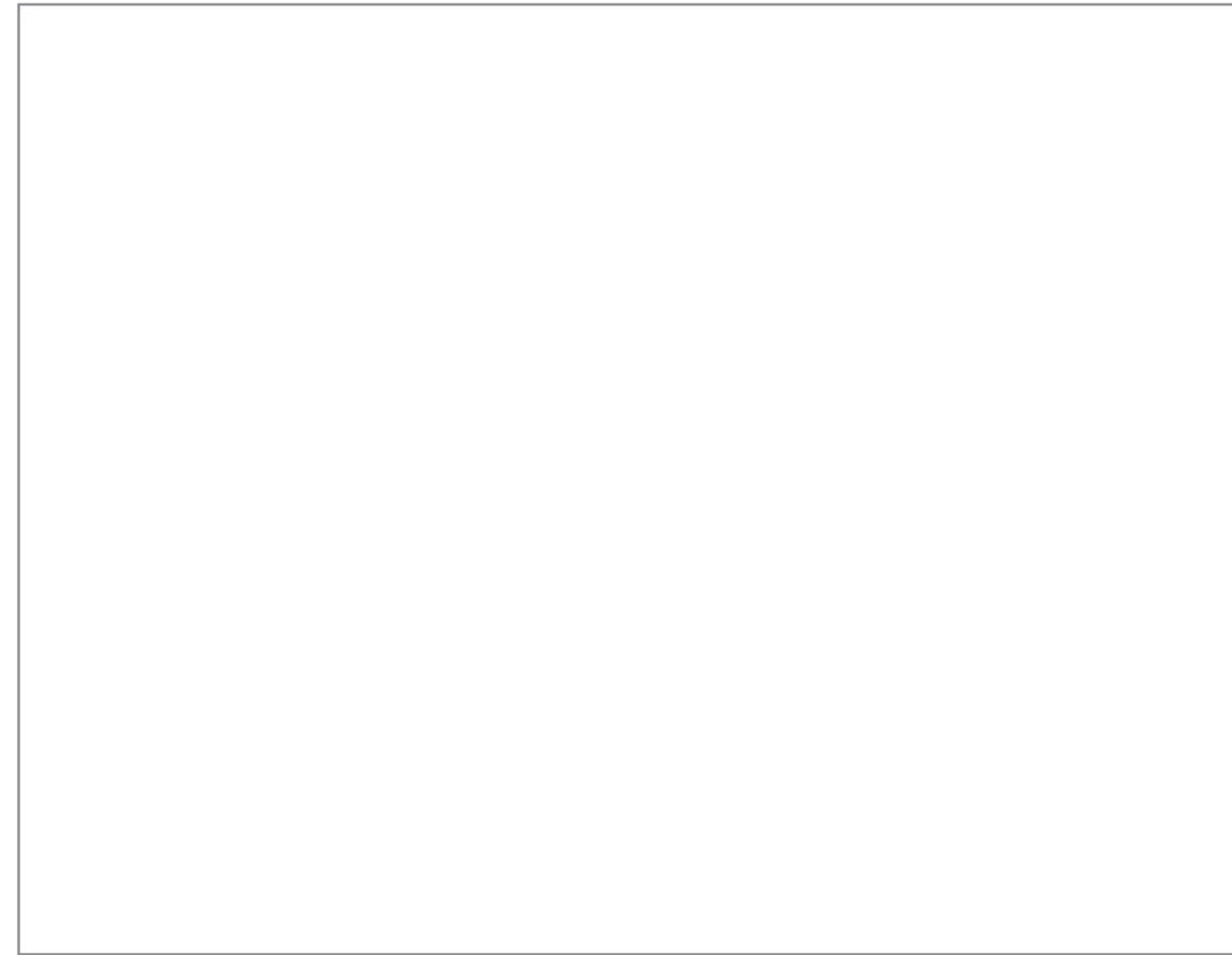
Event-based



Request/Response



Request/Response



Event-based

Request/Response

Event-based

Synchronous

Request/Response

Event-based

Synchronous
Asynchronous

Synchronous

Asynchronous

Request/Response

Event-based

RPC

Client

issueInvoice(...)

Server

Client

```
issueInvoice(...)
```

Server

```
public void issueInvoice(..) {  
    ...  
}
```

Client

```
issueInvoice(...)
```

Server

```
public void issueInvoice(...) {  
    ...  
}
```

FOWLER'S FIRST LAW OF DISTRIBUTED OBJECTS

FirstLaw



Martin Fowler

My First Law of Distributed Object Design: Don't distribute your objects (From P of EAA).

The relevant chapter is available online.

<https://martinfowler.com/bliki/FirstLaw.html>

FOWLER'S FIRST LAW OF DISTRIBUTED OBJECTS

FirstLaw



Martin Fowler

My First Law of Distributed Object Design: Don't distribute objects (From P of EAA).

The relevant chapter is available online.

<https://martinfowler.com/bliki/FirstLaw.html>

Microservices and the First Law of Distributed Objects



Martin Fowler

13 August 2014

When I wrote [Patterns of Enterprise Application Architecture](#), I coined what I called the **First Law of Distributed Object Design: "don't distribute your objects"**. In recent months there's been a lot of interest in [microservices](#), which has led a few people to ask whether microservices are in contravention to this law, and if so why I am in favor of them?

It's important to note that in this first law statement, I use the phrase "distributed objects". This reflects an idea that was rather in vogue in the late 90's early 00's but since has (rightly) fallen out of favor. The idea of distributed objects is that you could design objects and choose to use these same objects either in-process or remote, where remote might mean another process in the same machine, or on a different machine. Clever middleware, such as DCOM or a CORBA implementation, would handle the in-process/remote distinction so your system could be written and you could break it up into processes independently of how the application was designed.

<https://martinfowler.com/articles/distributed-objects-microservices.html>

Beware tight coupling with RPC

SCHEMA BINDING



SCHEMA BINDING

Code

```
public class Customer {  
    private int id;  
    private String name;  
    private int age;  
    ...  
}
```



SCHEMA BINDING

Code

```
public class Customer {  
    private int id;  
    private String name;  
    private int age;  
    ...  
}
```

JSON

```
{  
    "id" : 123,  
    "name" : "sam",  
    "age" : 15  
}
```



TOLERANT READER PATTERN

Postel's Law:
“Be conservative in what you do, liberal in what you expect”

TOLERANT READER PATTERN

Postel's Law:
“Be conservative in what you do, liberal in what you expect”

JSON v1

```
{  
  "id" : 123,  
  "name" : "sam",  
  "age" : 15  
}
```

TOLERANT READER PATTERN

Postel's Law:
“Be conservative in what you do, liberal in what you expect”

JSON v1

```
{  
  "id" : 123,  
  "name" : "sam",  
  "age" : 15  
}
```

JSON v2

```
{  
  "id" : 123,  
  "fullname" : "sam",  
  "dob" : 15/08/1938  
}
```

TOLERANT READER PATTERN

Postel's Law:
“Be conservative in what you do, liberal in what you expect”

JSON v1

```
{  
  "id" : 123,  
  "name" : "sam",  
  "age" : 15  
}
```

JSON v2

```
{  
  "id" : 123,  
  "fullname" : "sam",  
  "dob" : 15/08/1938  
}
```

Breaking change

TOLERANT READER PATTERN

Postel's Law:
“Be conservative in what you do, liberal in what you expect”

JSON v1

```
{  
  "id" : 123,  
  "name" : "sam",  
  "age" : 15  
}
```

JSON v2

```
{  
  "id" : 123,  
  "fullname" : "sam",  
  "dob" : 15/08/1938  
}
```

But what if we only wanted “ID”?

Breaking change

SCHEMAS - TYPE

```
"namespace": "com.acme",
"protocol": "HelloWorld",
"doc": "Protocol Greetings",

"types": [
    {"name": "Greeting", "type": "record", "fields": [
        {"name": "message", "type": "string"}]},
    {"name": "Curse", "type": "error", "fields": [
        {"name": "message", "type": "string"}]}
],
```

SCHEMAS - MESSAGE

```
"messages": {  
    "hello": {  
        "doc": "Say hello.",  
        "request": [{"name": "greeting", "type": "Greeting"}],  
        "response": "Greeting",  
        "errors": ["Curse"]  
    }  
}
```

GOOD OPTIONS...

GOOD OPTIONS...



The Apache Avro™ 1.8.2 Documentation

Apache Avro™ 1.8.2 Documentation

Introduction
Schemas
Comparison with other systems

Introduction

Apache Avro™ is a data serialization system.

Avro provides:

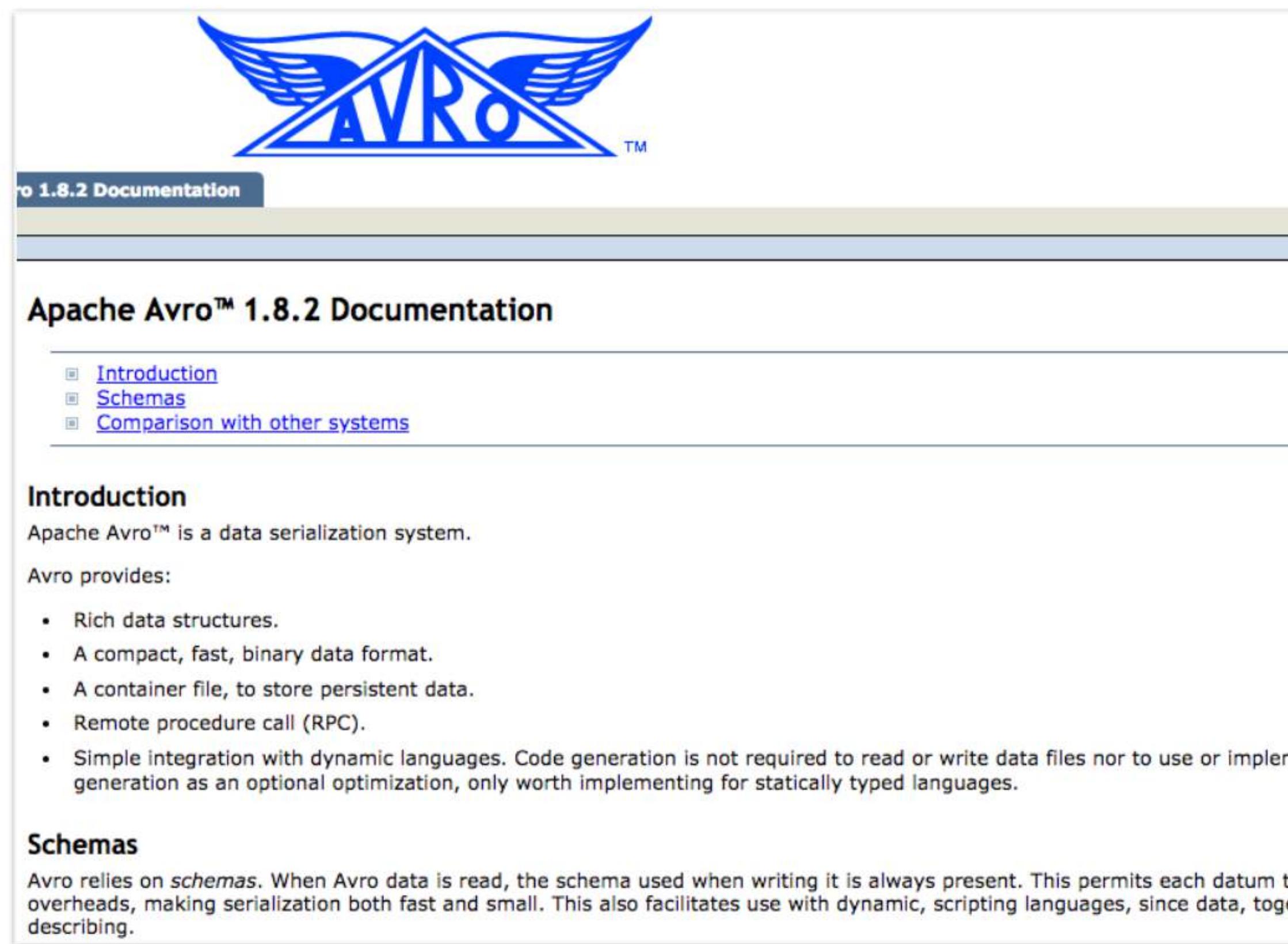
- Rich data structures.
- A compact, fast, binary data format.
- A container file, to store persistent data.
- Remote procedure call (RPC).
- Simple integration with dynamic languages. Code generation is not required to read or write data files nor to use or implement generation as an optional optimization, only worth implementing for statically typed languages.

Schemas

Avro relies on *schemas*. When Avro data is read, the schema used when writing it is always present. This permits each datum to be overheads, making serialization both fast and small. This also facilitates use with dynamic, scripting languages, since data, together describing.

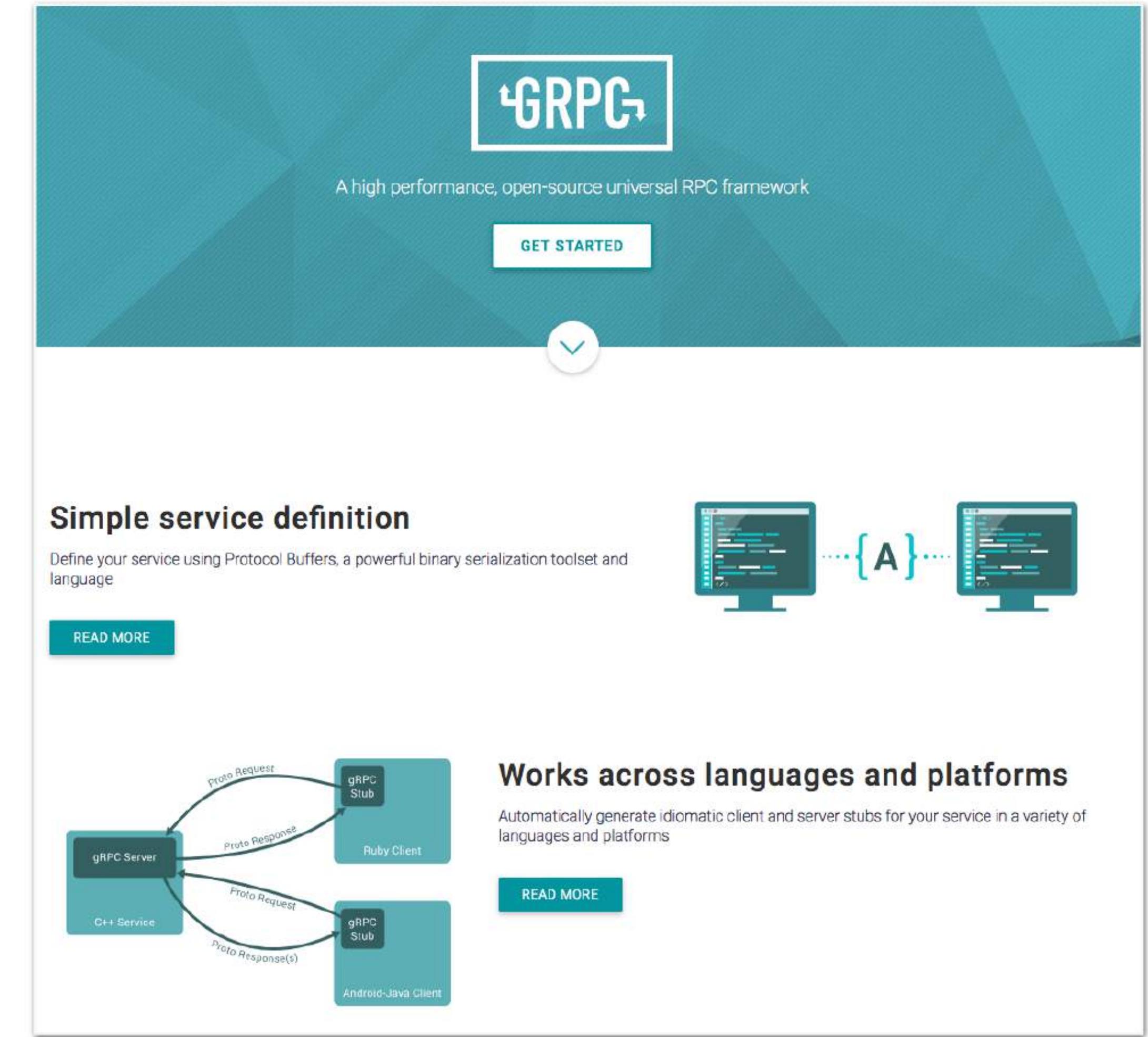
<http://avro.apache.org/docs/current/>

GOOD OPTIONS...



The screenshot shows the Apache Avro 1.8.2 Documentation homepage. At the top is the Avro logo with wings. Below it is a navigation bar with a "GET STARTED" button. The main content area is titled "Apache Avro™ 1.8.2 Documentation". It includes sections for "Introduction", "Schemas", and "Comparison with other systems". The "Introduction" section describes Avro as a data serialization system and lists its features: Rich data structures, A compact, fast, binary data format, A container file, to store persistent data, Remote procedure call (RPC), and Simple integration with dynamic languages. The "Schemas" section notes that Avro relies on schemas for data serialization.

<http://avro.apache.org/docs/current/>



The screenshot shows the gRPC website homepage. At the top is the gRPC logo. Below it is a sub-headline: "A high performance, open-source universal RPC framework". There is a "GET STARTED" button and a "Simple service definition" section with a "READ MORE" button. The "Simple service definition" section contains a diagram showing a "gRPC Server" connected to two clients: a "Ruby Client" and an "Android-Java Client". The server sends "Proto Request" messages to both clients, which respond with "Proto Response" and "Proto Response(s)" messages respectively. To the right of this section is a "Works across languages and platforms" section with a "READ MORE" button. This section features an illustration of three computer monitors connected by a dashed line labeled "A", representing the跨语言和平台的特性。

<https://grpc.io/>

Synchronous

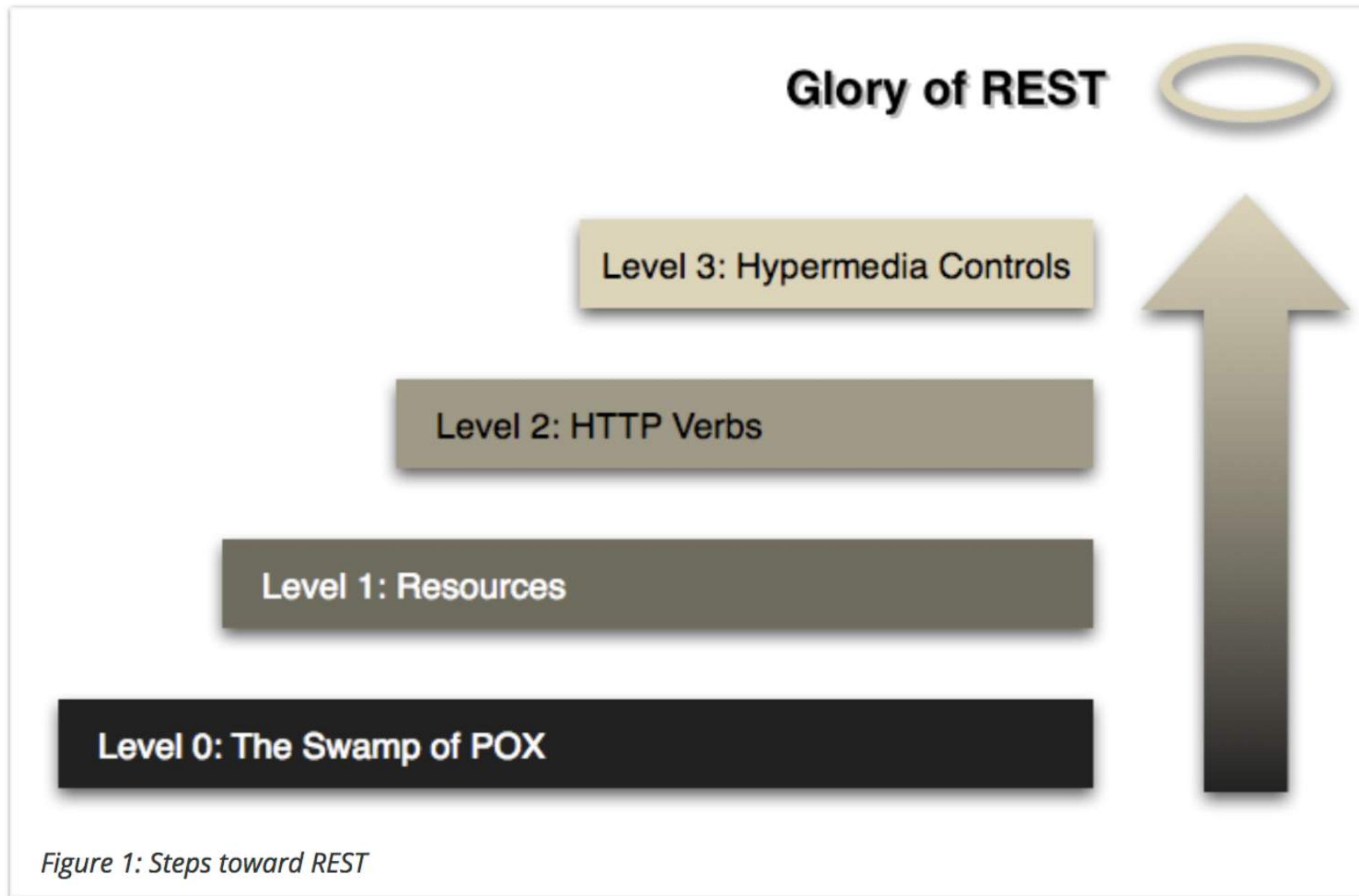
Asynchronous

Request/Response

RPC

REST

Event-based



<https://martinfowler.com/articles/richardsonMaturityModel.html>

Richardson Maturity Model

steps toward the glory of REST

A model (developed by Leonard Richardson) that breaks down the principal elements of a REST approach into three steps. These introduce resources, http verbs, and hypermedia controls.

18 March 2010



Martin Fowler

Contents

- [Level 0](#)
- [Level 1 - Resources](#)
- [Level 2 - HTTP Verbs](#)
- [Level 3 - Hypermedia Controls](#)
- [The Meaning of the Levels](#)

Translations: [Korean](#) · [Portuguese](#) · [Persian](#)

Find **similar articles** to this by looking at these tags: [application integration](#) · [web services](#)

Recently I've been reading drafts of [Rest In Practice](#): a book that a couple of my colleagues have been working on. Their aim is to explain how to use Restful web

<https://martinfowler.com/articles/richardsonMaturityModel.html>

Synchronous

Asynchronous

Request/Response

RPC

REST

Event-based

Synchronous

Request/Response

RPC

REST

Event-based

REST?

Synchronous

Asynchronous

Request/Response

RPC

REST

Akka & Orleans

Event-based

REST?

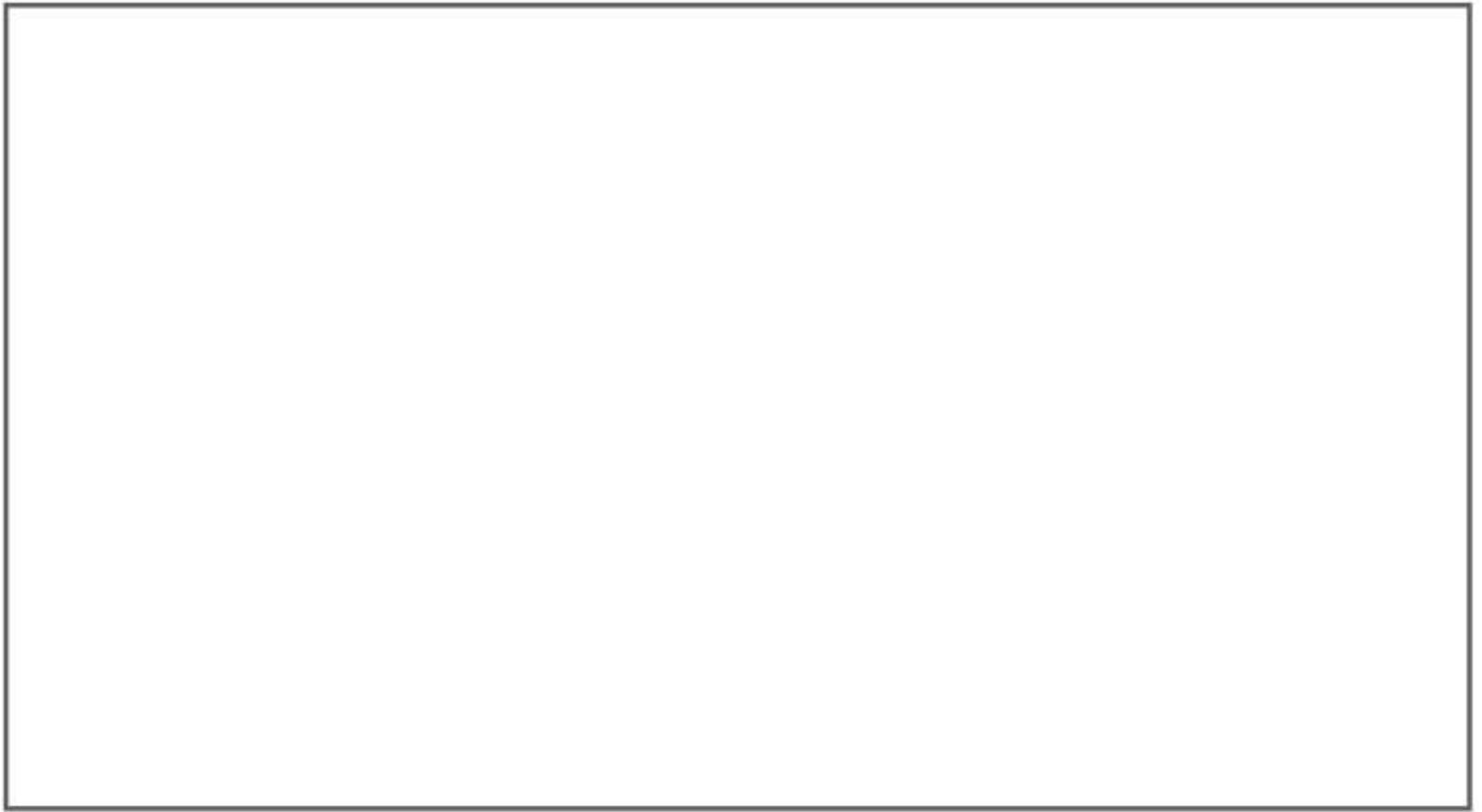
Akka & Orleans

ACTOR-BASED SYSTEMS



ACTOR-BASED SYSTEMS

```
doSomething();
```

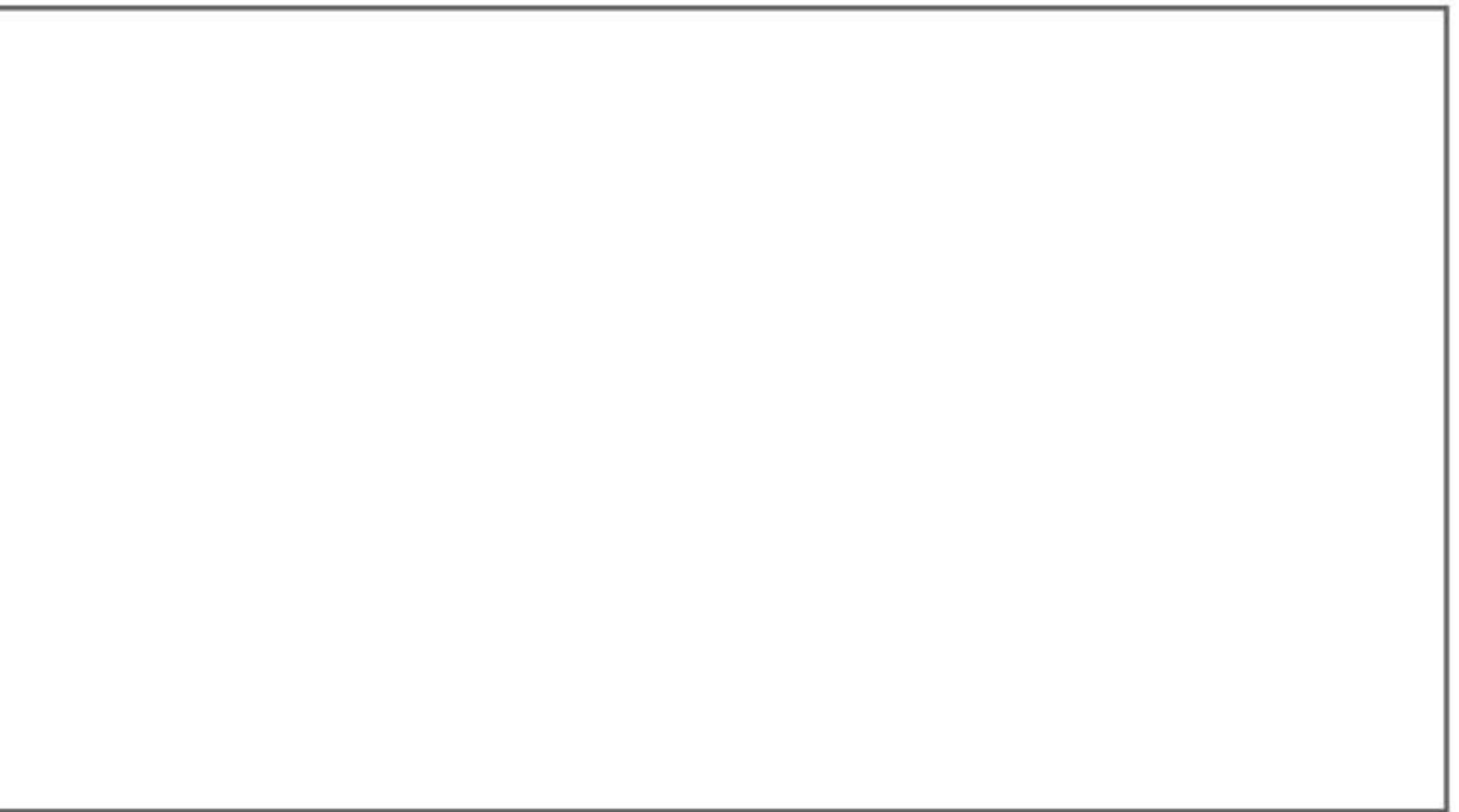


ACTOR-BASED SYSTEMS

```
doSomething();  
doAnotherThing();
```

ACTOR-BASED SYSTEMS

```
doSomething();  
doAnotherThing();  
thenDoThis...
```



ACTOR-BASED SYSTEMS

```
doSomething();  
doAnotherThing();  
thenDoThis...
```

**Each line executed in turn
before we run the next line**

ACTOR-BASED SYSTEMS

```
doSomething();  
doAnotherThing();  
thenDoThis...
```

```
doSomething();  
doAnotherThing();  
thenDoThis...
```

**Each line executed in turn
before we run the next line**

ACTOR-BASED SYSTEMS

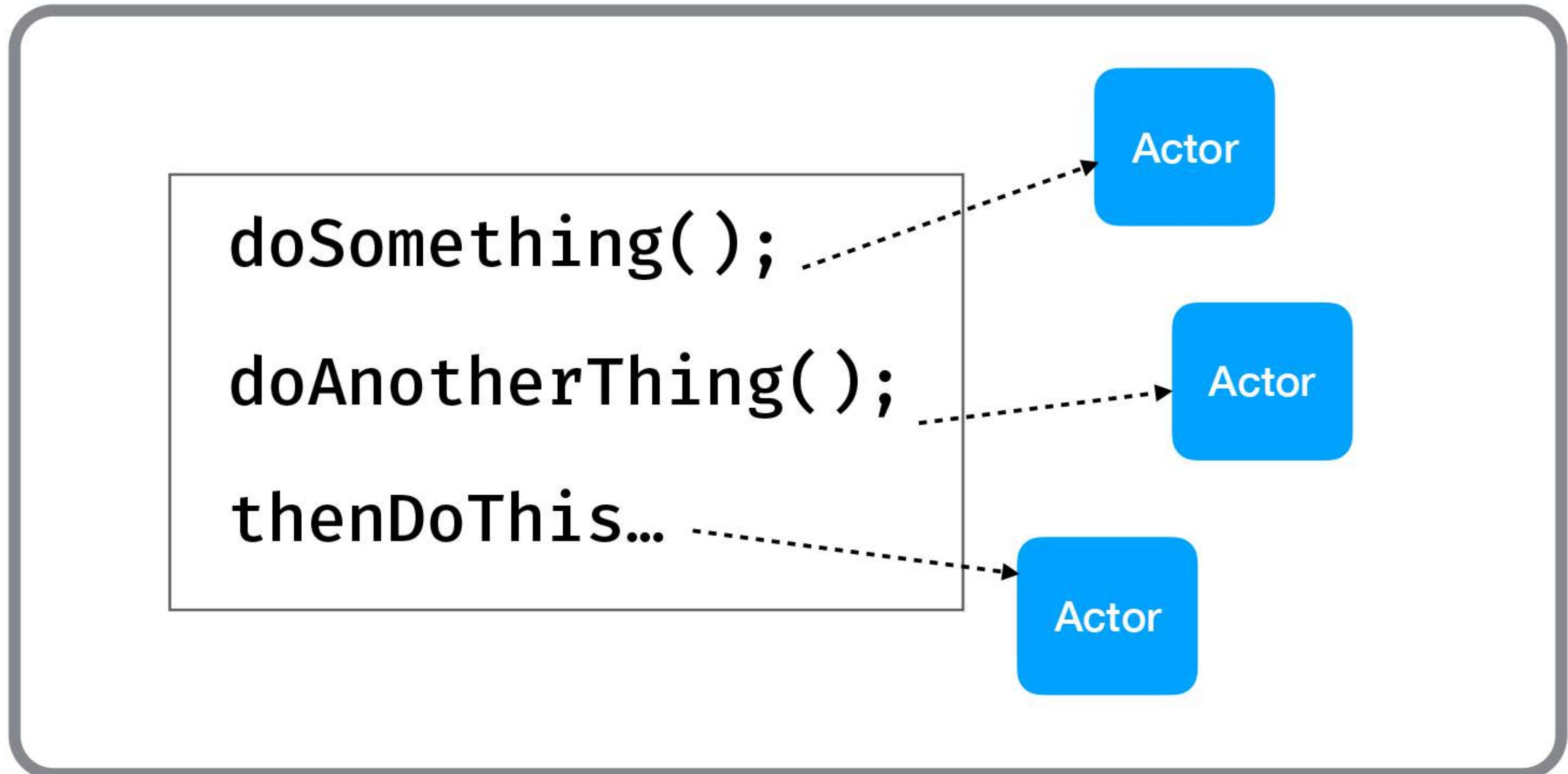
```
doSomething();  
doAnotherThing();  
thenDoThis...
```

Each line executed in turn
before we run the next line

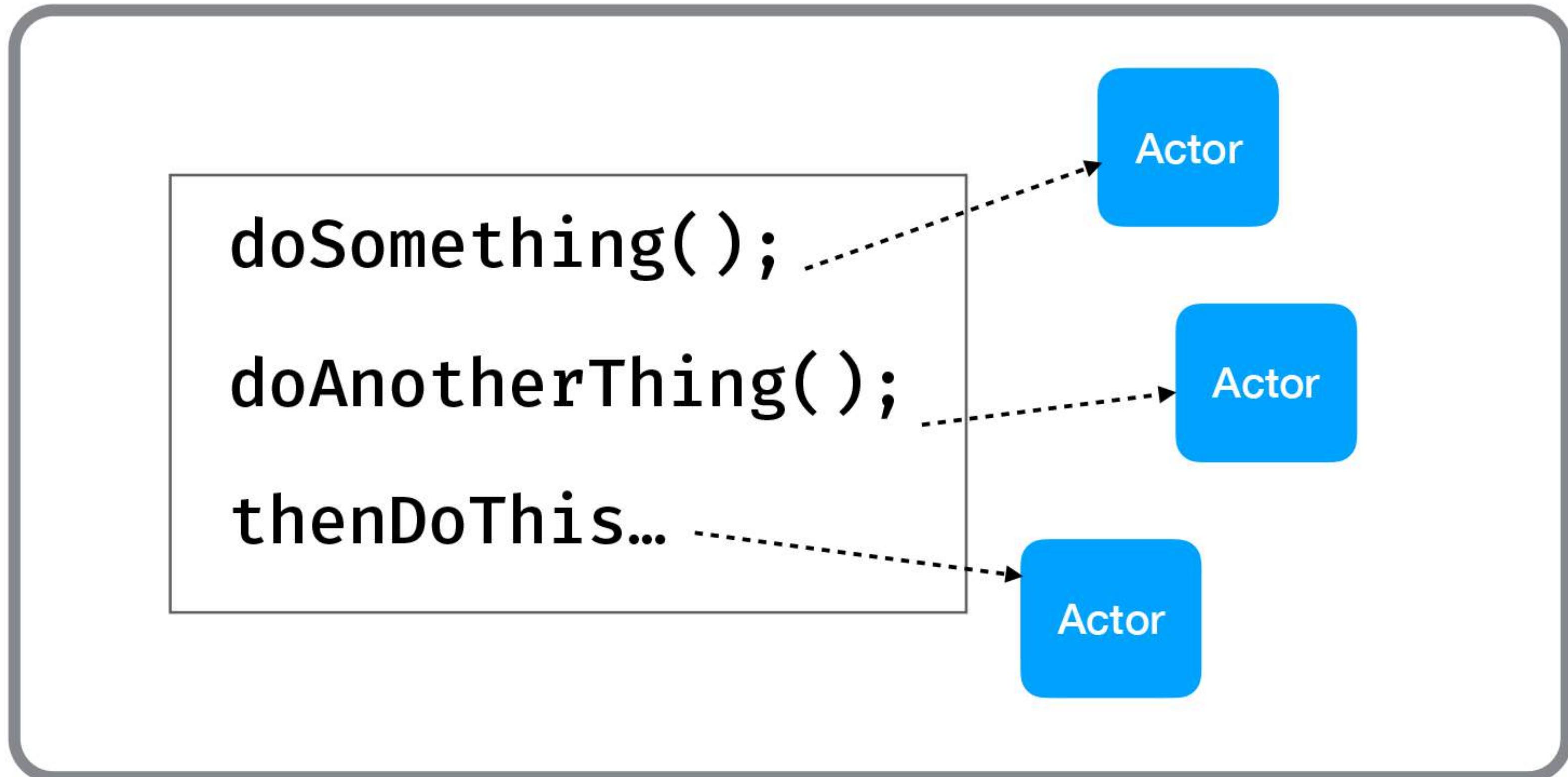
```
doSomething();  
doAnotherThing();  
thenDoThis...
```

With Actor-based systems,
we're passing messages to
actors...

ACTORS CONT.

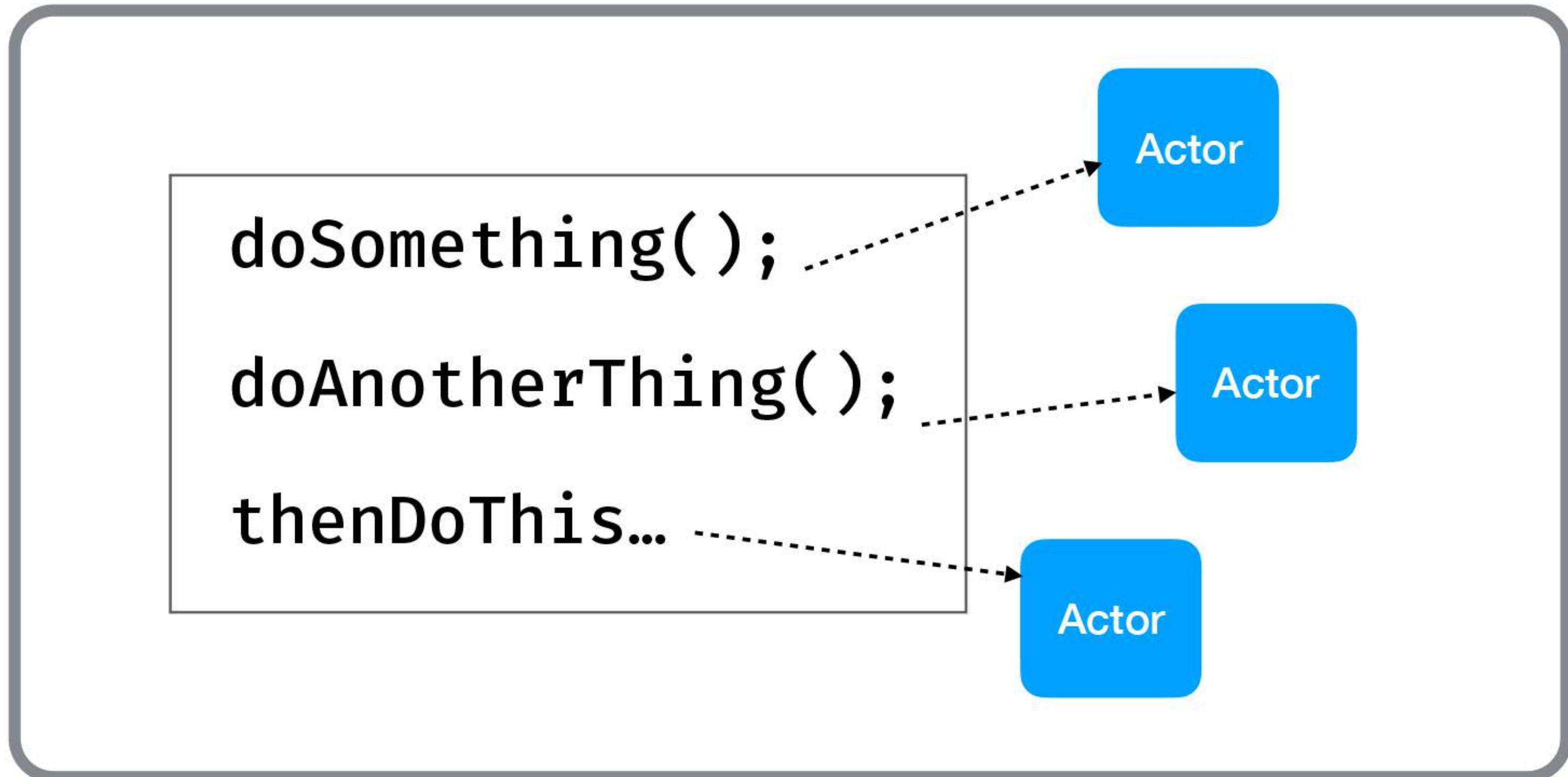


ACTORS CONT.



Actors run on their
own thread of
execution

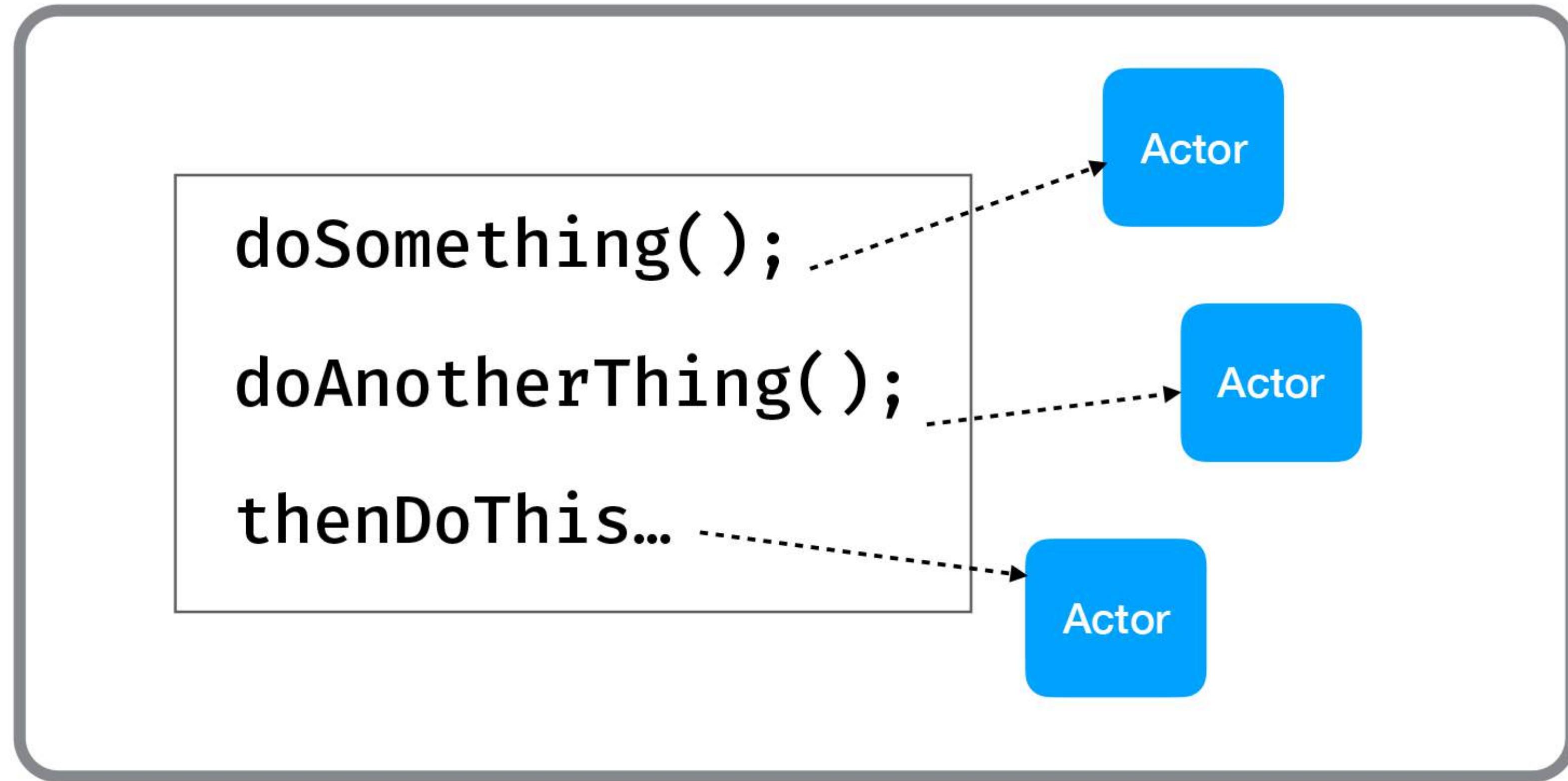
ACTORS CONT.



Actors run on their own thread of execution

Inherently async

ACTORS CONT.

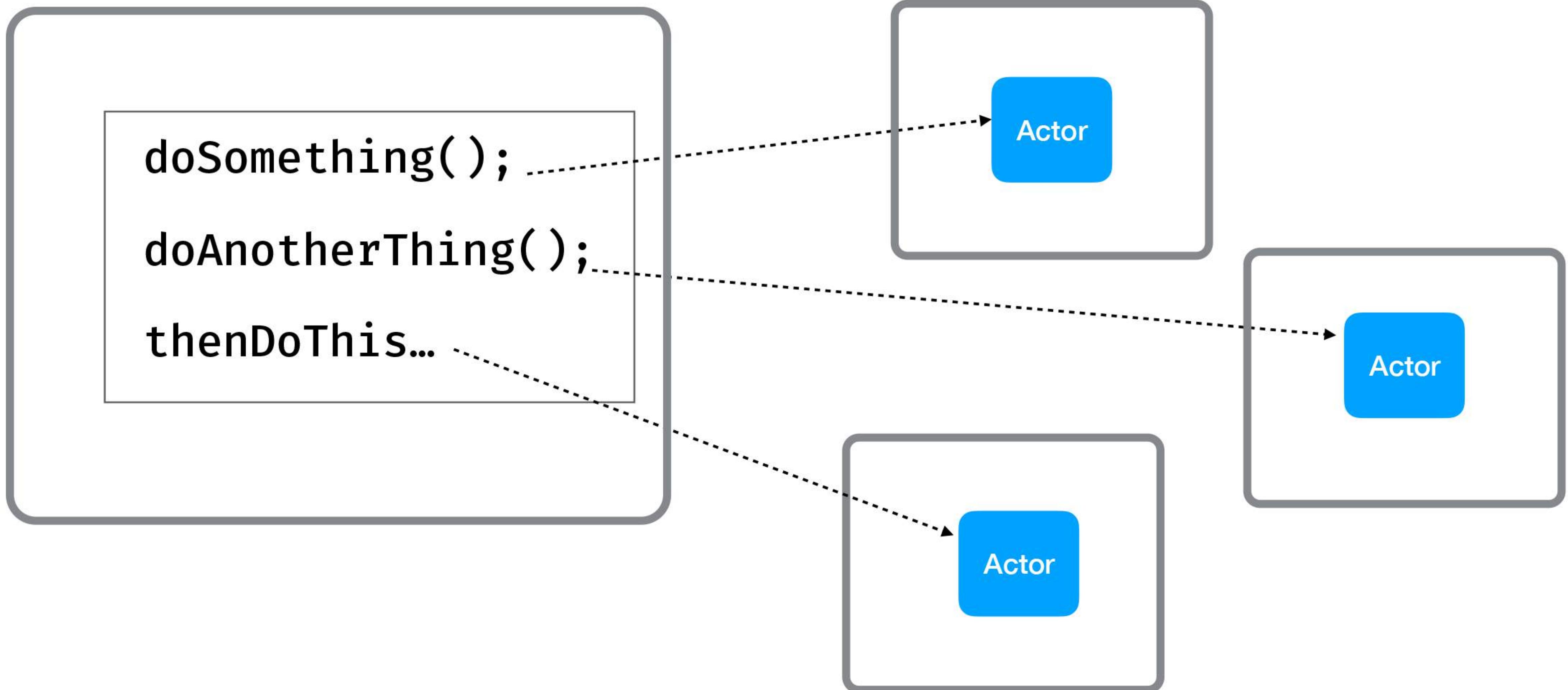


Actors run on their own thread of execution

Inherently async

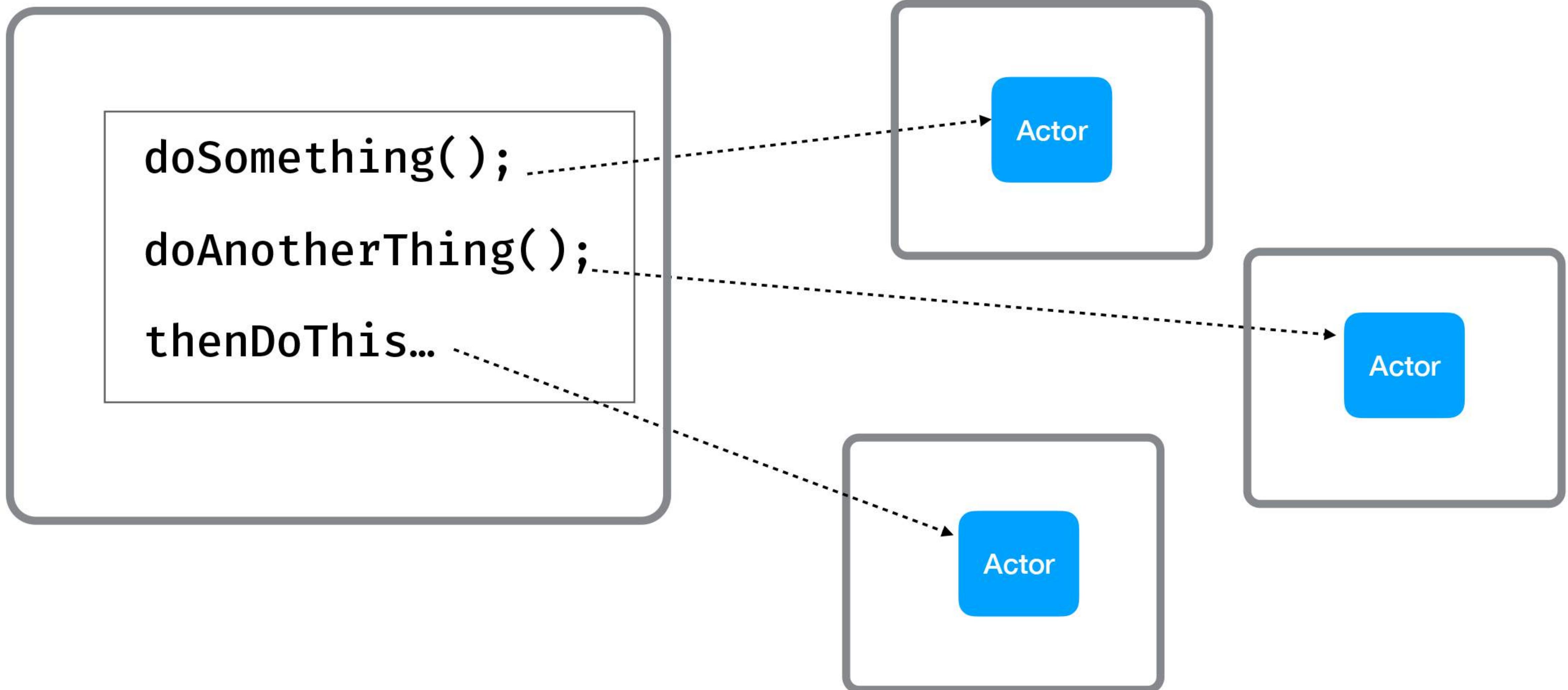
Makes better use of multi-core systems

DISTRIBUTED ACTORS

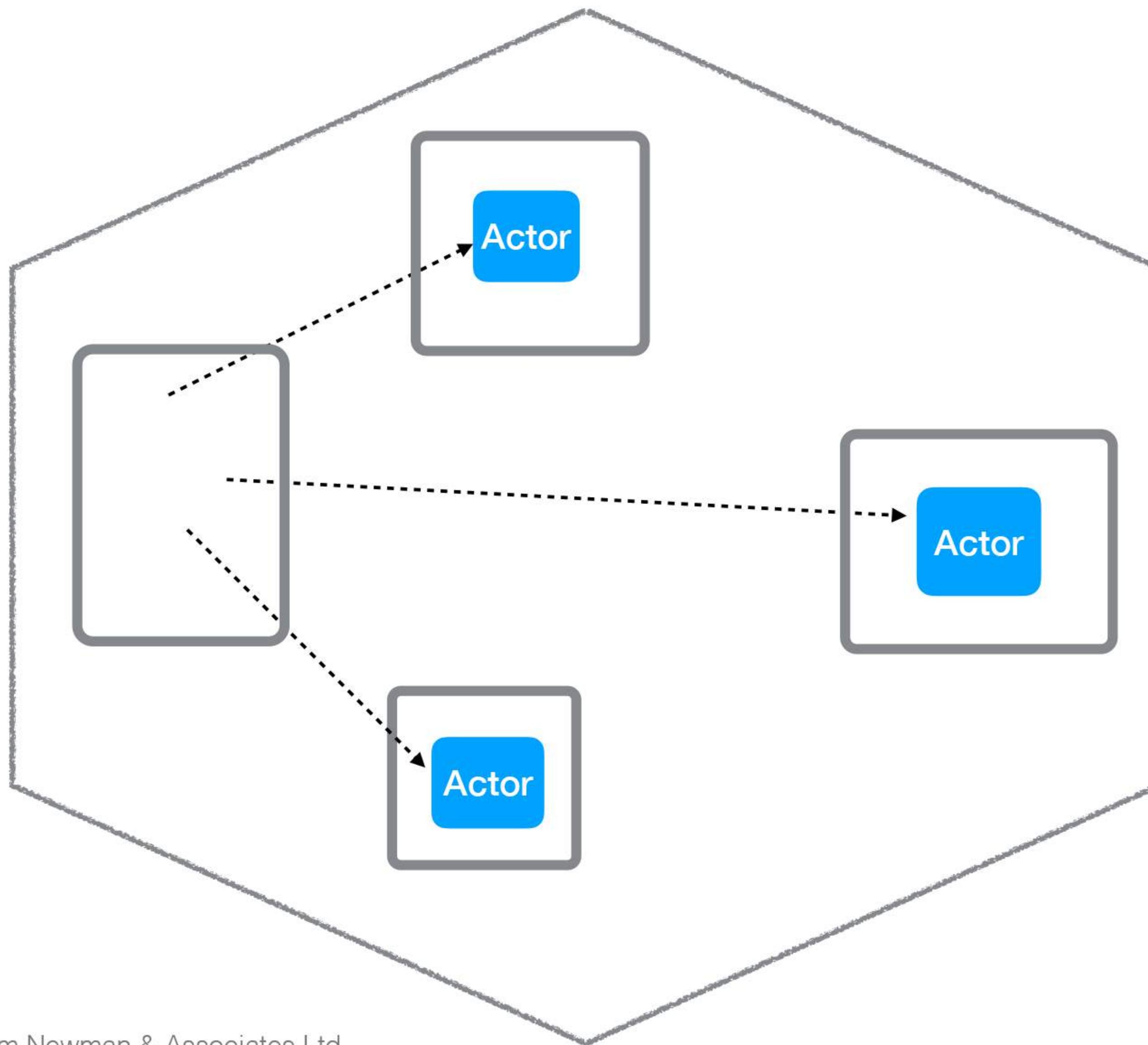


Err, don't distribute your objects?

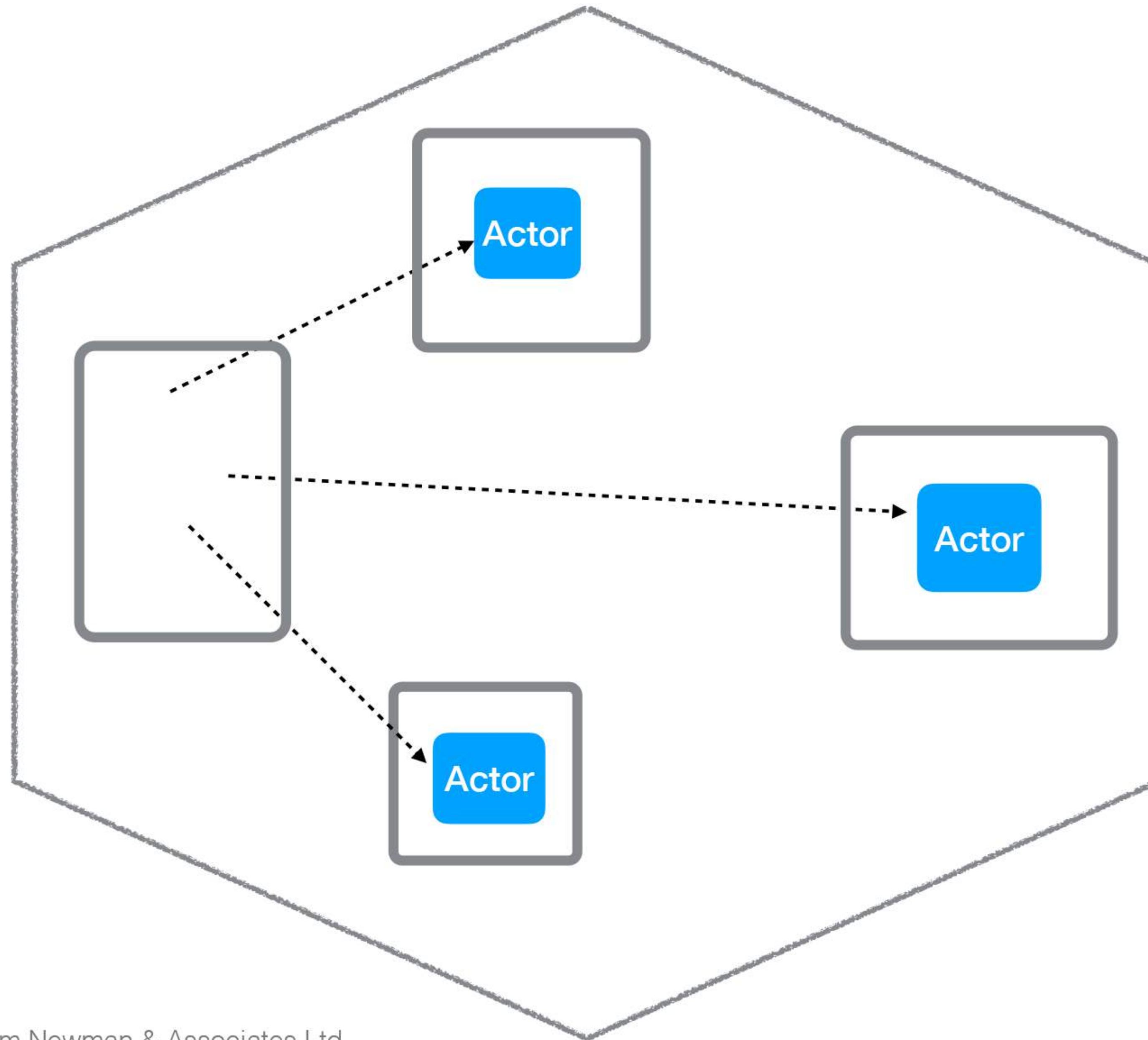
DISTRIBUTED ACTORS



DISTRIBUTED ACTORS & MICROSERVICES?

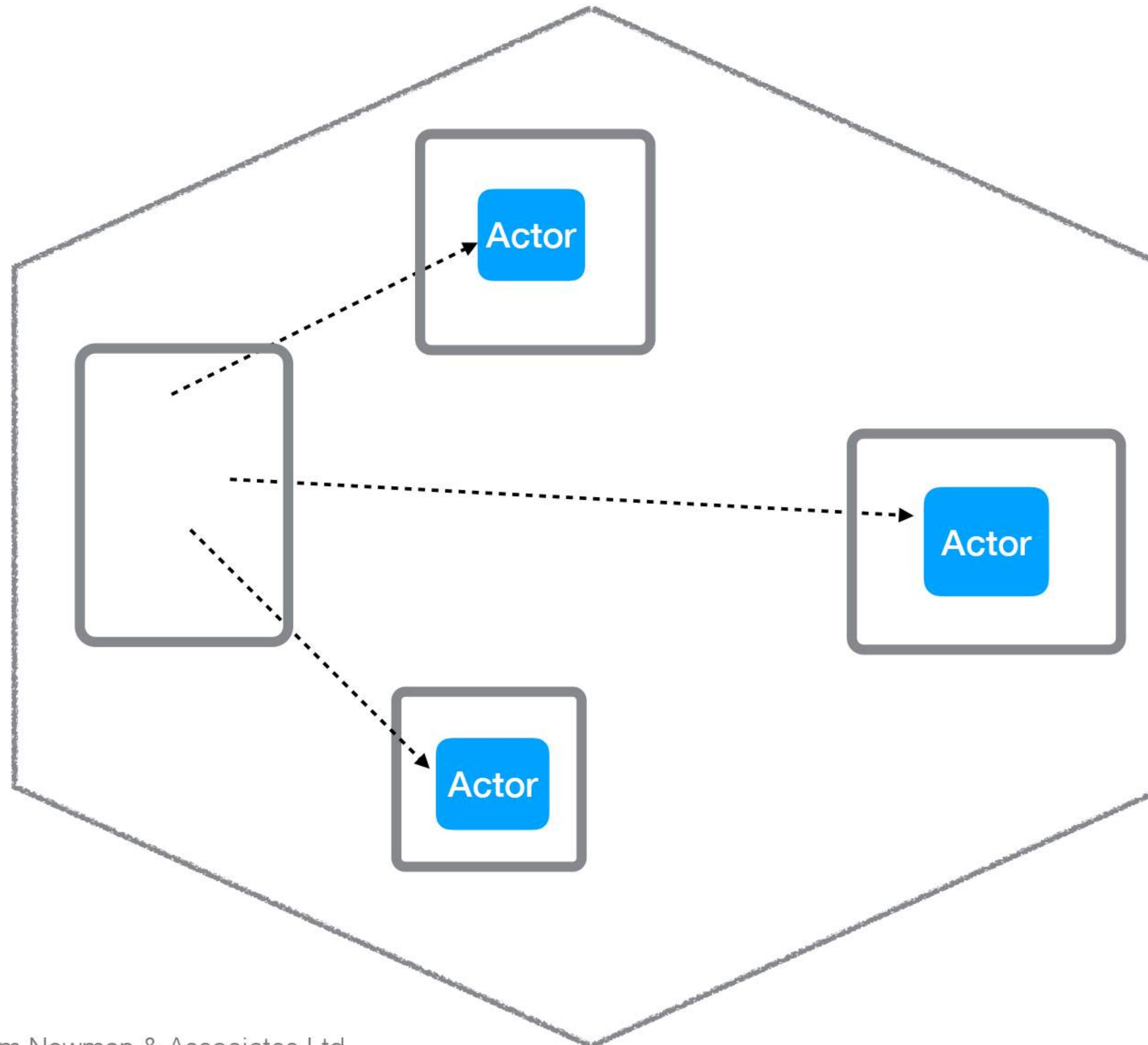


DISTRIBUTED ACTORS & MICROSERVICES?



**Keep actor cluster
inside a logical
boundary**

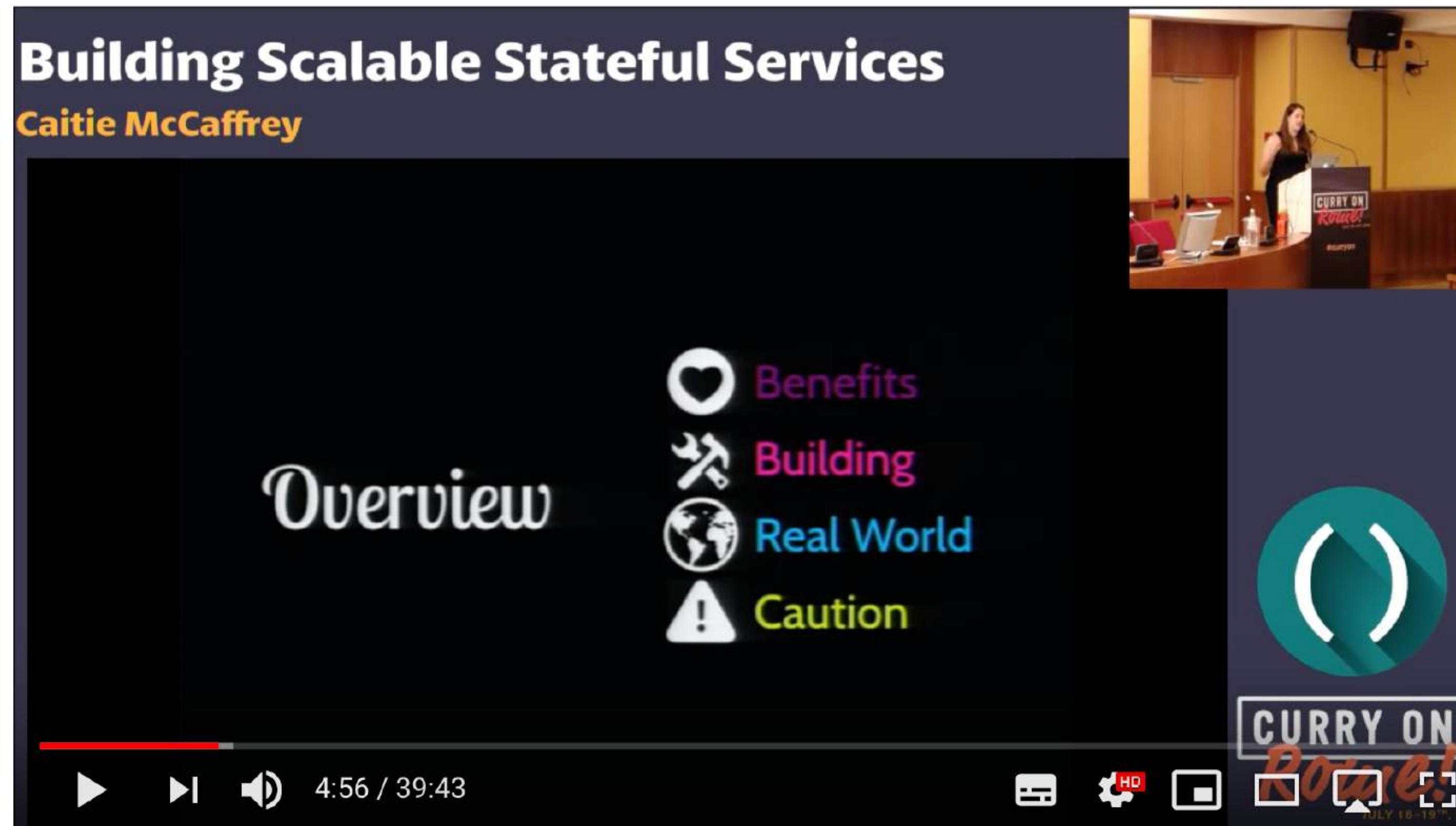
DISTRIBUTED ACTORS & MICROSERVICES?



**Keep actor cluster
inside a logical
boundary**

**Switch to more
coarse-grained
comms for inter-
service
communication**

MORE INFO



Caitie McCaffrey - Building Scalable Stateful Services - Curry On

908 views

1 like

0 dislikes

SHARE

SAVE

...

<https://www.youtube.com/watch?v=aJFxQAAMAQc>

Synchronous

Asynchronous

Request/Response

RPC

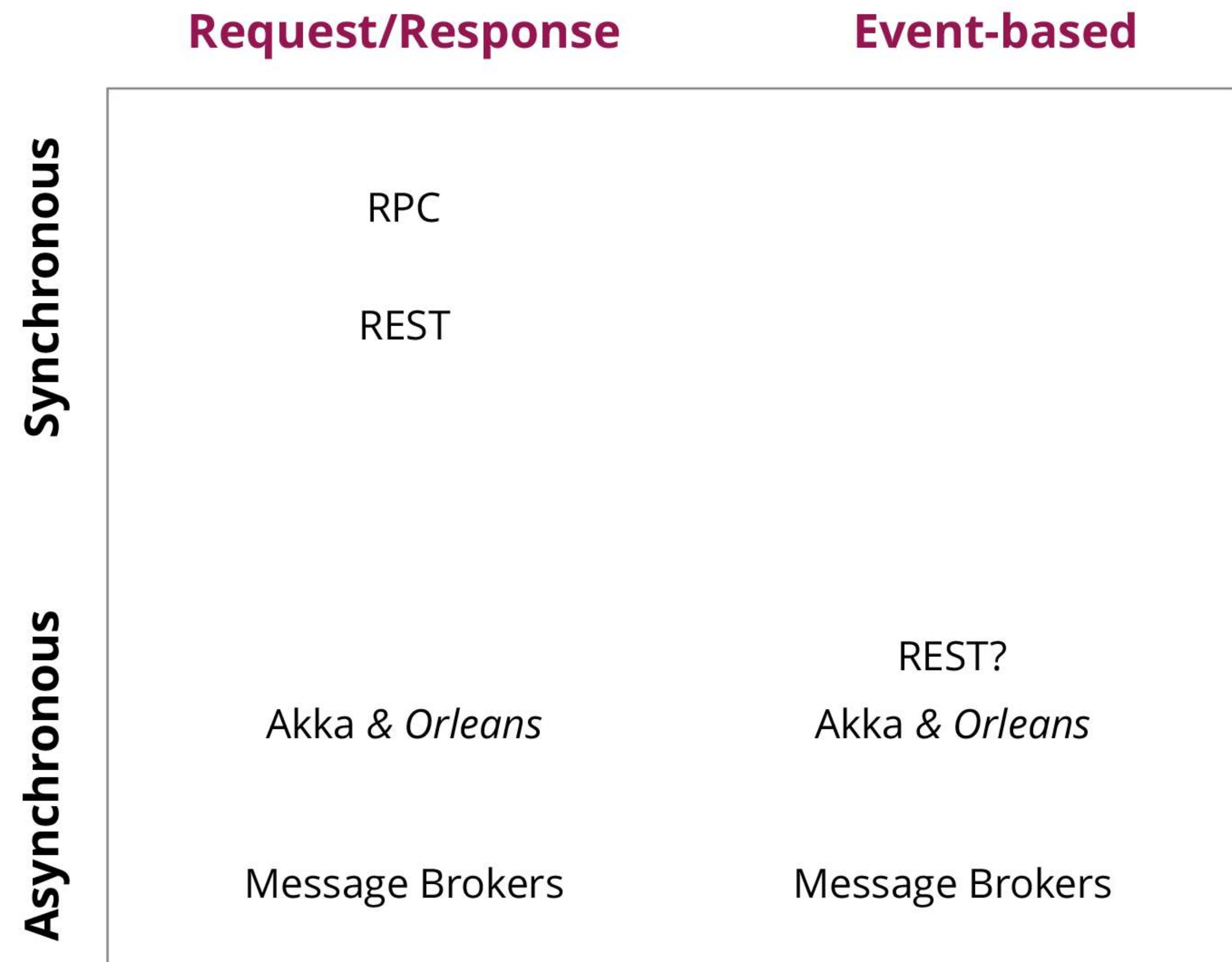
REST

Akka & Orleans

Event-based

REST?

Akka & Orleans





<https://www.flickr.com/photos/mikecogh/11092157174/>

MESSAGE BROKER CONCERNS

Guaranteed Delivery

MESSAGE BROKER CONCERNS

Guaranteed Delivery*

MESSAGE BROKER CONCERNS

Guaranteed Delivery*

Transactions

MESSAGE BROKER CONCERNS

Guaranteed Delivery*

Transactions

Throughput

MESSAGE BROKER CONCERNS

Guaranteed Delivery*

Transactions

Throughput

Ordering

MESSAGE BROKER CONCERNS

Guaranteed Delivery*

Transactions

Throughput

Ordering

Exactly Once Delivery?

Poll

If you use a message broker, which do you use?

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ
- b.) Kafka

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ
- b.) Kafka

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ
- b.) Kafka
- c.) RabbitMQ

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ
- b.) Kafka
- c.) RabbitMQ

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ
- b.) Kafka
- c.) RabbitMQ
- d.) Cloud service (SQS, SNS, or kinesis)

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ
- b.) Kafka
- c.) RabbitMQ
- d.) Cloud service (SQS, SNS, or kinesis)

Poll

If you use a message broker, which do you use?

- a.) ActiveMQ
- b.) Kafka
- c.) RabbitMQ
- d.) Cloud service (SQS, SNS, or kinesis)
- e.) Something else?

CONSENSUS PROBLEMS...

The Byzantine Generals Problem

July 5, 1982

[Download PDF](#)

[Abstract](#) [Related Info](#)

[BibTex](#)

Authors

[Leslie Lamport](#)
Robert Shostak
Marshall Pease

Publication Type

Article

Journal

ACM Transactions on
Programming Languages and
Systems

Pages

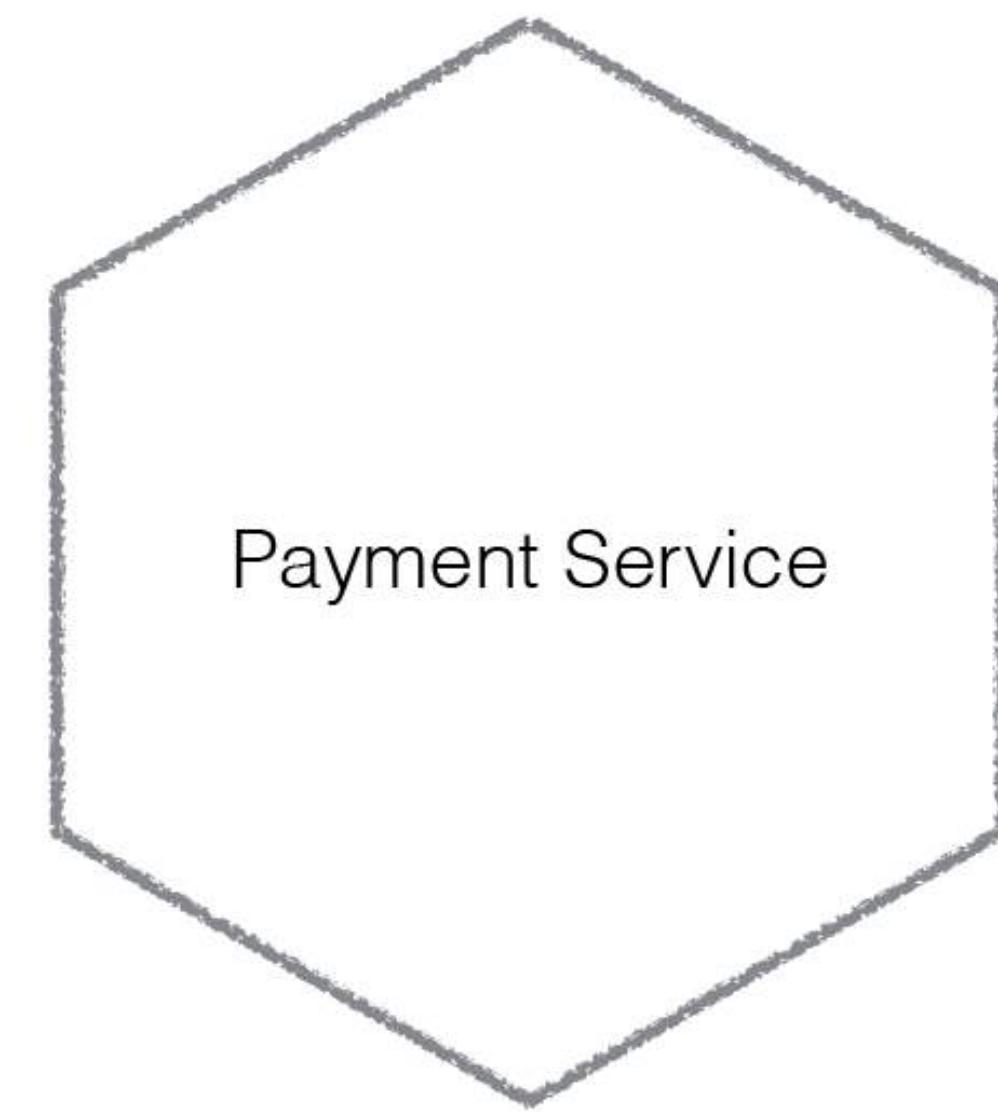
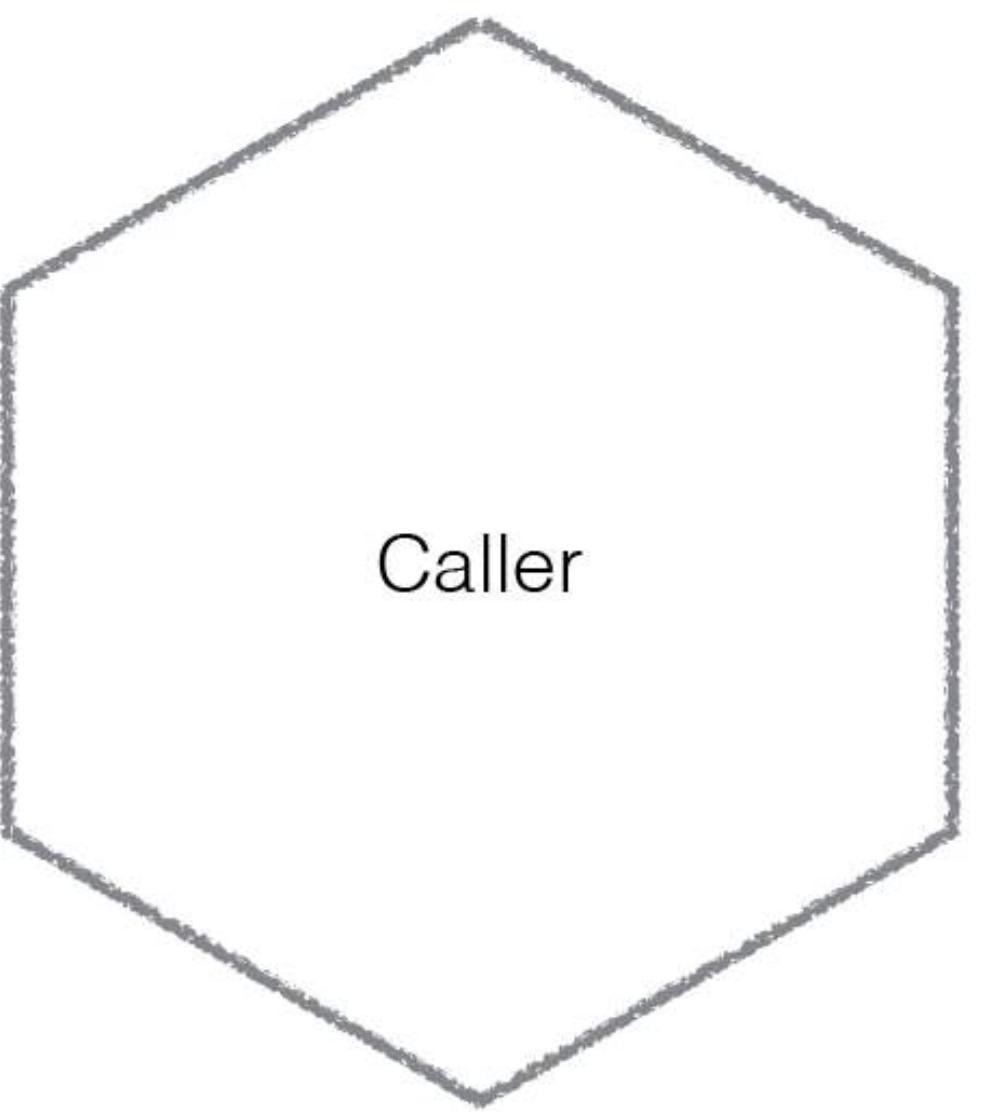
382-401

I have long felt that, because it was posed as a cute problem about philosophers seated around a table, Dijkstra's dining philosopher's problem received much more attention than it deserves. (For example, it has probably received more attention in the theory community than the readers/writers problem, which illustrates the same principles and has much more practical importance.) I believed that the problem introduced in [41] was very important and deserved the attention of computer scientists. The popularity of the dining philosophers problem taught me that the best way to attract attention to a problem is to present it in terms of a story.

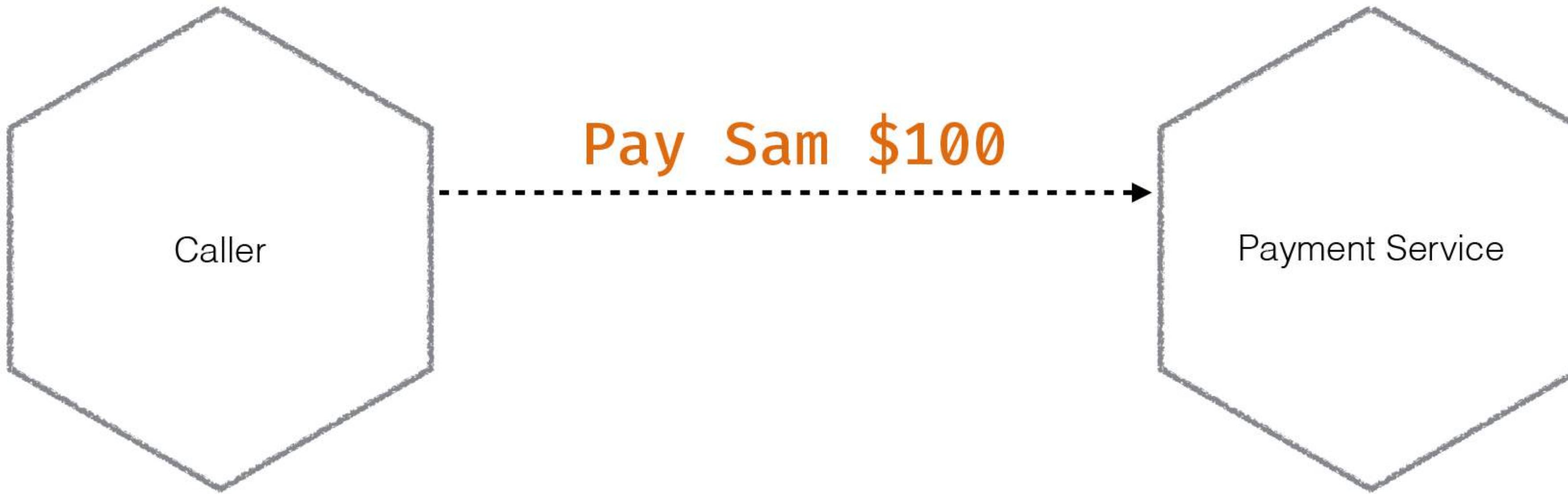
There is a problem in distributed computing that is sometimes called the Chinese Generals Problem, in which two generals have to come to a common agreement on whether to attack or retreat, but can communicate only by sending messengers who might never arrive. I stole the idea of the generals and posed the problem in terms of a group of generals, some of whom may be traitors, who have to reach a common decision. I wanted to assign the generals a nationality that would not offend any readers. At the time, Albania was a completely closed society, and I felt it unlikely that there would be any Albanians around to object, so the original title of this paper was The Albanian Generals Problem. Jack Goldberg was smart enough to realize that there were Albanians in the world outside Albania, and Albania might not always be a black hole, so he suggested that I find another name. The obviously more appropriate Byzantine generals then occurred to me.

<http://research.microsoft.com/en-us/um/people/lamport/pubs/byz.pdf>

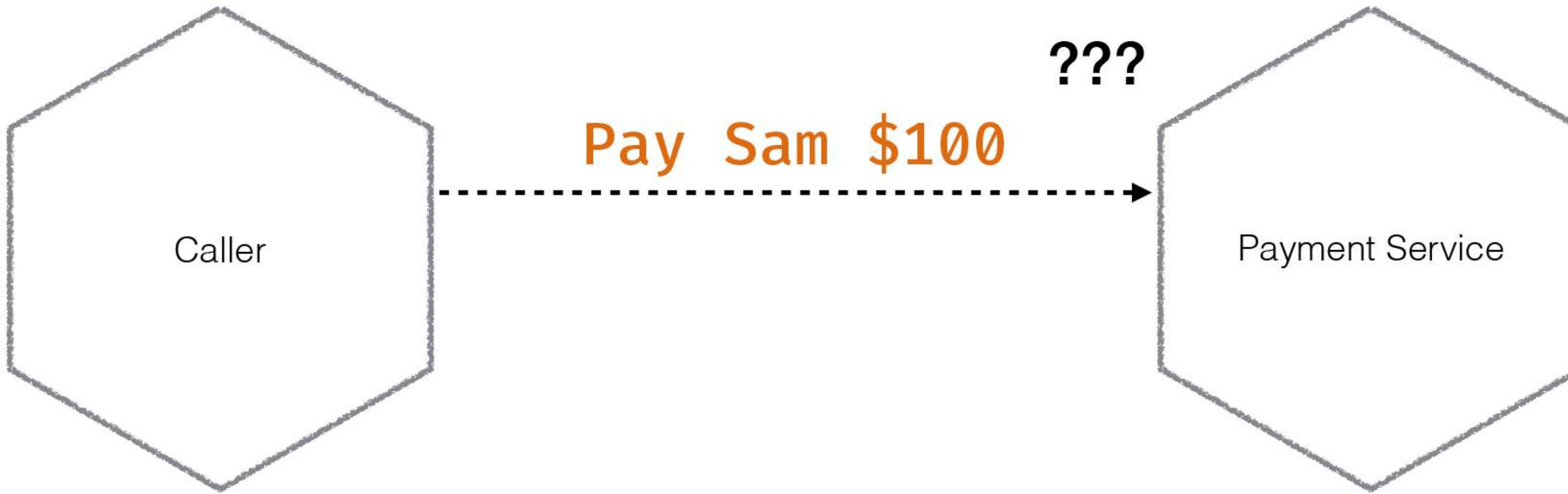
ONCE OR MORE DELIVERY



ONCE OR MORE DELIVERY



ONCE OR MORE DELIVERY



ONCE OR MORE DELIVERY



ONCE OR MORE DELIVERY

Consumer intent:
Sam should have \$100 more



ONCE OR MORE DELIVERY

Consumer intent:
Sam should have \$100 more

Actual result:
Sam gets \$200!!!



ONCE OR MORE DELIVERY

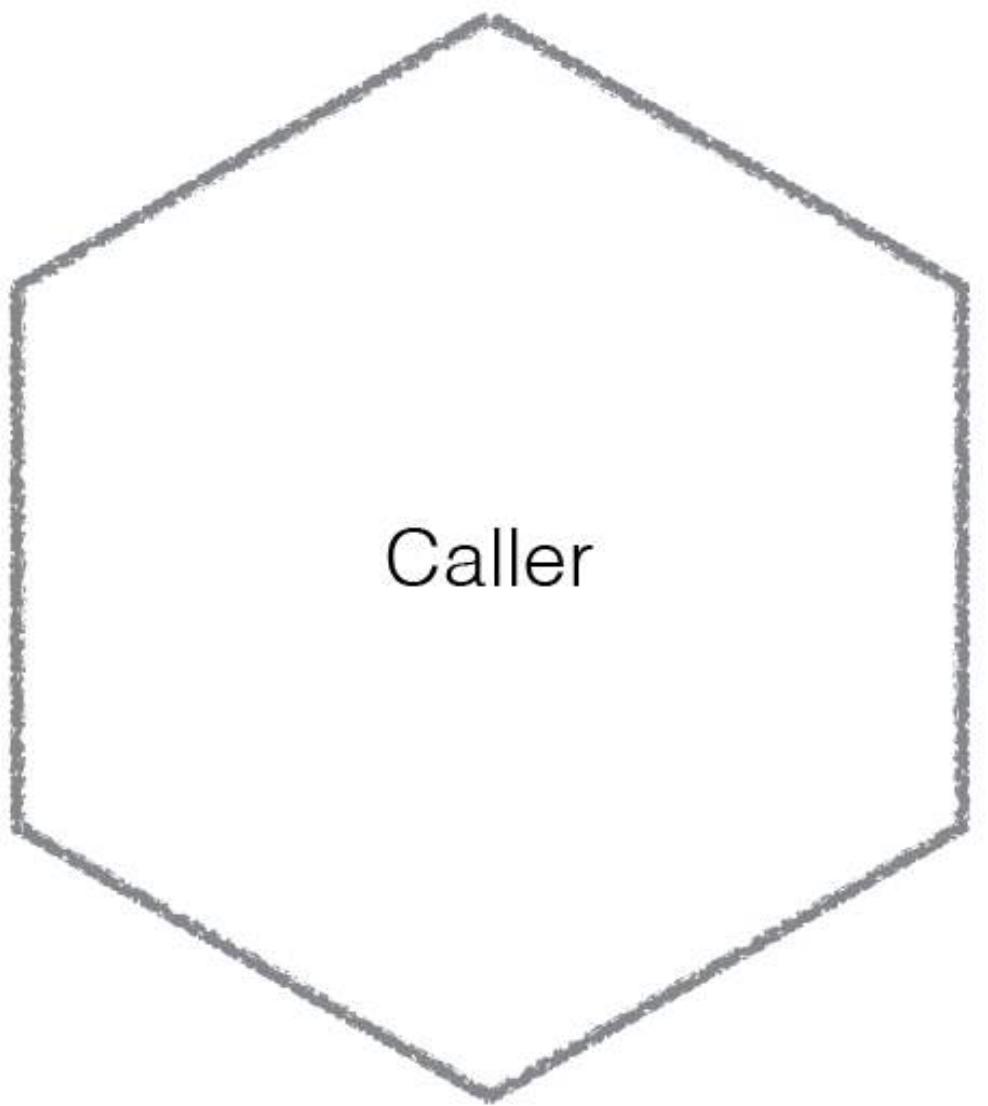
Consumer intent:
Sam should have \$100 more

Actual result:
Sam gets \$200!!!

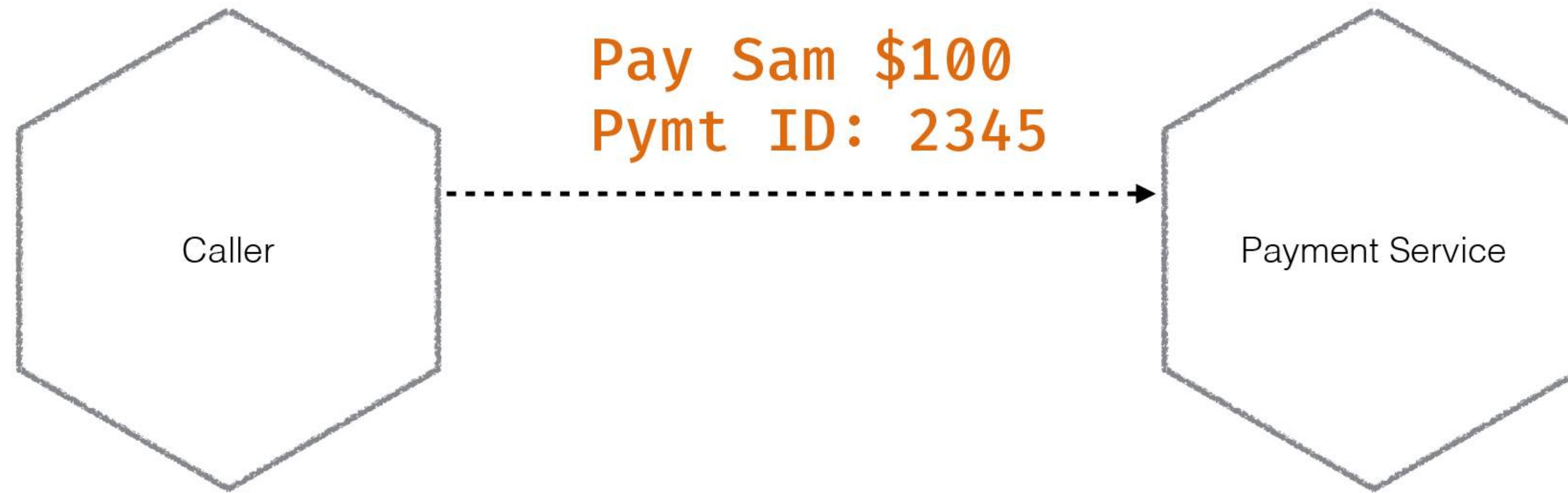


Exactly once delivery is ...erm...tricky

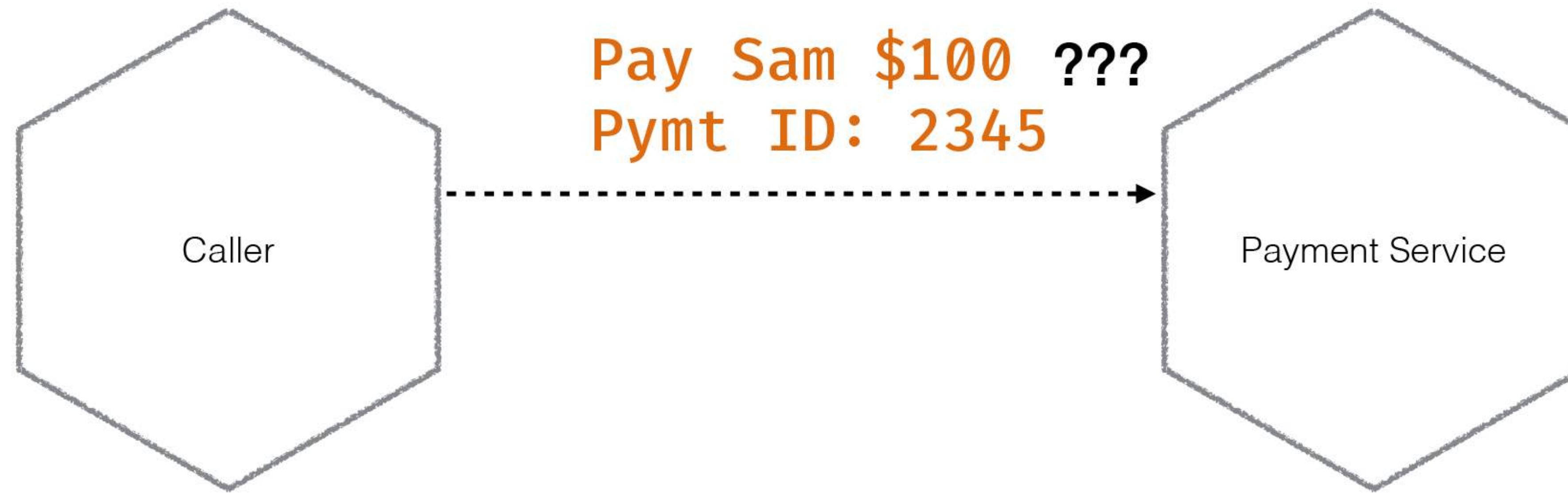
IDEMPOTENCY



IDEMPOTENCY



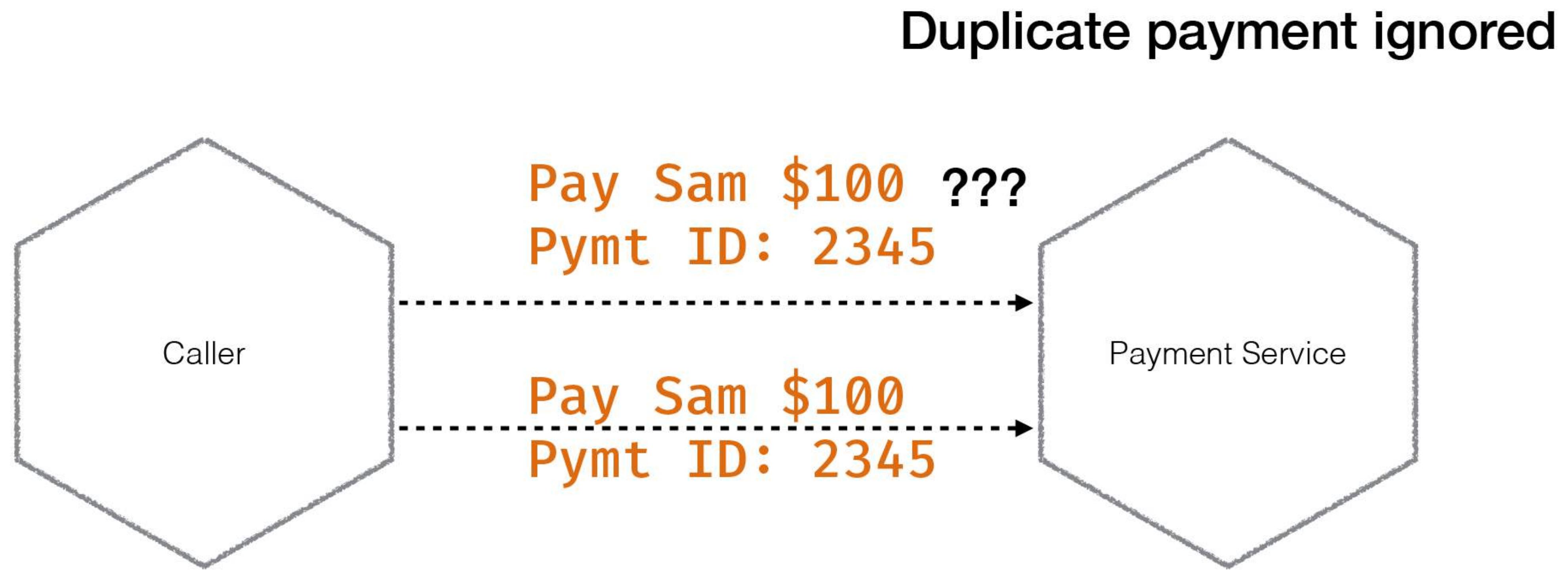
IDEMPOTENCY



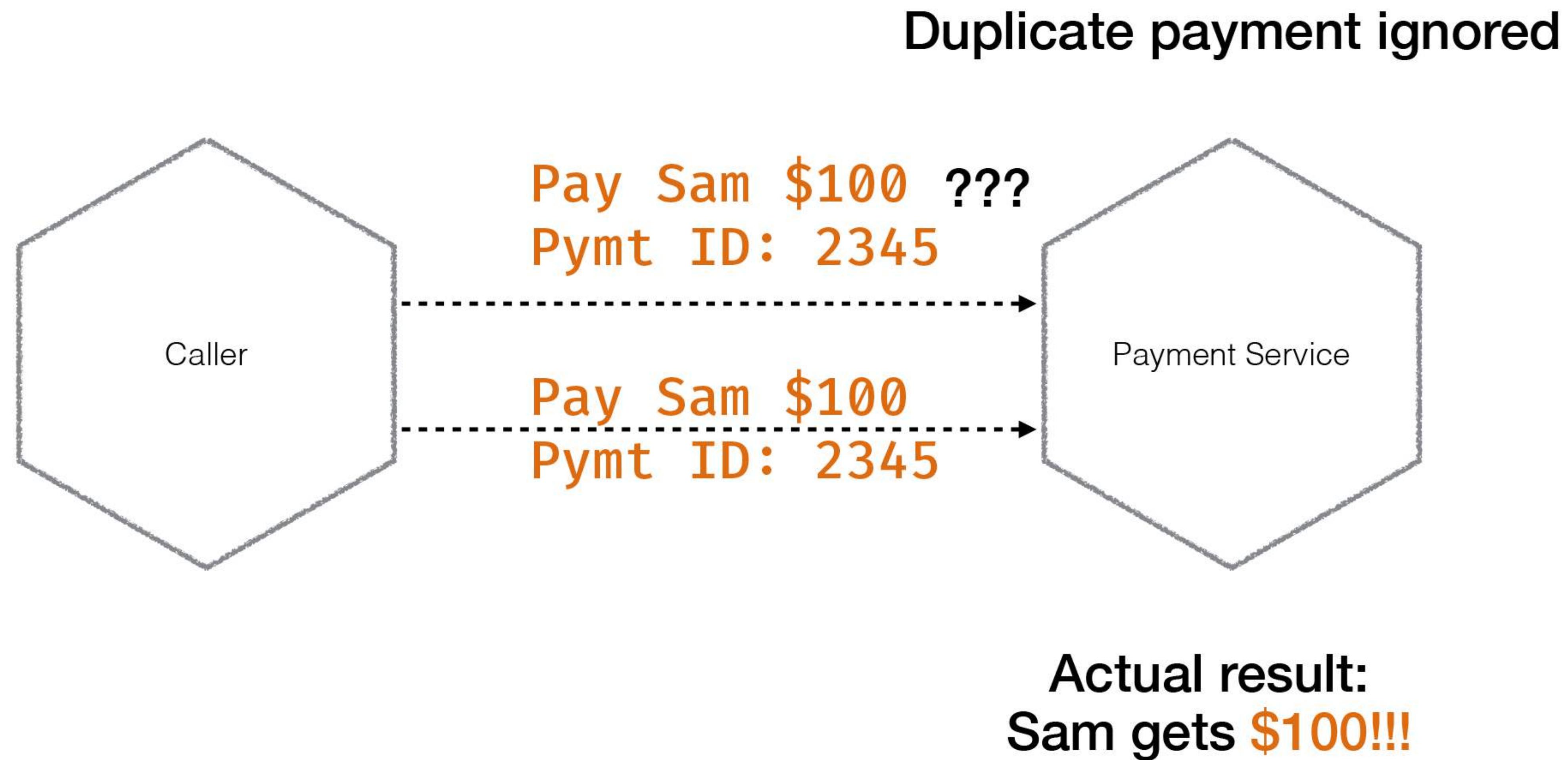
IDEMPOTENCY



IDEMPOTENCY



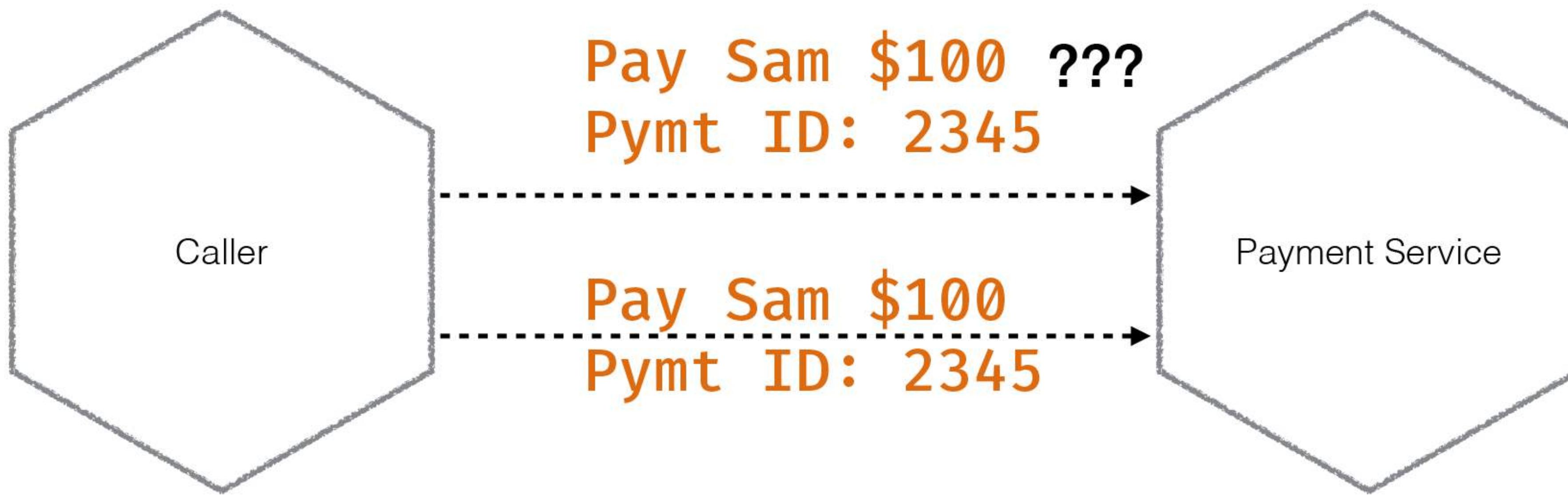
IDEMPOTENCY



IDEMPOTENCY

Idempotency means we don't need to worry about "exactly once" delivery

Duplicate payment ignored

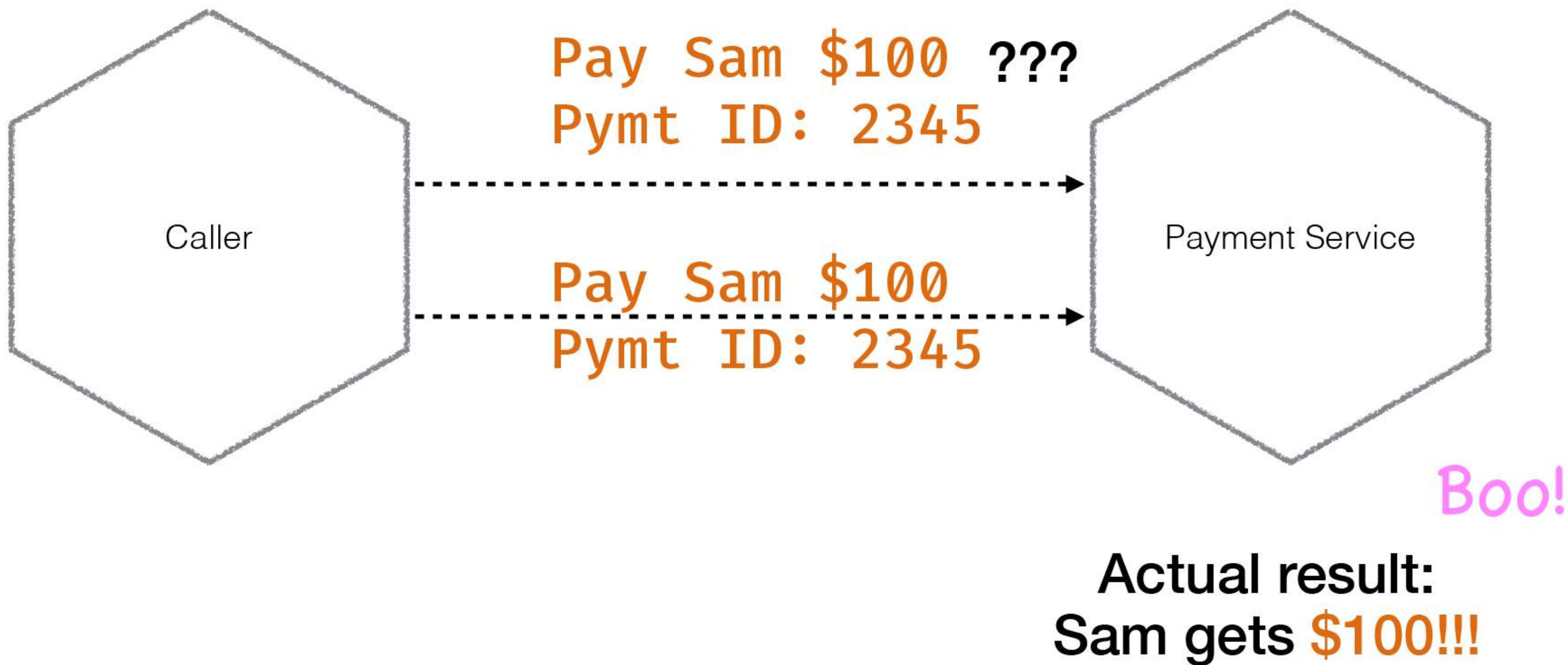


Actual result:
Sam gets **\$100!!!**

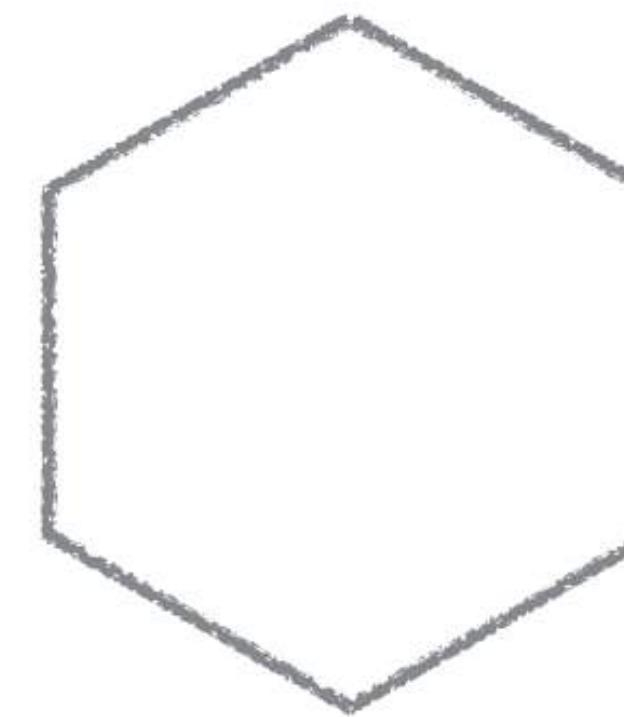
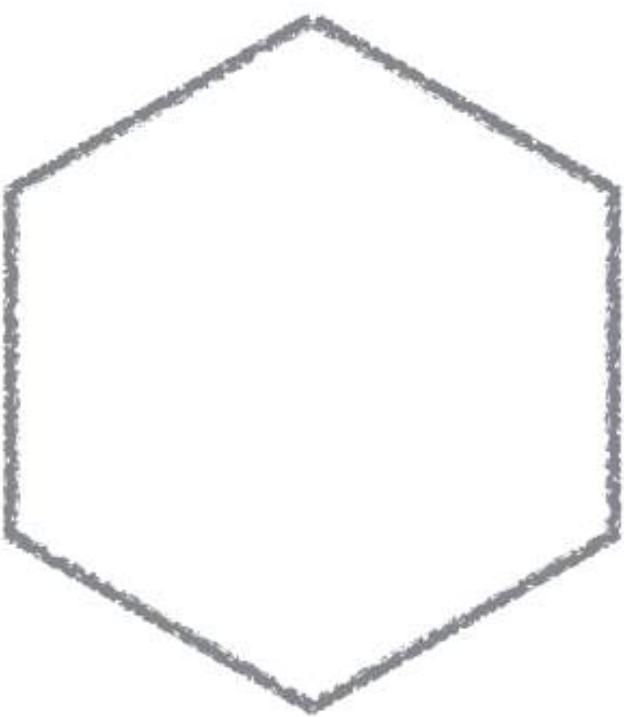
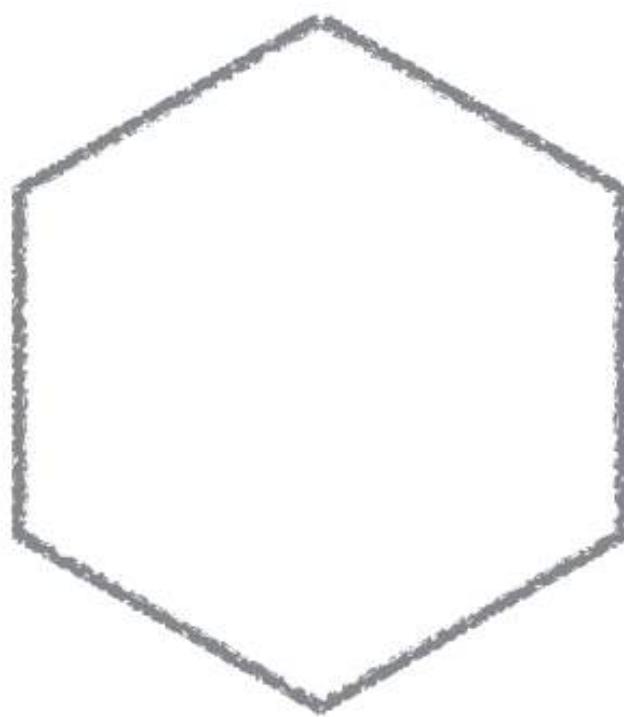
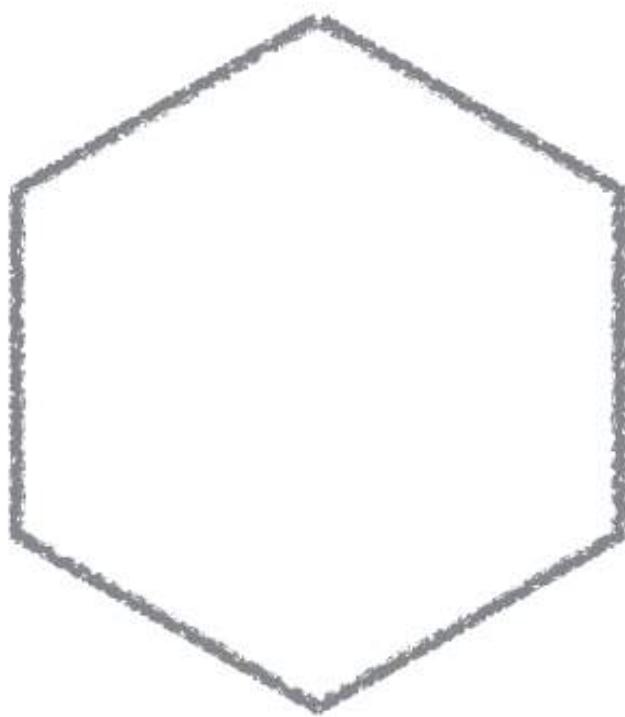
IDEMPOTENCY

Idempotency means we don't need to worry about "exactly once" delivery

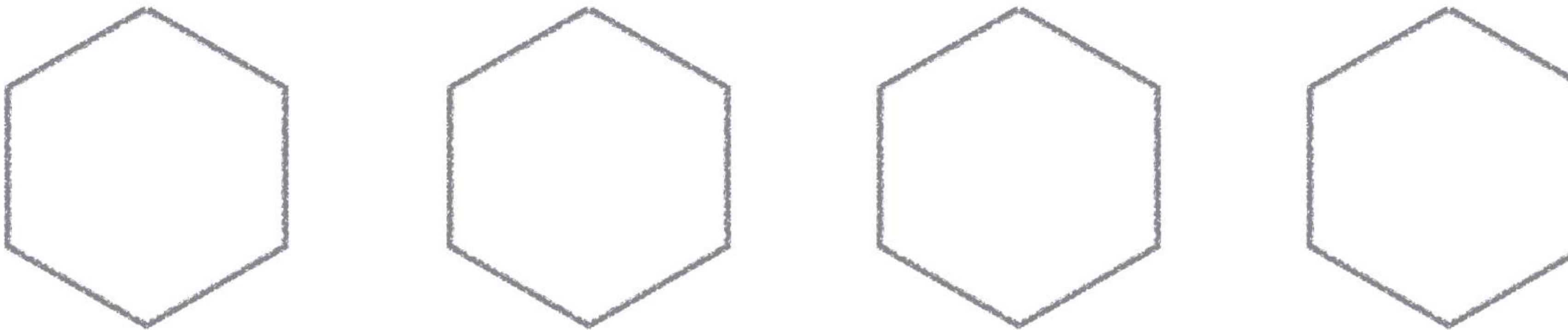
Duplicate payment ignored



ENTERPRISE SERVICE BUS!

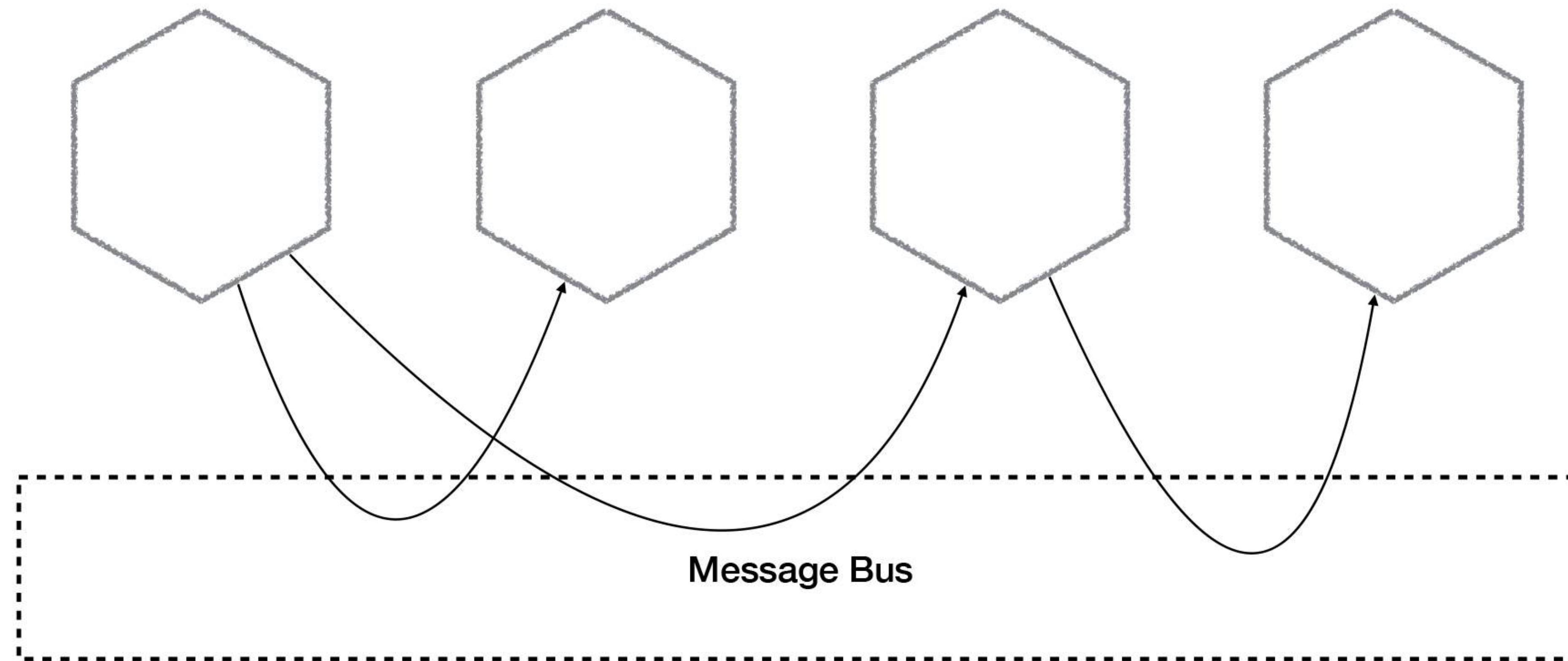


ENTERPRISE SERVICE BUS!

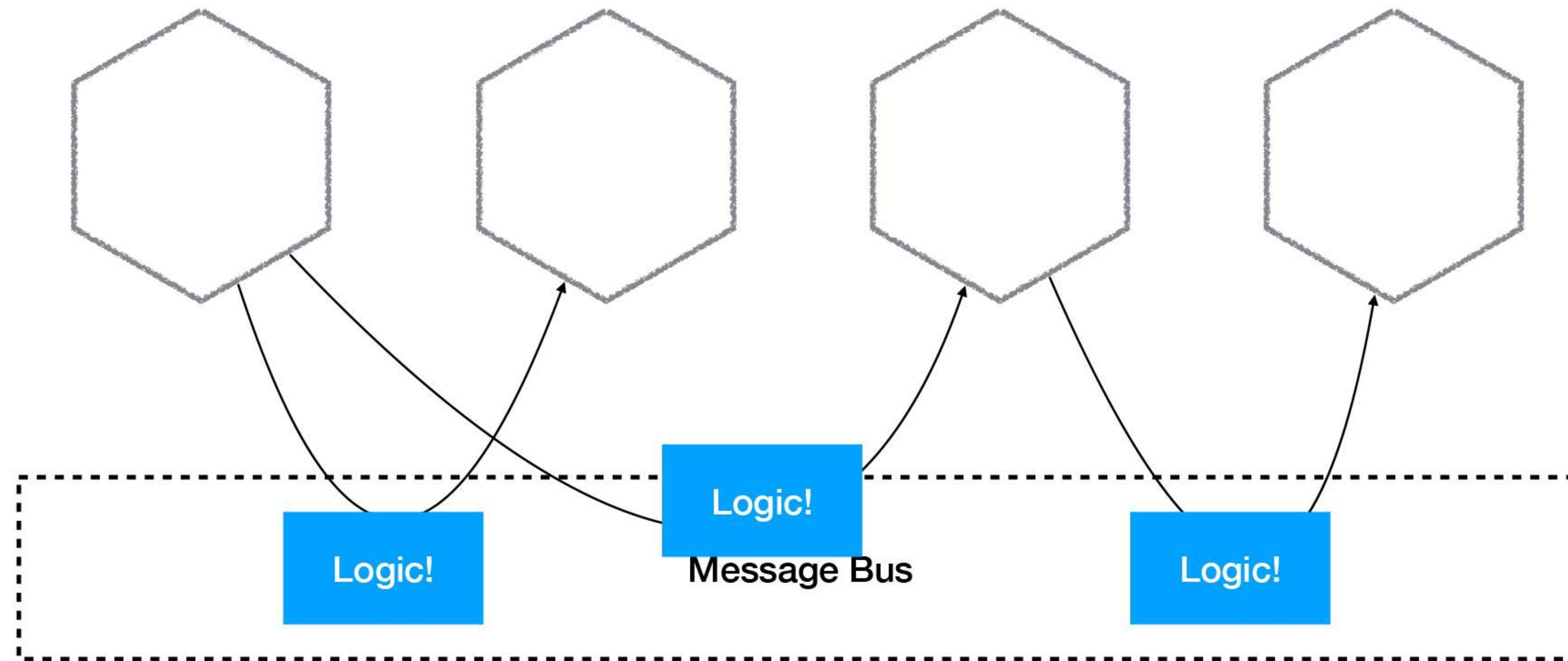


Message Bus

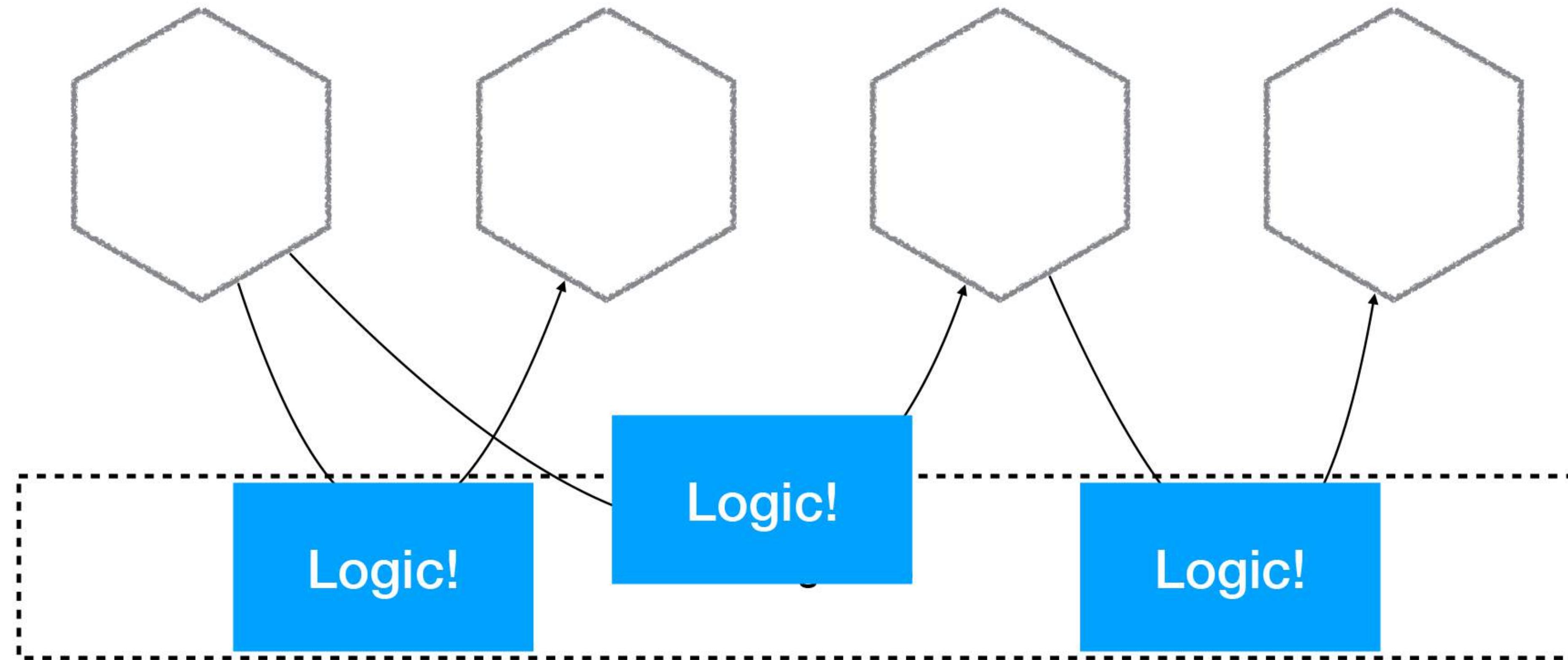
ENTERPRISE SERVICE BUS!



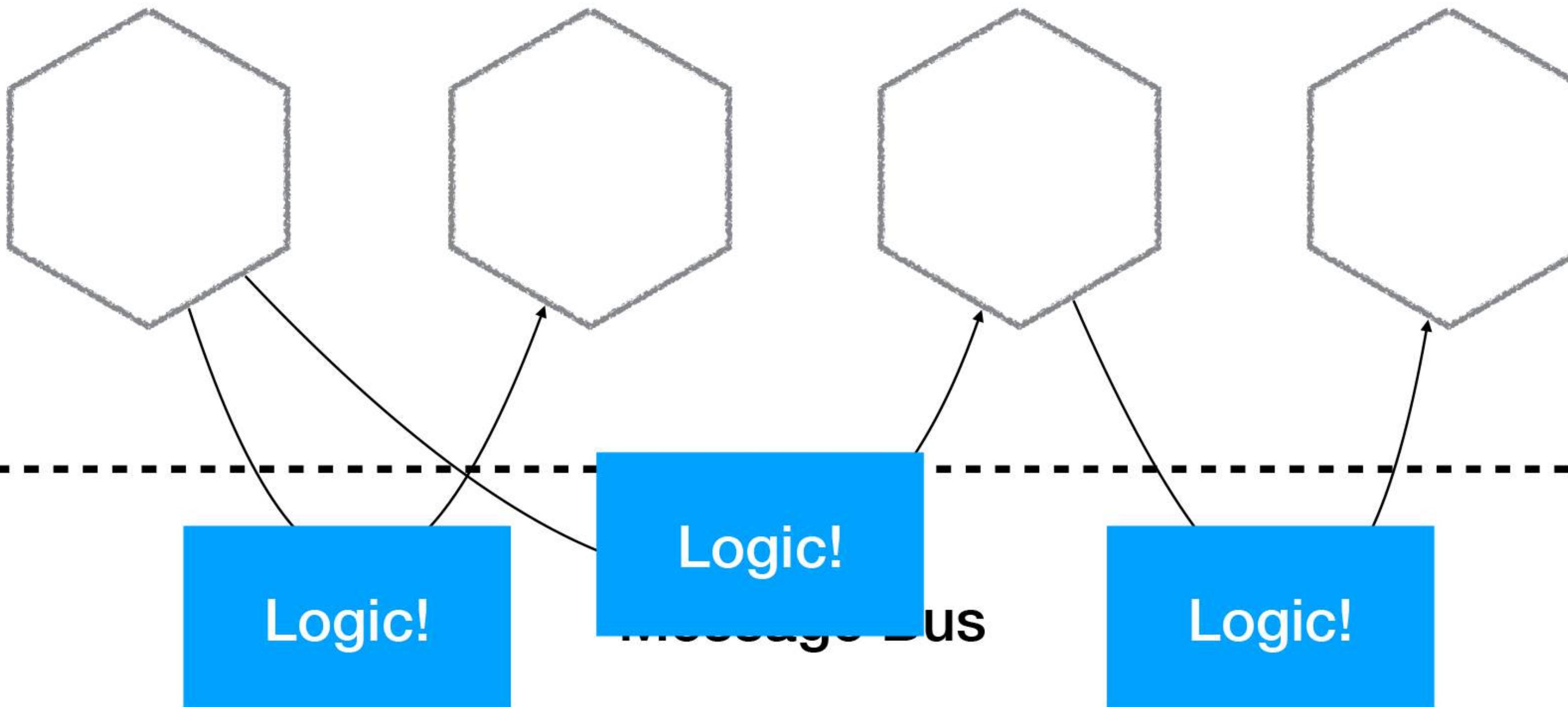
ENTERPRISE SERVICE BUS!



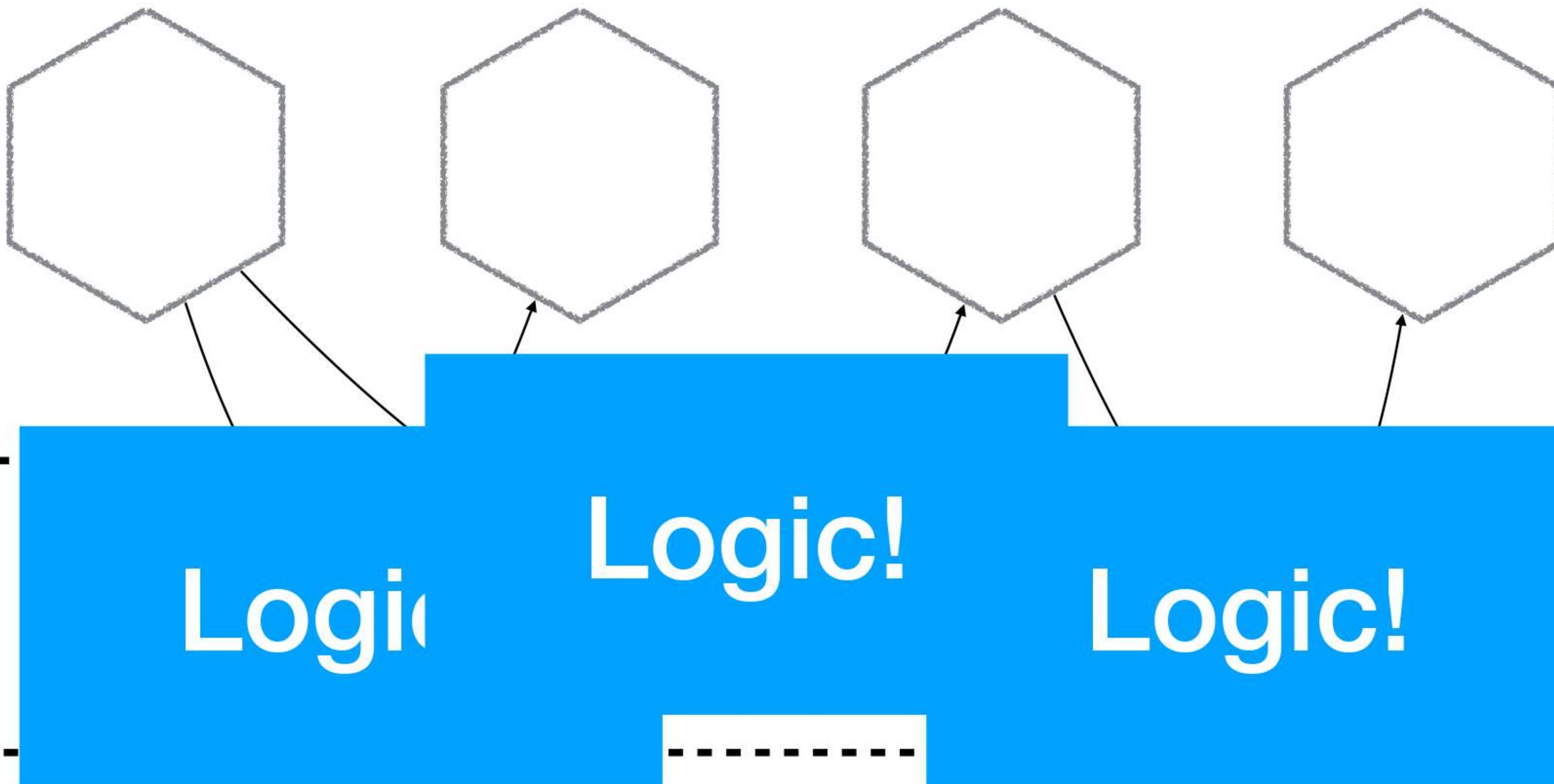
ENTERPRISE SERVICE BUS!



ENTERPRISE SERVICE BUS!



ENTERPRISE SERVICE BUS!



**Keep your pipes dumb, and your
endpoints smart**

One other thing. Message brokers aren't simple to build, nor are they simply to run







<http://kafka.apache.org>

High Volume



<http://kafka.apache.org>

High Volume

Message Permanence



<http://kafka.apache.org>

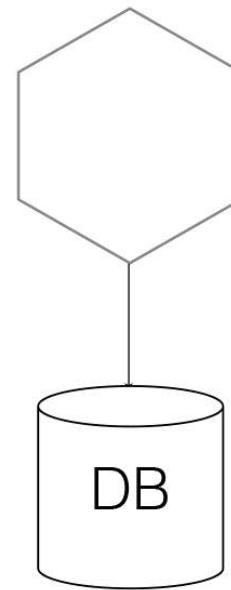
High Volume

Message Permanence

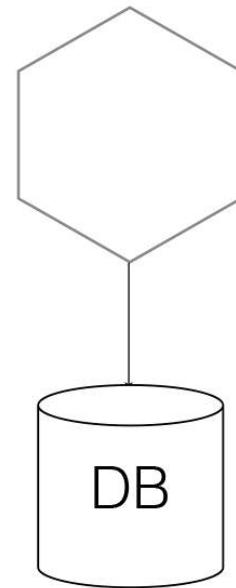


<http://kafka.apache.org>

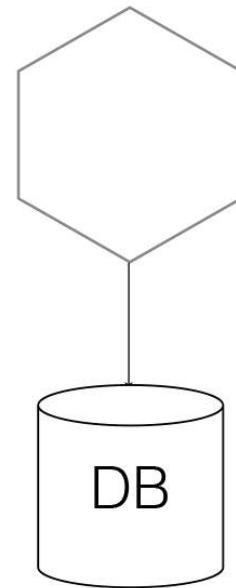
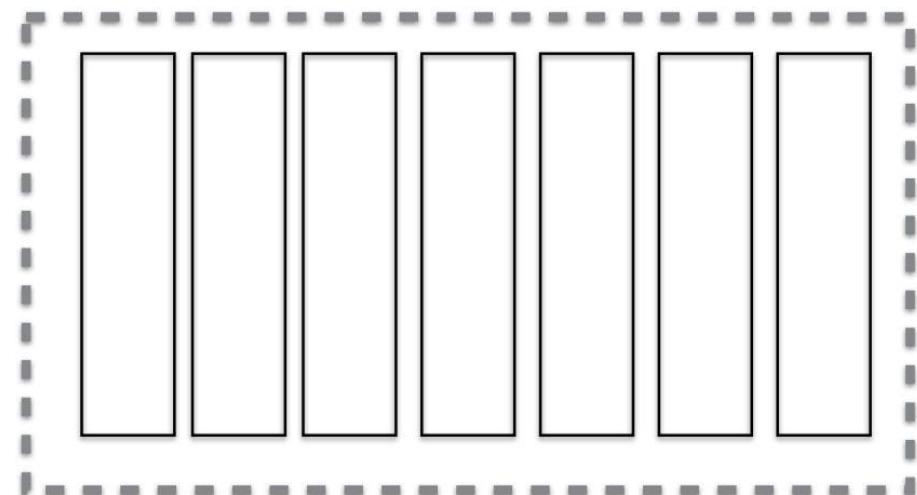
TYPICAL MESSAGE BROKERS



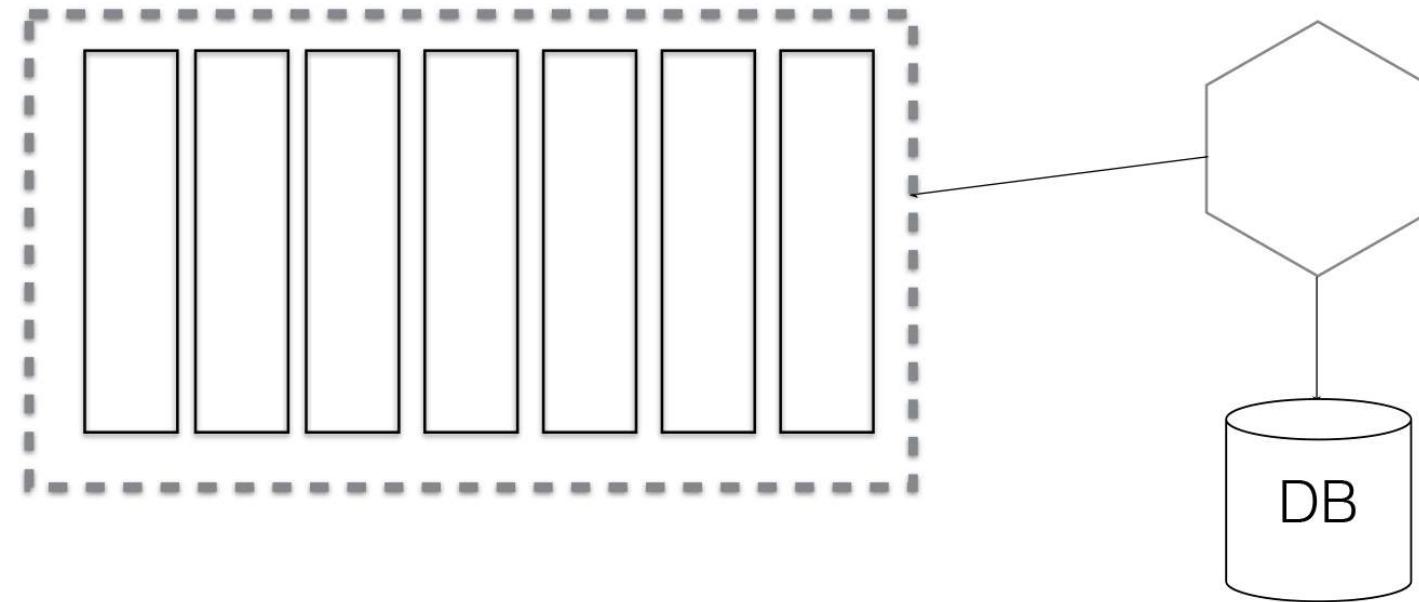
TYPICAL MESSAGE BROKERS



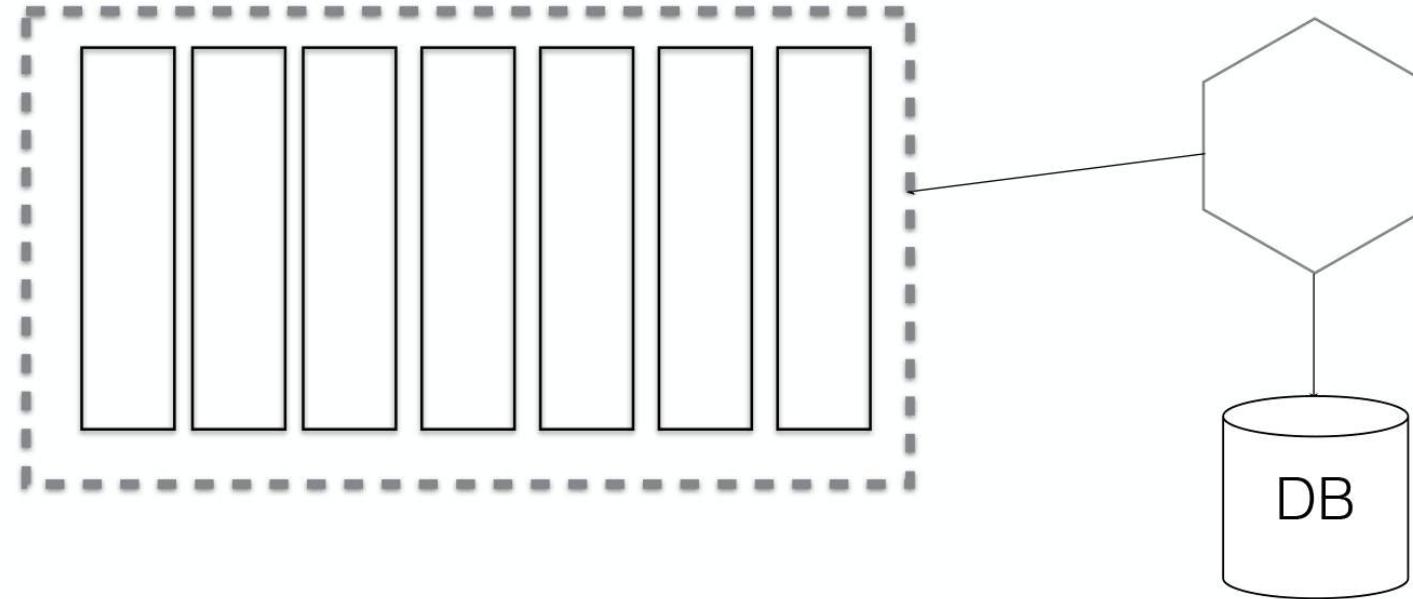
TYPICAL MESSAGE BROKERS



TYPICAL MESSAGE BROKERS



TYPICAL MESSAGE BROKERS

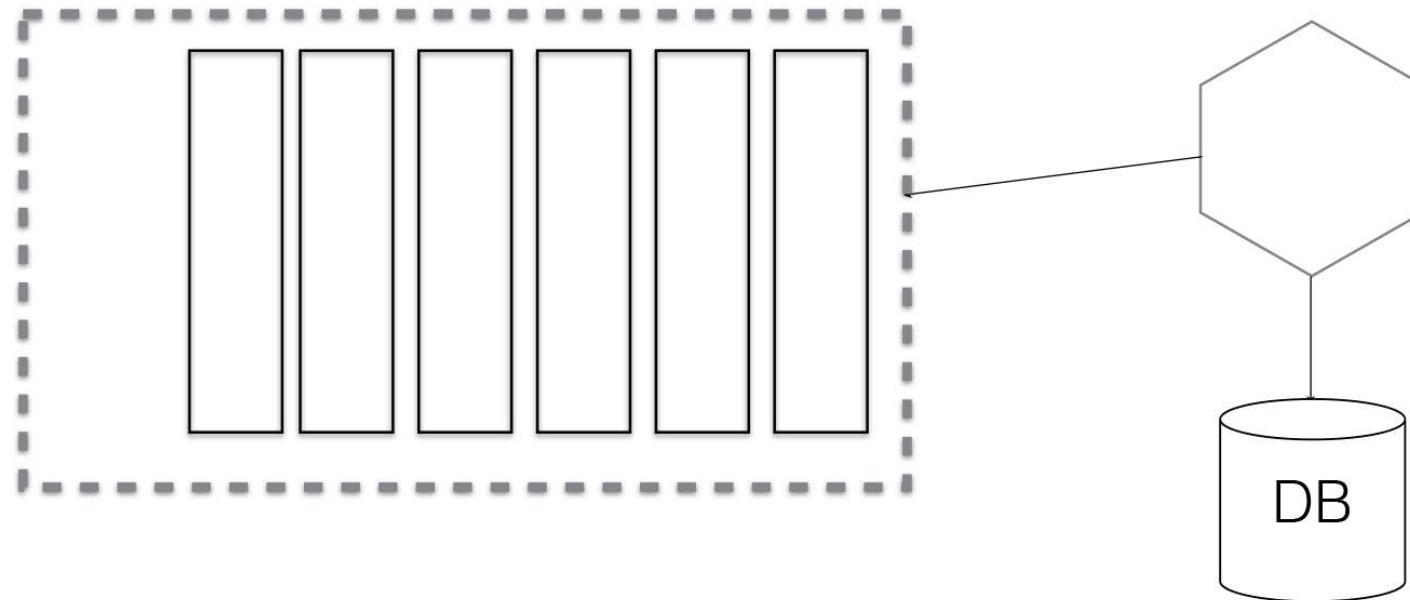


sam@samnewman.io

15 My Place, UK

No Email

TYPICAL MESSAGE BROKERS

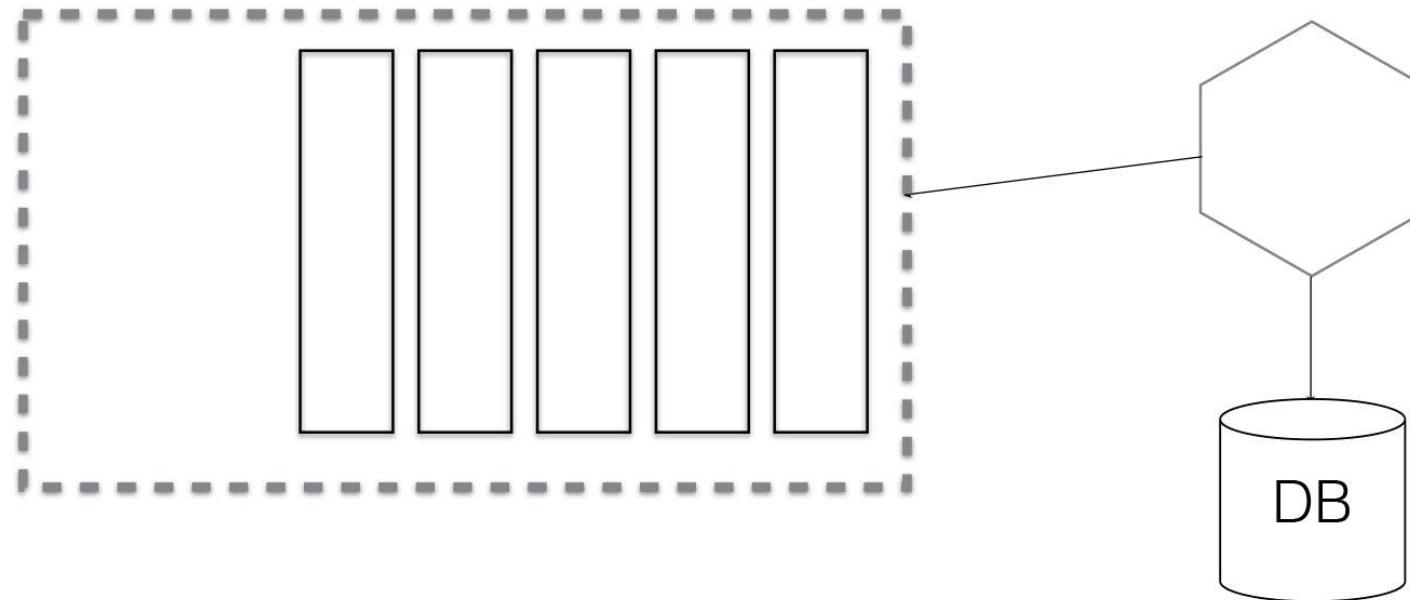


sam@samnewman.io

15 My Place, UK

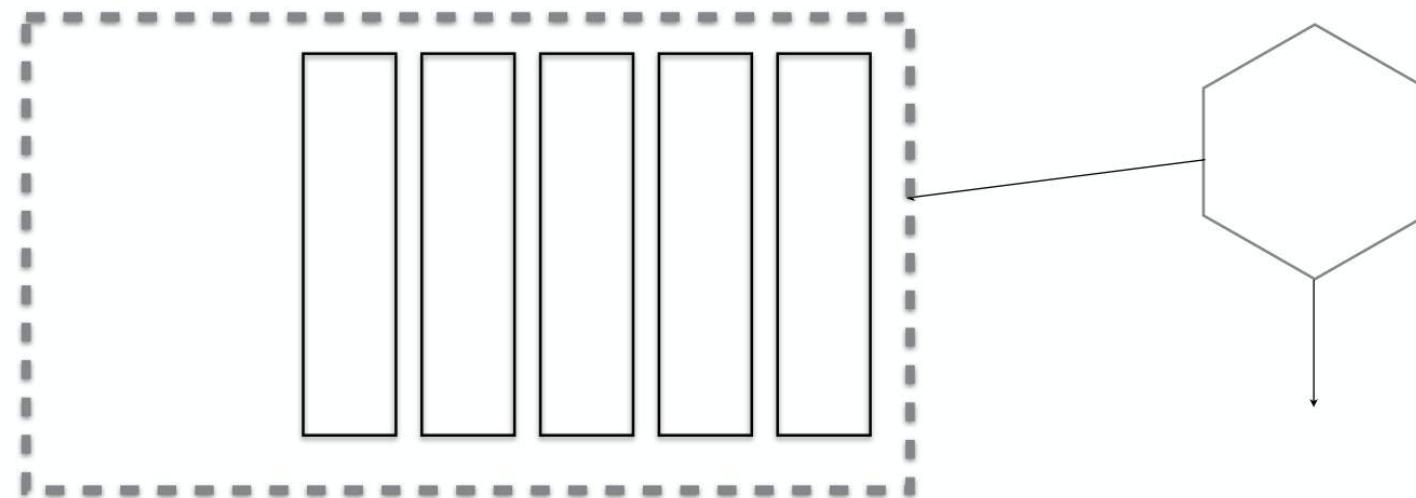
Email

TYPICAL MESSAGE BROKERS



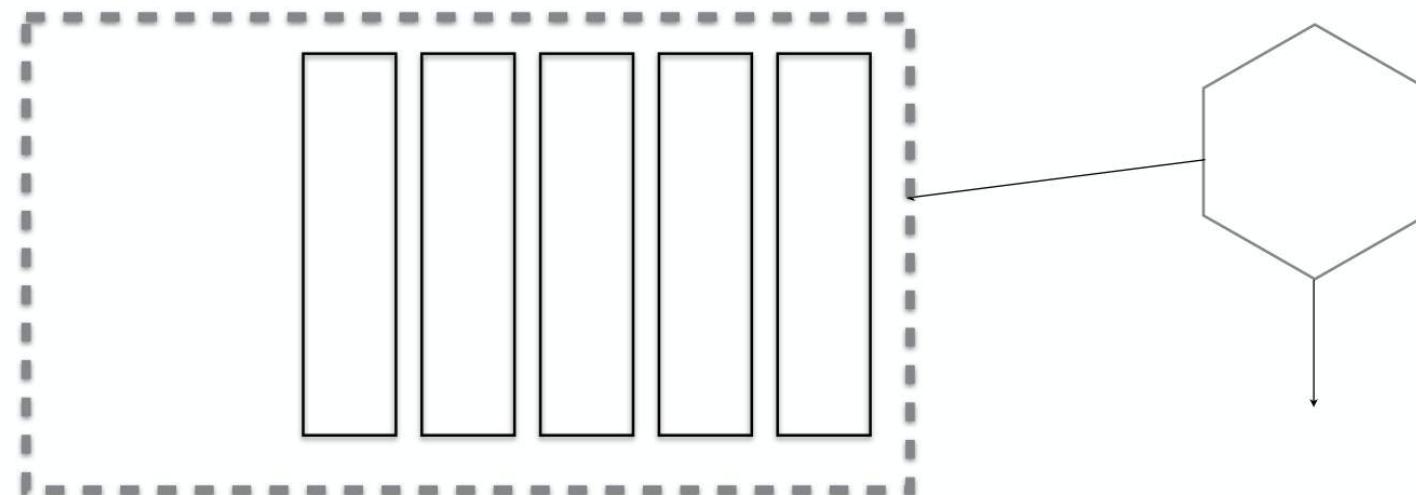
sam@gmail.com	15 My Place, UK	Email
----------------------	-----------------	--------------

TYPICAL MESSAGE BROKERS

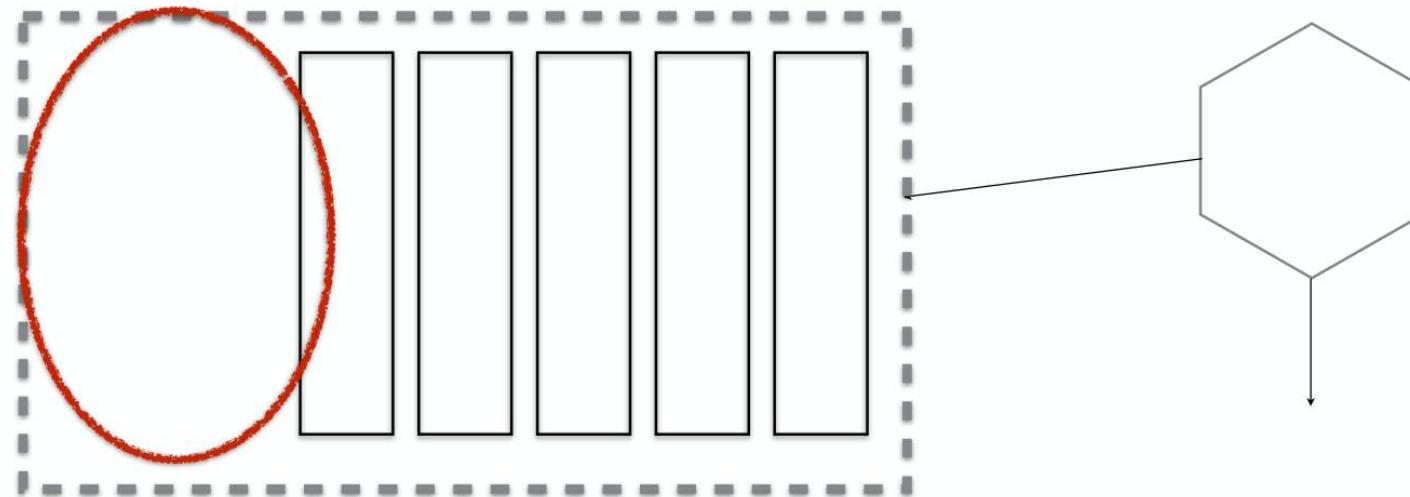


sam@gmail.com	15 My Place, UK	Email
----------------------	-----------------	--------------

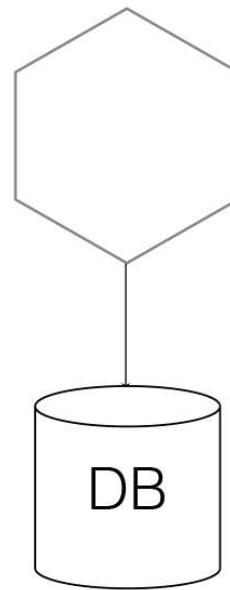
TYPICAL MESSAGE BROKERS



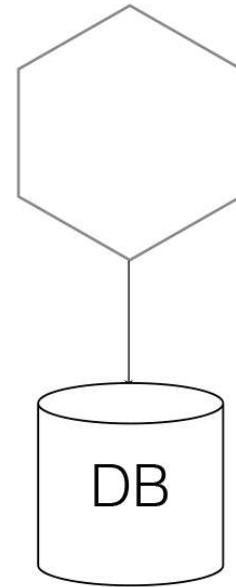
TYPICAL MESSAGE BROKERS



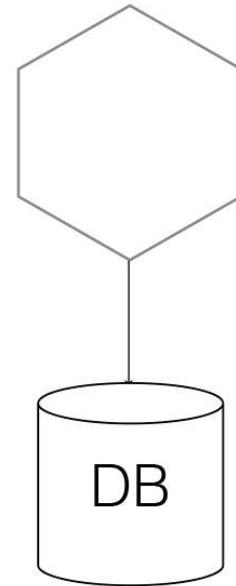
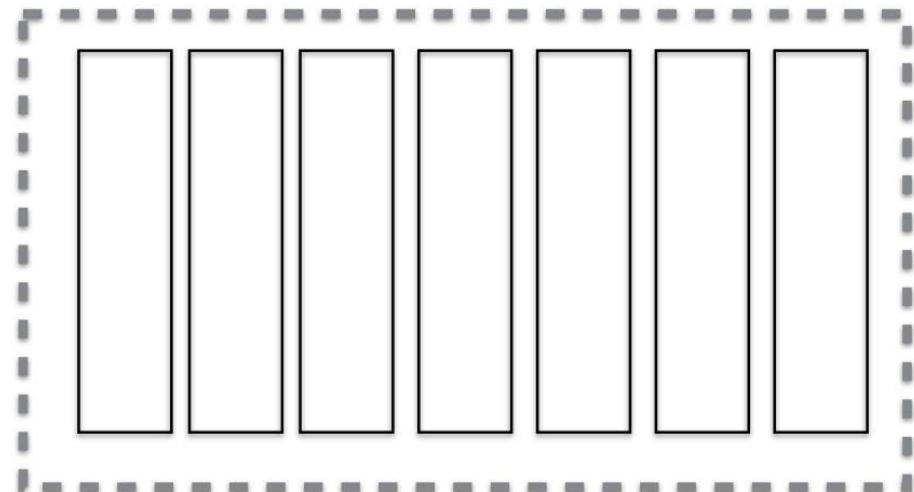
KAFKA - MESSAGE PERMANENCE



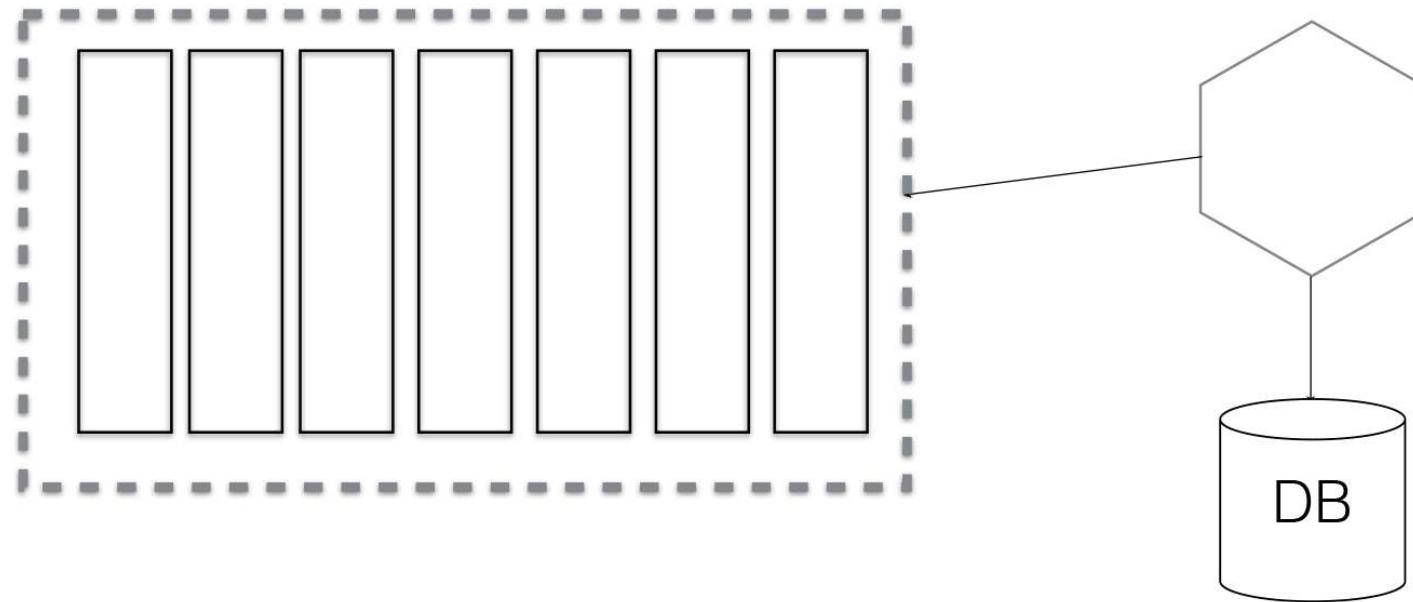
KAFKA - MESSAGE PERMANENCE



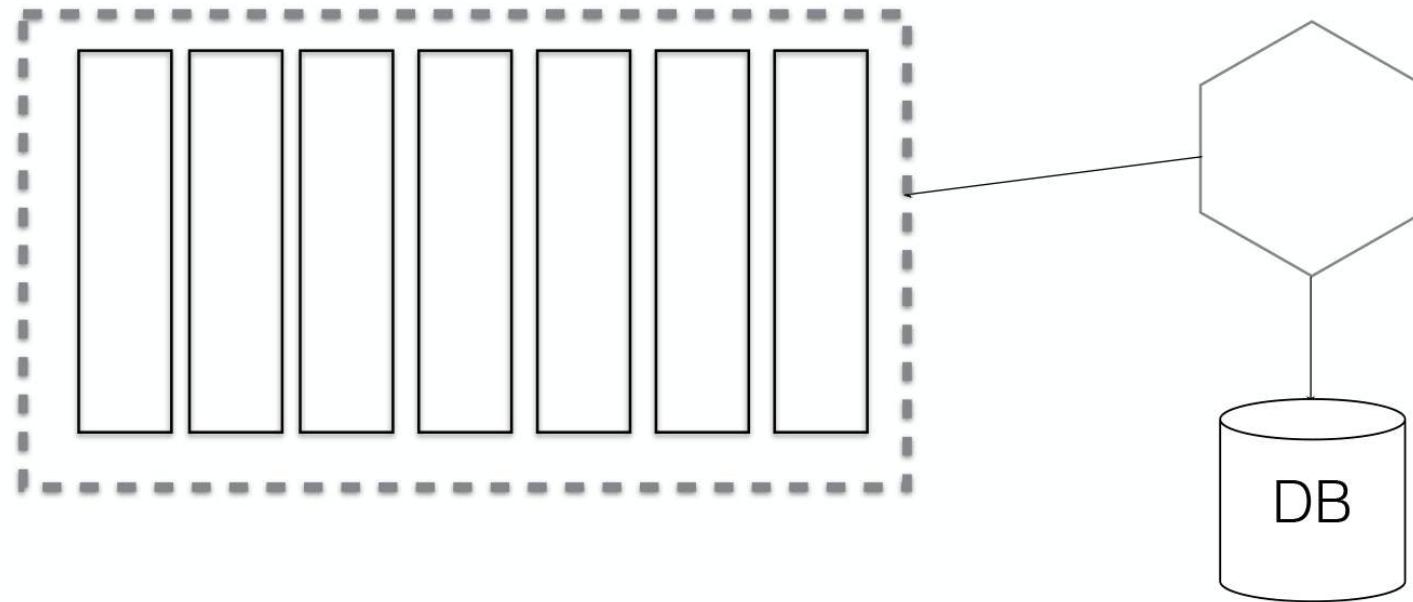
KAFKA - MESSAGE PERMANENCE



KAFKA - MESSAGE PERMANENCE



KAFKA - MESSAGE PERMANENCE

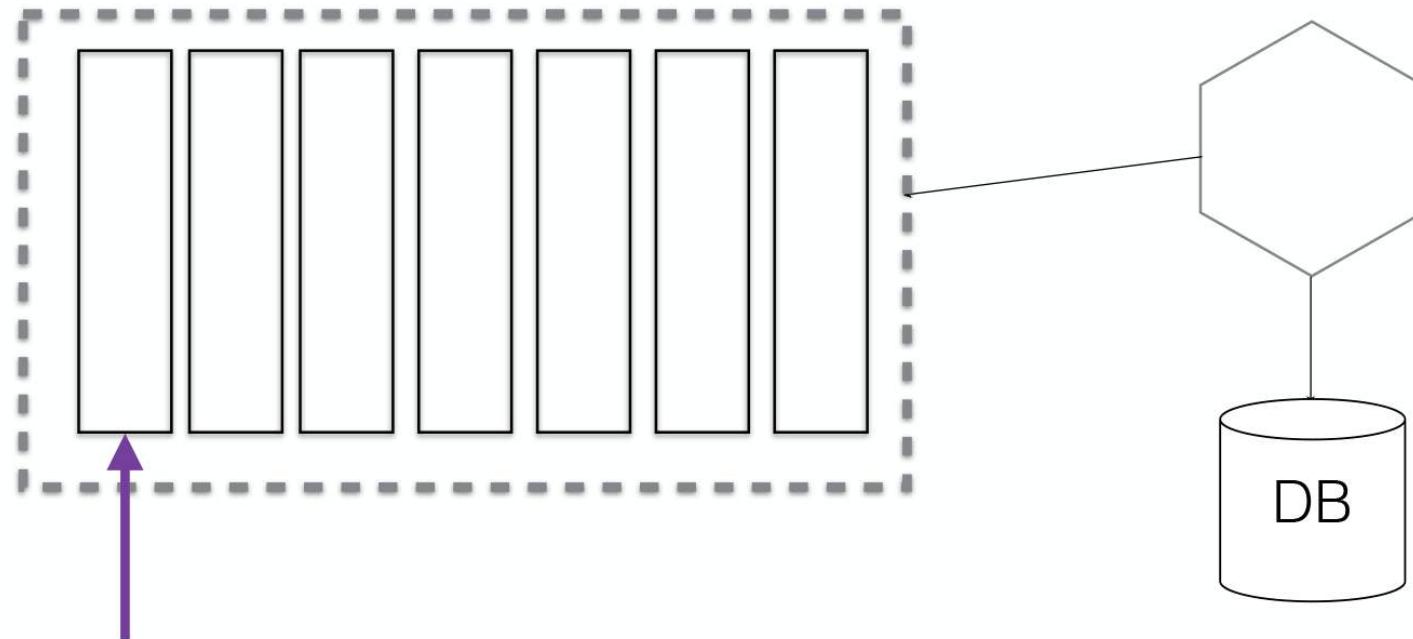


sam@samnewman.io

15 My Place, UK

No Email

KAFKA - MESSAGE PERMANENCE

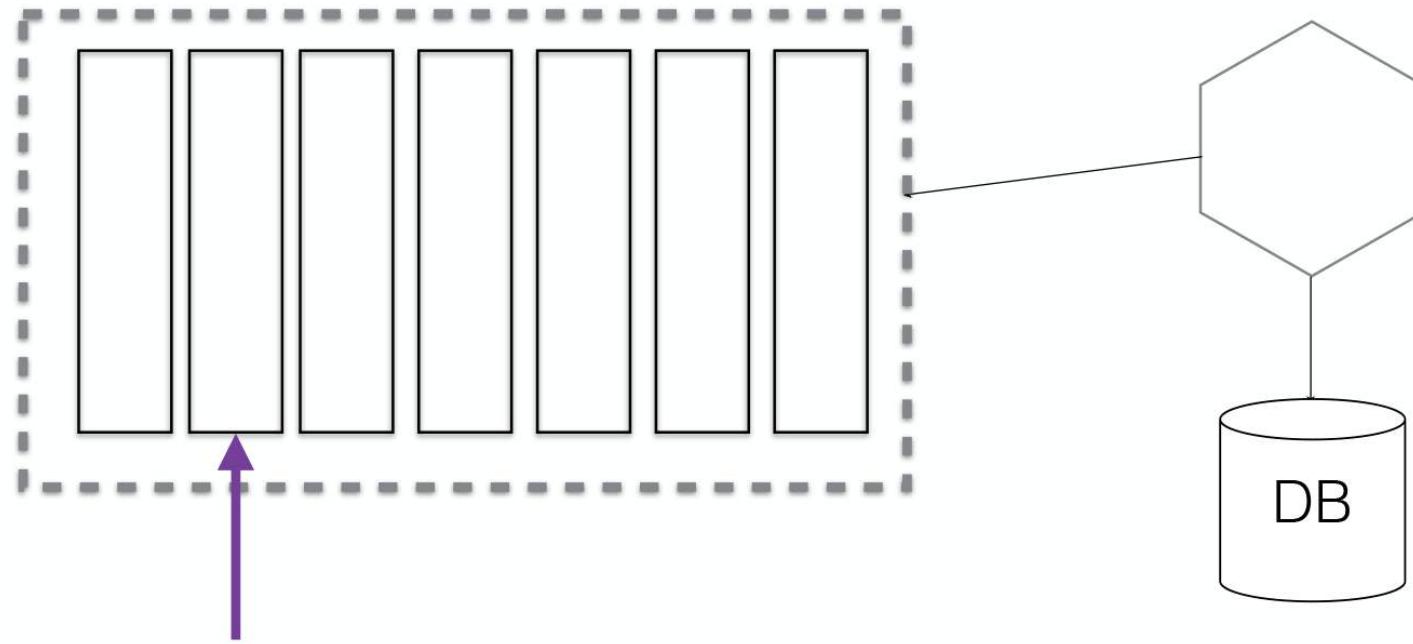


sam@samnewman.io

15 My Place, UK

No Email

KAFKA - MESSAGE PERMANENCE

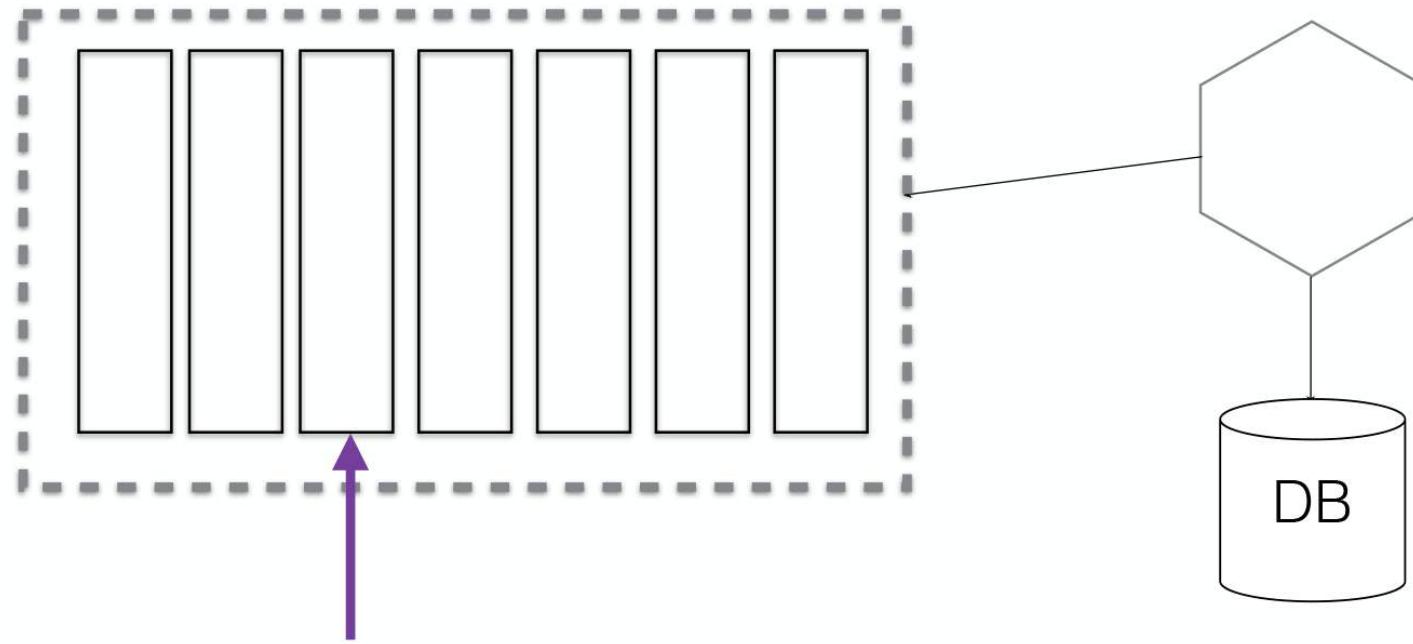


sam@samnewman.io

15 My Place, UK

Email

KAFKA - MESSAGE PERMANENCE

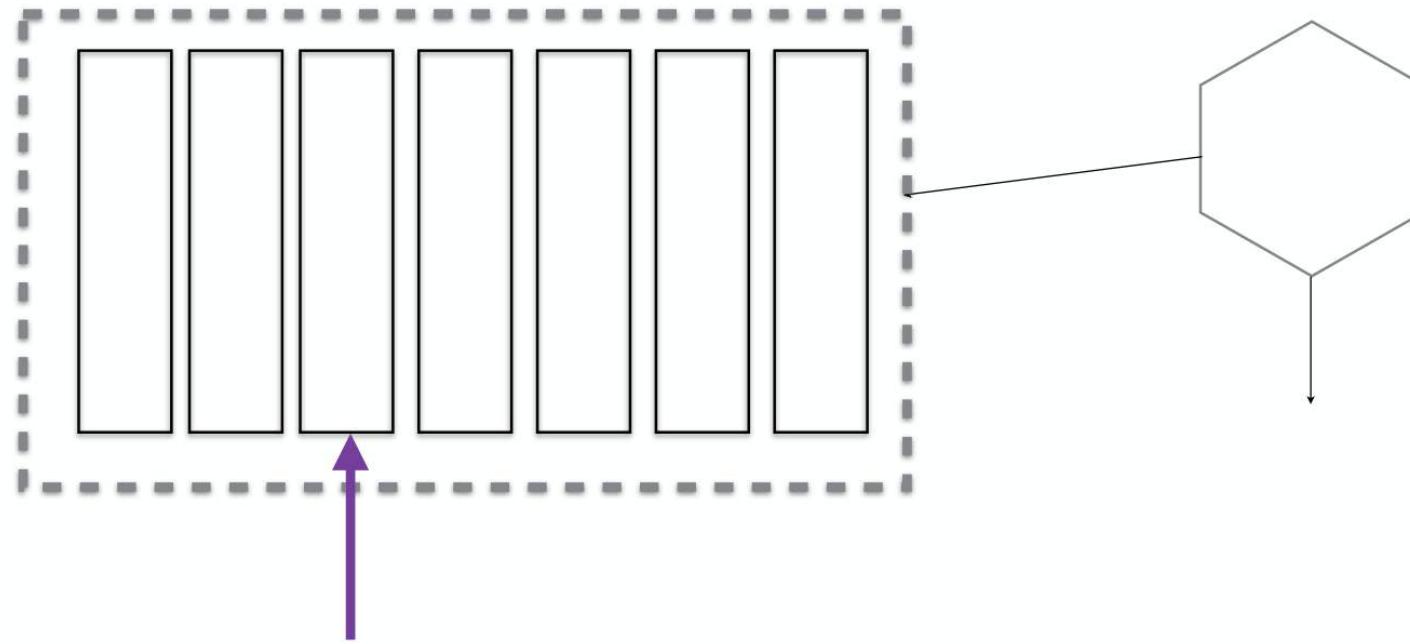


sam@gmail.com

15 My Place, UK

Email

KAFKA - MESSAGE PERMANENCE

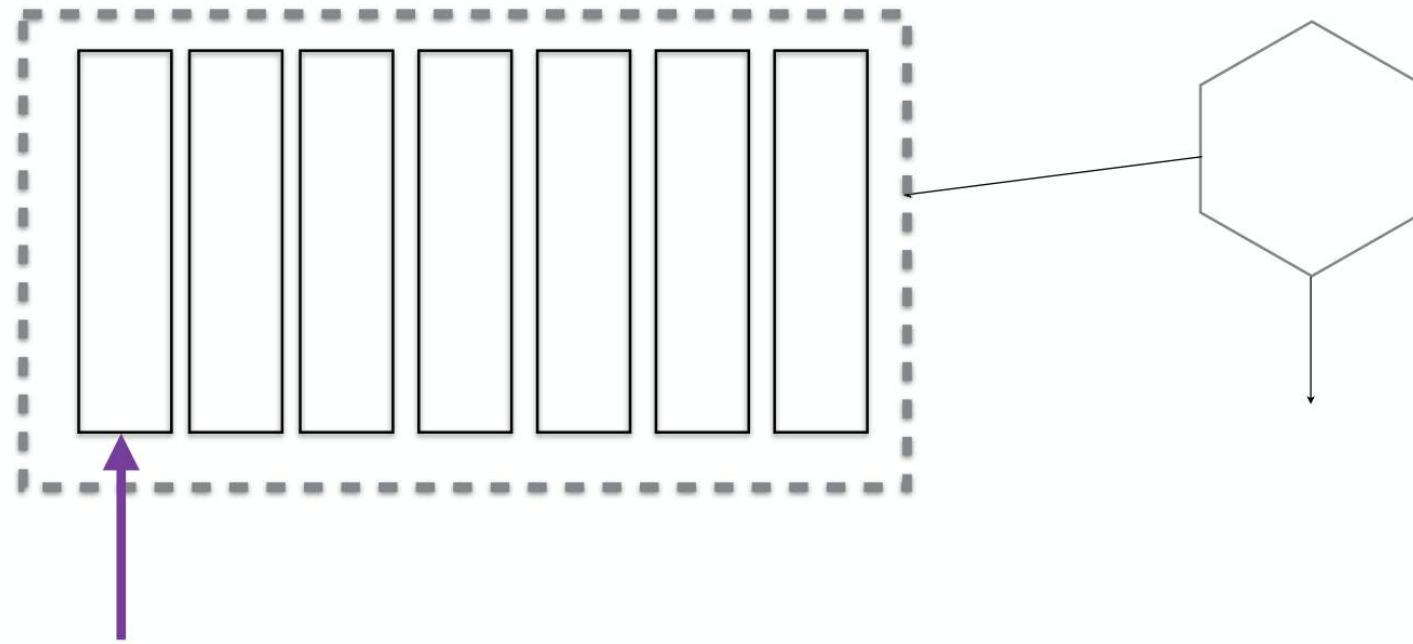


sam@gmail.com

15 My Place, UK

Email

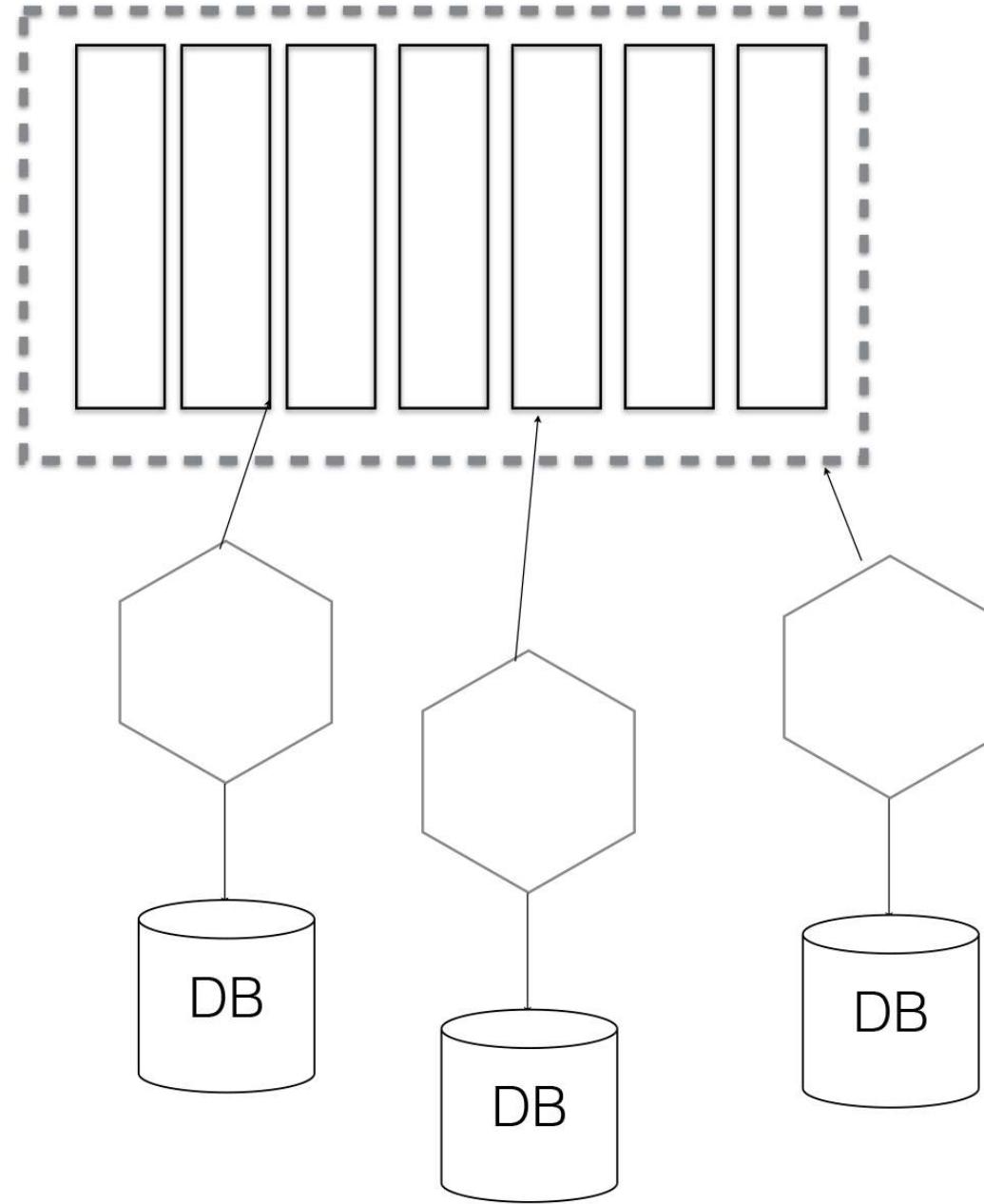
KAFKA - MESSAGE PERMANENCE



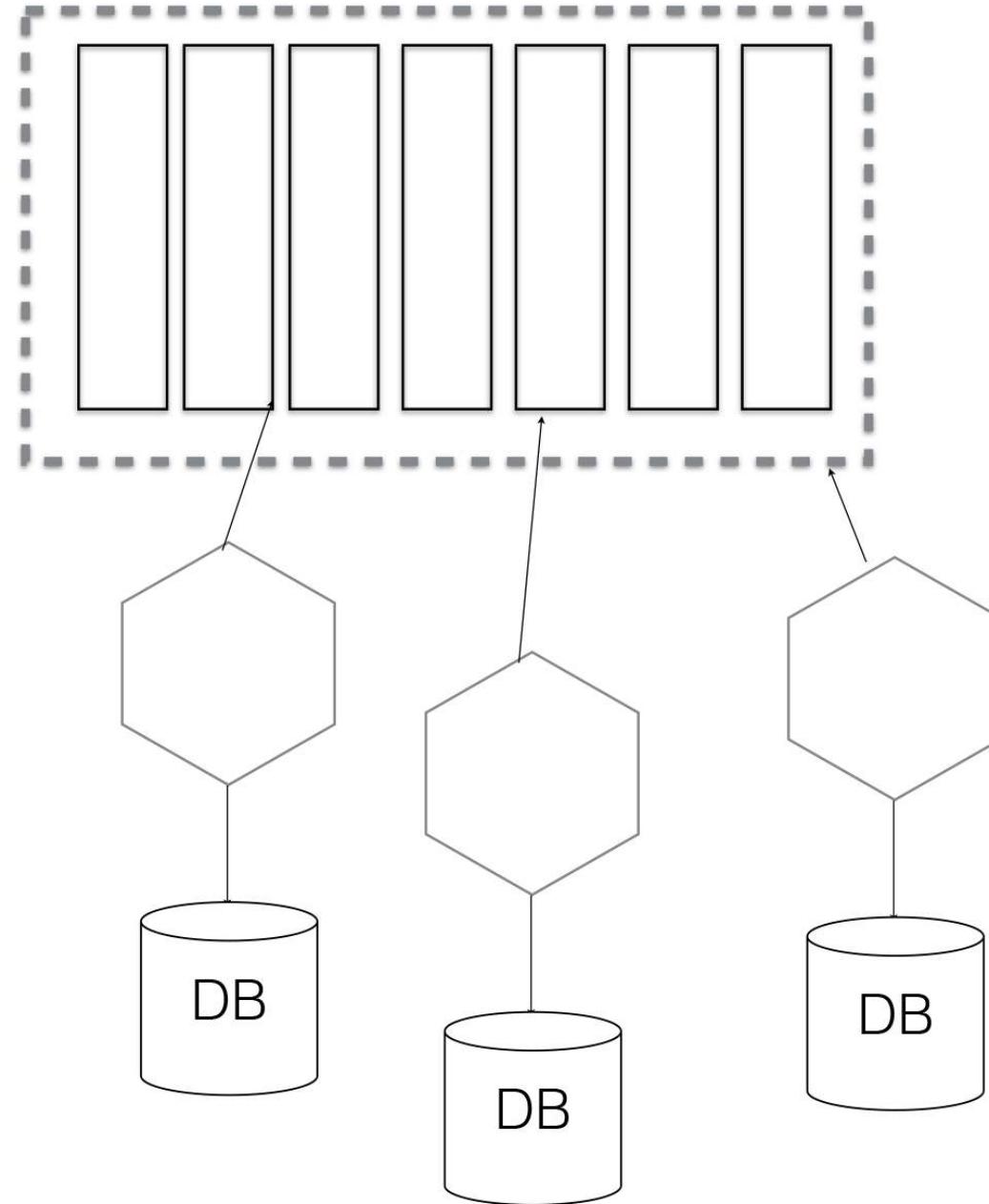
sam@gmail.com

15 My Place, UK

Email

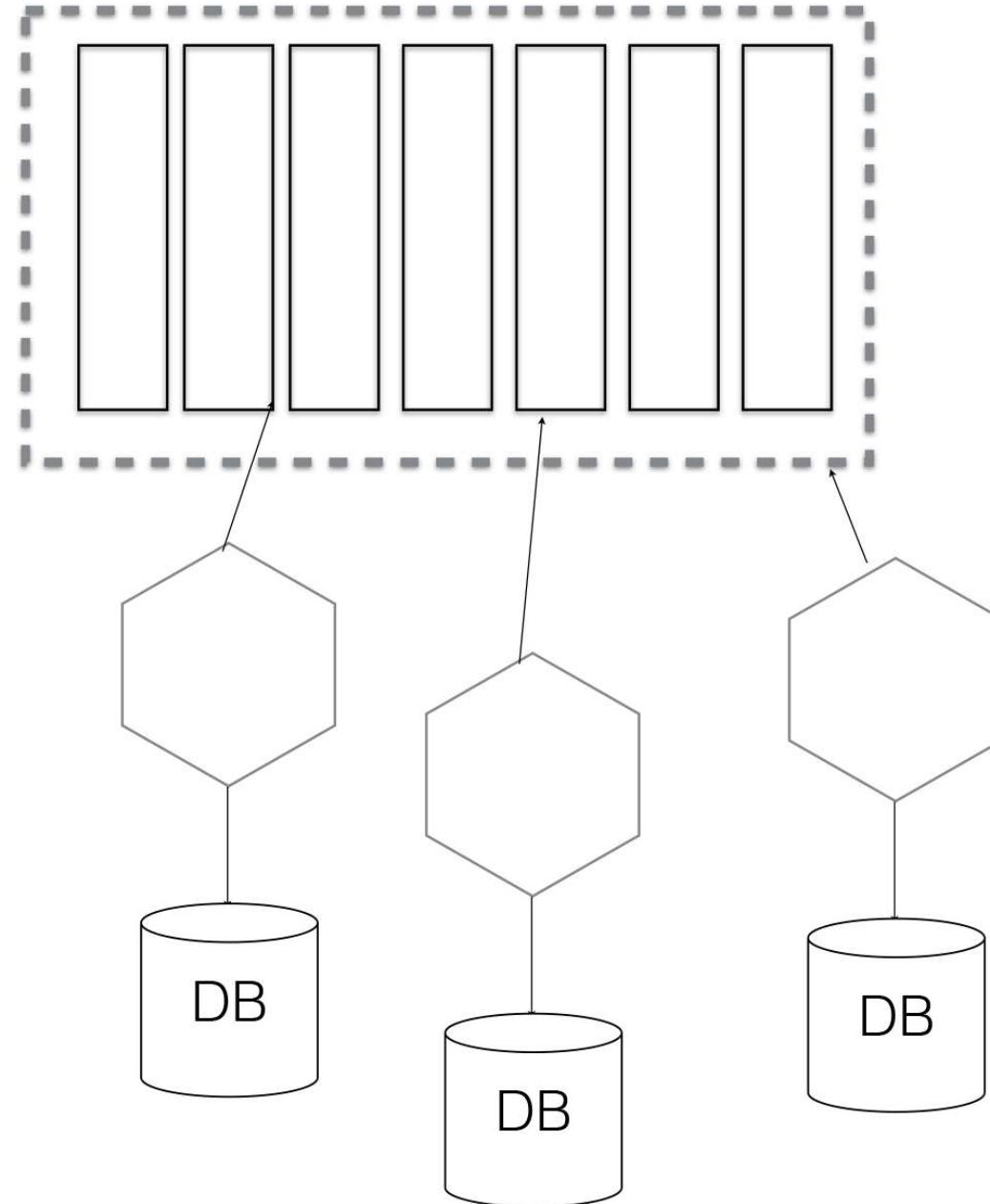


Source Of Truth



Source Of Truth

Service Of Record



confluent

O'REILLY®

Compliments of
confluent

Designing Event-Driven Systems

Concepts and Patterns for Streaming Services with Apache Kafka

Concepts and Patterns for Streaming Services with Apache Kafka

Download Book

First Name

Last Name

Company Name

Email

Phone Number

Job Role ▾

Get Book

<https://www.confluent.io/designing-event-driven-systems>

O'REILLY®

Designing Data-Intensive Applications

THE BIG IDEAS BEHIND RELIABLE, SCALABLE,
AND MAINTAINABLE SYSTEMS



Martin Kleppmann

Don't just hack it together

NoSQL... Big Data... Scalability... CAP Theorem... Eventual Consistency... Sharding...

Nice buzzwords, but how does the stuff actually *work*?

As software engineers, we need to build applications that are reliable, scalable and maintainable in the long run. We need to understand the range of available tools and their trade-offs. For that, we have to dig deeper than buzzwords.

This book will help you navigate the diverse and fast-changing landscape of technologies for storing and processing data. We compare a broad variety of tools and approaches, so that you can see the strengths and weaknesses of each, and decide what's best for your application.

[Get the book »](#)

 Tweet

<https://dataintensive.net/>

**Simple = Synchronous, Request
Response Communication**

**Simple = Synchronous, Request
Response Communication**

REST over HTTP

Very well supported

Very well supported

Good error handling semantics

Very well supported

Good error handling semantics

Cache controls

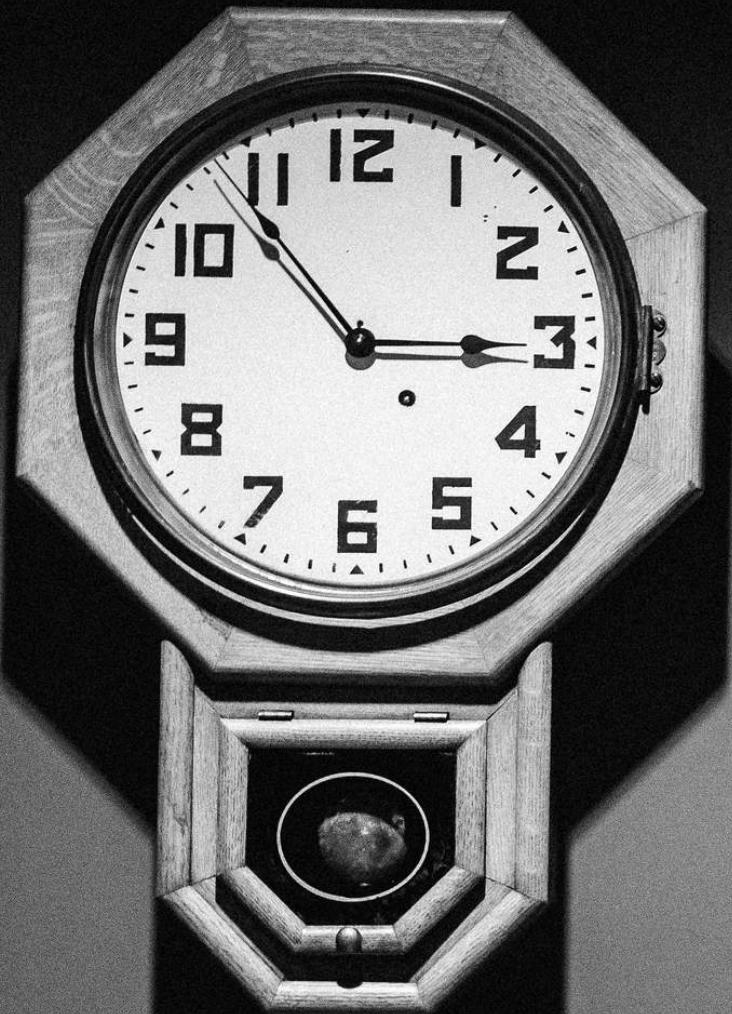
Very well supported

Good error handling semantics

Cache controls

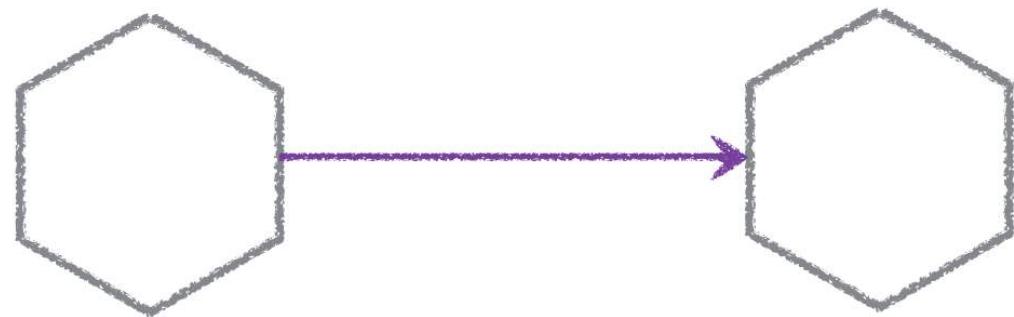
**Easy to scale - good, reliable
(boring?) technology**

**Where does “simple” stop
being good enough?**





Long-lived processes



Call overhead

DIFFERENT PROTOCOLS

DIFFERENT PROTOCOLS

Simple Binary Encoding (SBE)

javadoc 1.8.7

SBE is an OSI layer 6 presentation for encoding and decoding binary application messages for low-latency financial applications. This repository contains the reference implementations in Java, C++, Golang, and C#.

Further details on the background and usage of SBE can be found on the [Wiki](#).

An XSD for SBE specs can be found [here](#). Please address questions about the specification to the [SBE FIX community](#).

For the latest version information and changes see the [Change Log](#) with [downloads](#) at [Maven Central](#).

The Java and C++ SBE implementations are designed to work very efficiently with the [Aeron](#) messaging system for low-latency and high-throughput communications. The Java SBE implementation has a dependency on [Agrona](#) for its buffer implementations.

<https://github.com/real-logic/simple-binary-encoding>

DIFFERENT PROTOCOLS

Simple Binary Encoding (SBE)

javadoc 1.8.7

SBE is an OSI layer 6 presentation for encoding and decoding binary applications. This repository contains the reference implementations in Java.

Further details on the background and usage of SBE can be found on the [SBE website](#).

An XSD for SBE specs can be found [here](#). Please address questions about SBE to the [mailing list](#).

For the latest version information and changes see the [Change Log](#) with the [diffs](#).

The Java and C++ SBE implementations are designed with work very efficient, low-latency and high-throughput communications. The Java SBE implementation is based on the [Aeron](#) codebase.

<https://github.com/real-logic/simple-binary-encoding>

Aeron

chat on [glitter](#) To chat with other Aeron users and the contributors.

javadoc 1.10.5

Efficient reliable UDP unicast, UDP multicast, and IPC message transport. Java and C++ clients are available in this repository, and a [.NET client](#) is available from a 3rd party. All three clients can exchange messages across machines, or on the same machine via IPC, very efficiently. Message streams can be recorded by the [Archive](#) module to persistent storage for later or real-time replay.

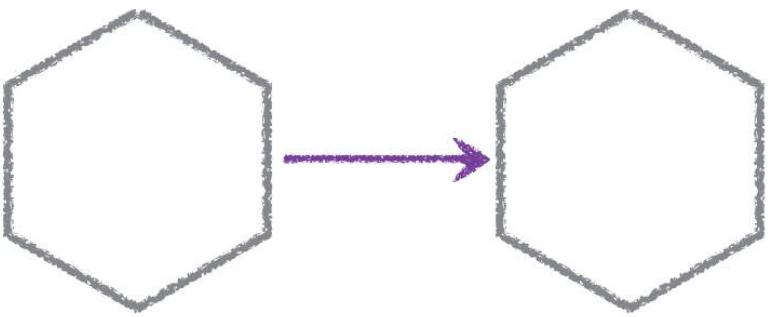
Performance is the key focus. Aeron is designed to be the highest throughput with the lowest and most predictable latency possible of any messaging system. Aeron integrates with [Simple Binary Encoding \(SBE\)](#) for the best possible performance in message encoding and decoding. Many of the data structures used in the creation of Aeron have been factored out to the [Agrona](#) project.

For details of usage, protocol specification, FAQ, etc. please check out the [Wiki](#).

For those who prefer to watch a video then try [Aeron Messaging](#) from StrangeLoop 2014. Things have moved on quite a bit with performance and some features but the basic design still applies.

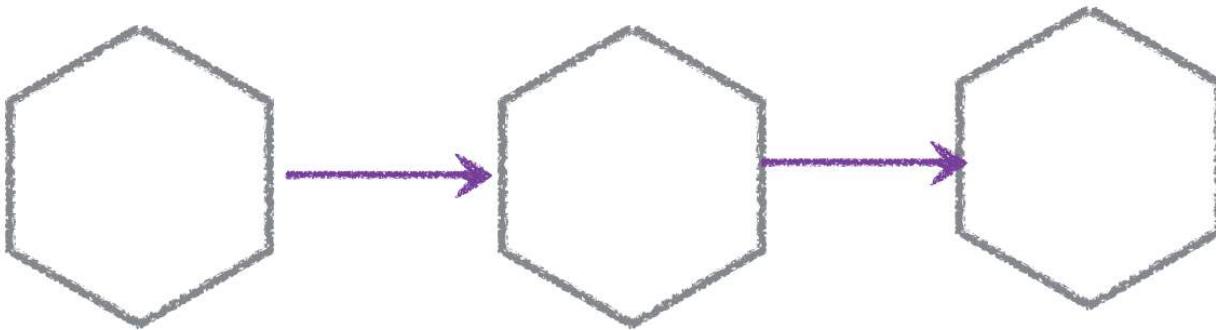
<https://github.com/real-logic/aeron>

BE CAREFUL ABOUT LATENCY BUILDUP



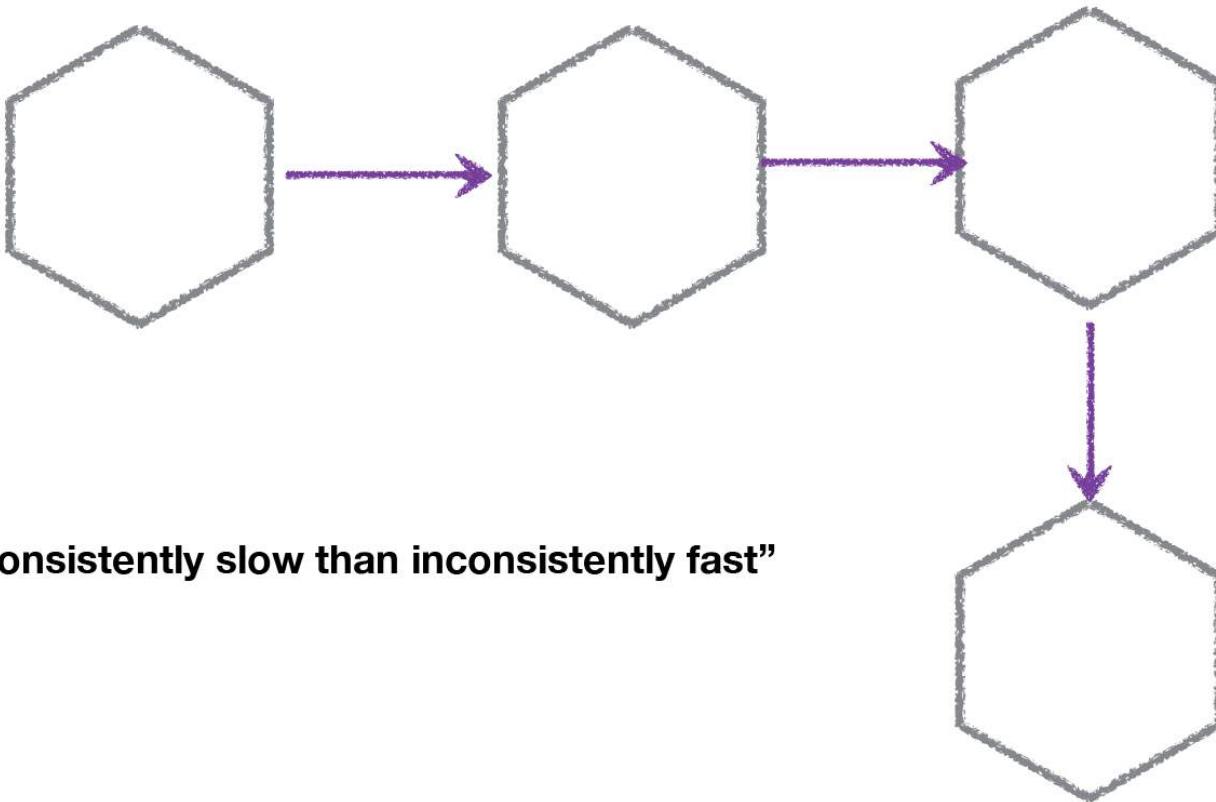
“It’s better to be consistently slow than inconsistently fast”

BE CAREFUL ABOUT LATENCY BUILDUP



“It’s better to be consistently slow than inconsistently fast”

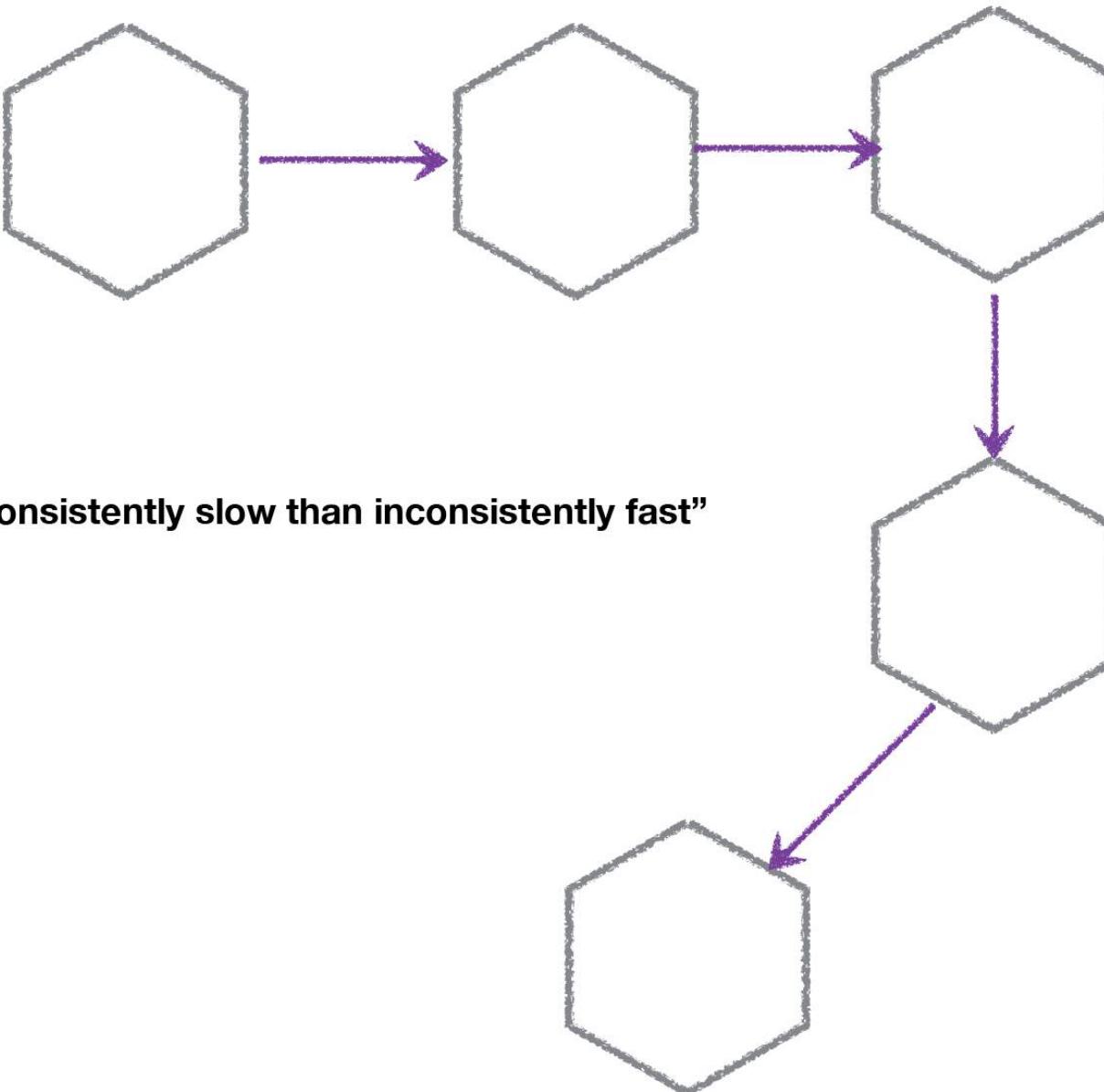
BE CAREFUL ABOUT LATENCY BUILDUP



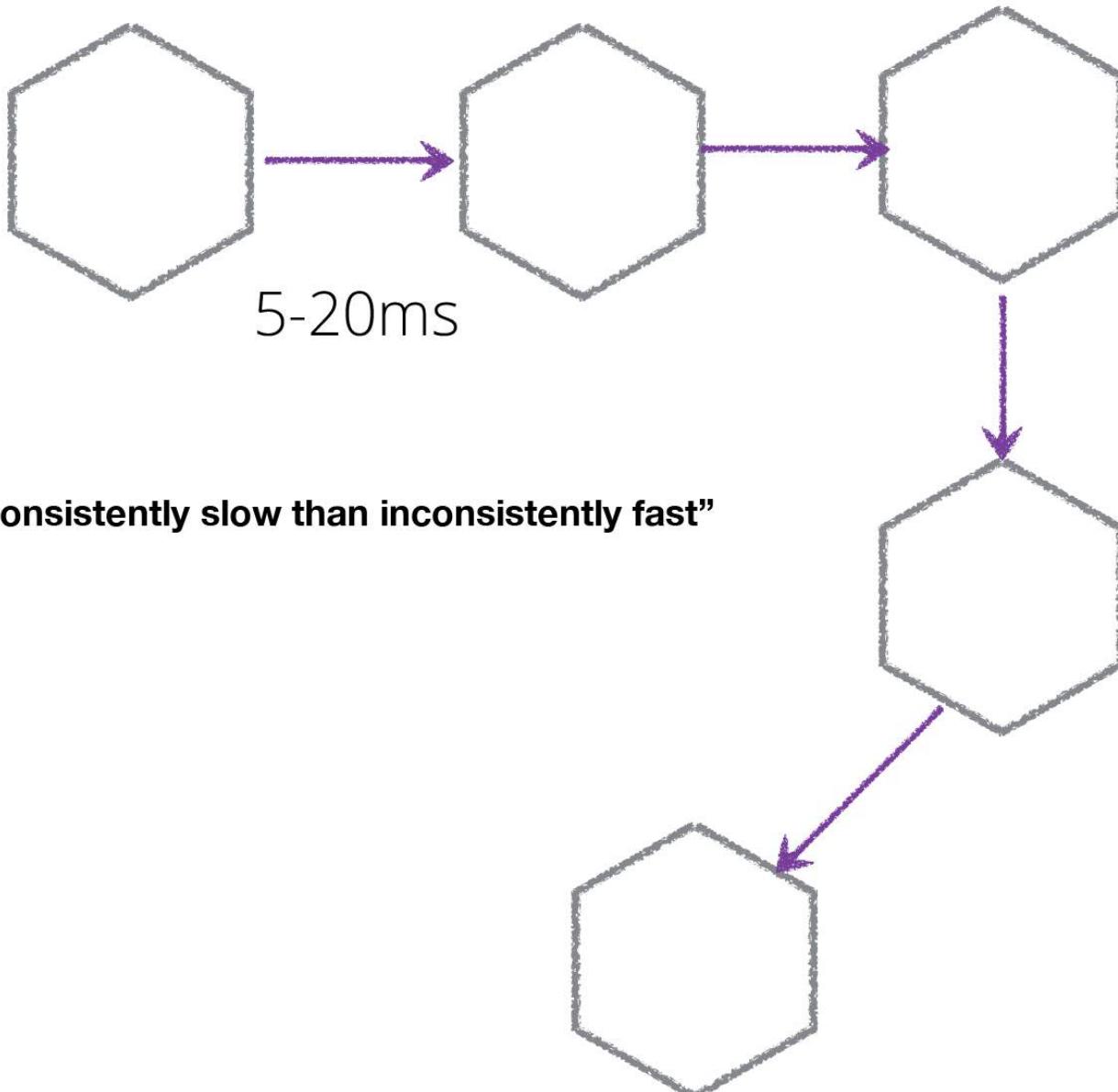
“It’s better to be consistently slow than inconsistently fast”

BE CAREFUL ABOUT LATENCY BUILDUP

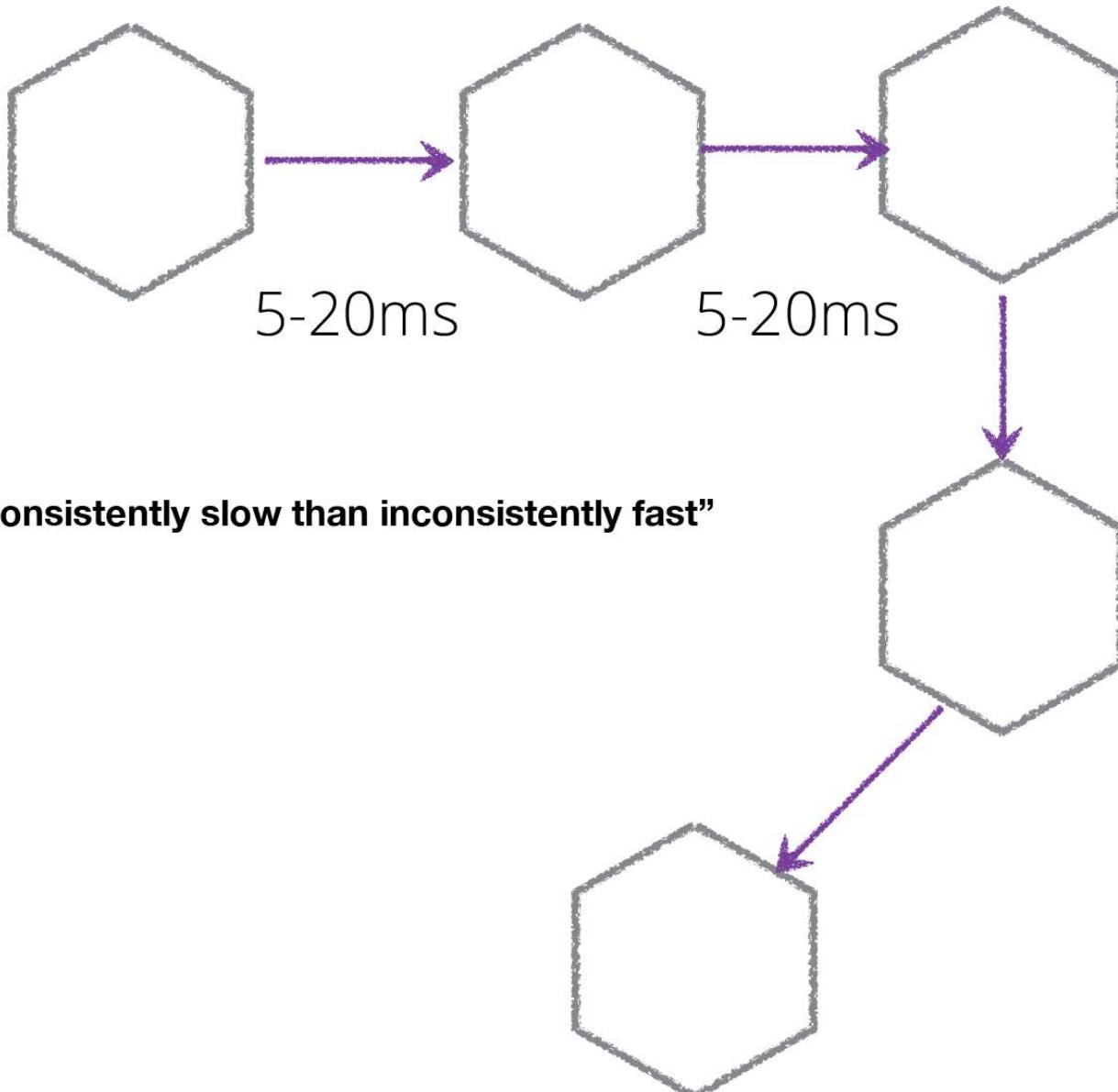
“It’s better to be consistently slow than inconsistently fast”



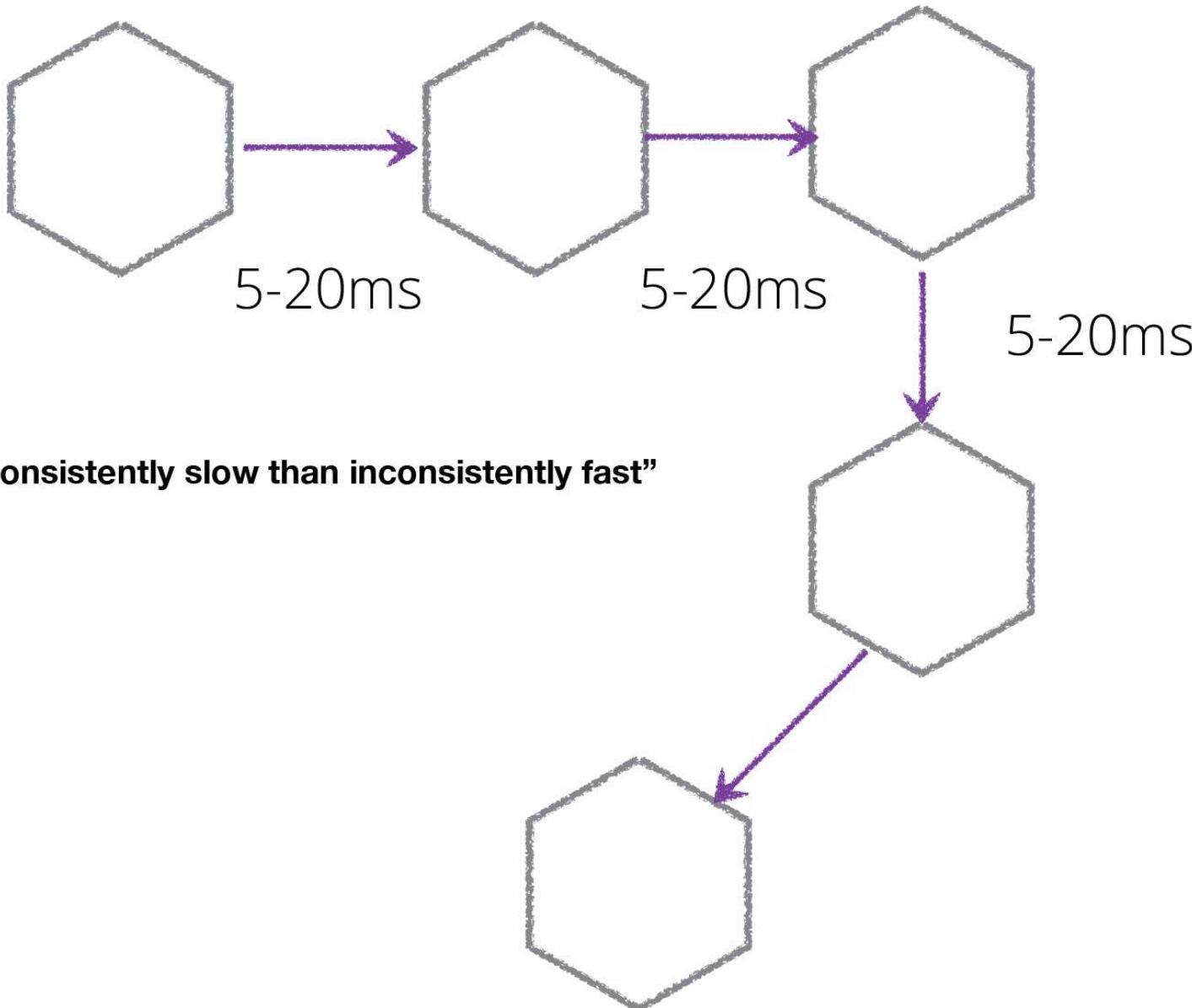
BE CAREFUL ABOUT LATENCY BUILDUP



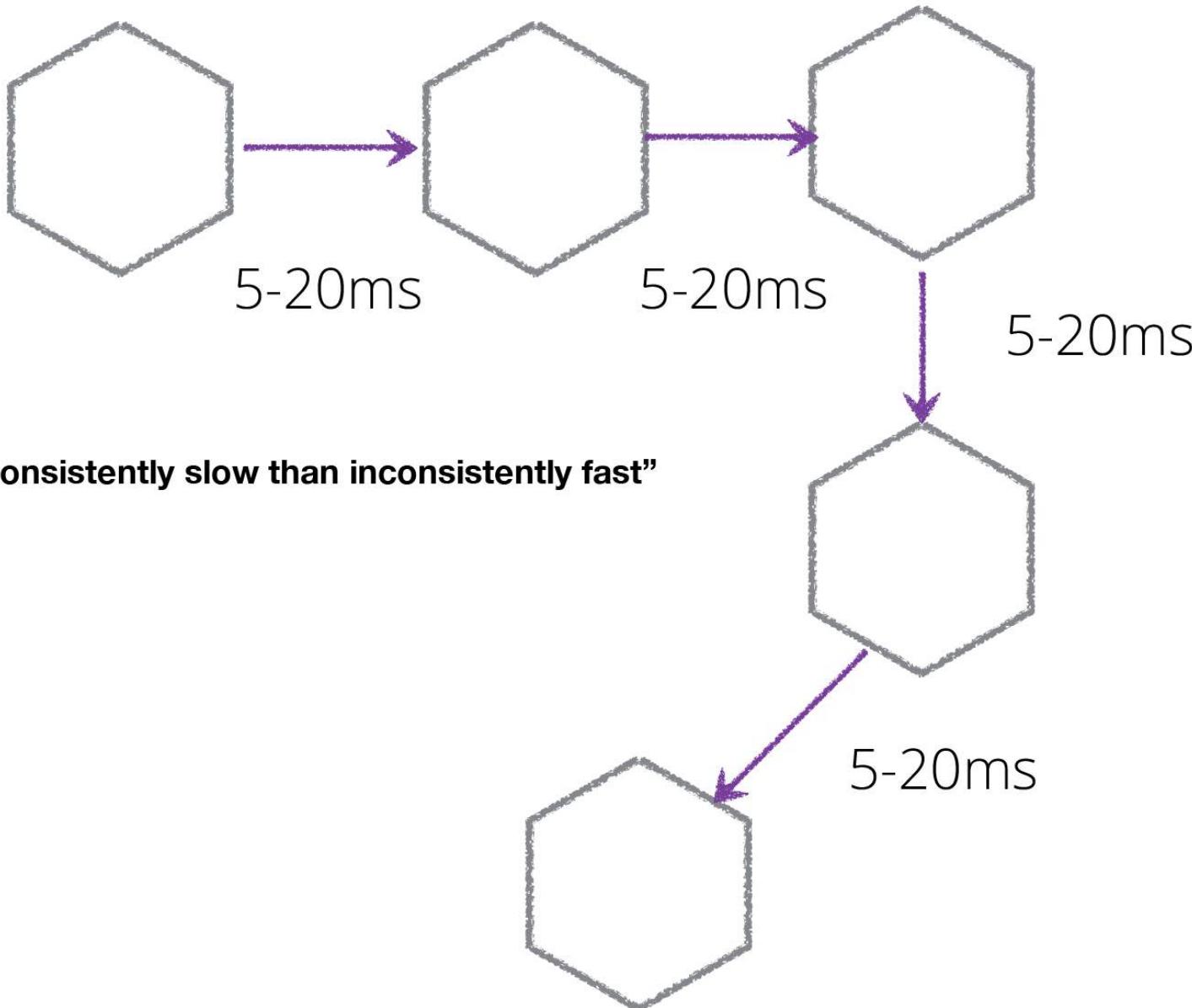
BE CAREFUL ABOUT LATENCY BUILDUP



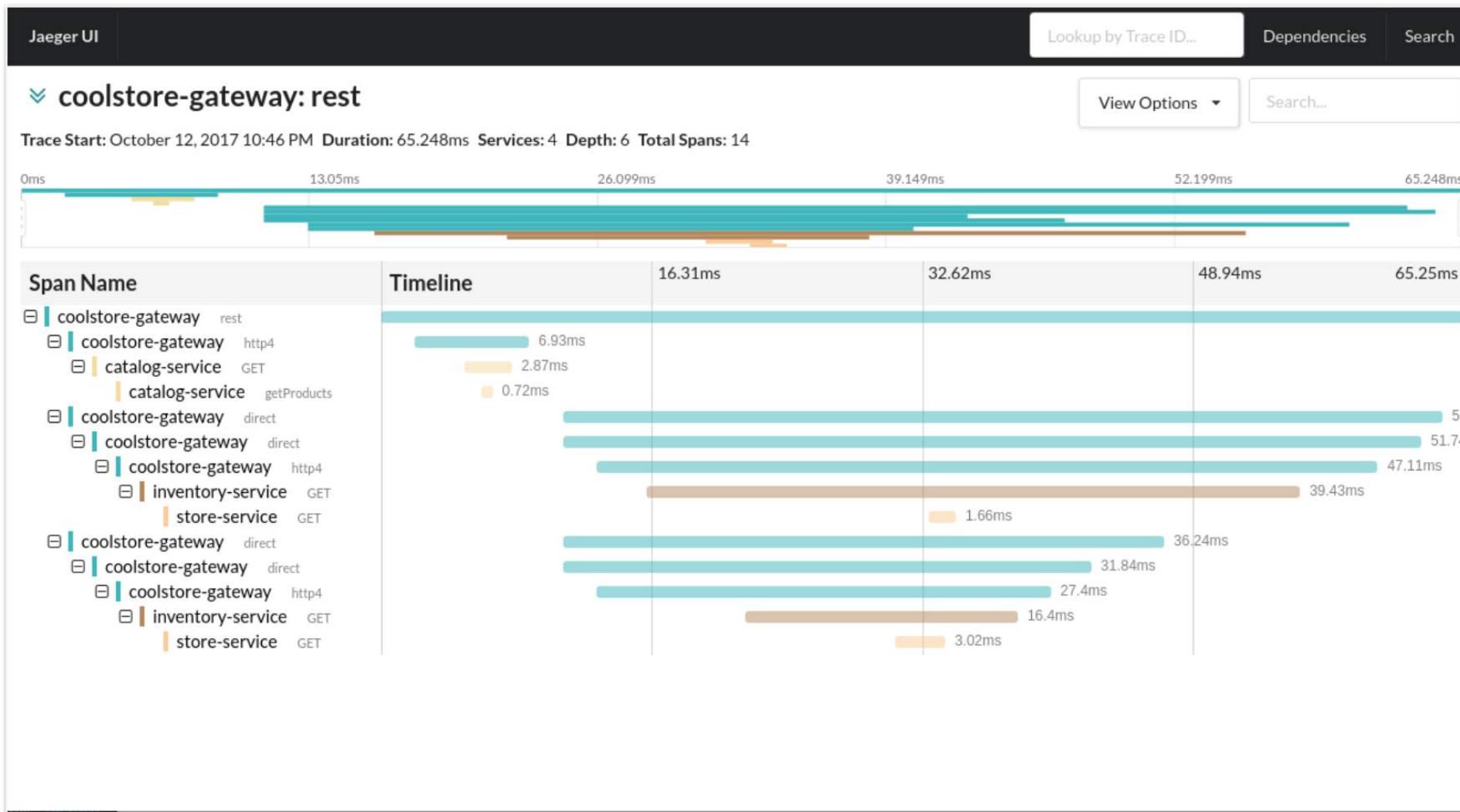
BE CAREFUL ABOUT LATENCY BUILDUP



BE CAREFUL ABOUT LATENCY BUILDUP



DISTRIBUTED TRACING



<https://github.com/jaegertracing/jaeger>



Yuri Shkuro

Follow

Staff Engineer at Uber; Tech Lead for Jaeger, Uber's distributed tracing system; co-founder (co-conspirator) of OpenTracing. <https://github.com/yurishkuro>

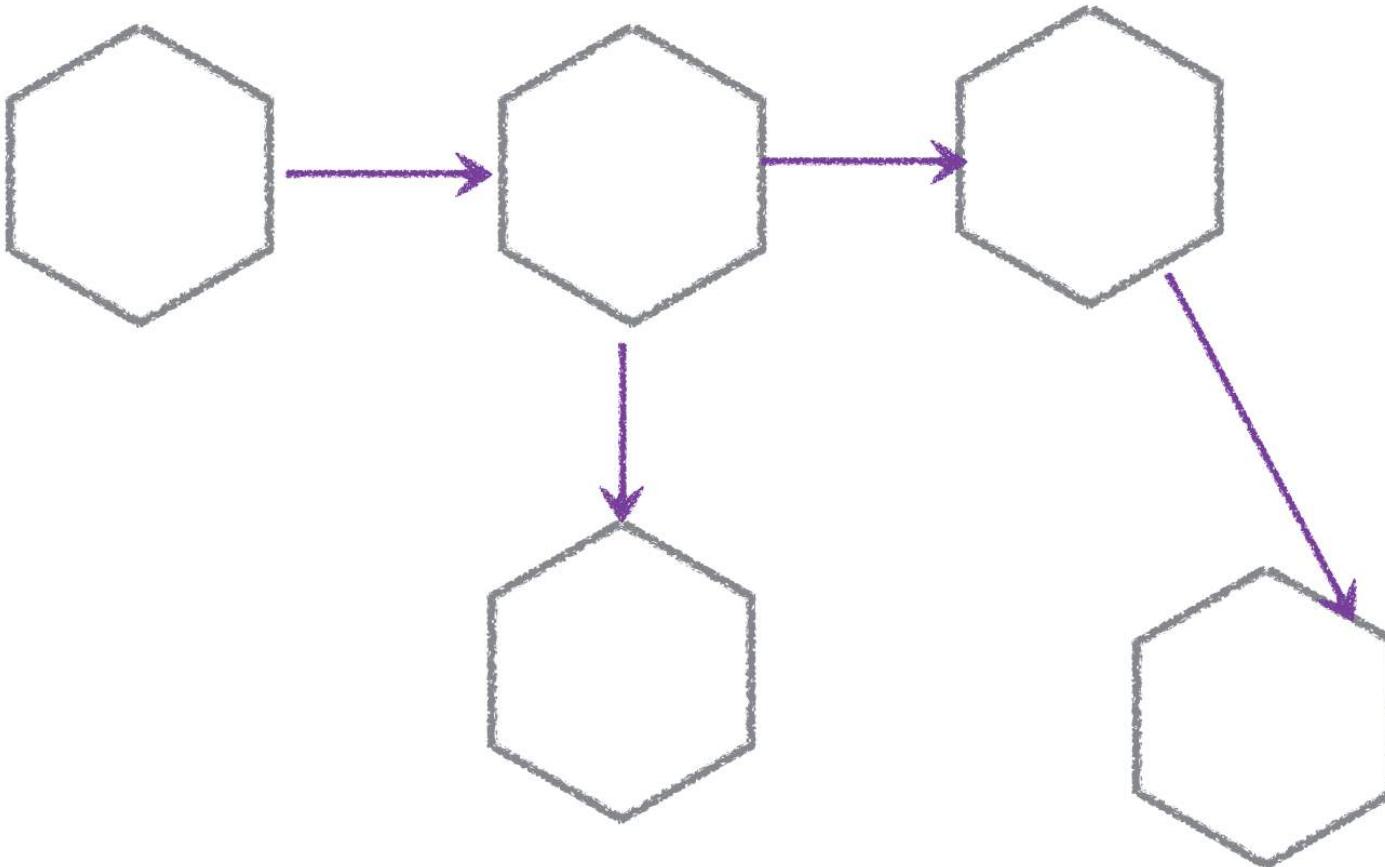
May 6, 2017 · 19 min read

Take OpenTracing for a HotROD ride

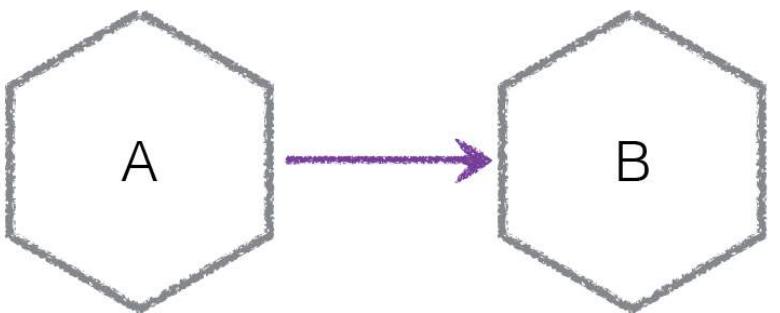


<https://medium.com/opentracing/take-opentracing-for-a-hotrod-ride-f6e3141f7941>

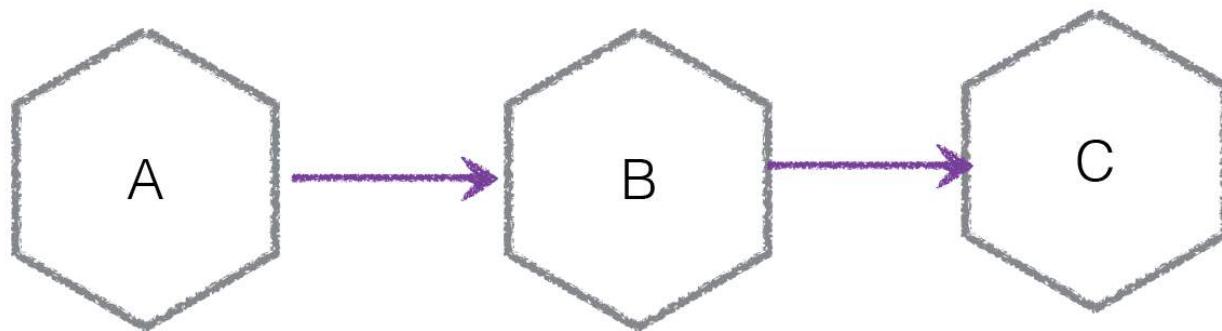
DO MORE IN PARALLEL



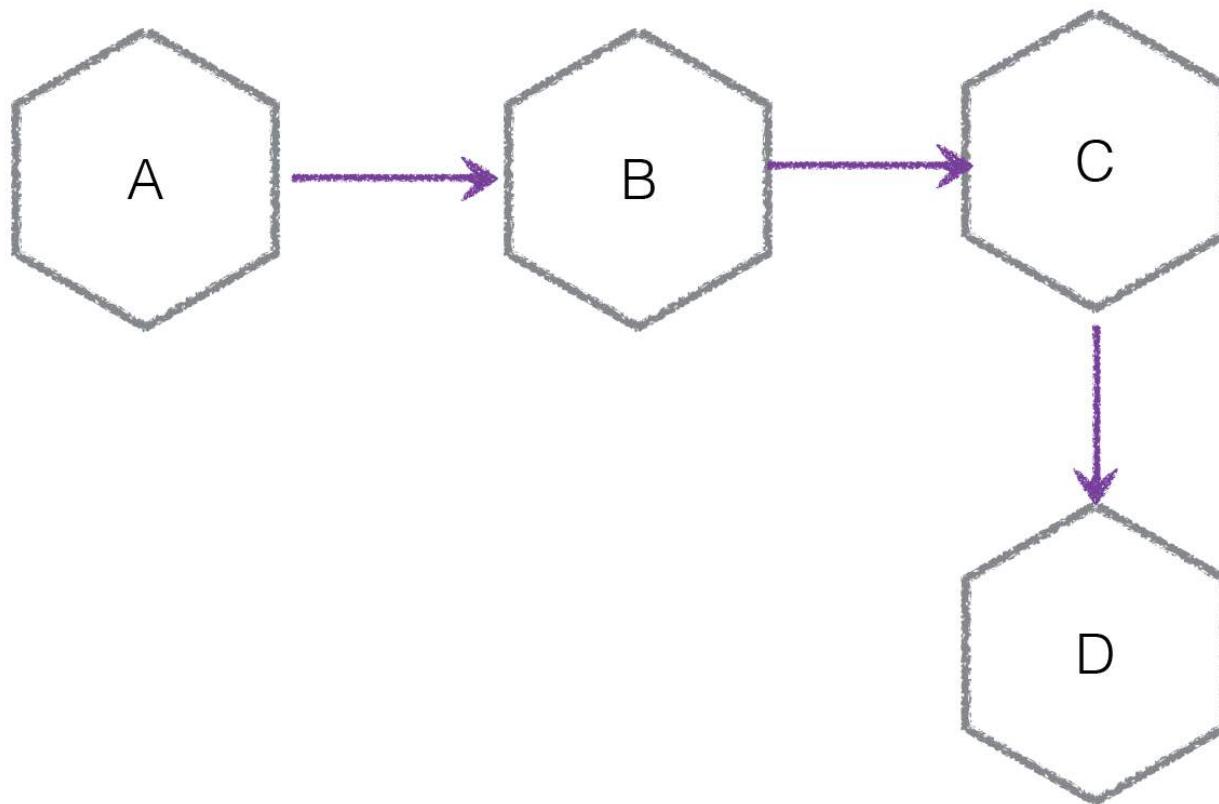
TEMPORAL COUPLING



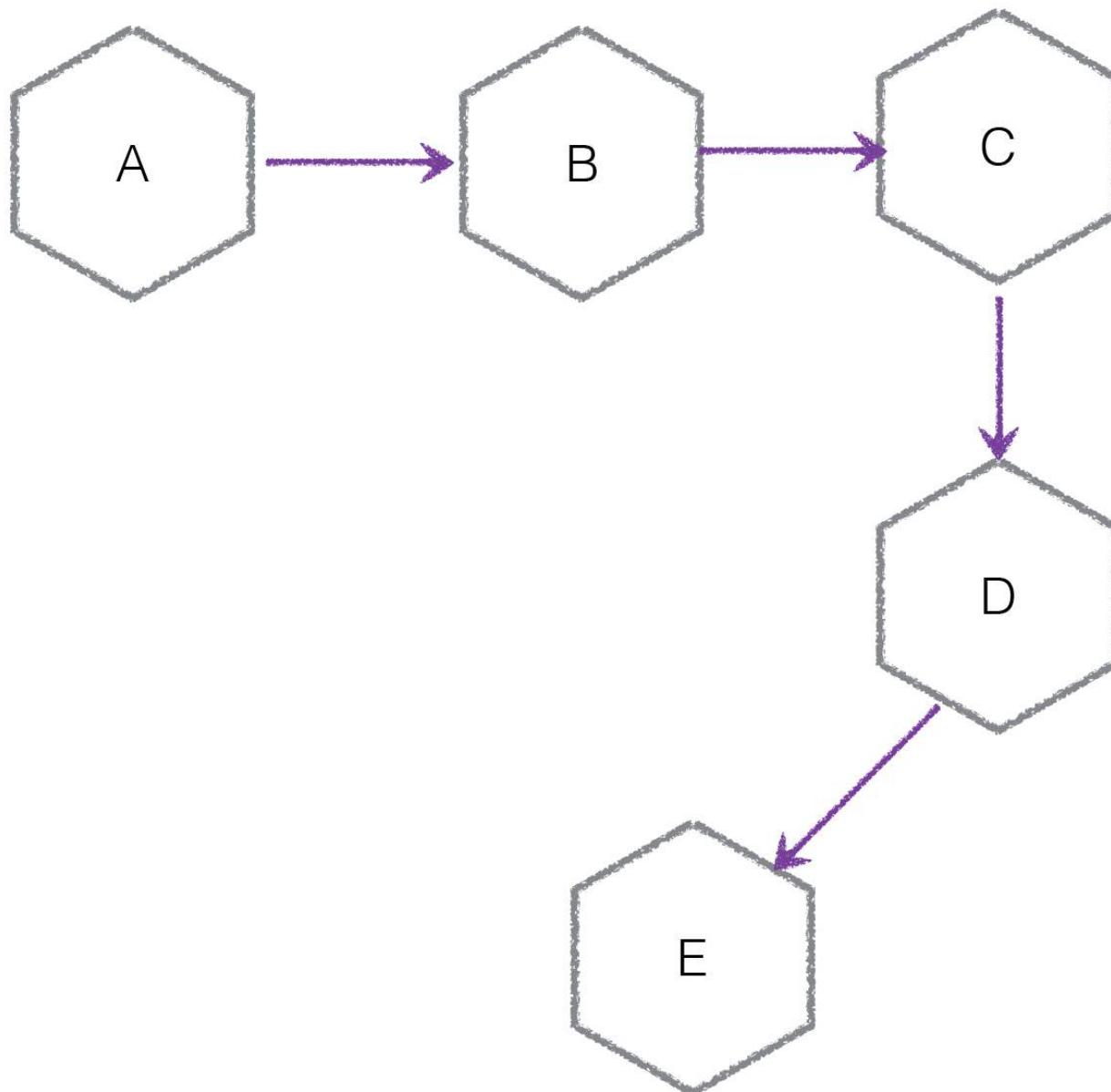
TEMPORAL COUPLING



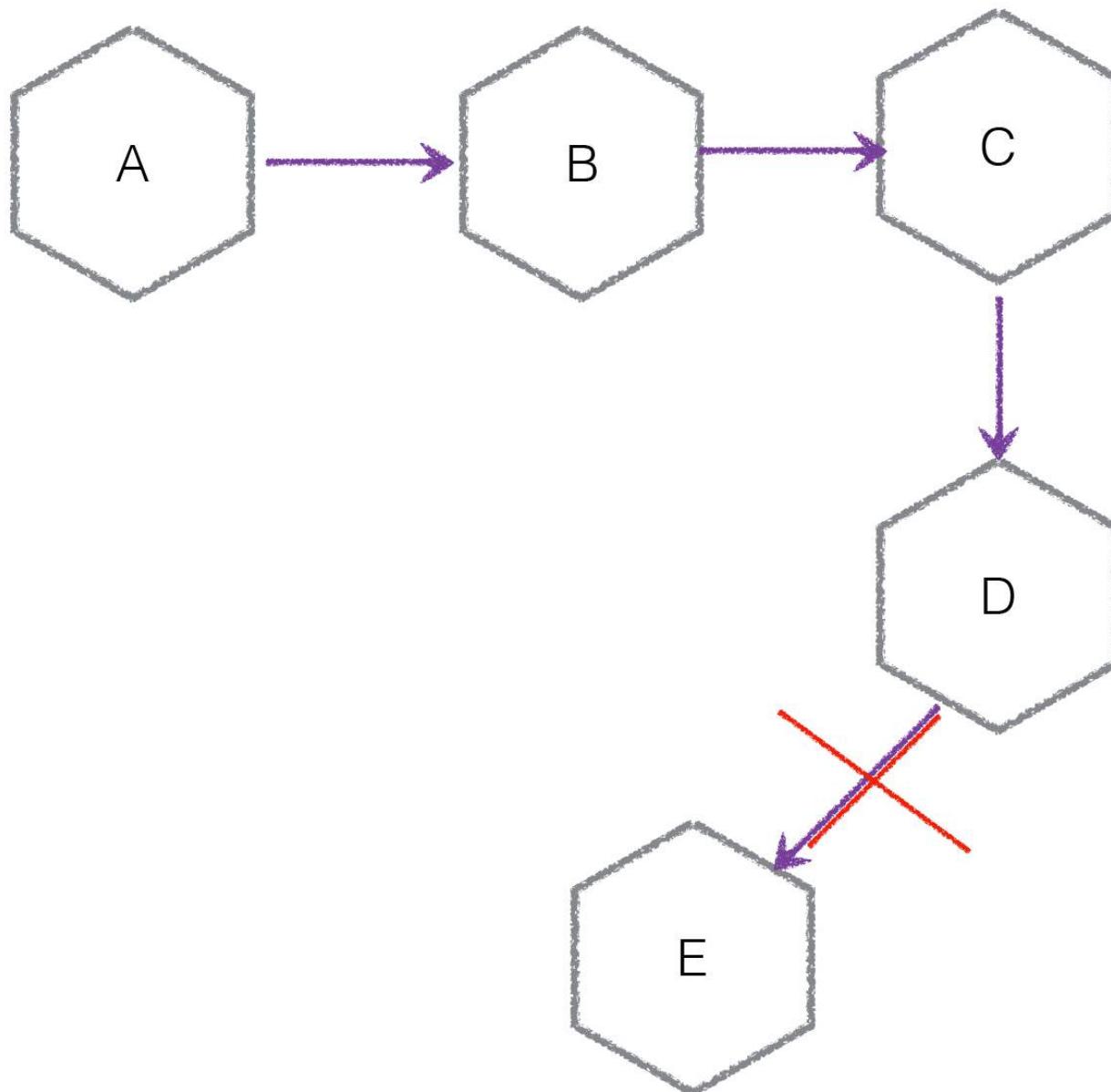
TEMPORAL COUPLING



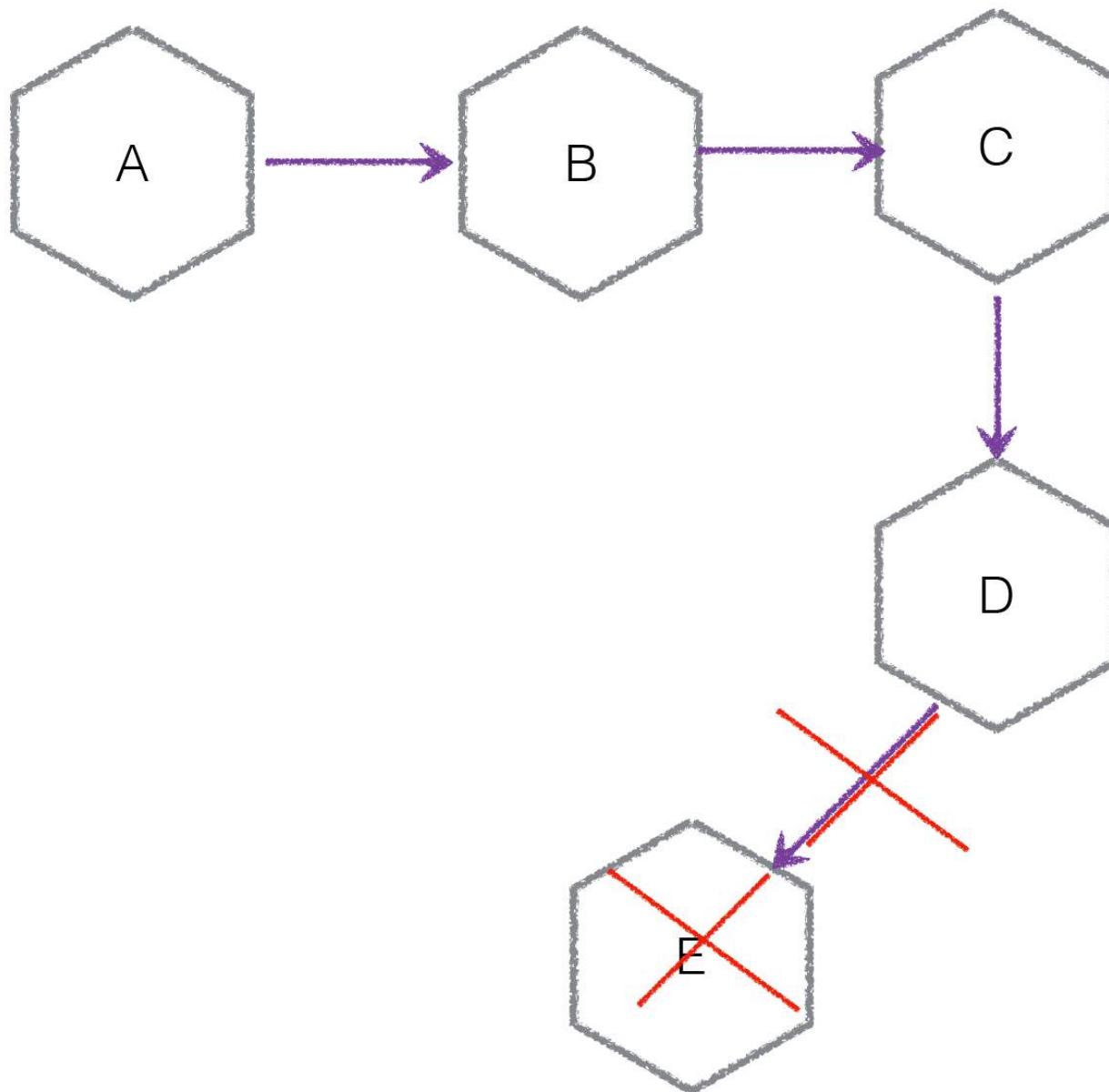
TEMPORAL COUPLING



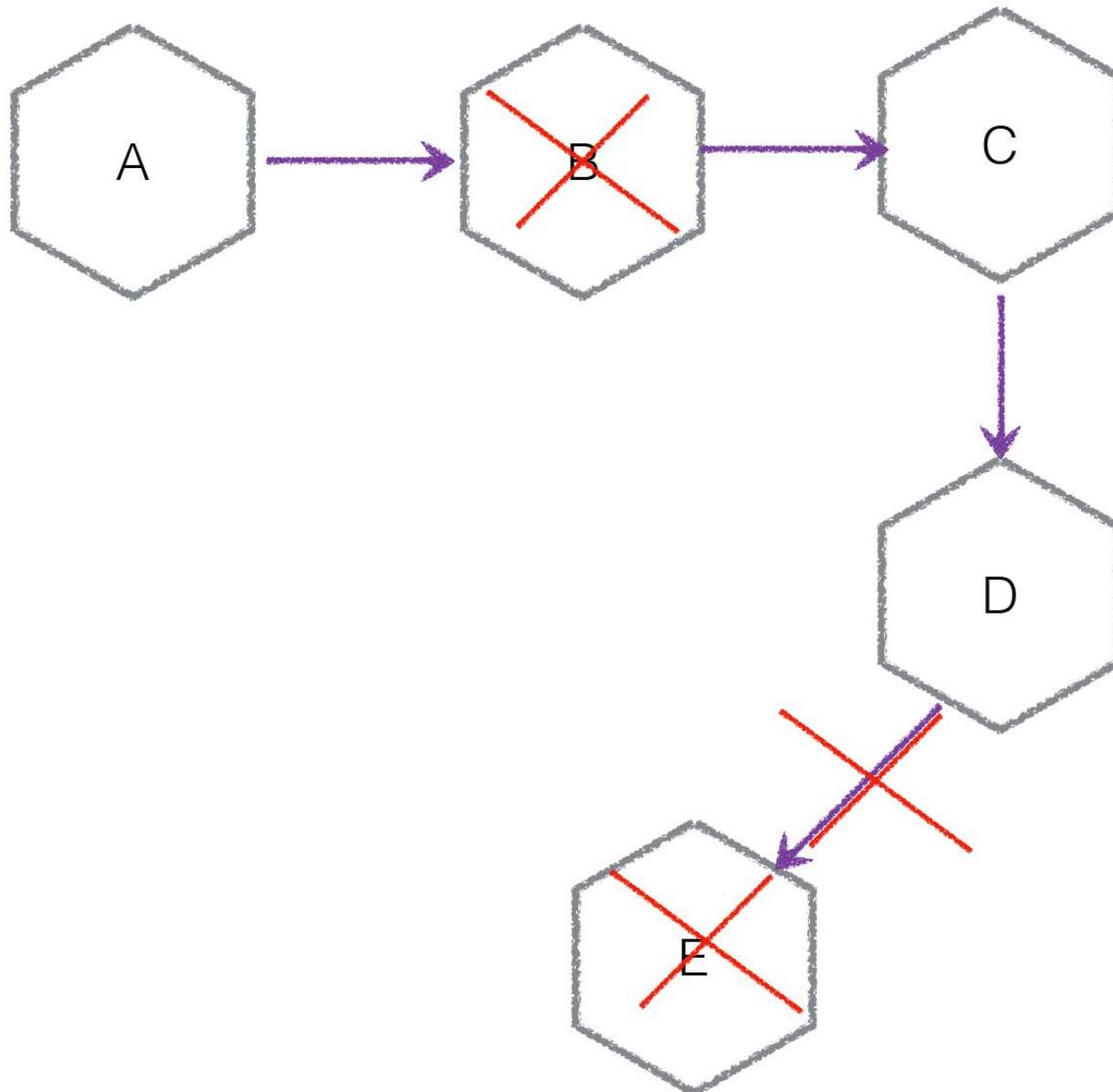
TEMPORAL COUPLING



TEMPORAL COUPLING



TEMPORAL COUPLING



REACTIVE MANIFESTO

The Reactive Manifesto

Published on September 16 2014. (v2.0)

Organisations working in disparate domains are independently discovering patterns for building software that look the same. These systems are more robust, more resilient, more flexible and better positioned to meet modern demands.

These changes are happening because application requirements have changed dramatically in recent years. Only a few years ago a large application had tens of servers, seconds of response time, hours of offline maintenance and gigabytes of data. Today applications are deployed on everything from mobile devices to cloud-based clusters running thousands of multi-core processors. Users expect millisecond response times and 100% uptime. Data is measured in Petabytes. Today's demands are simply not met by yesterday's software architectures.

We believe that a coherent approach to systems architecture is needed, and we believe that all necessary aspects are already recognised individually: we want systems that are Responsive, Resilient, Elastic and Message Driven. We call these Reactive Systems.

Systems built as Reactive Systems are more flexible, loosely-coupled and scalable. This makes them easier to develop and amenable to change. They are significantly more tolerant of failure and when failure does occur they meet it with elegance rather than disaster. Reactive Systems are highly responsive, giving users effective interactive feedback.

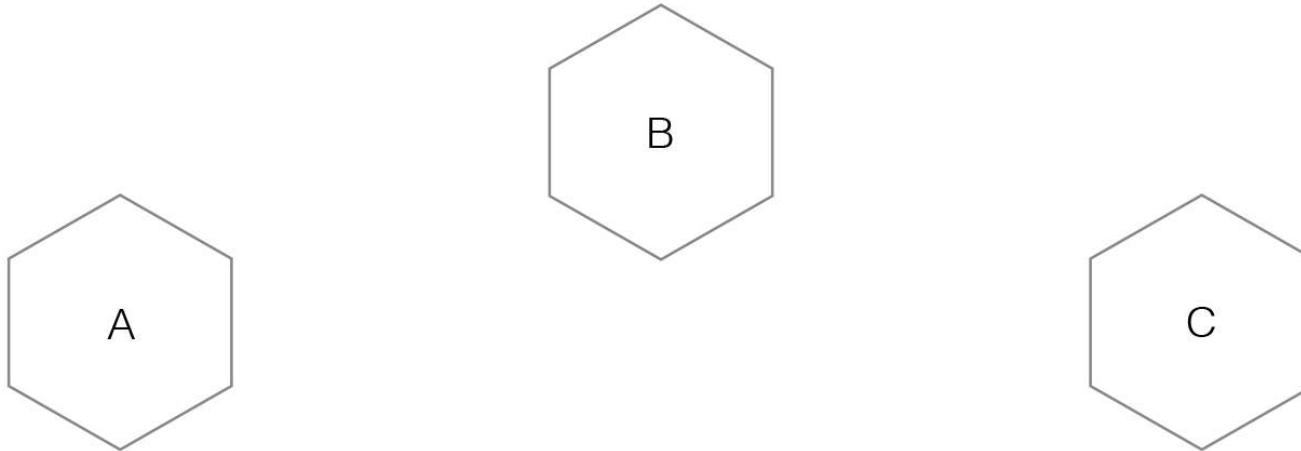
LOUDNESS

10



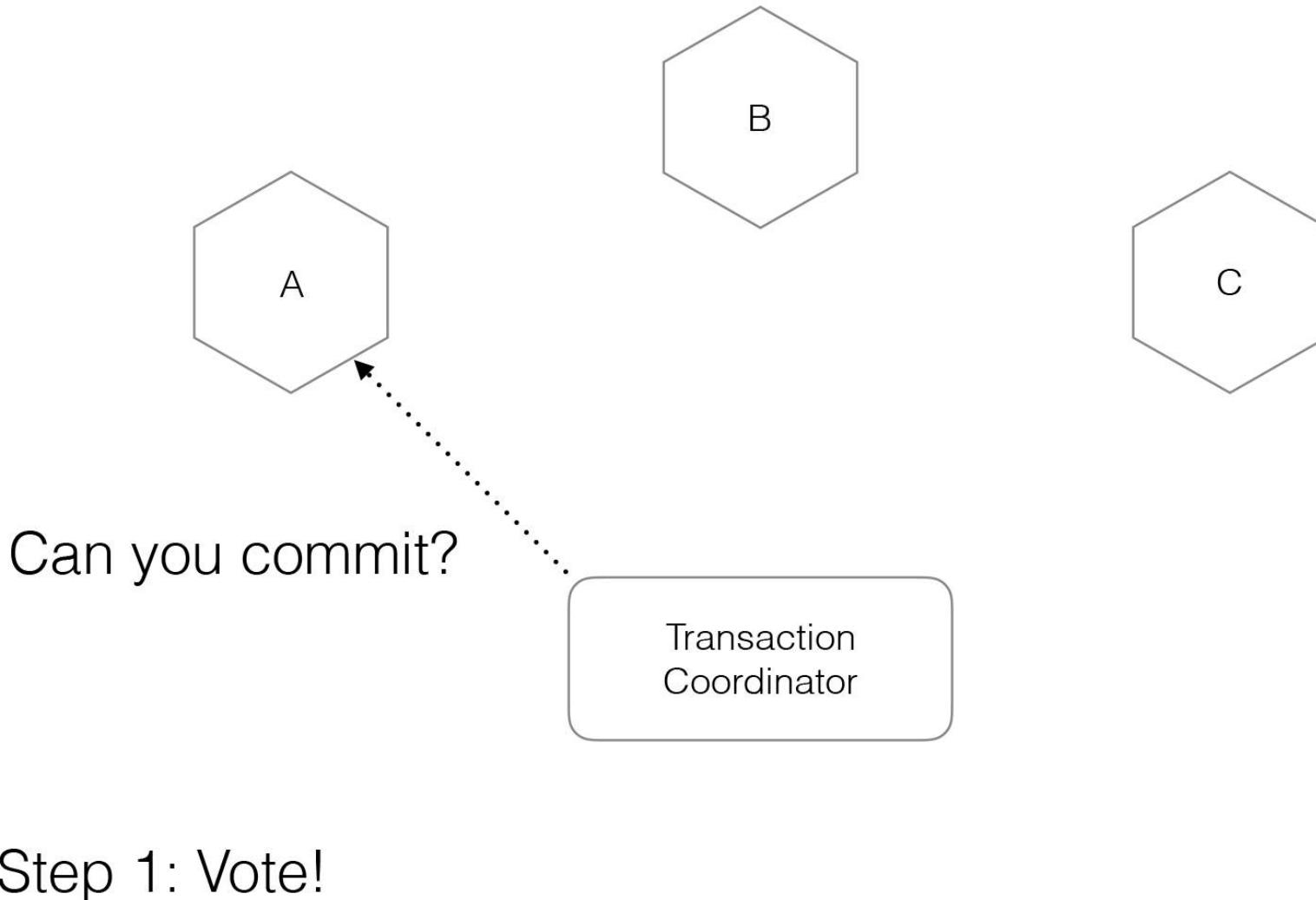
What about co-ordinating multiple services?

TWO PHASE COMMIT - SUCCESS

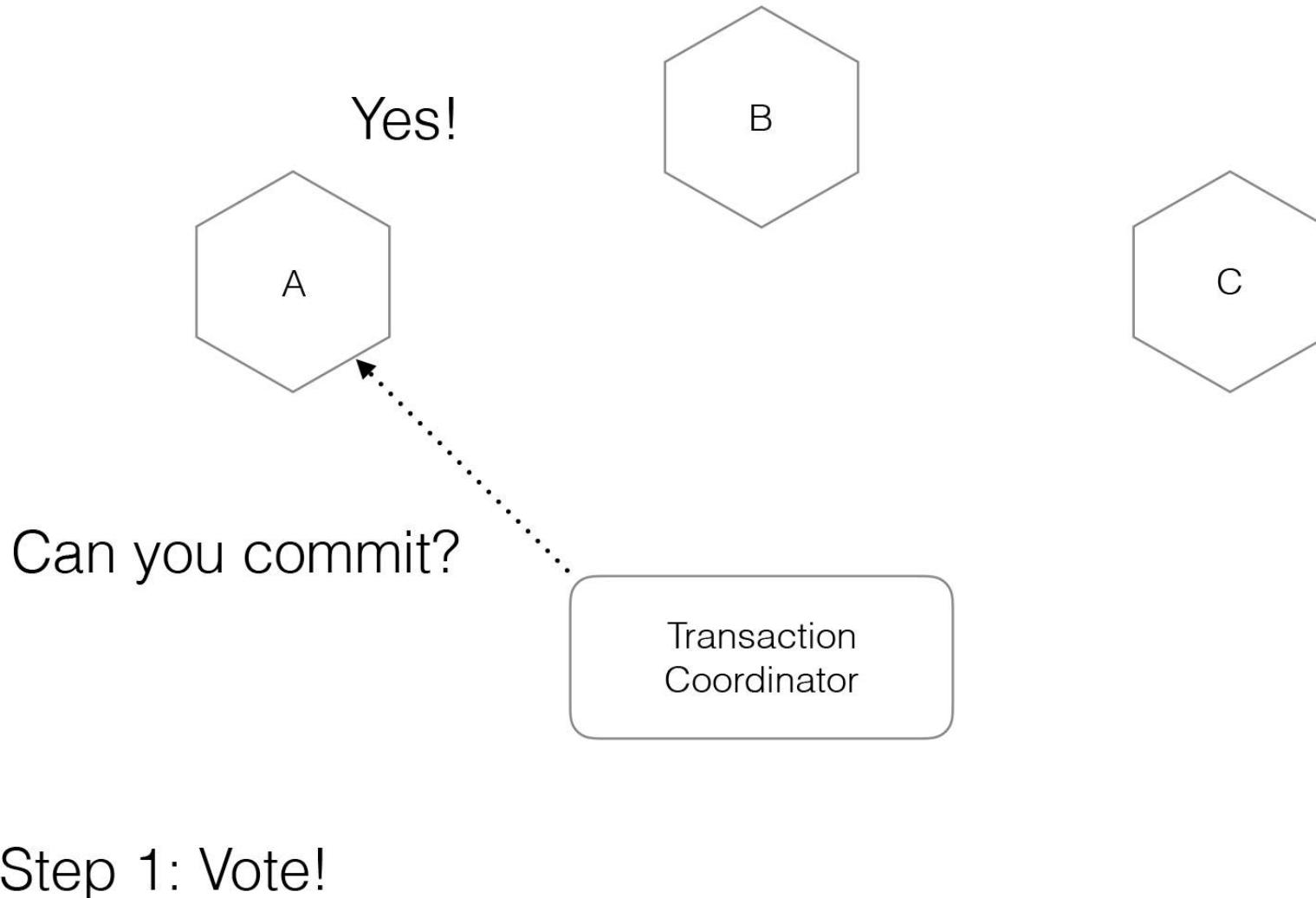


Step 1: Vote!

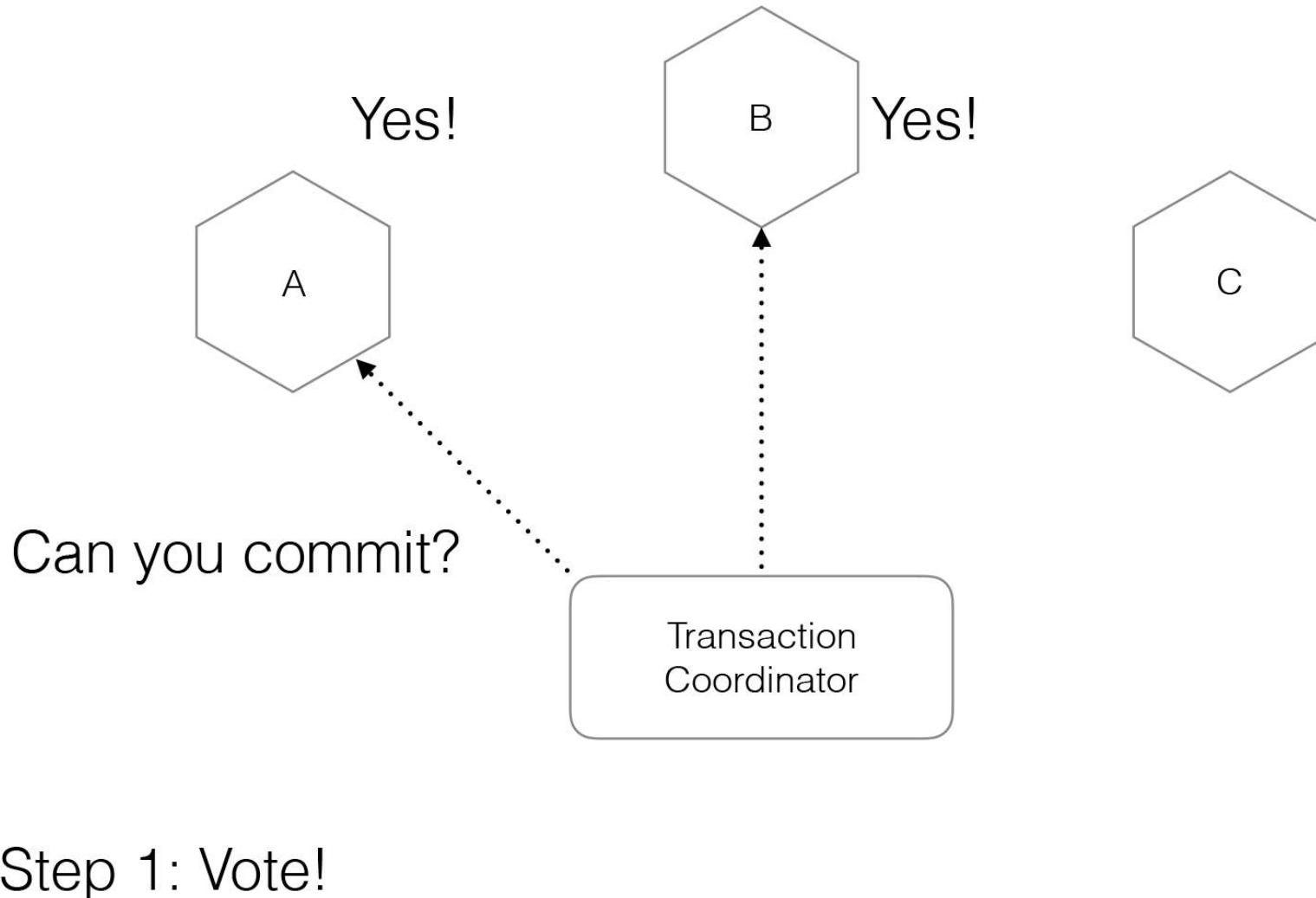
TWO PHASE COMMIT - SUCCESS



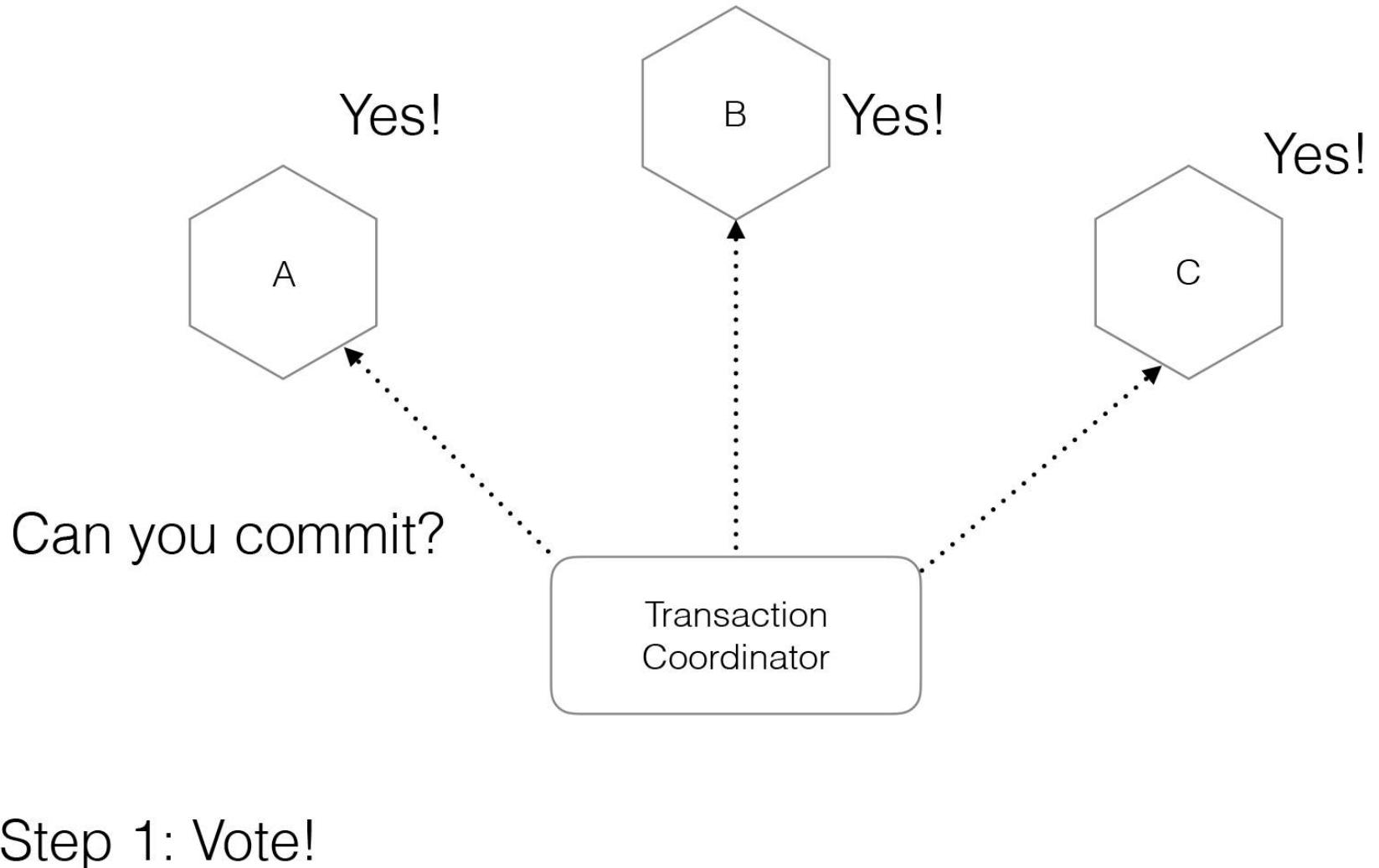
TWO PHASE COMMIT - SUCCESS



TWO PHASE COMMIT - SUCCESS

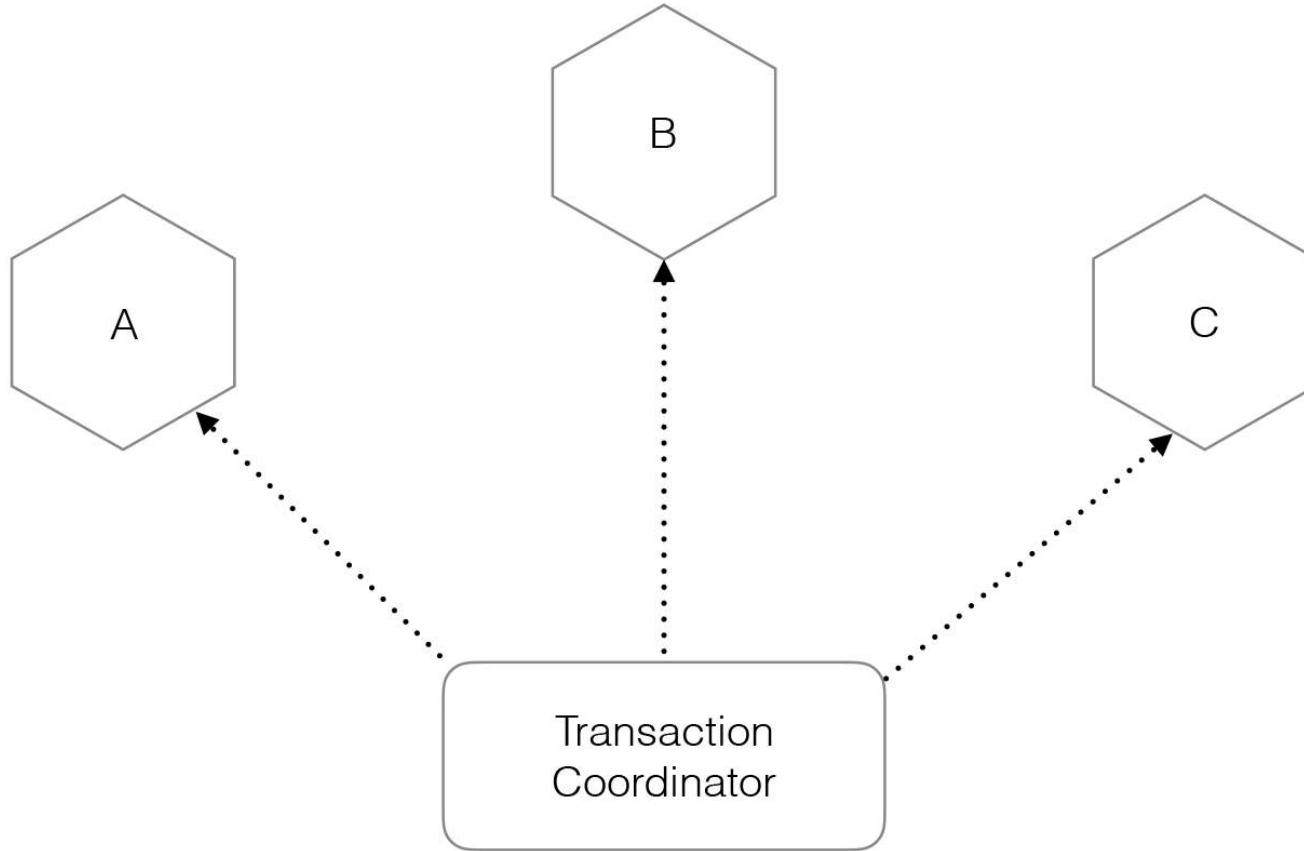


TWO PHASE COMMIT - SUCCESS



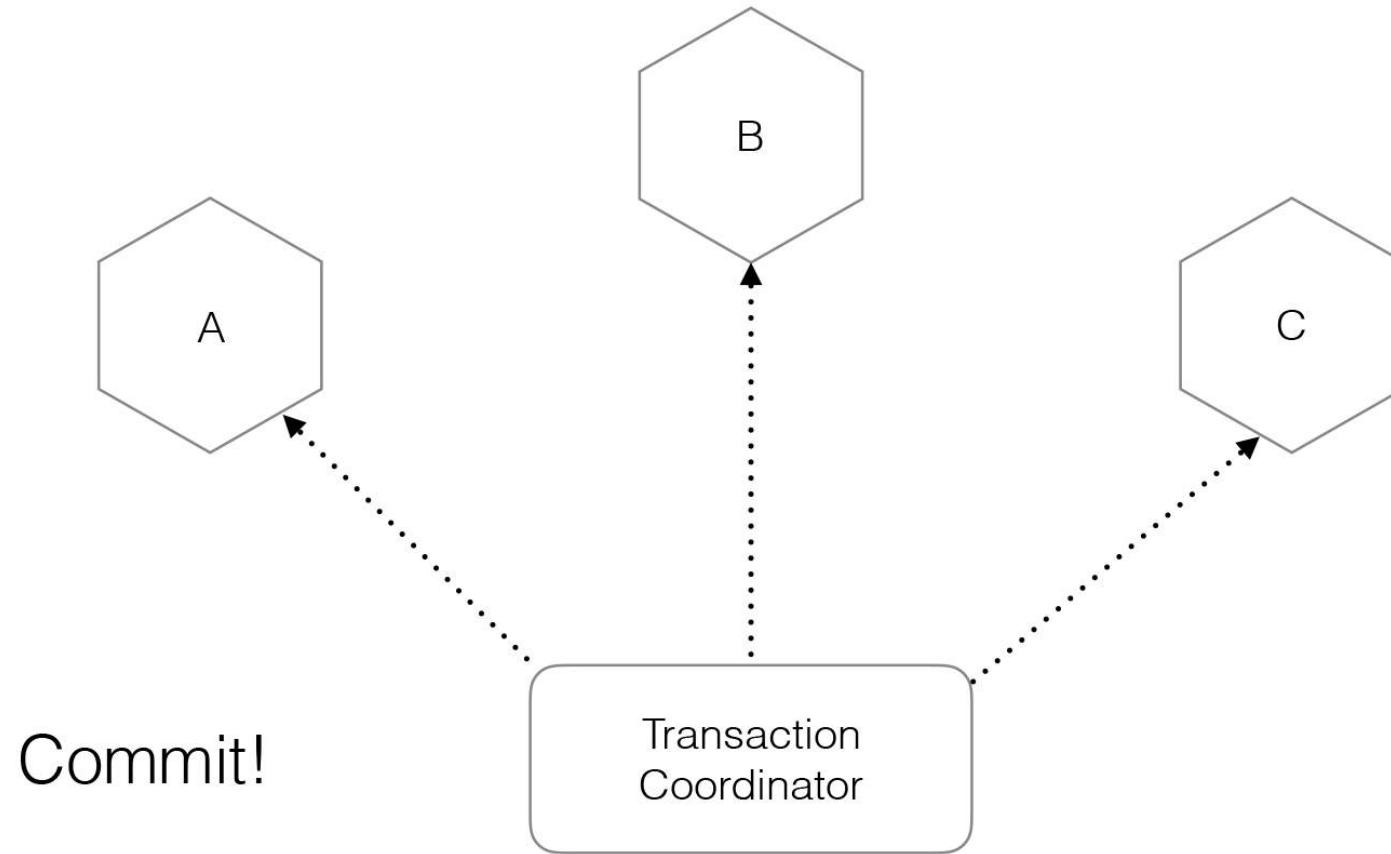
Step 1: Vote!

TWO PHASE COMMIT - SUCCESS



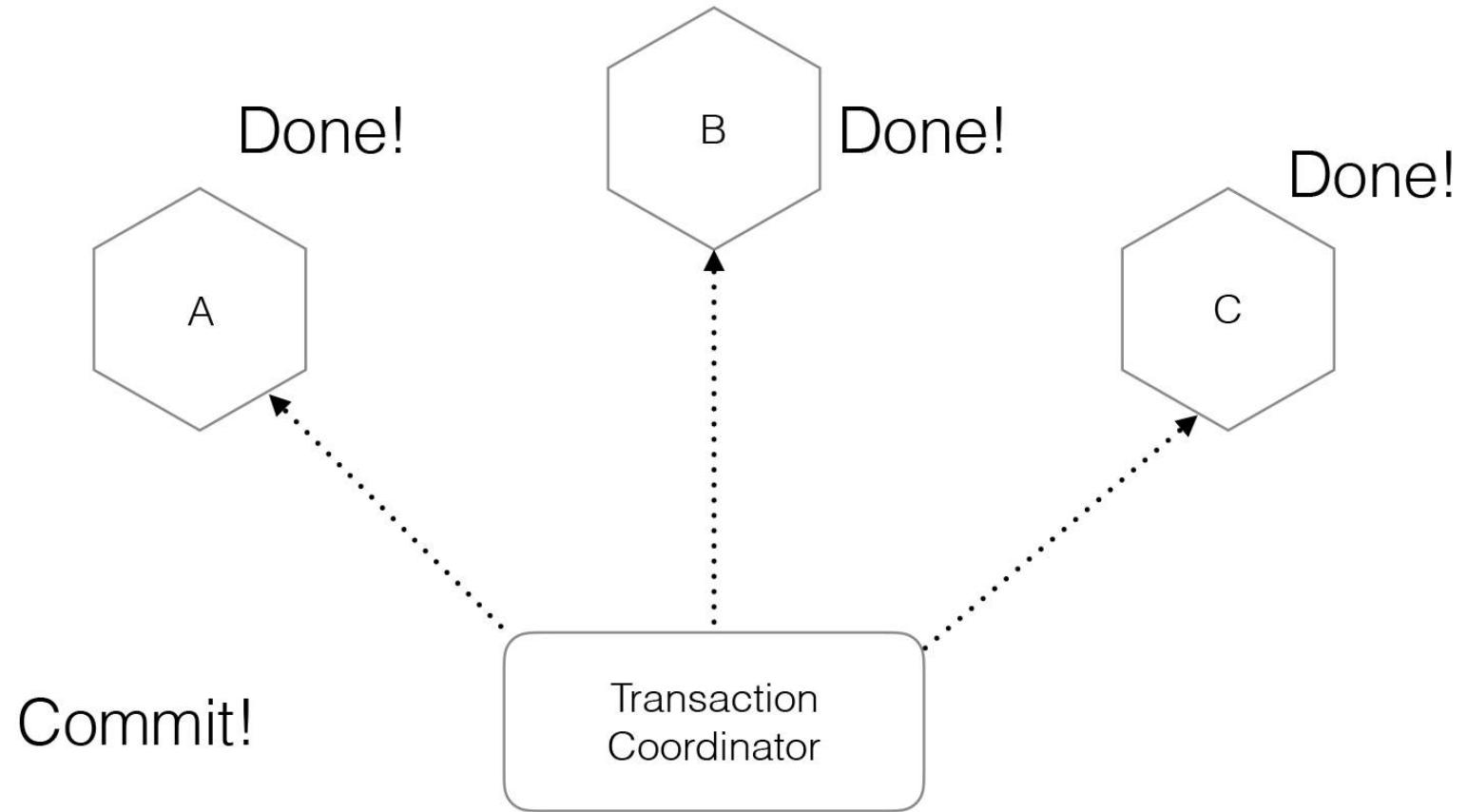
Step 2: Commit!

TWO PHASE COMMIT - SUCCESS



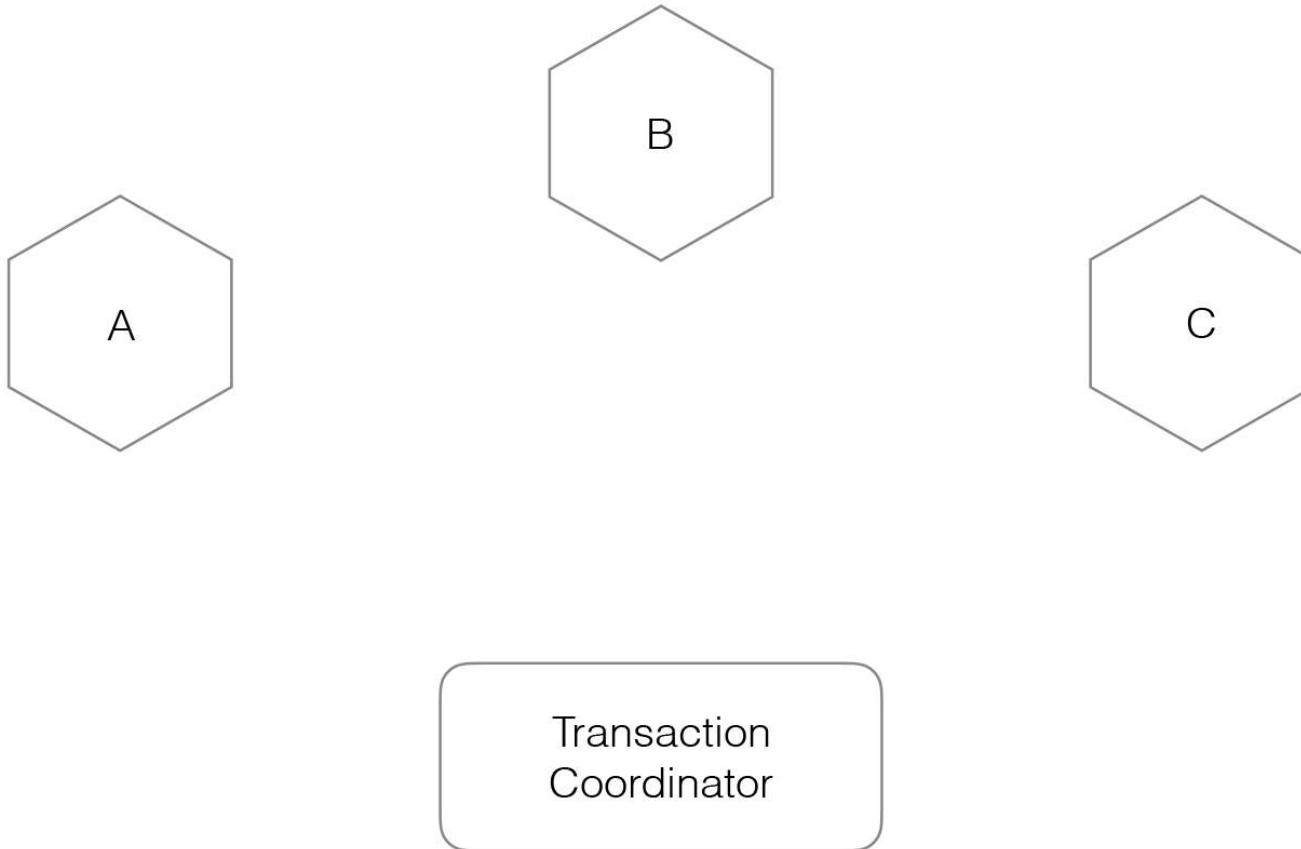
Step 2: Commit!

TWO PHASE COMMIT - SUCCESS



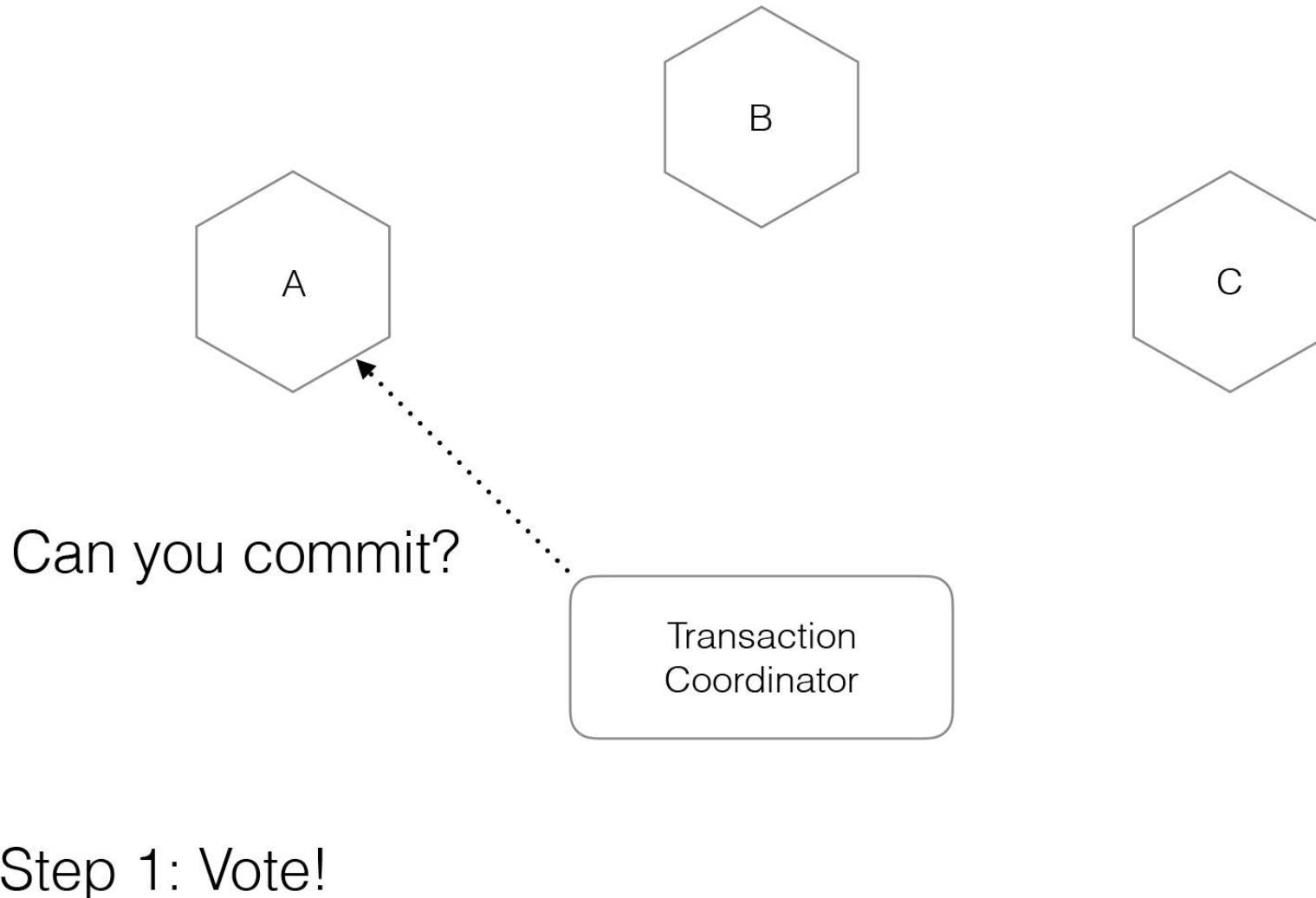
Step 2: Commit!

TWO PHASE COMMIT - FAILURE

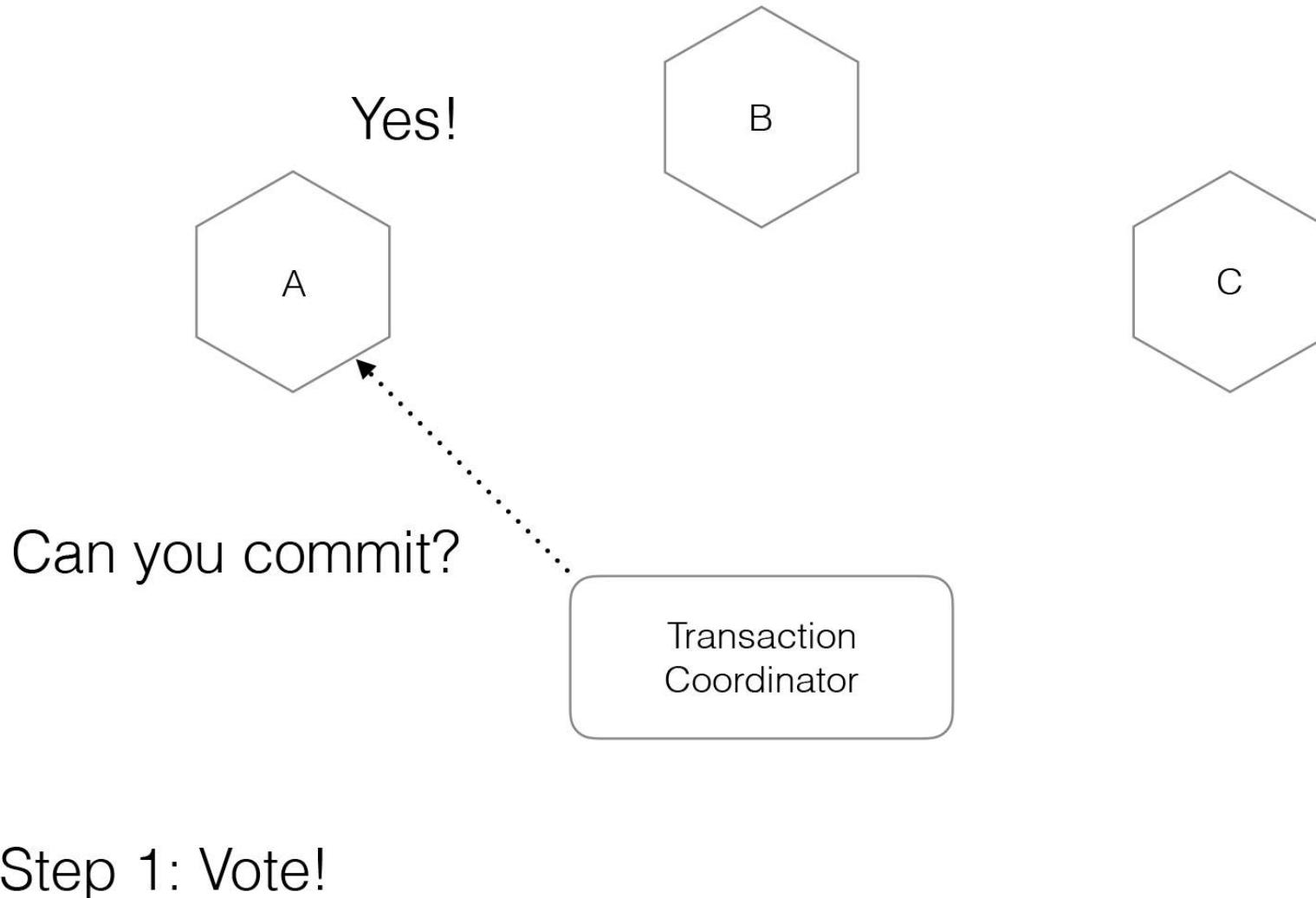


Step 1: Vote!

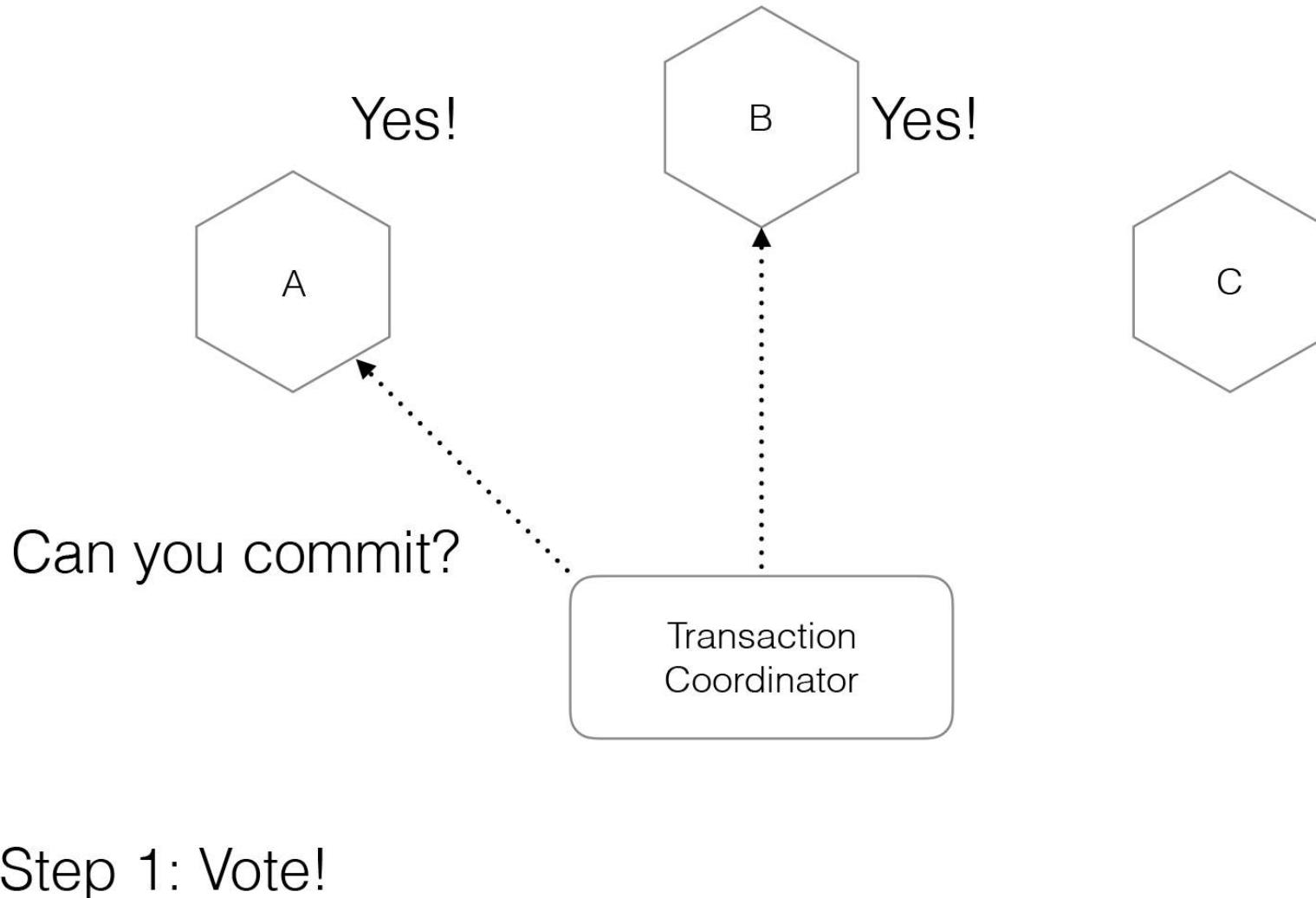
TWO PHASE COMMIT - FAILURE



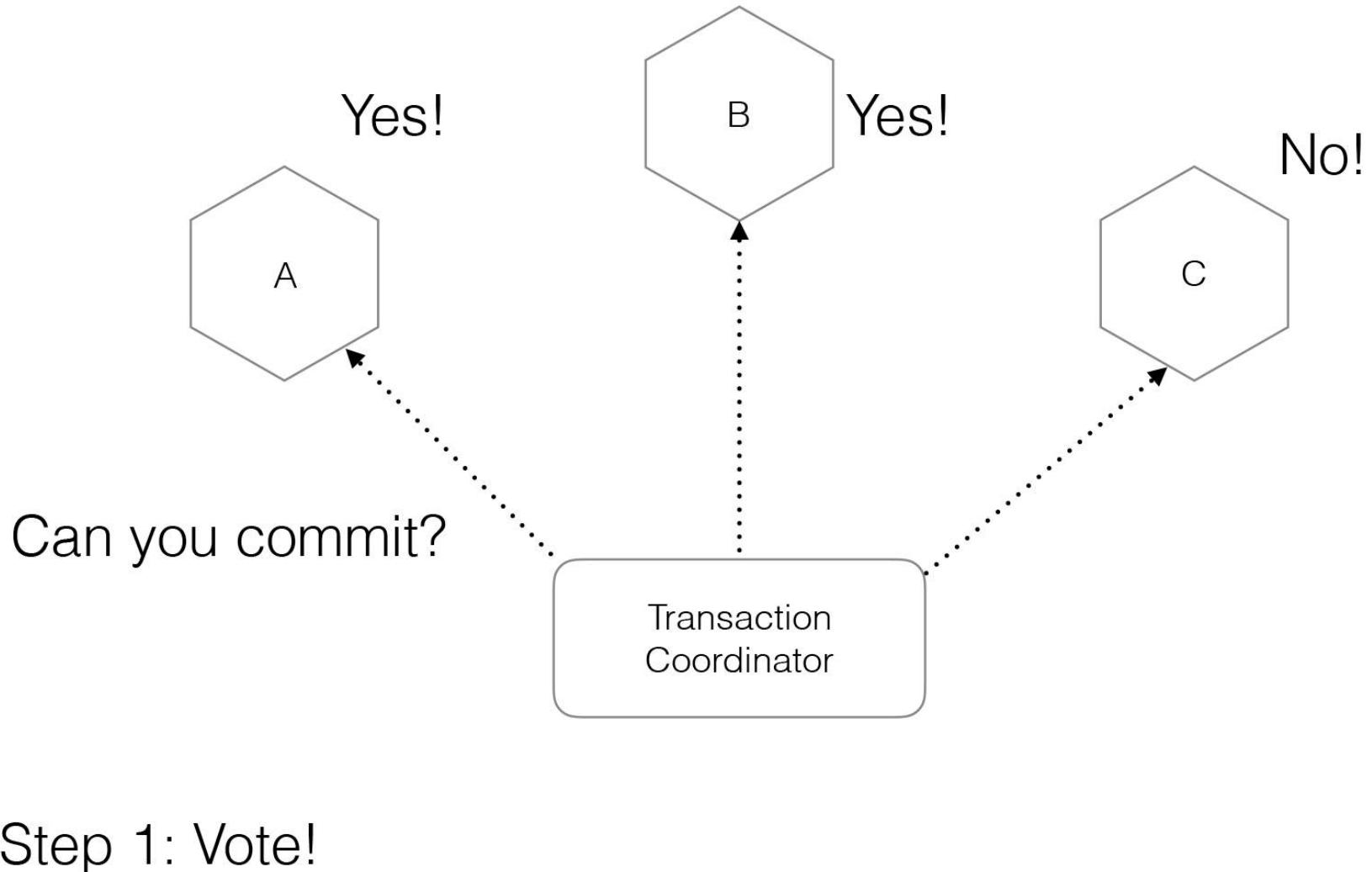
TWO PHASE COMMIT - FAILURE



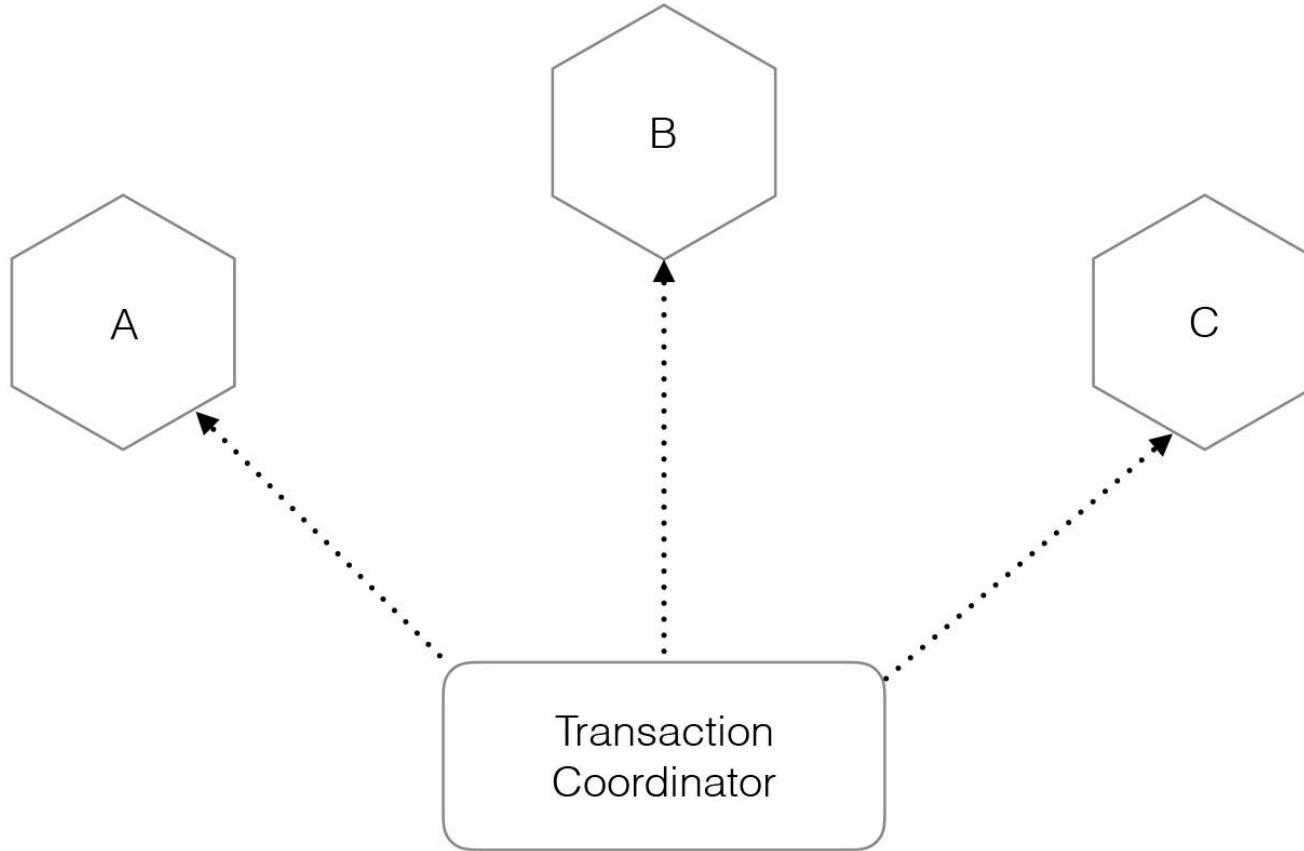
TWO PHASE COMMIT - FAILURE



TWO PHASE COMMIT - FAILURE

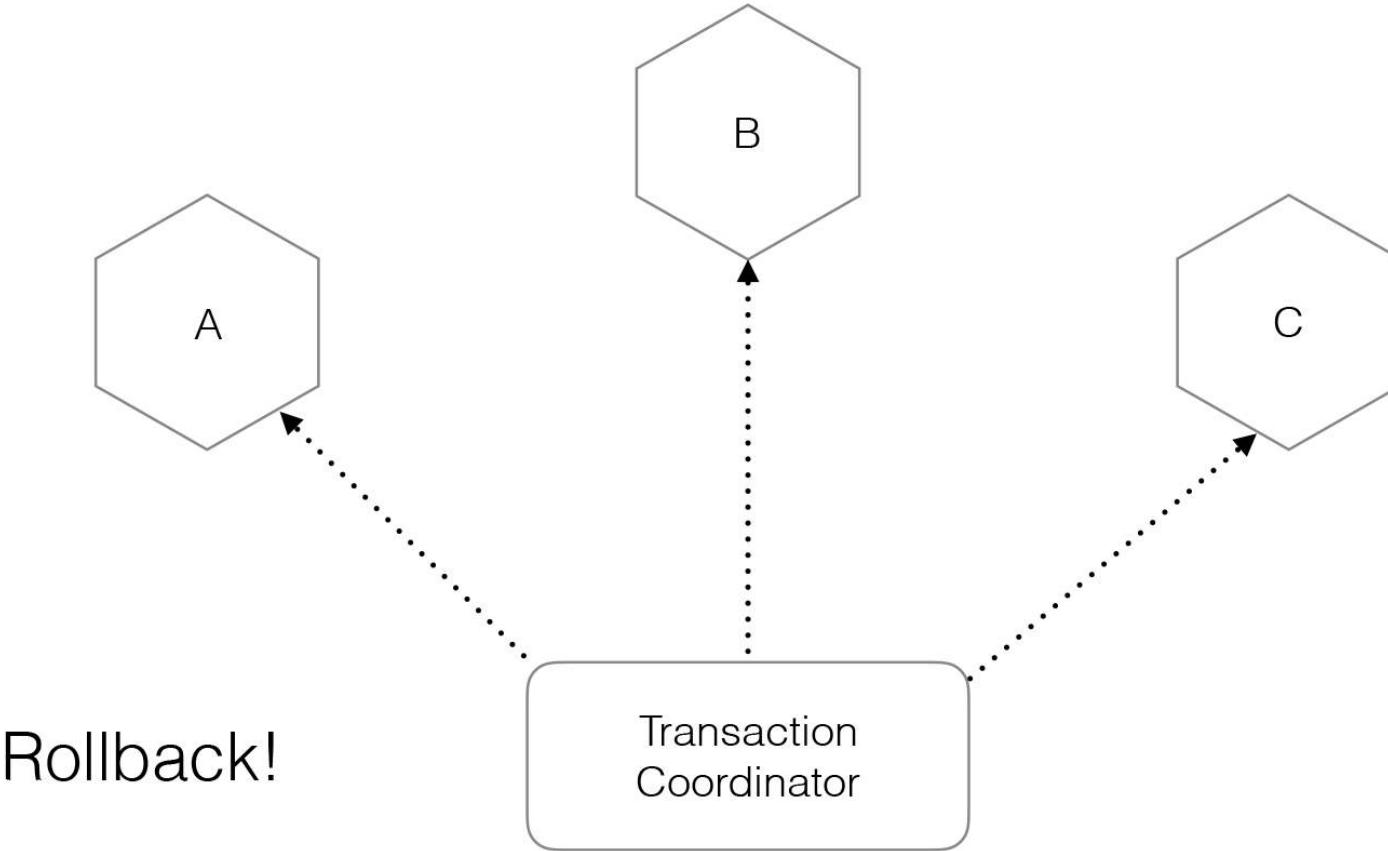


TWO PHASE COMMIT - FAILURE



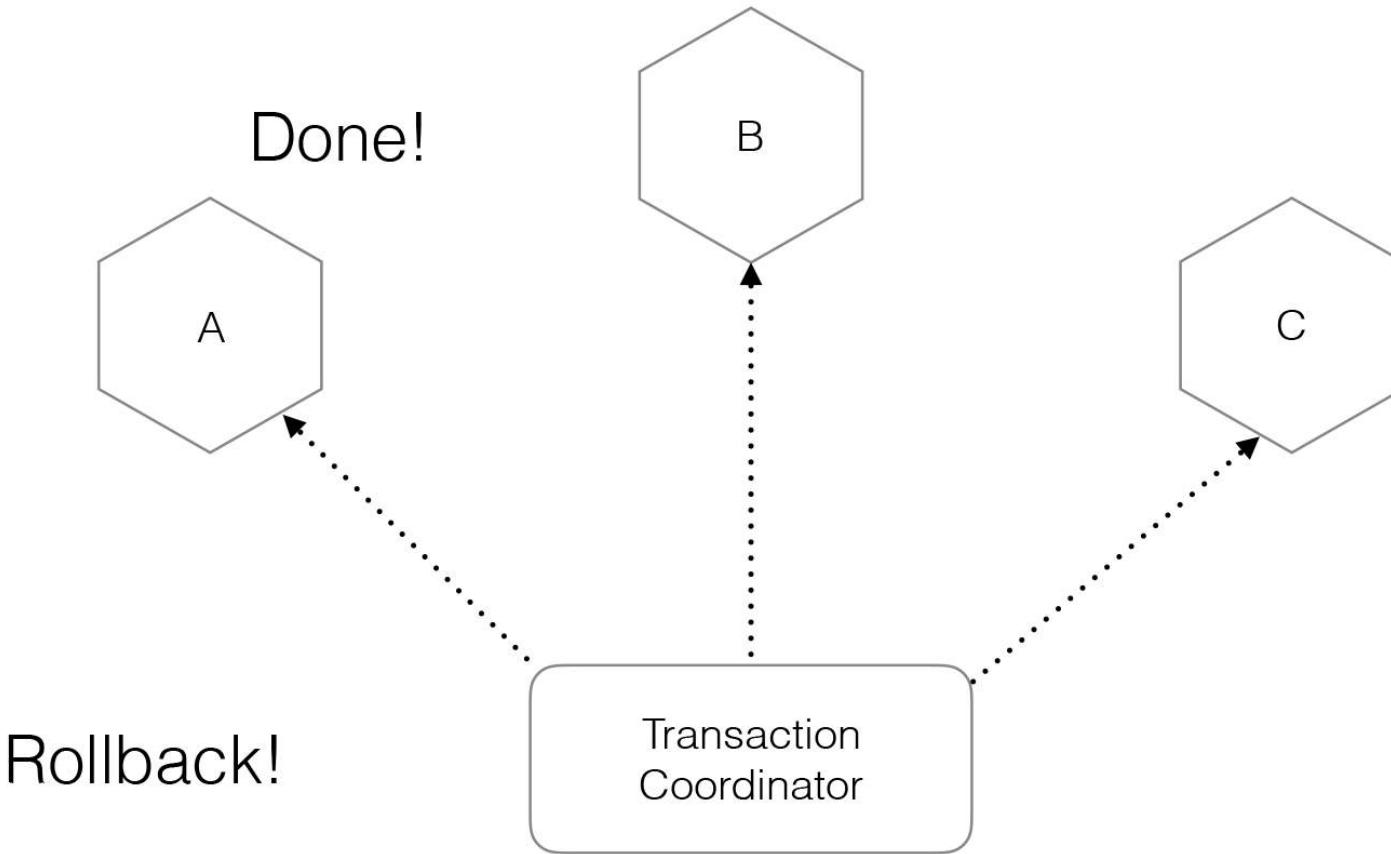
Step 2: Rollback

TWO PHASE COMMIT - FAILURE



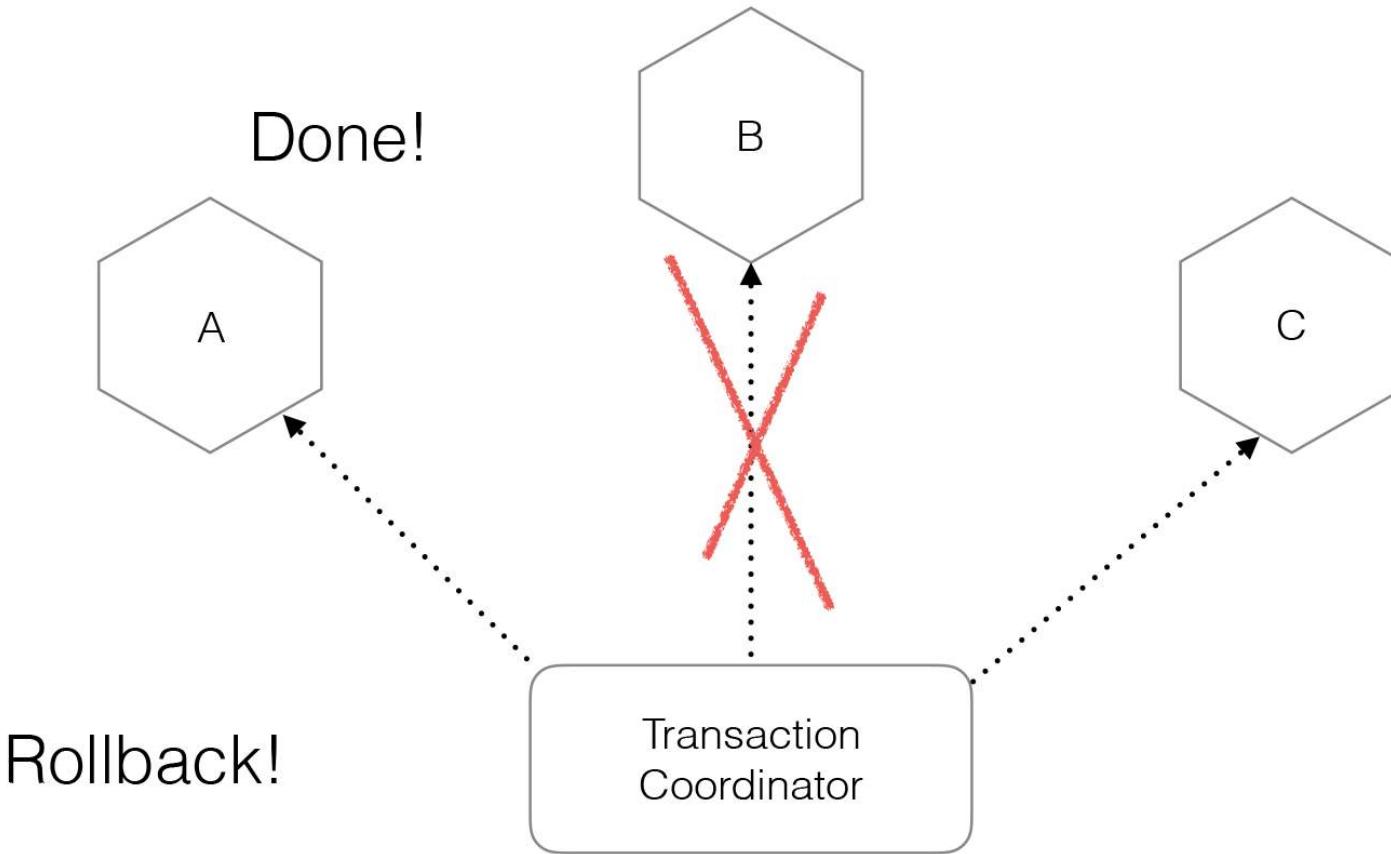
Step 2: Rollback

TWO PHASE COMMIT - FAILURE



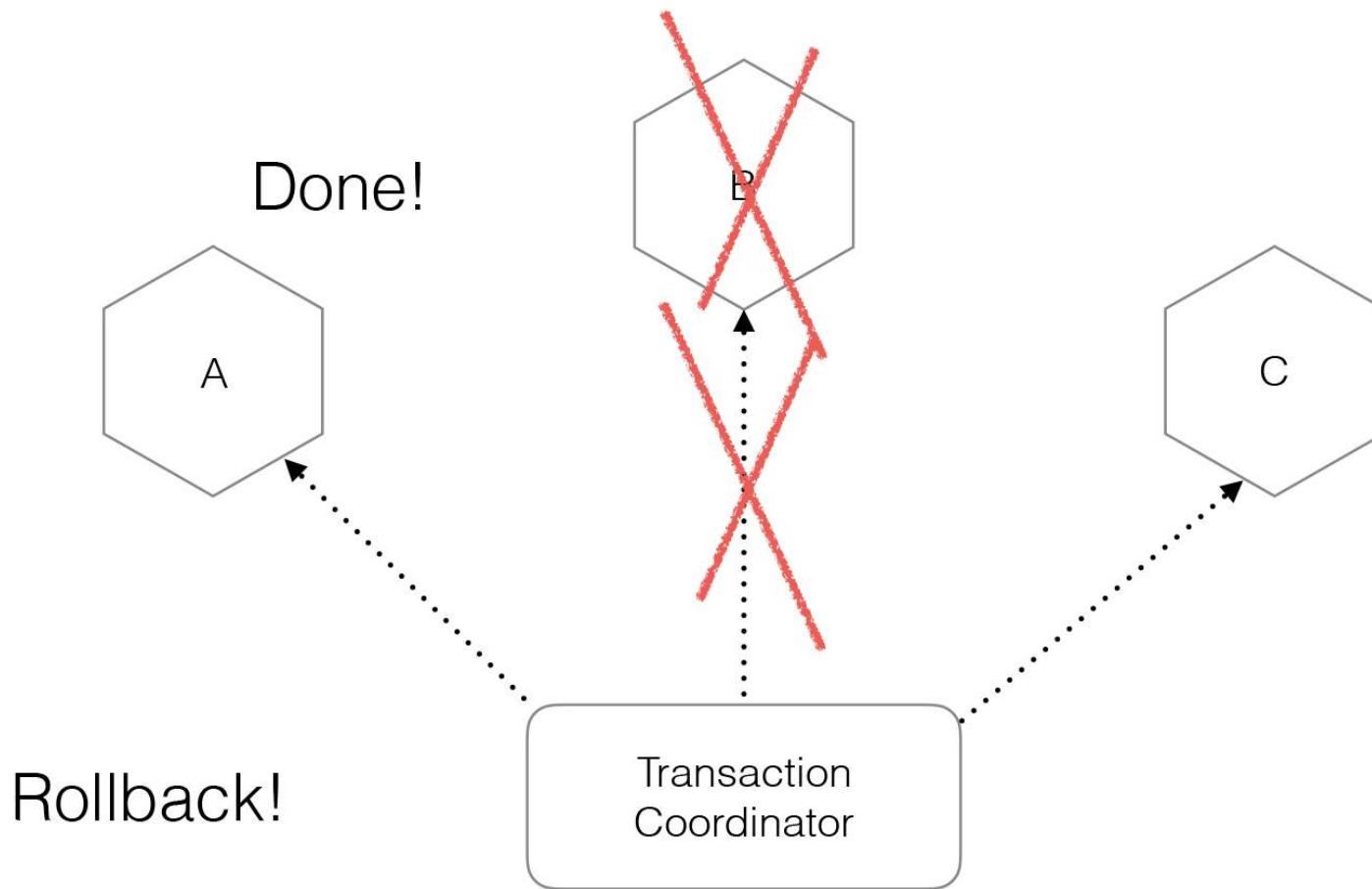
Step 2: Rollback

TWO PHASE COMMIT - FAILURE



Step 2: Rollback

TWO PHASE COMMIT - FAILURE



Step 2: Rollback

GOOGLE'S SPANNER

CLOUD SPANNER

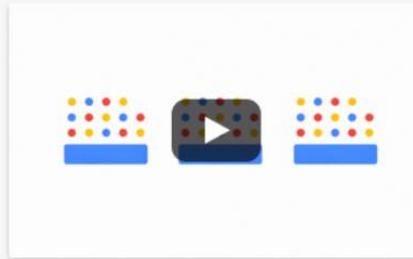
The first horizontally scalable, strongly consistent, relational database service



No-Compromise Relational Database Service

Cloud Spanner is the only enterprise-grade, globally-distributed, and strongly consistent database service built for the cloud specifically to combine the benefits of relational database structure with non-relational horizontal scale. This combination delivers high-performance transactions and strong consistency across rows, regions, and continents with an industry-leading 99.999% availability SLA, no planned downtime, and enterprise-grade security. Cloud Spanner revolutionizes database administration and management and makes application development more efficient.

In today's always-on, globally-distributed world, IT and developer efficiency, measured in app downtime and time to market, is one of an organization's most precious resources. The challenge of efficiently managing app database backends while at the same time giving developers the tools they need to build efficiently was previously a challenge.



<https://cloud.google.com/spanner/>

GOOGLE'S SPANNER

CLOUD SPANNER

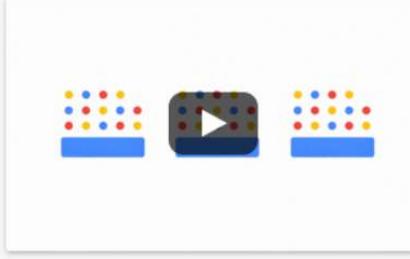
The first horizontally scalable, strongly consistent, relational database service

 TRY IT FREE

No-Compromise Relational Database Service

Cloud Spanner is the only enterprise-grade, globally-distributed, and strongly consistent database service built for the cloud specifically to combine the benefits of relational database structure with non-relational horizontal scale. This combination delivers high-performance transactions and strong consistency across rows, regions, and continents with an industry-leading 99.999% availability SLA, no planned downtime, and enterprise-grade security. Cloud Spanner revolutionizes database administration and management and makes application development more efficient.

In today's always-on, globally-distributed world, IT and developer efficiency, measured in app downtime and time to market, is one of an organization's most precious resources. The challenge of efficiently managing app database backends while at the same time giving developers the tools they need to build efficiently was previously a challenge.



Provides distributed transactions, but only within one data centre...

<https://cloud.google.com/spanner/>

GOOGLE'S SPANNER

CLOUD SPANNER

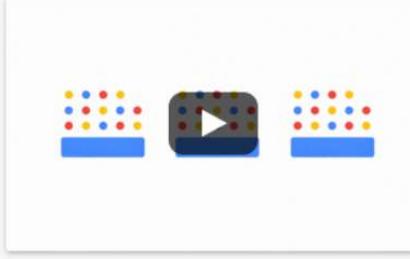
The first horizontally scalable, strongly consistent, relational database service

 TRY IT FREE

No-Compromise Relational Database Service

Cloud Spanner is the only enterprise-grade, globally-distributed, and strongly consistent database service built for the cloud specifically to combine the benefits of relational database structure with non-relational horizontal scale. This combination delivers high-performance transactions and strong consistency across rows, regions, and continents with an industry-leading 99.999% availability SLA, no planned downtime, and enterprise-grade security. Cloud Spanner revolutionizes database administration and management and makes application development more efficient.

In today's always-on, globally-distributed world, IT and developer efficiency, measured in app downtime and time to market, is one of an organization's most precious resources. The challenge of efficiently managing app database backends while at the same time giving developers the tools they need to build efficiently was previously a challenge.



Provides distributed transactions, but only within one data centre...

...and only in Google's data centres...

<https://cloud.google.com/spanner/>

GOOGLE'S SPANNER

CLOUD SPANNER

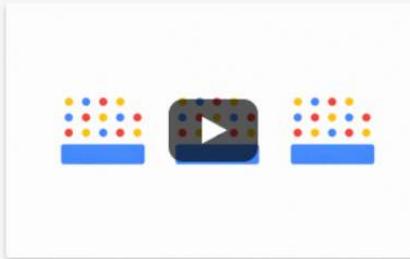
The first horizontally scalable, strongly consistent, relational database service

 TRY IT FREE

No-Compromise Relational Database Service

Cloud Spanner is the only enterprise-grade, globally-distributed, and strongly consistent database service built for the cloud specifically to combine the benefits of relational database structure with non-relational horizontal scale. This combination delivers high-performance transactions and strong consistency across rows, regions, and continents with an industry-leading 99.999% availability SLA, no planned downtime, and enterprise-grade security. Cloud Spanner revolutionizes database administration and management and makes application development more efficient.

In today's always-on, globally-distributed world, IT and developer efficiency, measured in app downtime and time to market, is one of an organization's most precious resources. The challenge of efficiently managing app database backends while at the same time giving developers the tools they need to build efficiently was previously a challenge.



Provides distributed transactions, but only within one data centre...

...and only in Google's data centres...

...and it makes use of atomic clocks in satellites

<https://cloud.google.com/spanner/>



Two phase commits attempt to give us DB-style transactions in a distributed system

But do we need that?

Sagas!

SAGAS

*Hector Garcia-Molina
Kenneth Salem*

Department of Computer Science
Princeton University
Princeton, N J 08544

Abstract

Long lived transactions (LLTs) hold on to database resources for relatively long periods of time, significantly delaying the termination of shorter and more common transactions. To alleviate these problems we propose the notion of a saga. A LLT is a saga if it can be written as a sequence of transactions that can be interleaved with other transactions. The database management system guarantees that either all the transactions in a saga are successfully completed or compensating transactions are run to amend a partial execution. Both the concept of saga and its implementation are relatively simple, but they have the potential to improve performance significantly. We analyze the various implementation issues related to sagas, including how they can be run on an existing system that does not directly support them. We also discuss techniques for database and LLT design that make it feasible to break up LLTs into sagas.

the majority of other transactions either because it accesses many database objects, it has lengthy computations, it pauses for inputs from the users, or a combination of these factors. Examples of LLTs are transactions to produce monthly account statements at a bank, transactions to process claims at an insurance company, and transactions to collect statistics over an entire database [Gray81a].

In most cases, LLTs present serious performance problems. Since they are transactions, the system must execute them as atomic actions, thus preserving the consistency of the database [Date81a, Ullm82a]. To make a transaction atomic, the system usually locks the objects accessed by the transaction until it commits, and this typically occurs at the end of the transaction. As a consequence, other transactions wishing to access the LLT's objects suffer a long locking delay. If LLTs are long because they access many database objects then other transactions are likely to suffer from an increased block-

SAGAS

*Hector Garcia-Molina
Kenneth Salem*

Department of Computer Science
Princeton University
Princeton, N J 08544

Abstract

Long lived transactions (LLTs) hold on to database resources for relatively long periods of time, significantly delaying the termination of shorter and more common transactions. To alleviate these problems we propose the notion of a saga. A LLT is a saga if it can be written as a sequence of transactions that can be interleaved with other transactions. The database management system guarantees that either all the transactions in a saga are successfully completed or compensating transactions are run to amend a partial execution. Both the concept of saga and its implementation are relatively simple, but they have the potential to improve performance significantly. We analyze the various implementation issues related to sagas, including how they can be run on an existing system that does not directly support them. We also discuss techniques for database and LLT design that make it feasible to break up LLTs into sagas.

the majority of other transactions either because it accesses many database objects, it has lengthy computations, it pauses for inputs from the users, or a combination of these factors. Examples of LLTs are transactions to produce monthly account statements at a bank, transactions to process claims at an insurance company, and transactions to collect statistics over an entire database [Gray81a].

In most cases, LLTs present serious performance problems. Since they are transactions, the system must execute them as atomic actions, thus preserving the consistency of the database [Date81a, Ullm82a]. To make a transaction atomic, the system usually locks the objects accessed by the transaction until it commits, and this typically occurs at the end of the transaction. As a consequence, other transactions wishing to access the LLT's objects suffer a long locking delay. If LLTs are long because they access many database objects then other transactions are likely to suffer from an increased block-

How to handle long-lived transactions on a single DB

SAGAS

*Hector Garcia-Molina
Kenneth Salem*

Department of Computer Science
Princeton University
Princeton, N J 08544

Abstract

Long lived transactions (LLTs) hold on to database resources for relatively long periods of time, significantly delaying the termination of shorter and more common transactions. To alleviate these problems we propose the notion of a saga. A LLT is a saga if it can be written as a sequence of transactions that can be interleaved with other transactions. The database management system guarantees that either all the transactions in a saga are successfully completed or compensating transactions are run to amend a partial execution. Both the concept of saga and its implementation are relatively simple, but they have the potential to improve performance significantly. We analyze the various implementation issues related to sagas, including how they can be run on an existing system that does not directly support them. We also discuss techniques for database and LLT design that make it feasible to break up LLTs into sagas.

the majority of other transactions either because it accesses many database objects, it has lengthy computations, it pauses for inputs from the users, or a combination of these factors. Examples of LLTs are transactions to produce monthly account statements at a bank, transactions to process claims at an insurance company, and transactions to collect statistics over an entire database [Gray81a].

In most cases, LLTs present serious performance problems. Since they are transactions, the system must execute them as atomic actions, thus preserving the consistency of the database [Date81a, Ullm82a]. To make a transaction atomic, the system usually locks the objects accessed by the transaction until it commits, and this typically occurs at the end of the transaction. As a consequence, other transactions wishing to access the LLT's objects suffer a long locking delay. If LLTs are long because they access many database objects then other transactions are likely to suffer from an increased block-

How to handle long-lived transactions on a single DB

Trying to avoid locking for the duration of the transaction

SAGAS - OVERVIEW

Start LLT

End LLT

SAGAS - OVERVIEW

Start LLT

Start T1
End T1

End LLT

SAGAS - OVERVIEW

Start LLT

Start T1
End T1

Start T2
End T2

End LLT

SAGAS - OVERVIEW

Start LLT

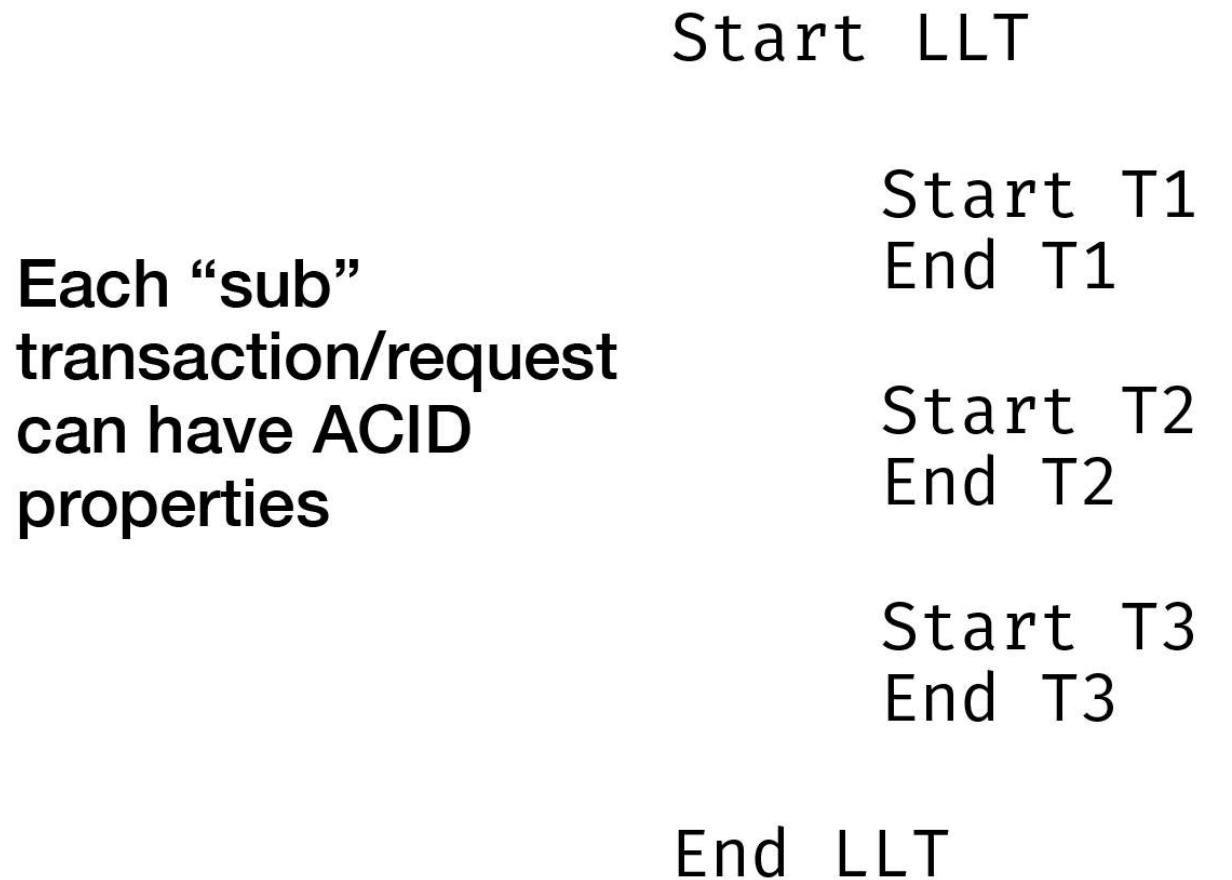
Start T1
End T1

Start T2
End T2

Start T3
End T3

End LLT

SAGAS - OVERVIEW



SAGAS - OVERVIEW

Each “sub” transaction/request can have ACID properties

Start LLT

Start T1
End T1

Start T2
End T2

~~Start T3~~
~~End T3~~

End LLT

But what happens if we have a failure?

SAGAS - OVERVIEW

Each “sub” transaction/request can have ACID properties

Start LLT

Start T1
End T1

Start T2
End T2

~~Start T3~~
~~End T3~~

End LLT

But what happens if we have a failure?

Each sub-transaction needs to have a matching compensating transaction

SAGAS - OVERVIEW

Each “sub” transaction/request can have ACID properties

Start LLT

Start T1
End T1

Start T2
End T2

~~Start T3
End T3~~

End LLT

But what happens if we have a failure?

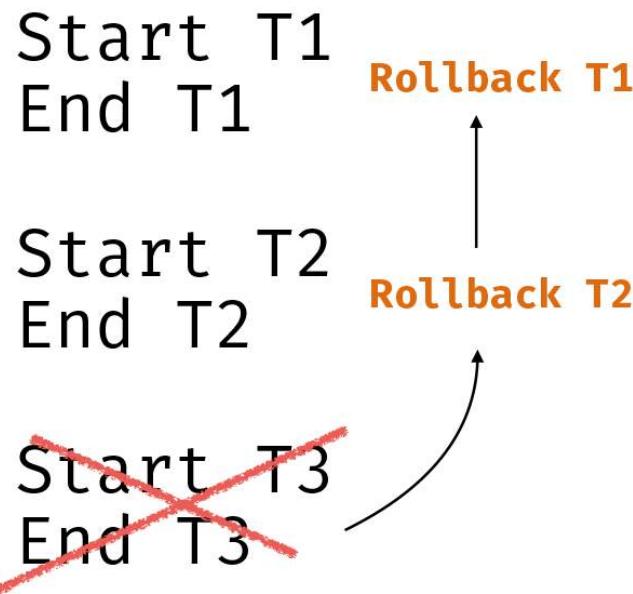
Each sub-transaction needs to have a matching compensating transaction

Rollback T2

SAGAS - OVERVIEW

Each “sub” transaction/request can have ACID properties

Start LLT

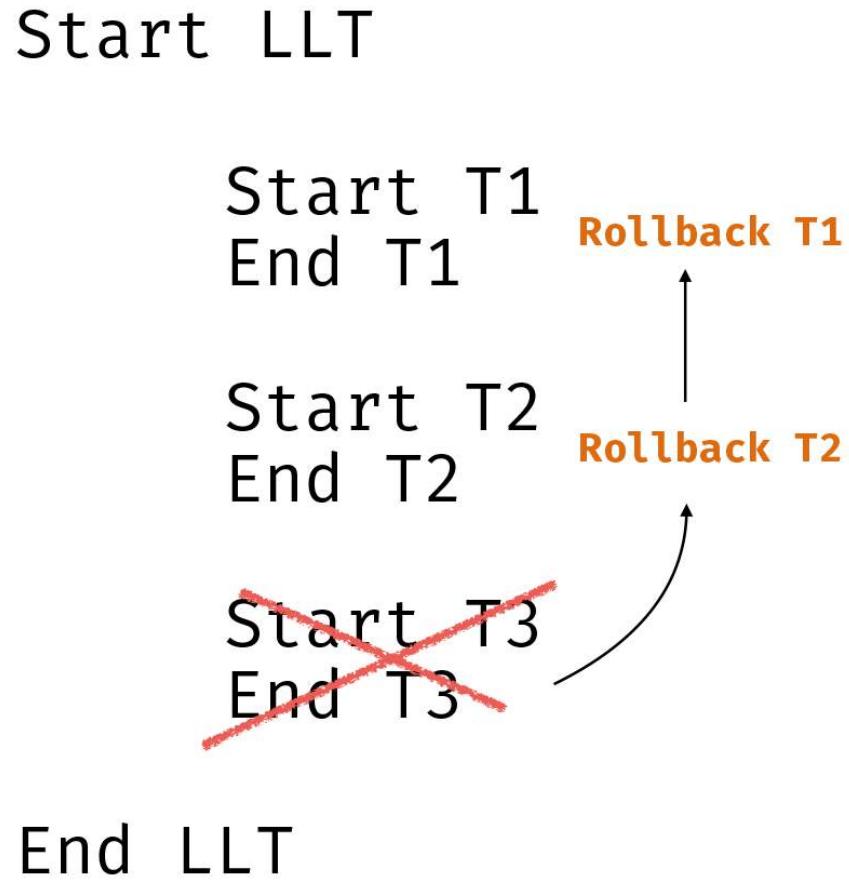


But what happens if we have a failure?

Each sub-transaction needs to have a matching compensating transaction

SAGAS - OVERVIEW

Each “sub” transaction/request can have ACID properties



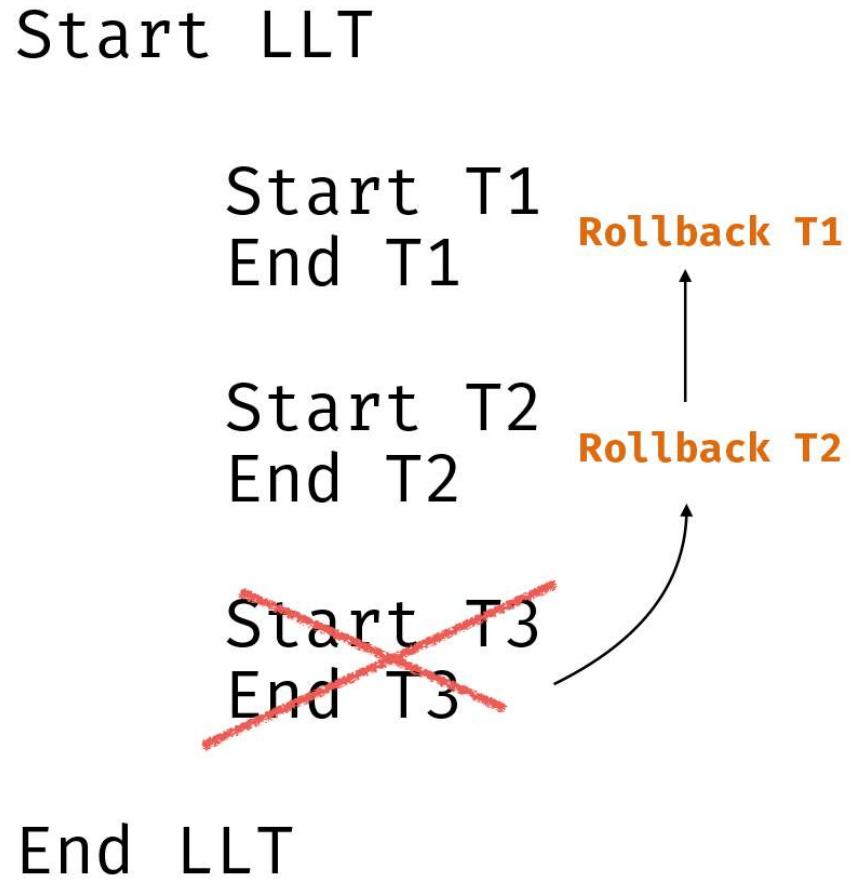
But what happens if we have a failure?

Each sub-transaction needs to have a matching compensating transaction

Rollback is a semantic rollback

SAGAS - OVERVIEW

Each “sub” transaction/request can have ACID properties



But what happens if we have a failure?

Each sub-transaction needs to have a matching compensating transaction

Rollback is a semantic rollback

Rollbacks need to be idempotent

EXAMPLE

Booking a holiday

End

EXAMPLE

Booking a holiday

Start Car Booking
End Car Booking

End

EXAMPLE

Booking a holiday

Start Car Booking
End Car Booking

Start Hotel Booking
End Hotel Booking

End

EXAMPLE

Booking a holiday

Start Car Booking
End Car Booking

Start Hotel Booking
End Hotel Booking

~~Start Flight Booking
End Flight Booking~~

End

EXAMPLE

Booking a holiday

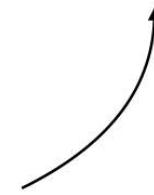
Start Car Booking
End Car Booking

Start Hotel Booking
End Hotel Booking

~~Start Flight Booking~~
~~End Flight Booking~~

End

Cancel Hotel



EXAMPLE

Booking a holiday

Start Car Booking
End Car Booking

Start Hotel Booking
End Hotel Booking

~~Start Flight Booking~~
~~End Flight Booking~~

End

Cancel Car

Cancel Hotel



FAILURES?

Fail Backward

Fail Forward

State machines across service boundaries

WHAT KEY THINGS DO YOU NEED FOR A SAGA PATTERN?

WHAT KEY THINGS DO YOU NEED FOR A SAGA PATTERN?

Execution Log

A record of what has happened

WHAT KEY THINGS DO YOU NEED FOR A SAGA PATTERN?

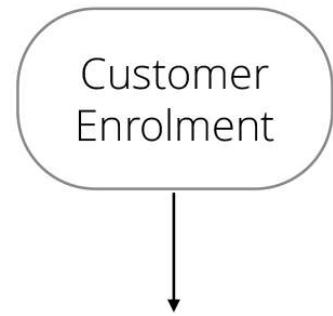
Execution Log

A record of what has happened

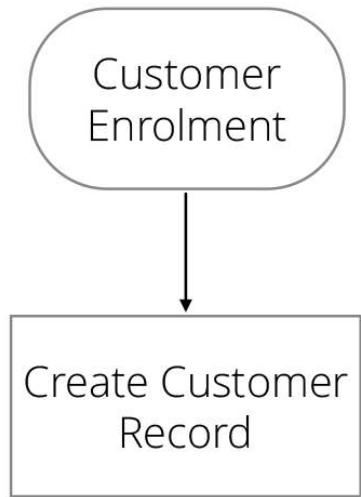
Execution Co-ordinator

Something to tell you what to do

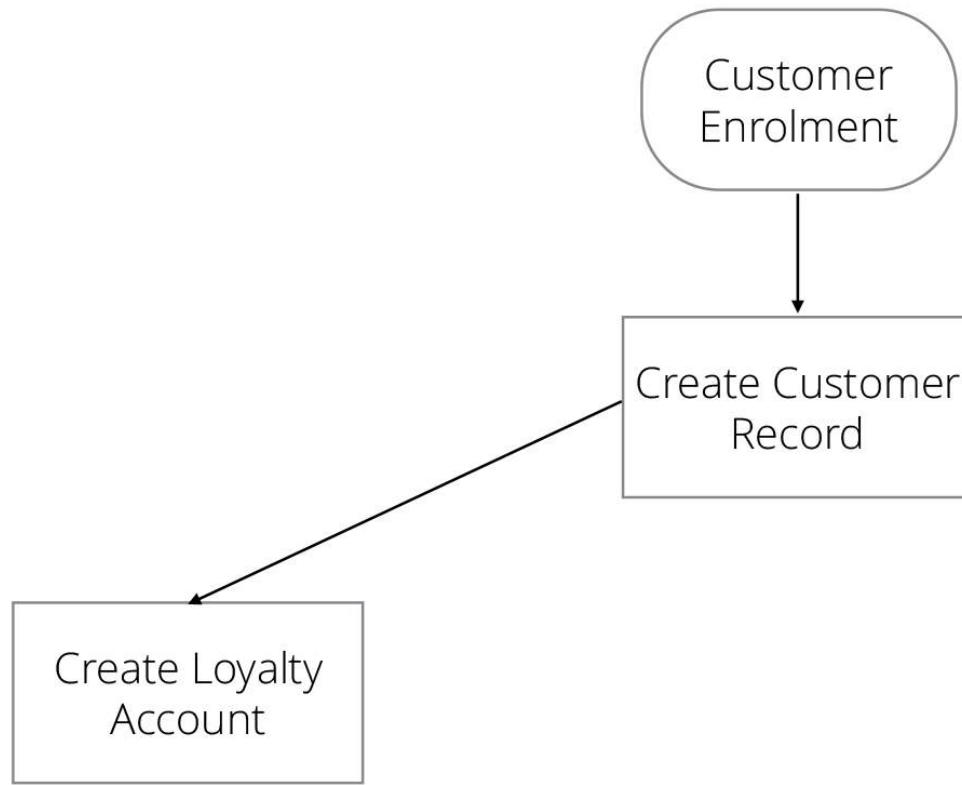
EXAMPLE BUSINESS PROCESS - CUSTOMER ENROLMENT



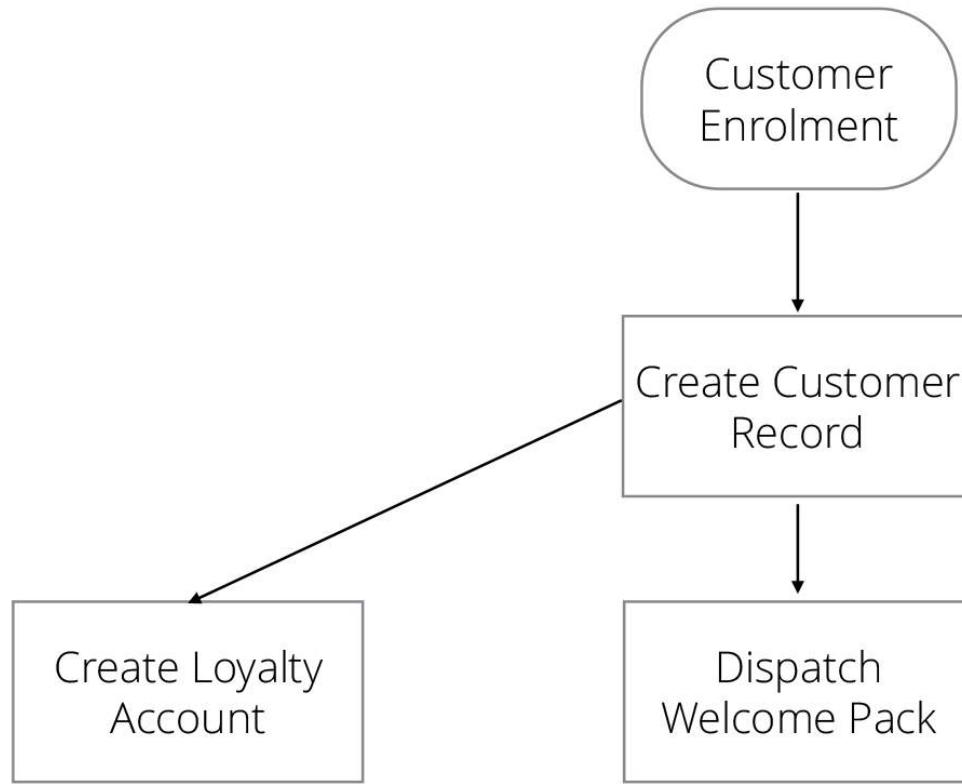
EXAMPLE BUSINESS PROCESS - CUSTOMER ENROLMENT



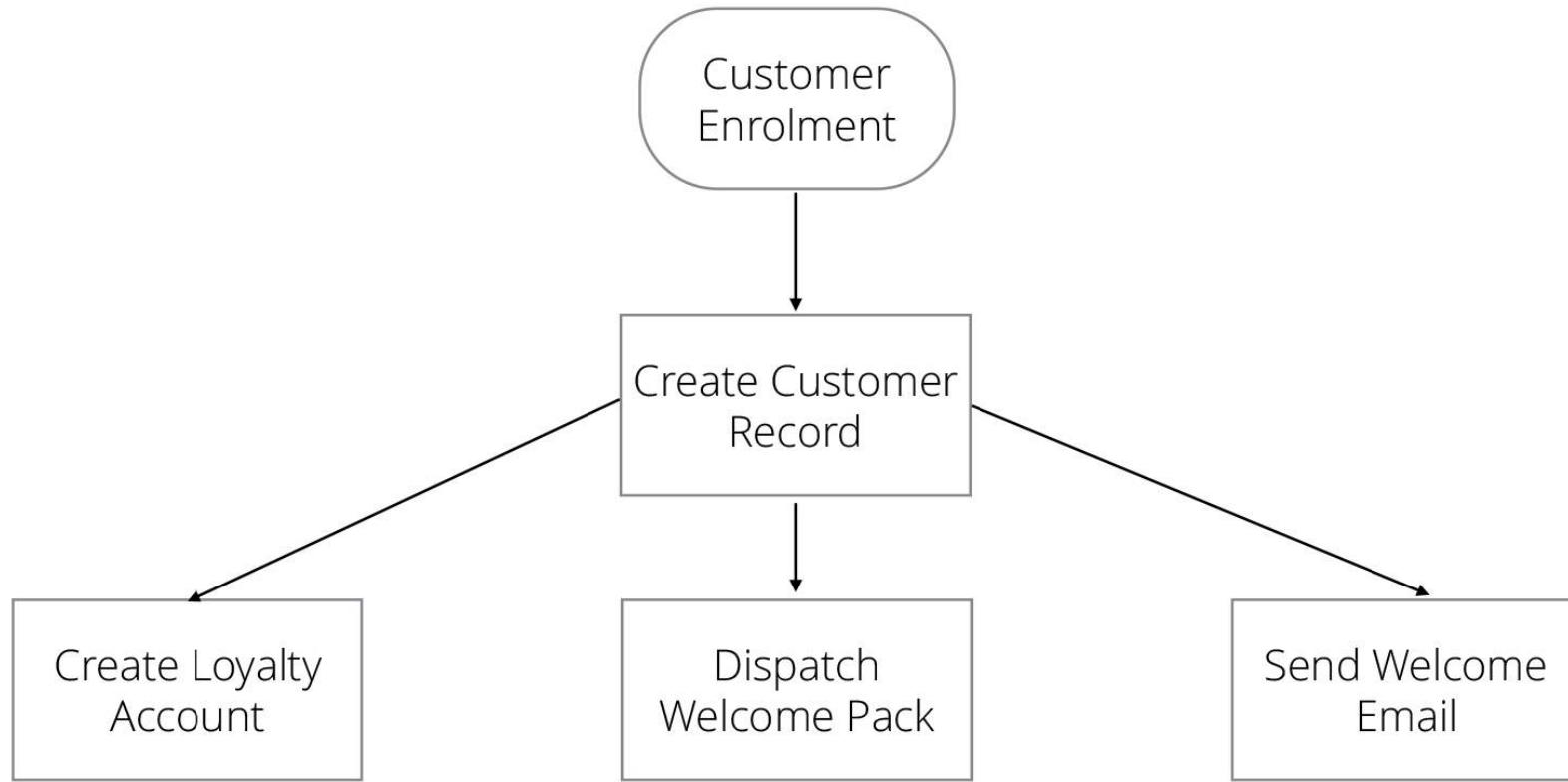
EXAMPLE BUSINESS PROCESS - CUSTOMER ENROLMENT



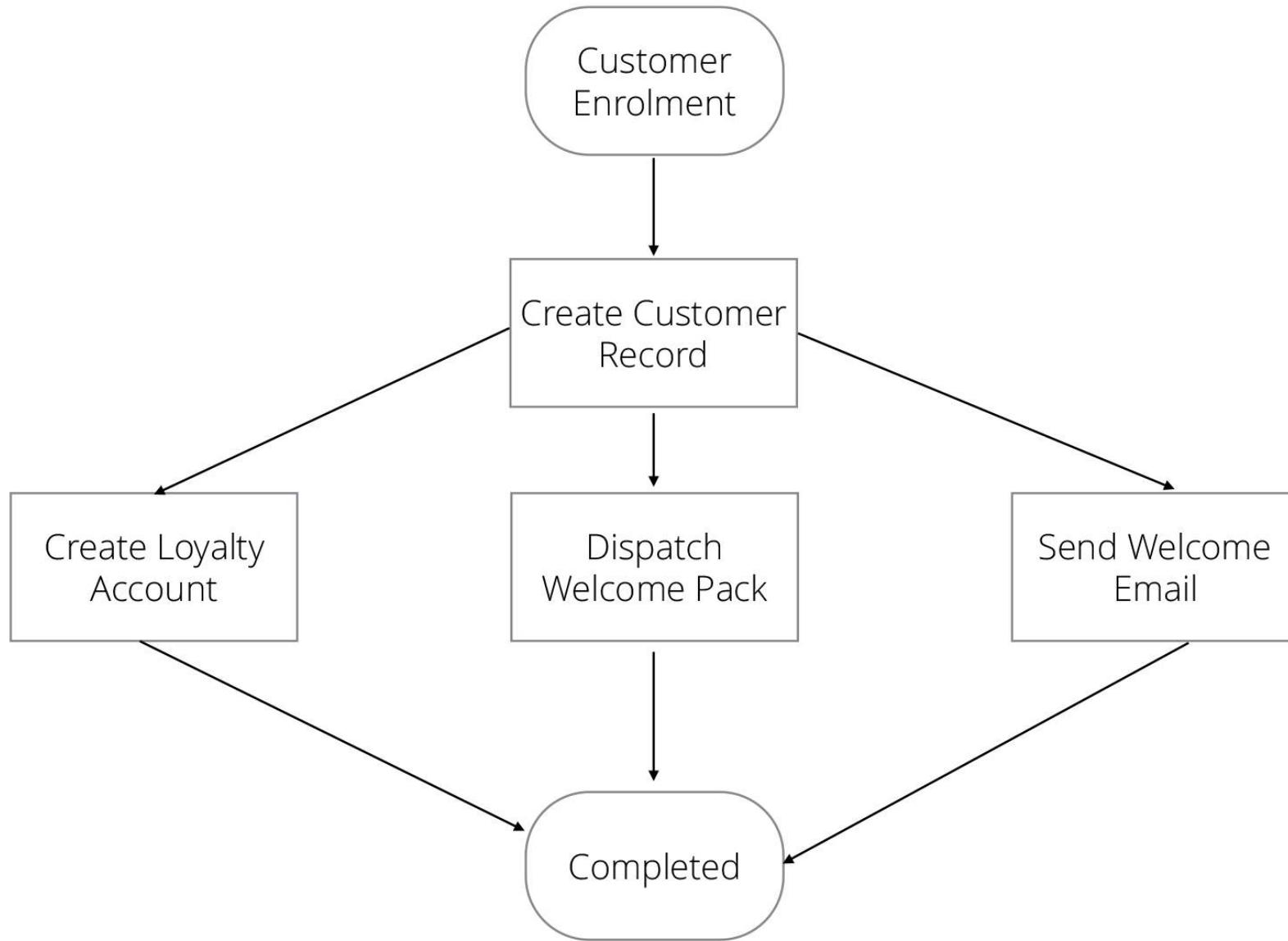
EXAMPLE BUSINESS PROCESS - CUSTOMER ENROLMENT



EXAMPLE BUSINESS PROCESS - CUSTOMER ENROLMENT



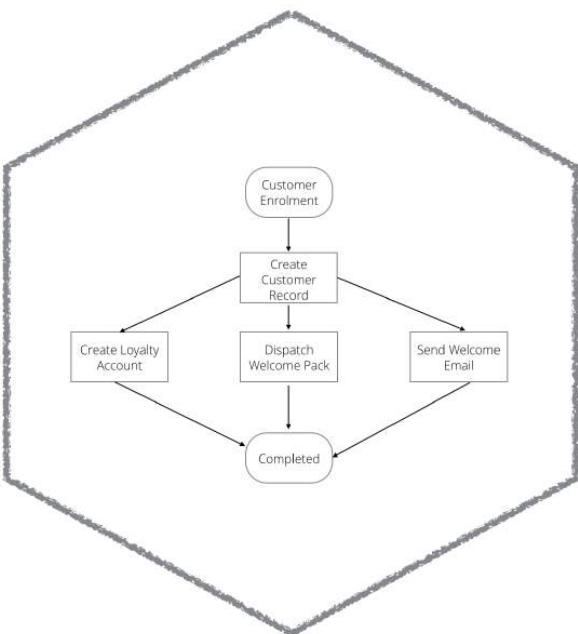
EXAMPLE BUSINESS PROCESS - CUSTOMER ENROLMENT



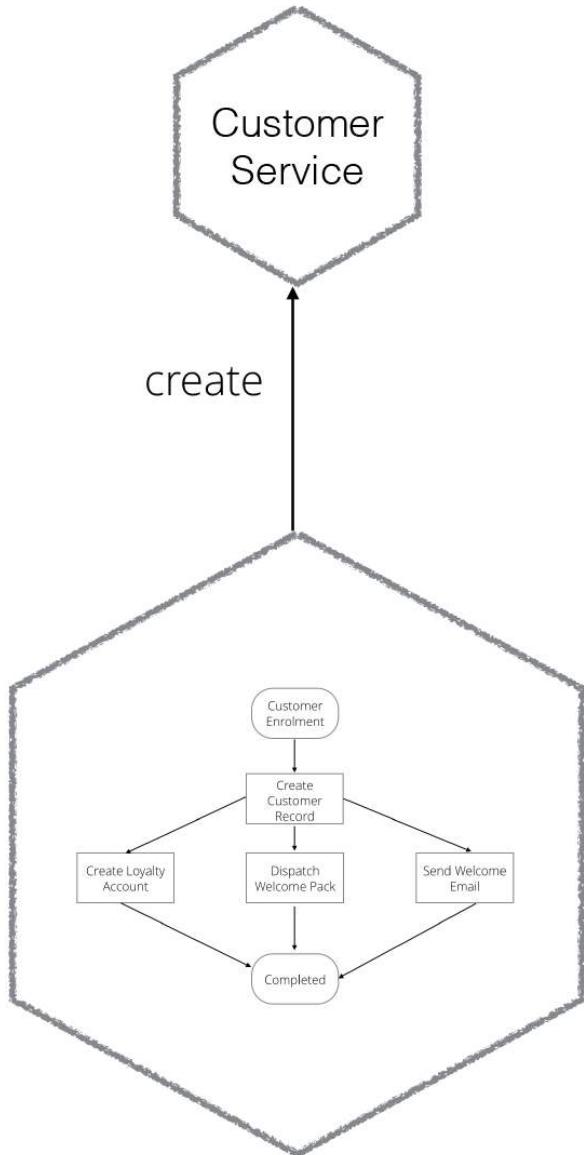
Two different styles of Sagas

Choreography vs Orchestration

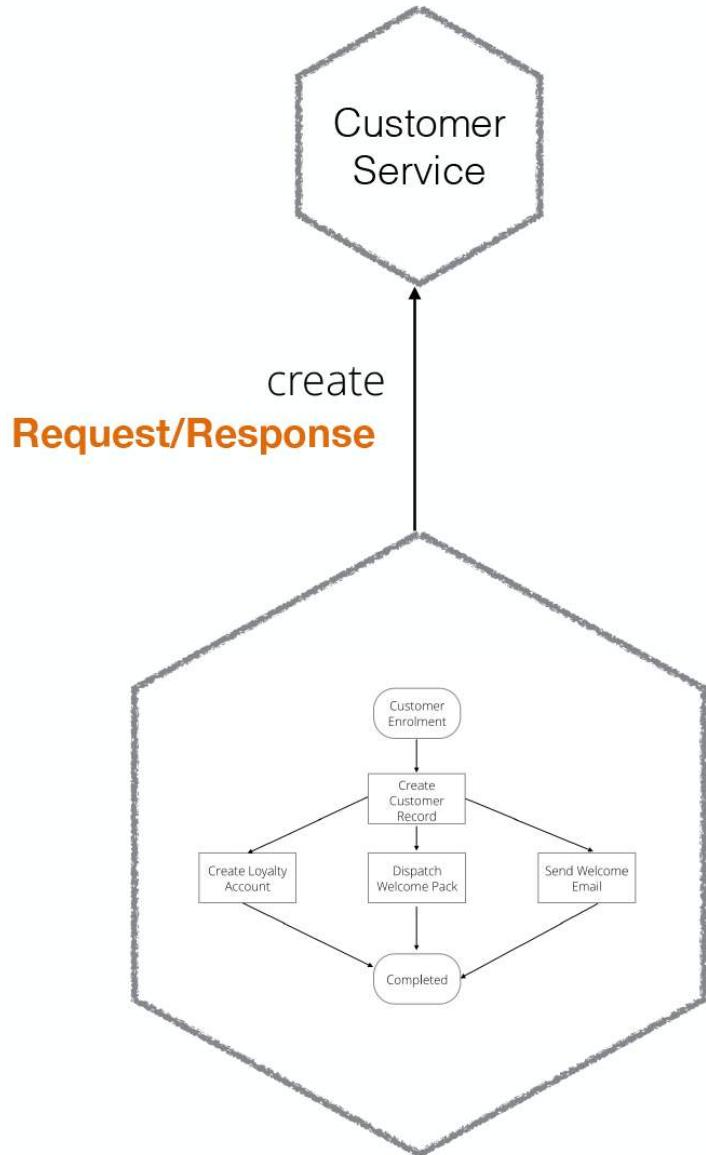
ORCHESTRATION



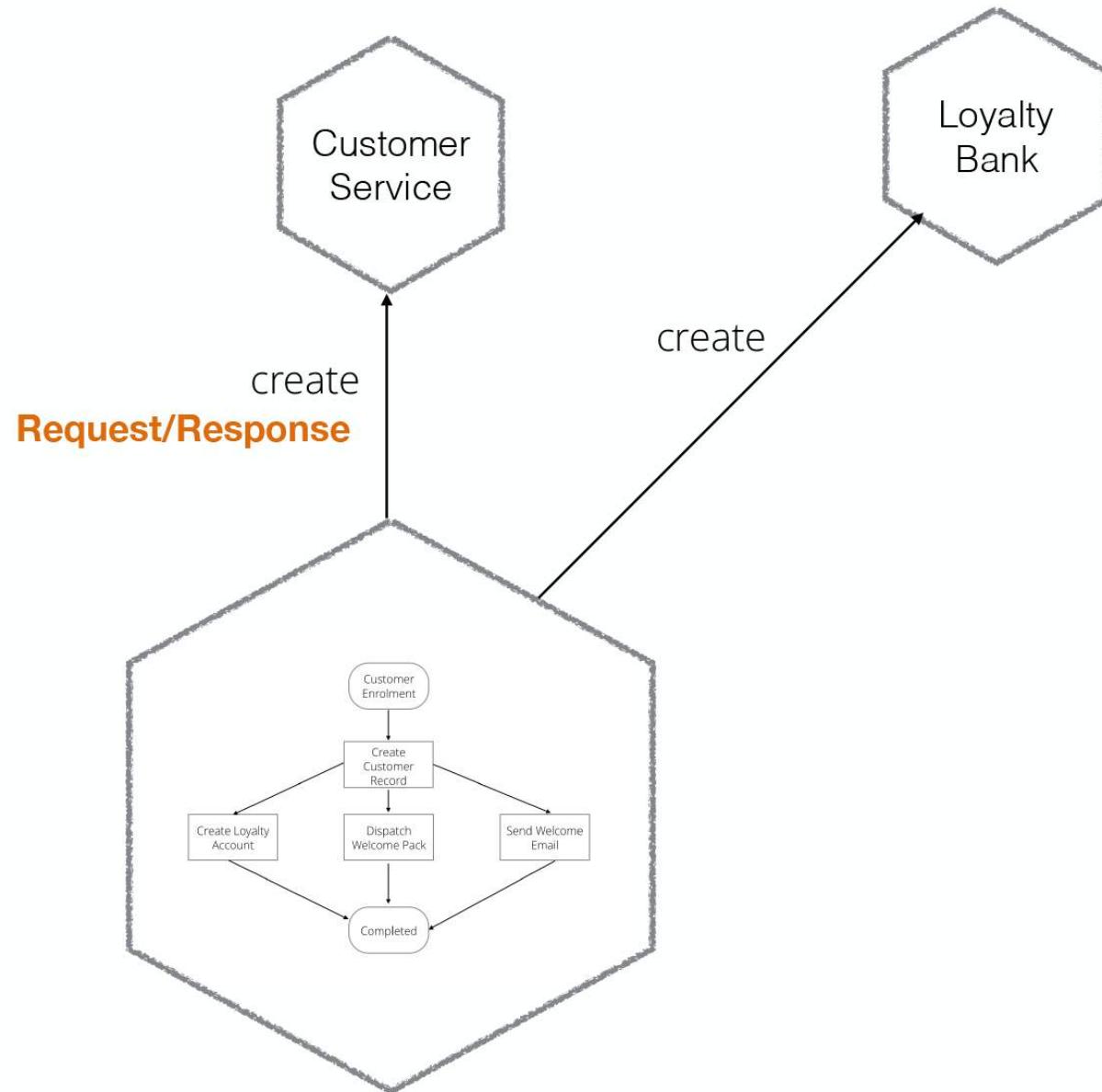
ORCHESTRATION



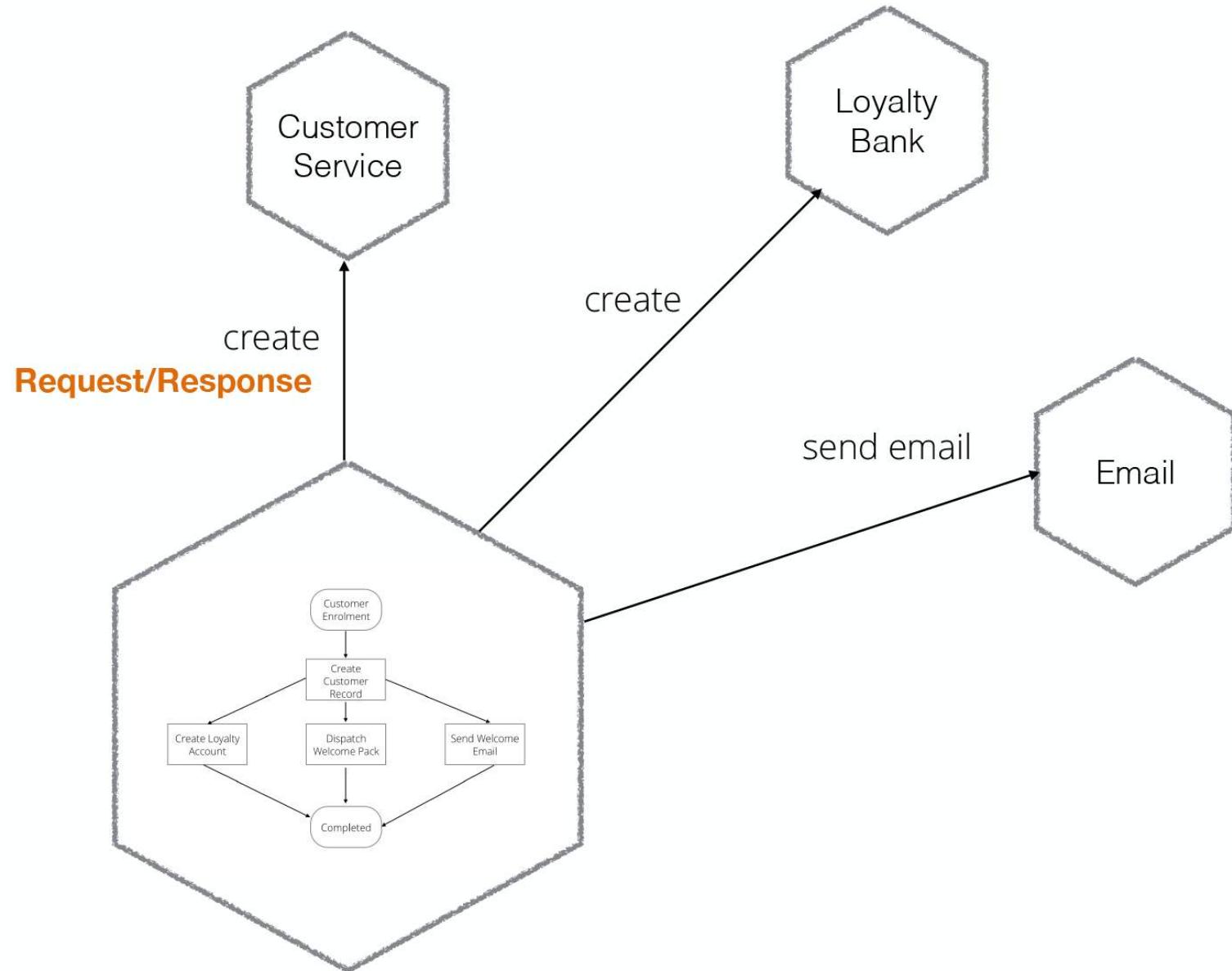
ORCHESTRATION



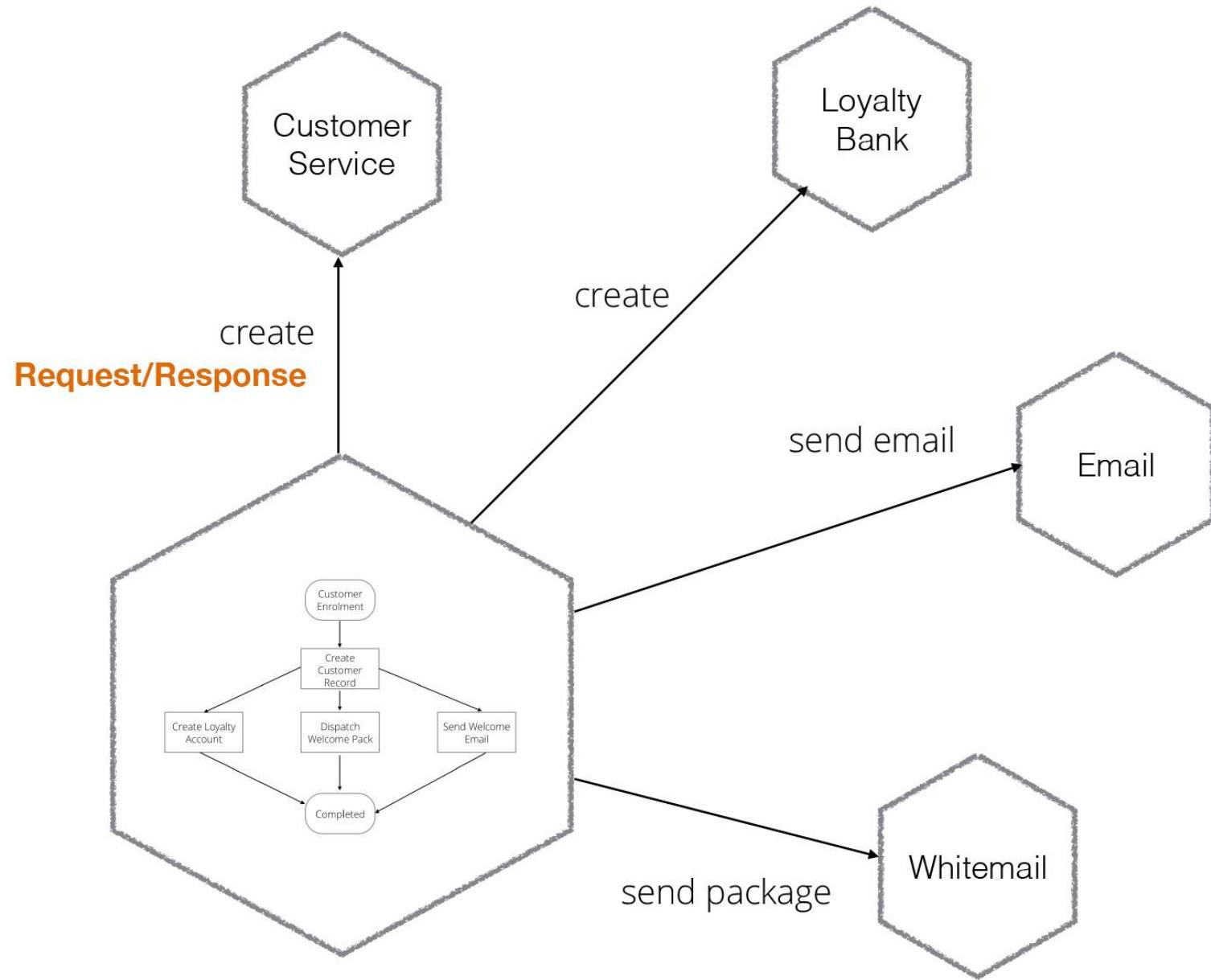
ORCHESTRATION



ORCHESTRATION



ORCHESTRATION



Beware of BPM!

Pros

Pros

Explicit representation of business process

Pros

Explicit representation of business process

Know in-line if there has been a problem

Pros

Explicit representation of business process

Know in-line if there has been a problem

Cons

Pros

Explicit representation of business process

Know in-line if there has been a problem

Cons

Can be fairly coupled

Pros

Explicit representation of business process

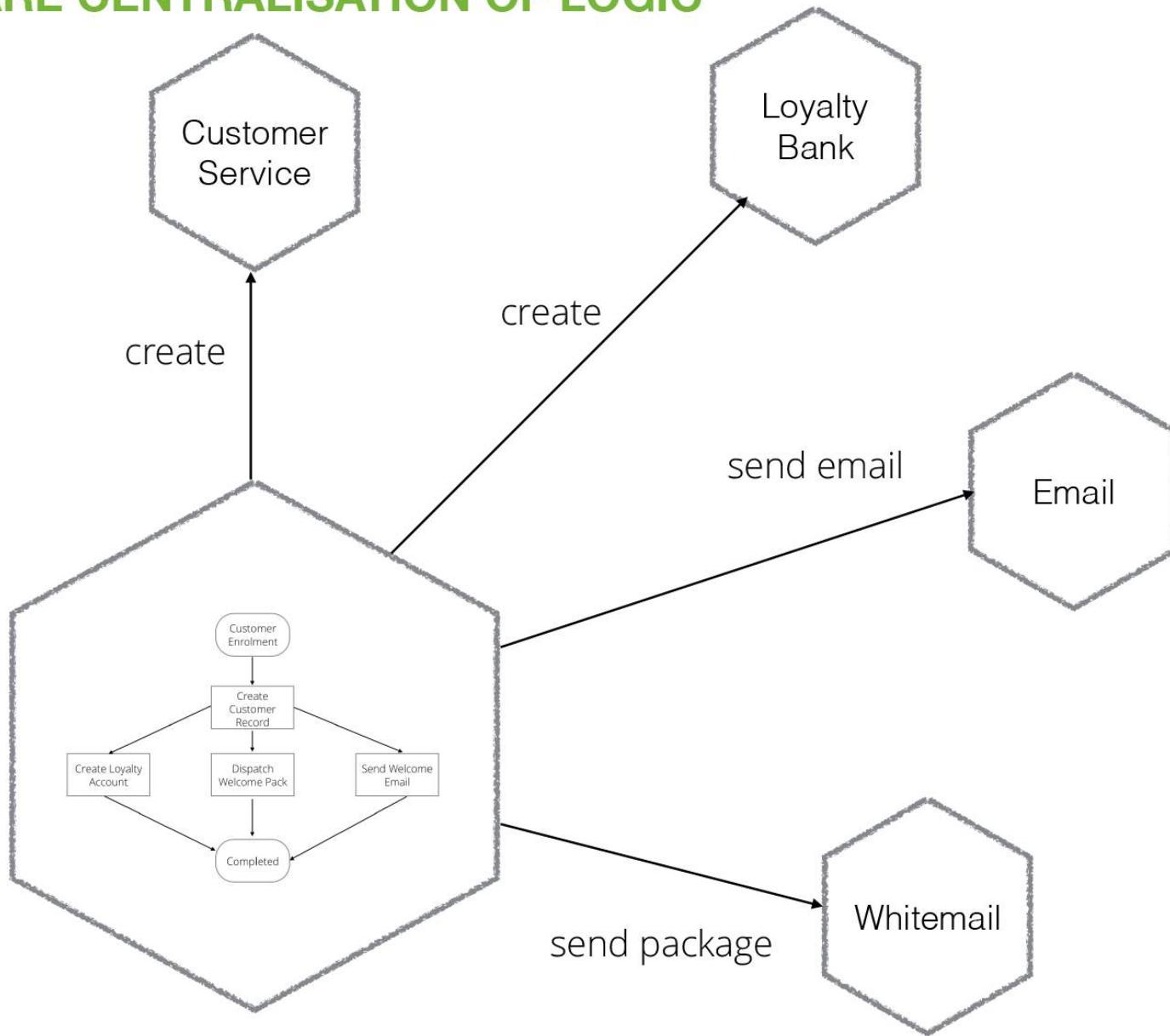
Know in-line if there has been a problem

Cons

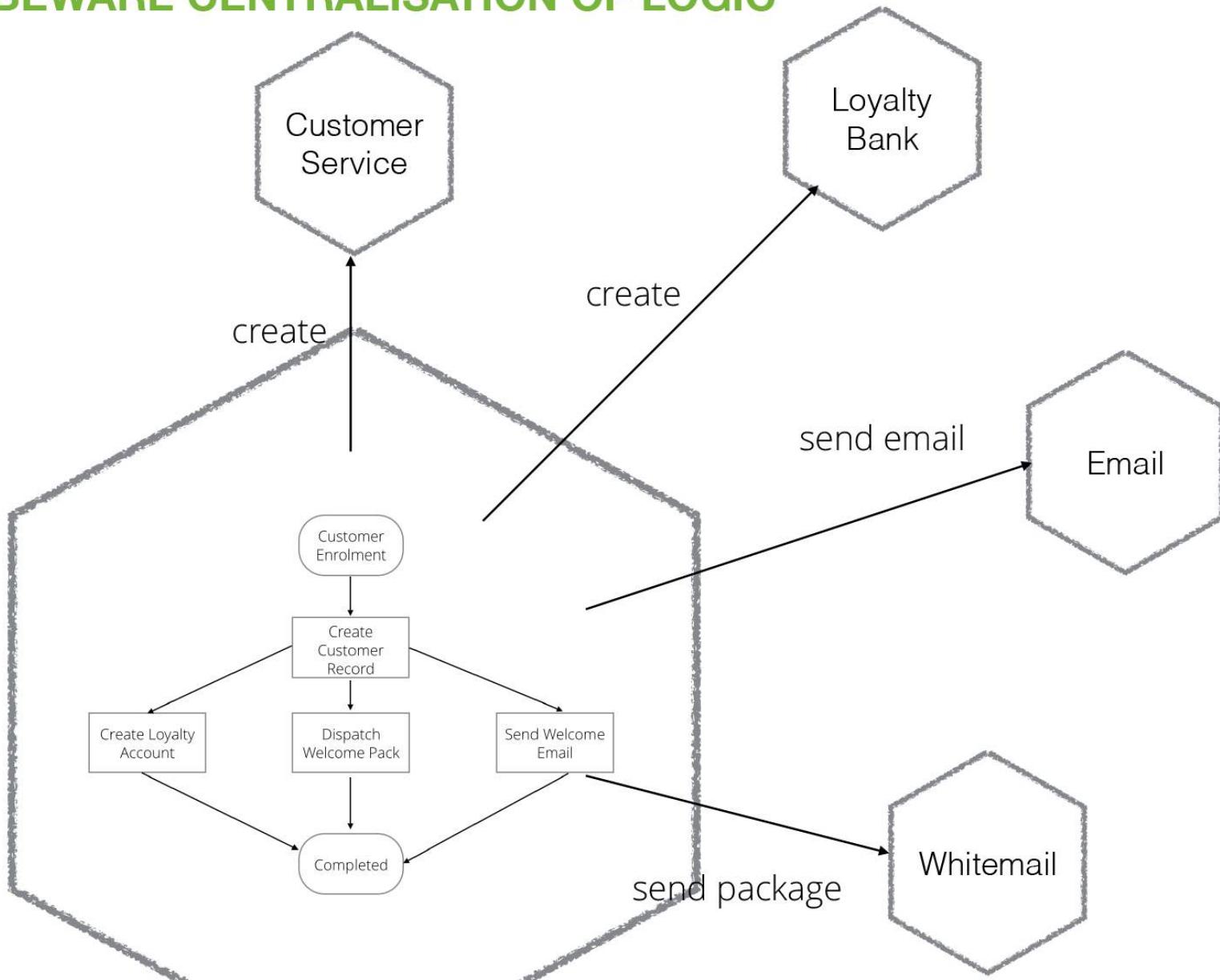
Can be fairly coupled

Can lead to overly smart (and dumb services)

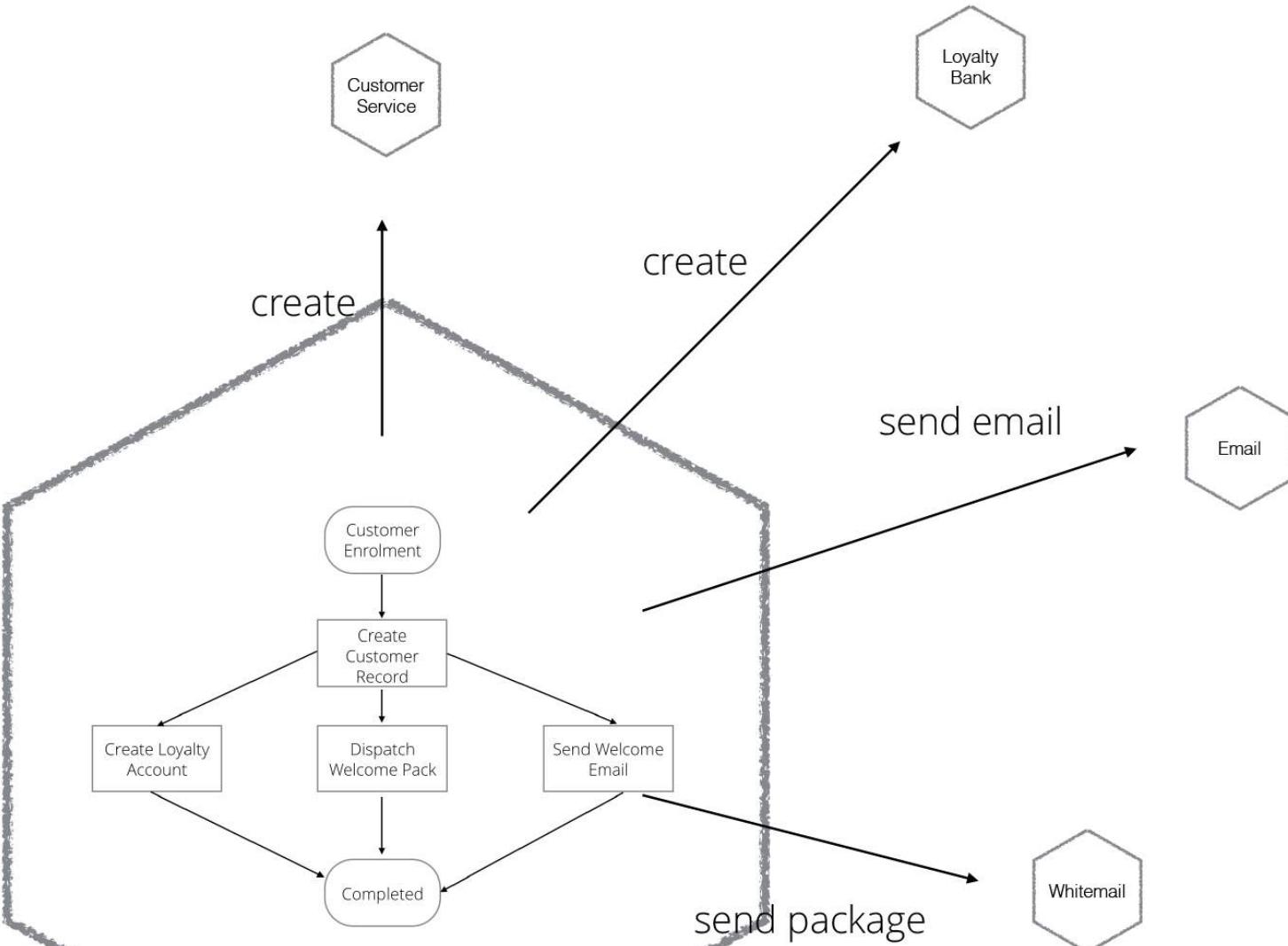
ORCHESTRATION - BEWARE CENTRALISATION OF LOGIC



ORCHESTRATION - BEWARE CENTRALISATION OF LOGIC

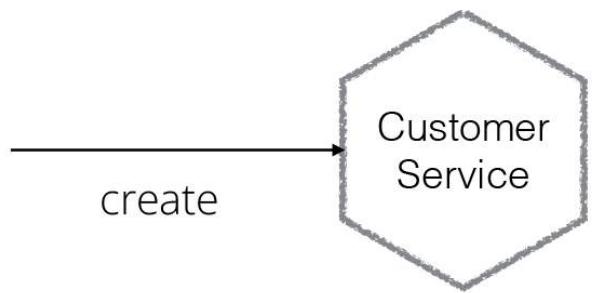


ORCHESTRATION - BEWARE CENTRALISATION OF LOGIC

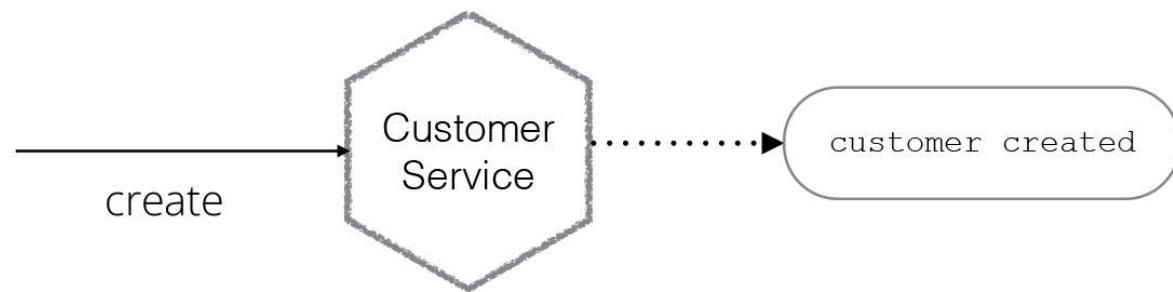


CHOREOGRAPHED

CHOREOGRAPHED



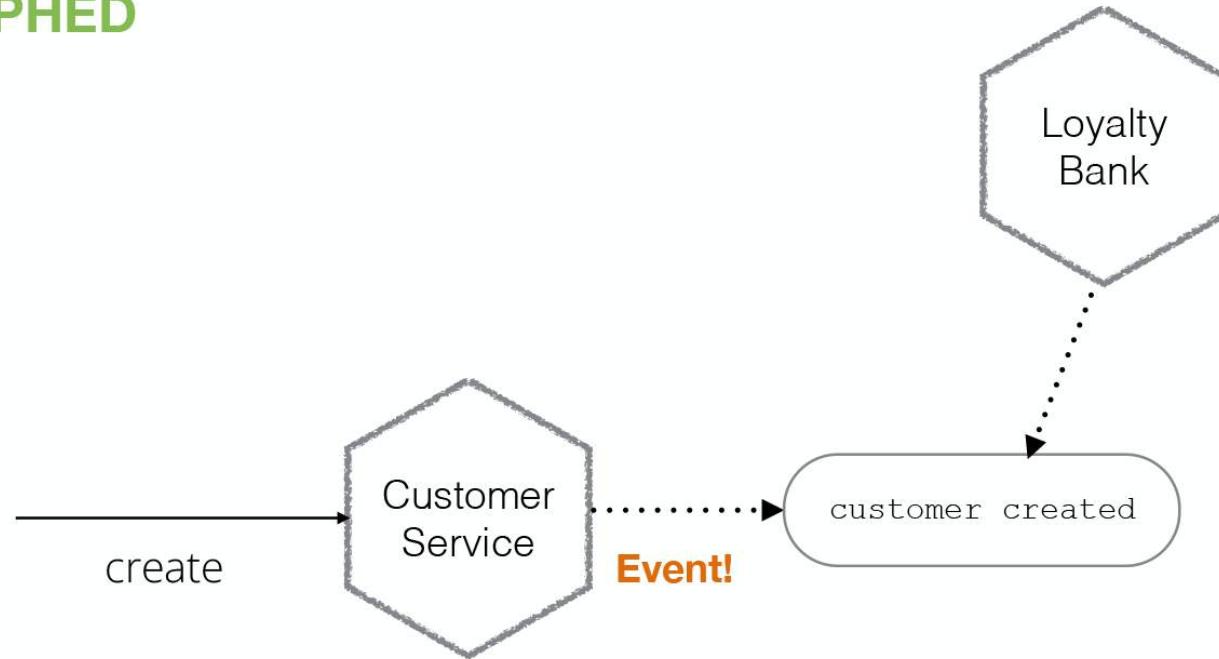
CHOREOGRAPHED



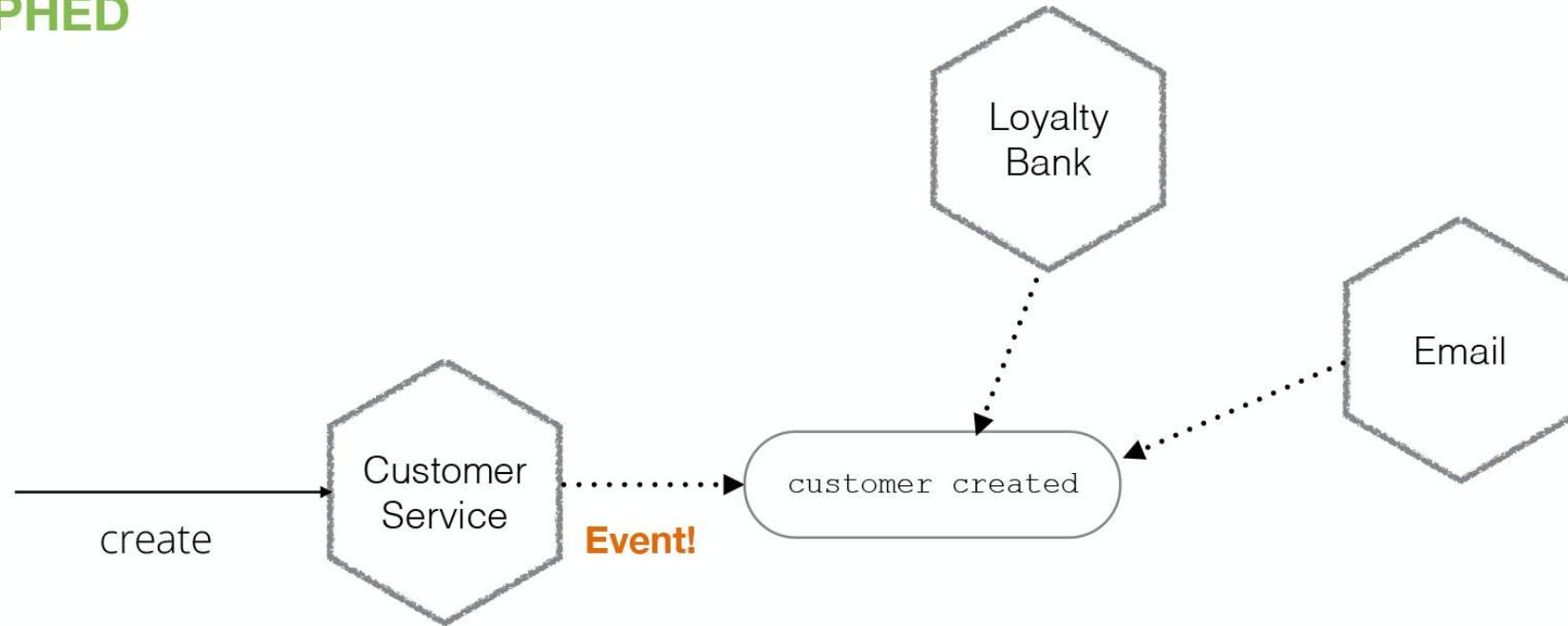
CHOREOGRAPHED



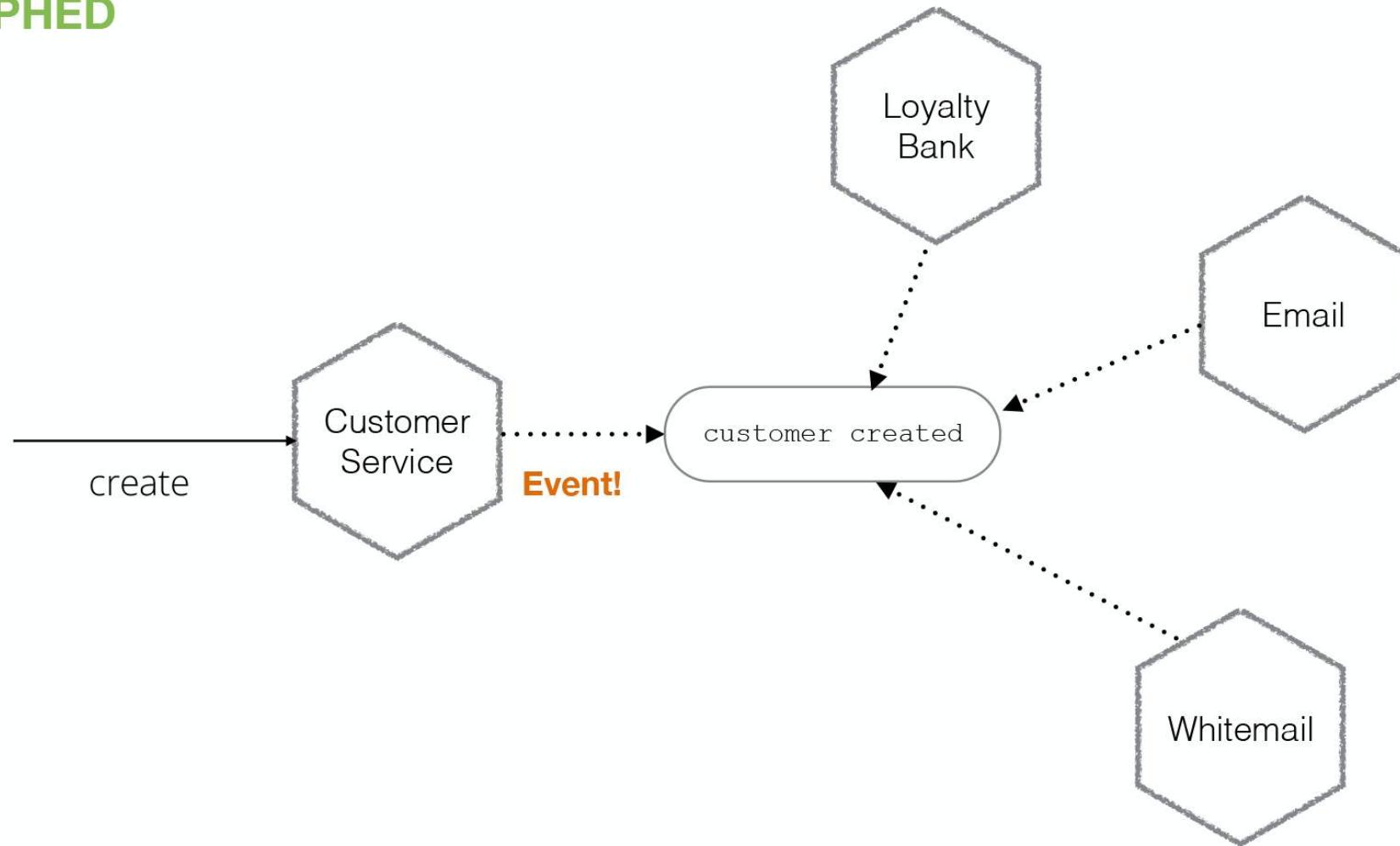
CHOREOGRAPHED



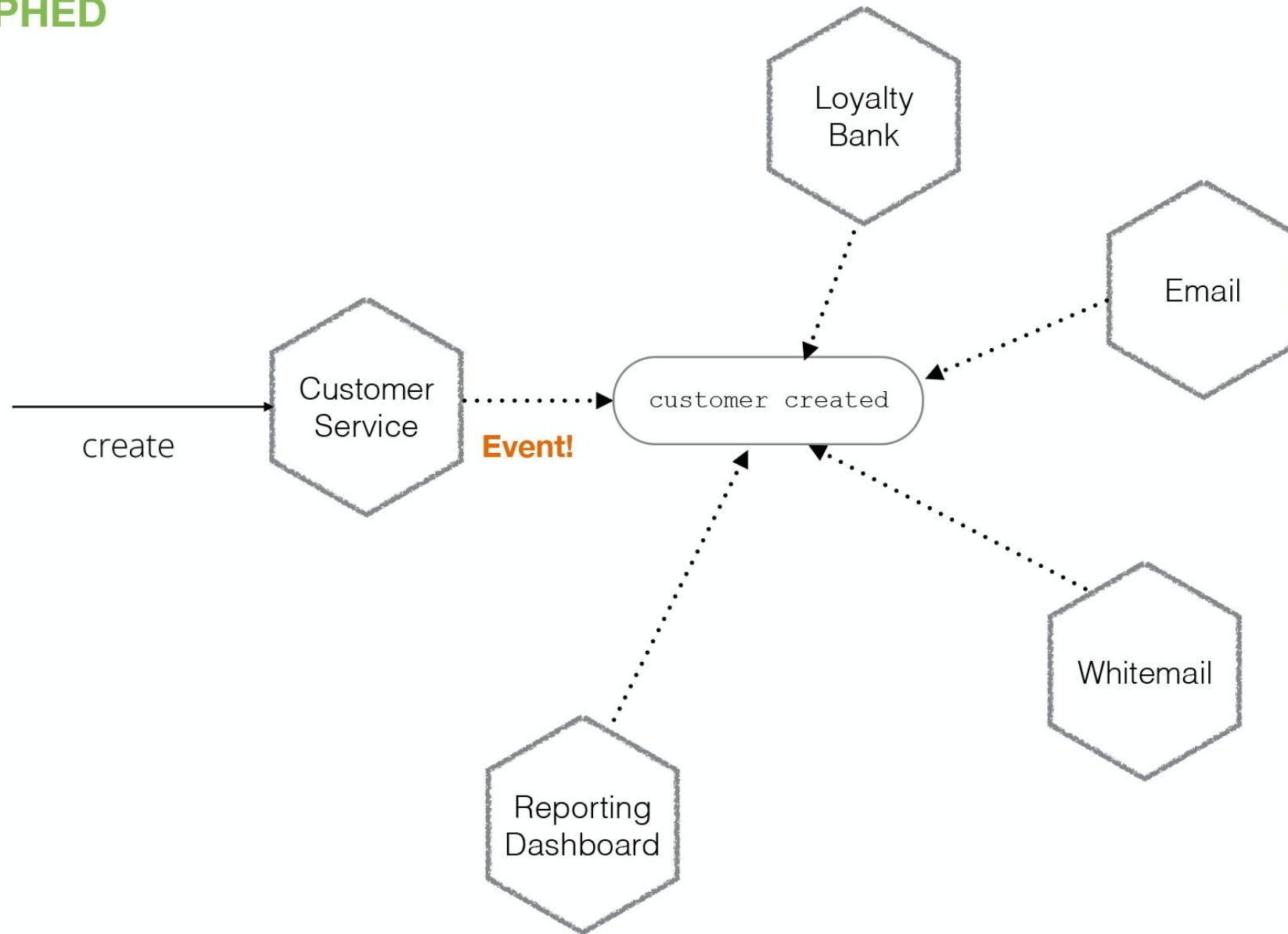
CHOREOGRAPHED



CHOREOGRAPHED



CHOREOGRAPHED



Pros

Pros

Highly decoupled

Pros

Highly decoupled

Evenly distributed smarts

Pros

Highly decoupled

Evenly distributed smarts

Cons

Pros

Highly decoupled

Evenly distributed smarts

Cons

Lost explicit business process mapping

Pros

Highly decoupled

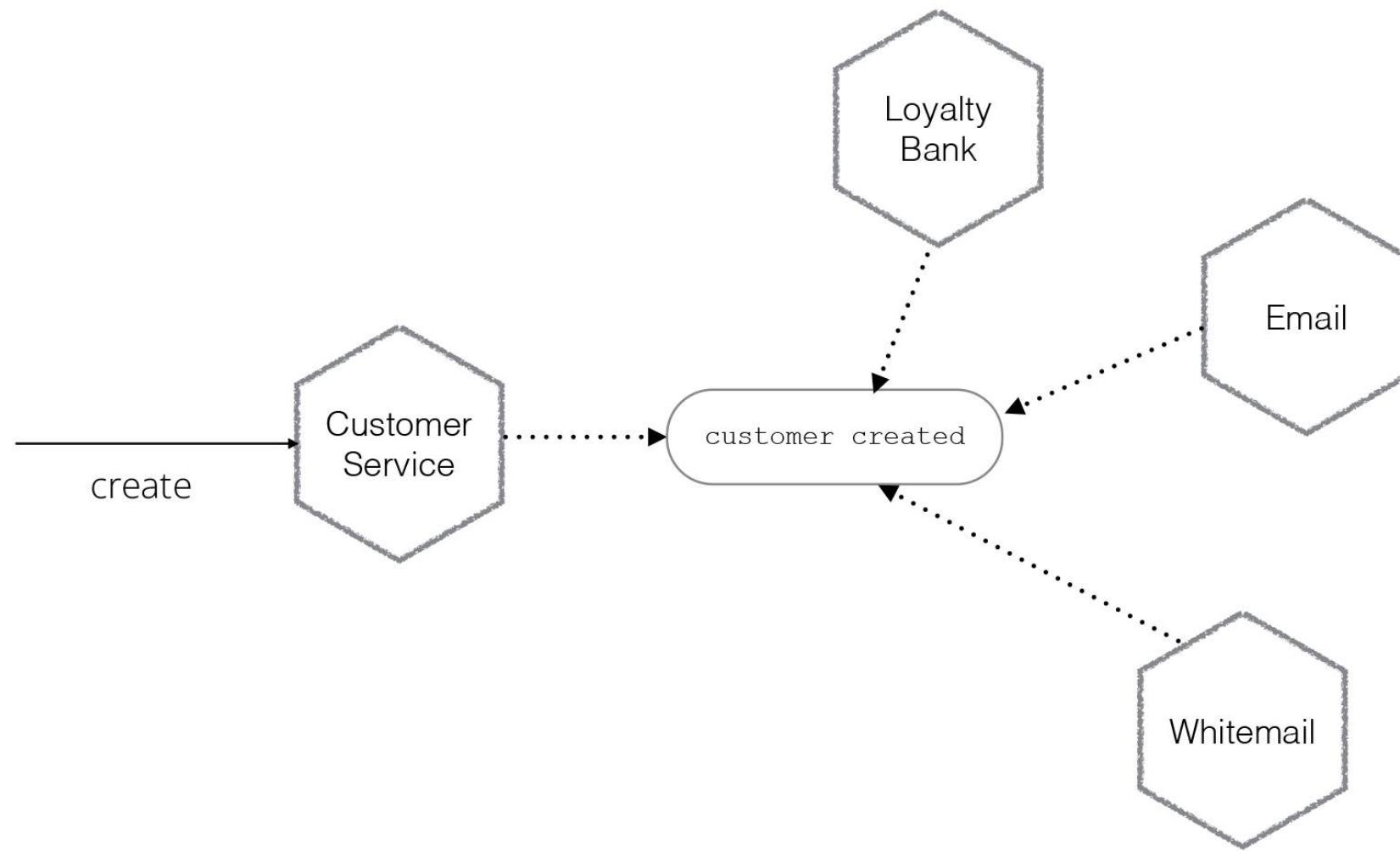
Evenly distributed smarts

Cons

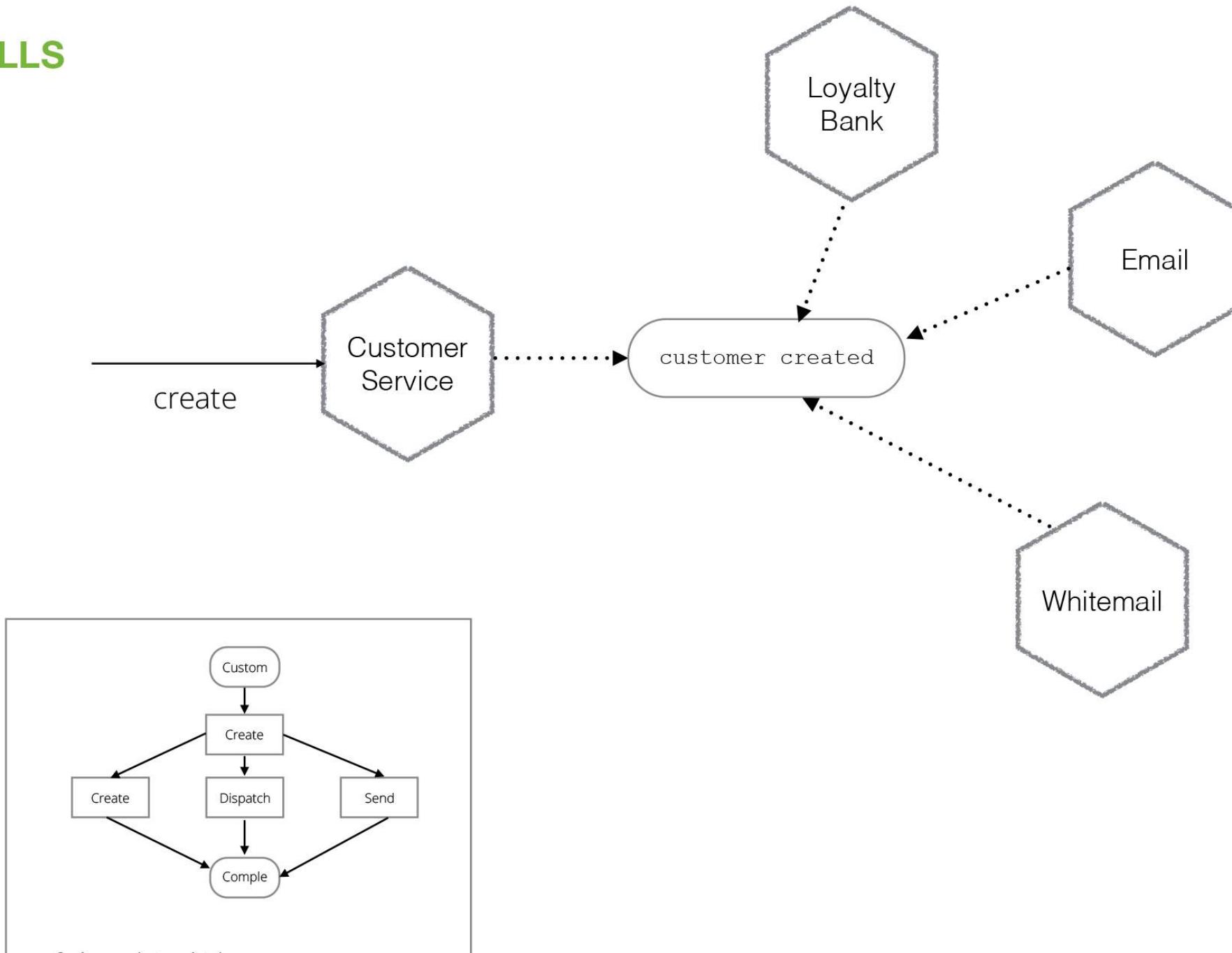
Lost explicit business process mapping

Understanding completion or error states is complex

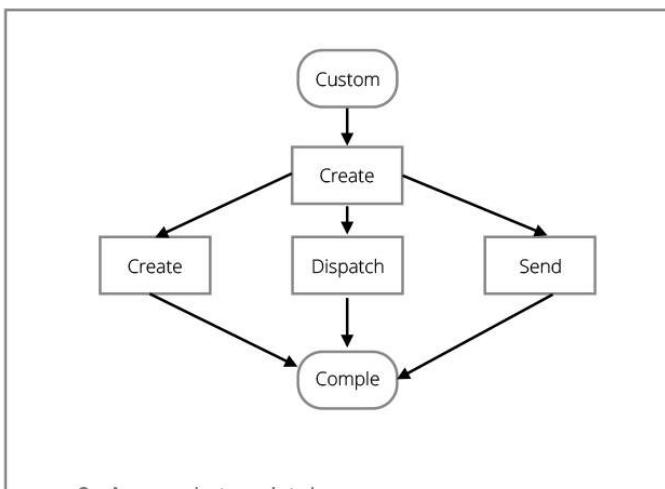
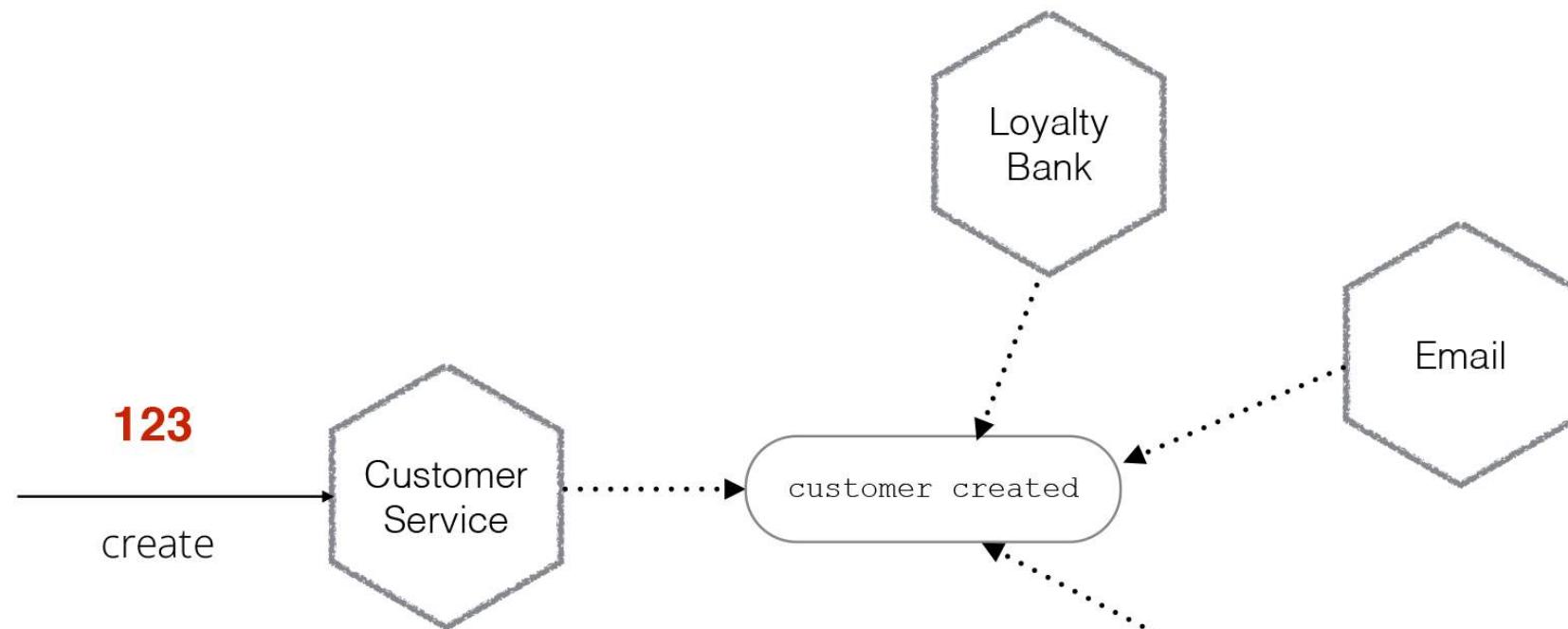
TRACING CALLS



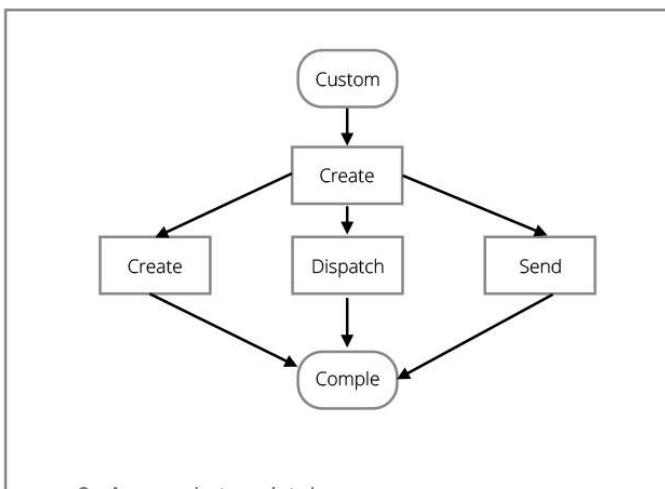
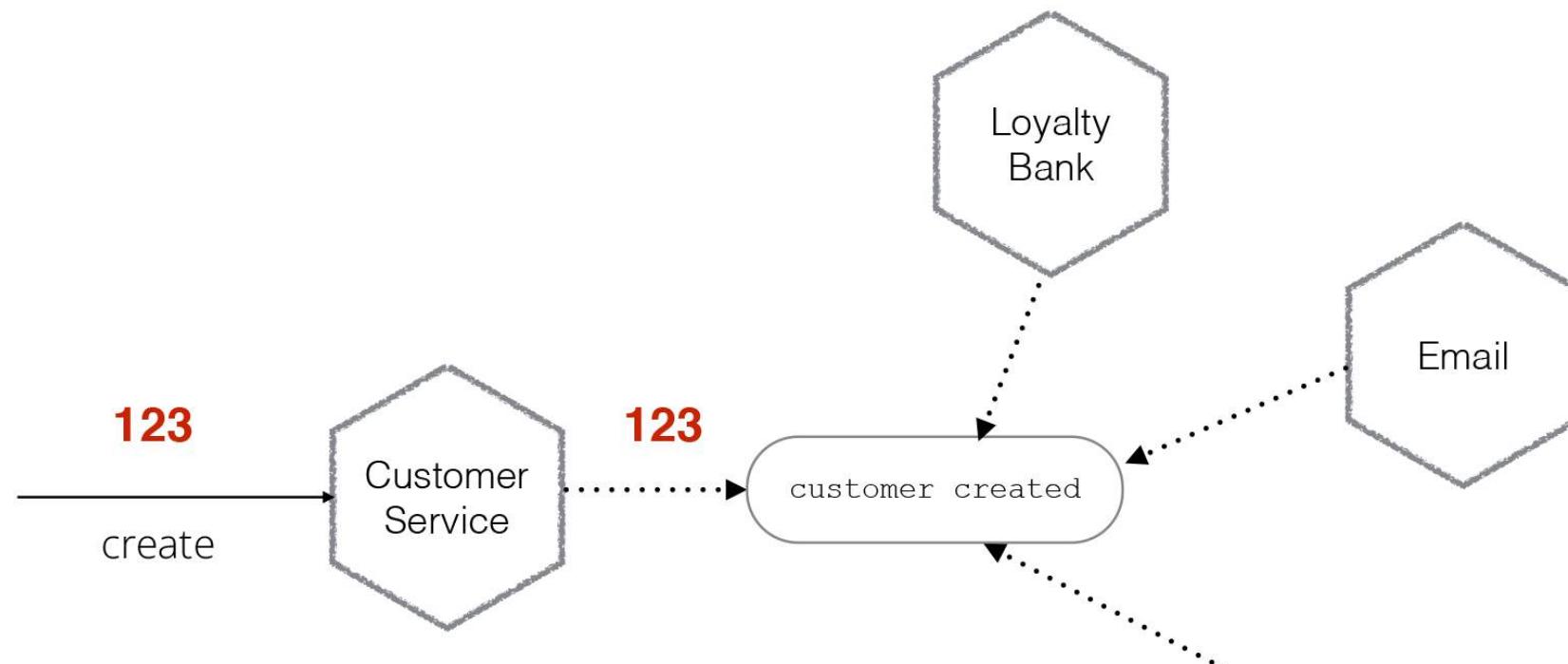
TRACING CALLS



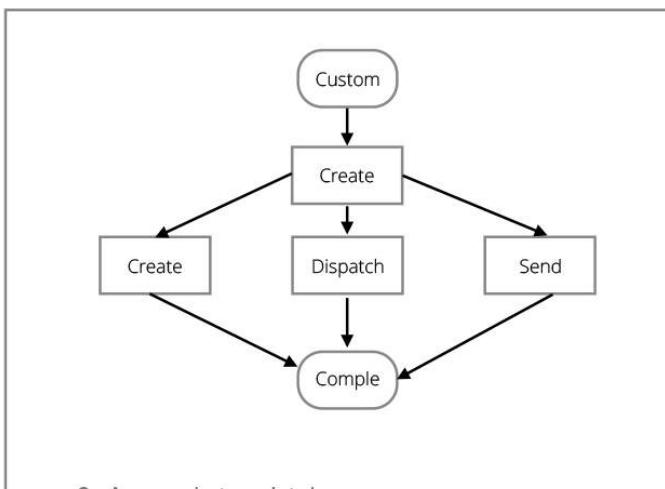
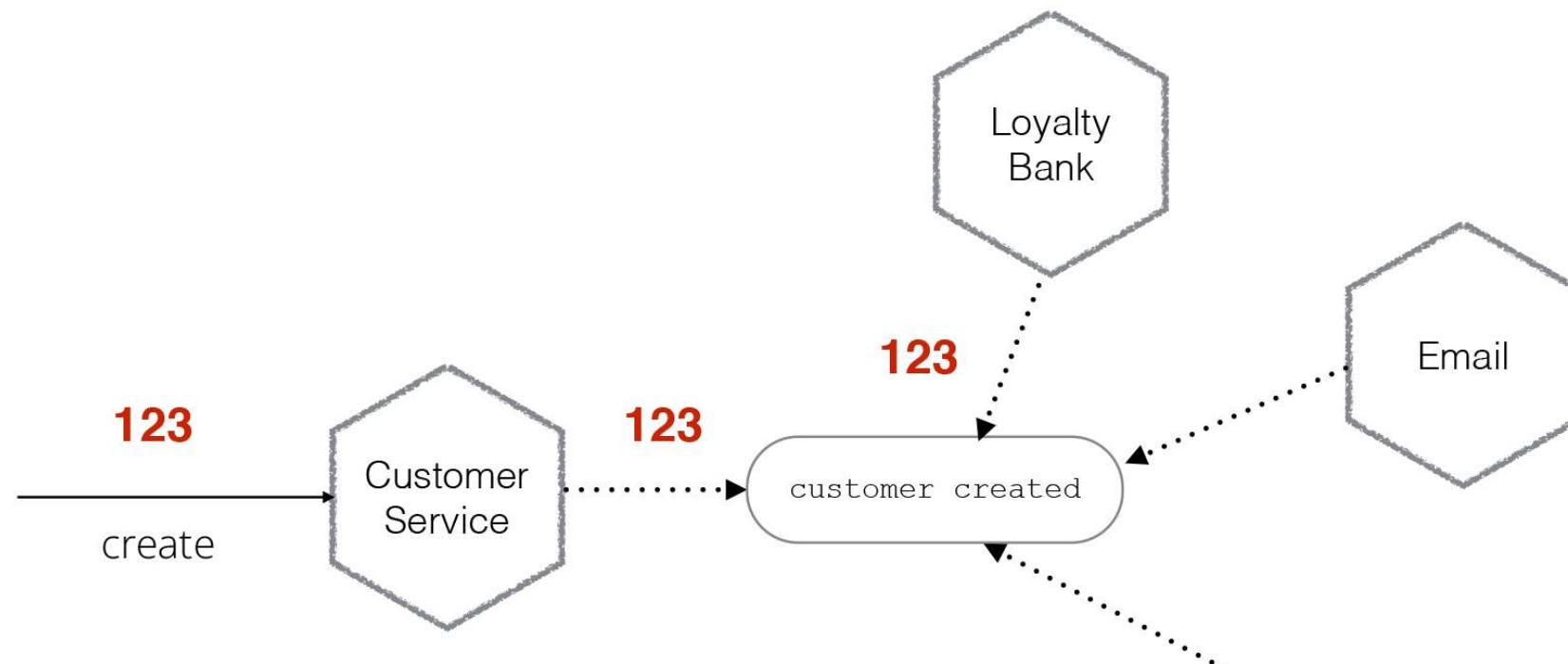
TRACING CALLS



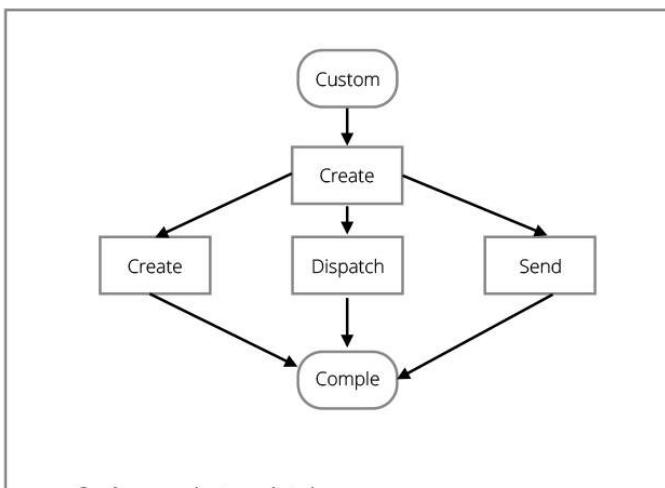
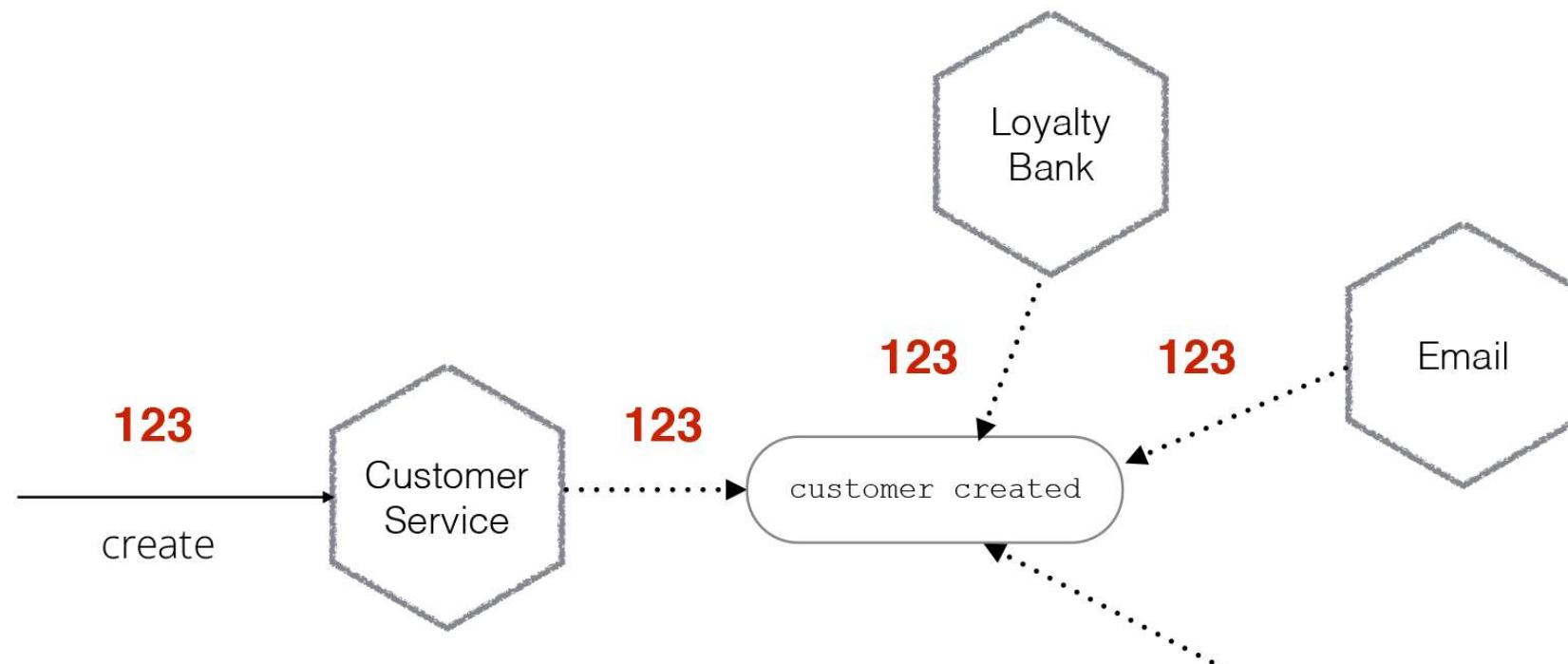
TRACING CALLS



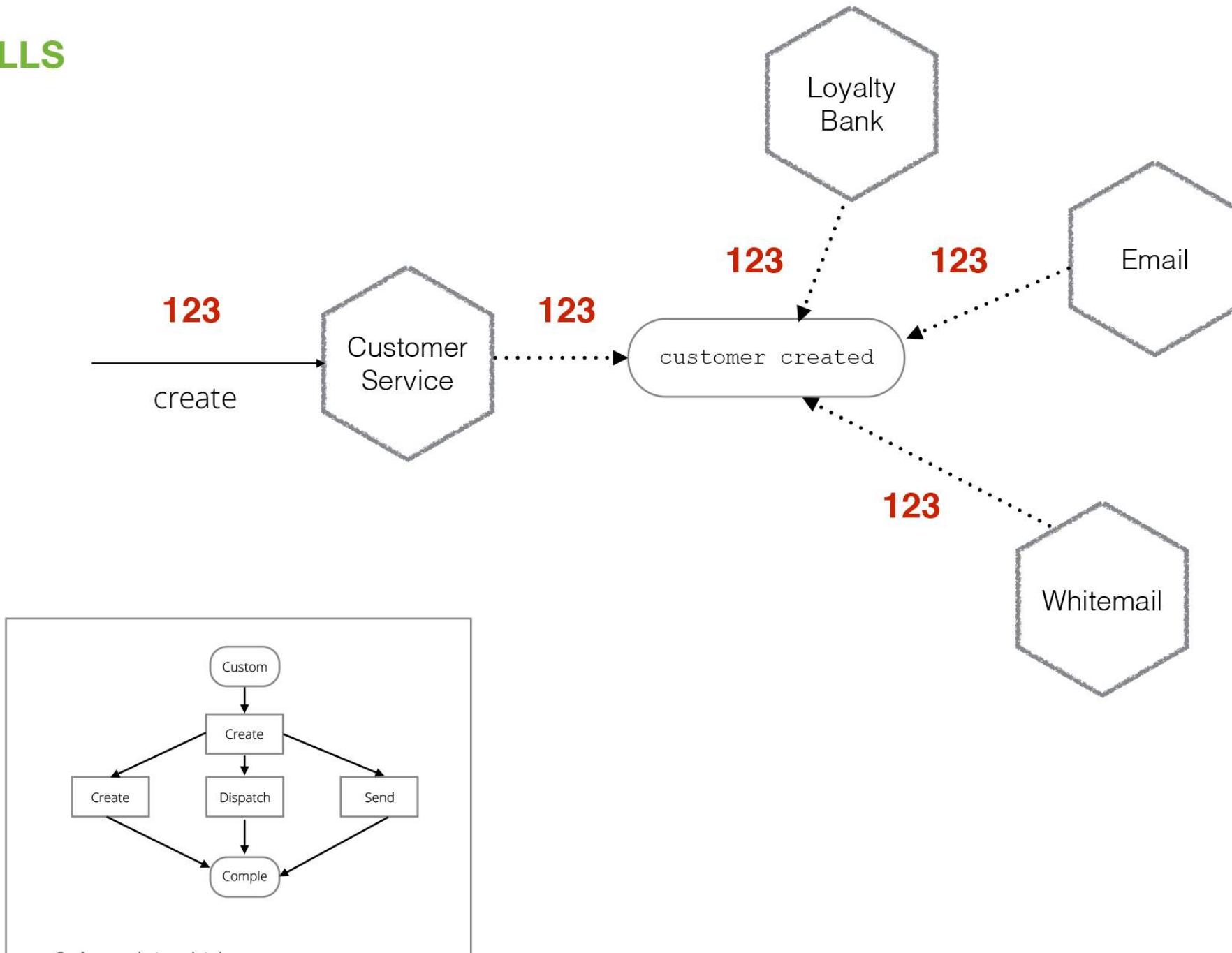
TRACING CALLS



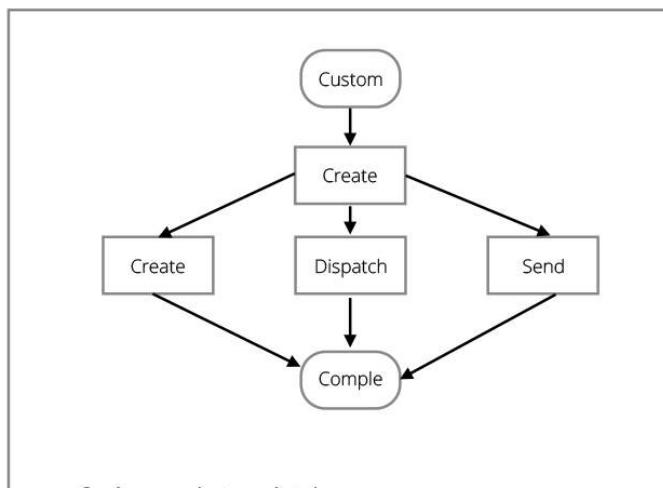
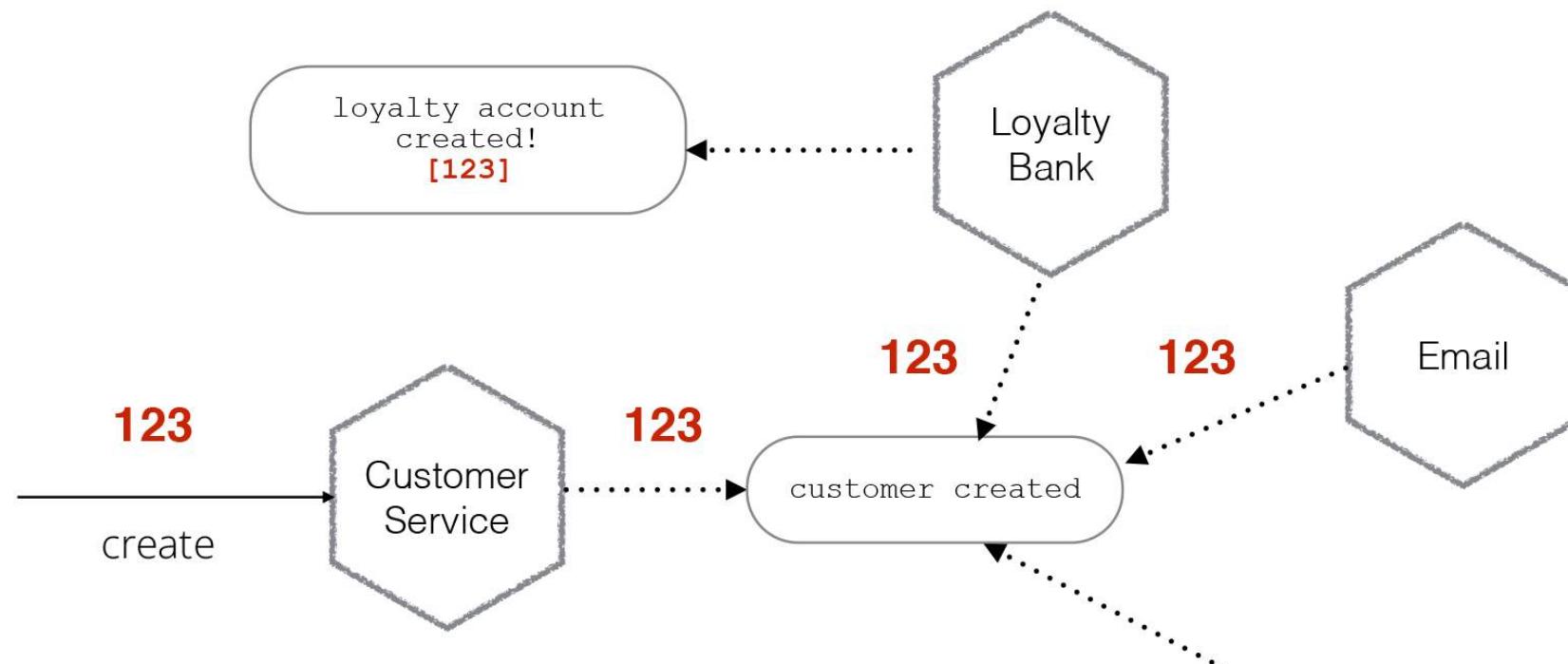
TRACING CALLS



TRACING CALLS

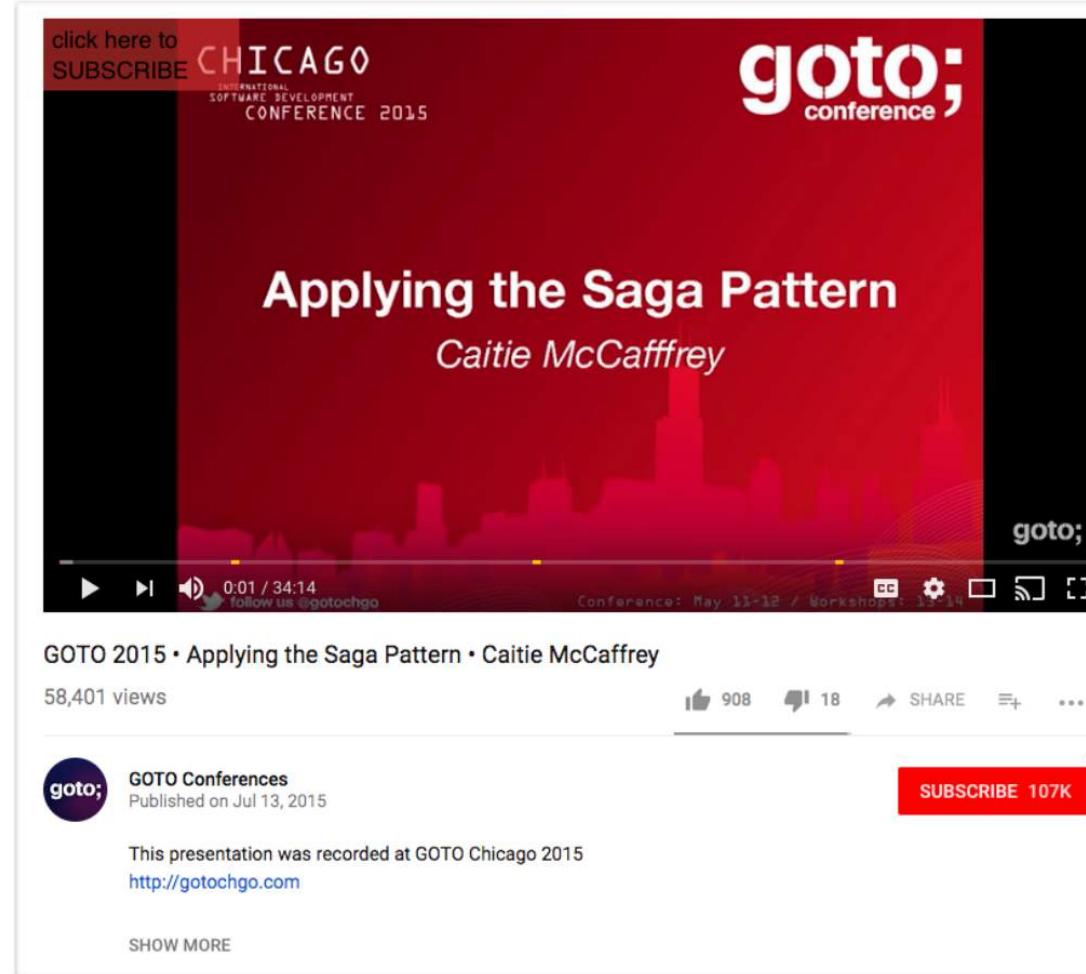


TRACING CALLS



You can of course mix styles!

MORE ON SAGAS...



<https://www.youtube.com/watch?v=xDuwrtwYHu8>

Pattern: Saga

Context

You have applied the [Database per Service](#) pattern. Each service has its own database. Some business transactions, however, span multiple services so you need a mechanism to ensure data consistency across services. For example, let's imagine that you are building an e-commerce store where customers have a credit limit. The application must ensure that a new order will not exceed the customer's credit limit. Since Orders and Customers are in different databases the application cannot simply use a local ACID transaction.

Problem

How to maintain data consistency across services?

Forces

- 2PC is not an option

Solution

Implement each business transaction that spans multiple services as a **saga**. A saga is a sequence of local transactions. Each local transaction updates the database and publishes a message or event to trigger the next local transaction in the saga. If a local transaction fails because it violates a business rule then the saga executes a series of compensating transactions that undo the changes that were made by the preceding local transactions.

<http://microservices.io/patterns/data/saga.html>

Copyrighted Material

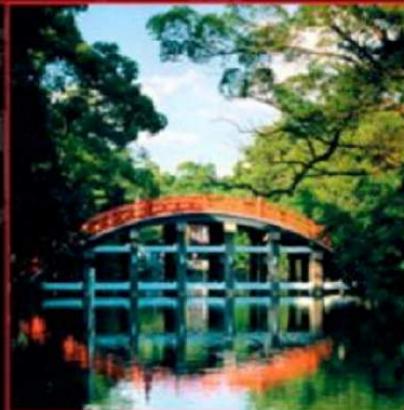
The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

DESIGNING, BUILDING, AND
DEPLOYING MESSAGING SOLUTIONS

GREGOR HOHPE
BOBBY WOOLF

WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON



Forewords by John Crupi and Martin Fowler

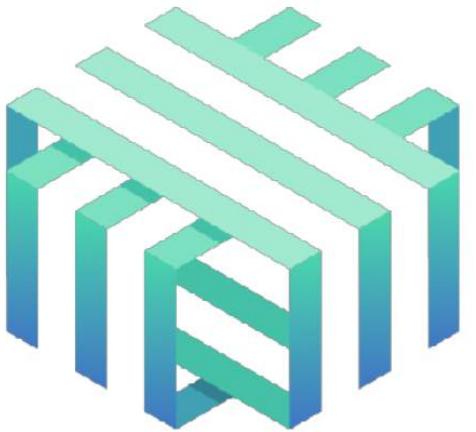
Copyrighted Material



A MARTIN FOWLER
SIGNATURE
BOOK

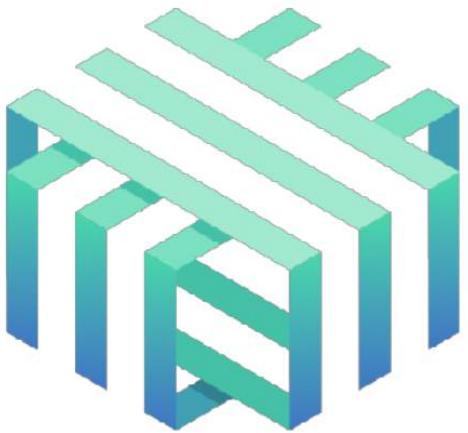


SERVICE MESSES

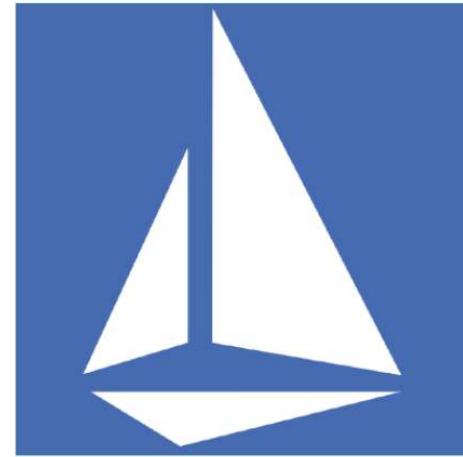


Linkerd

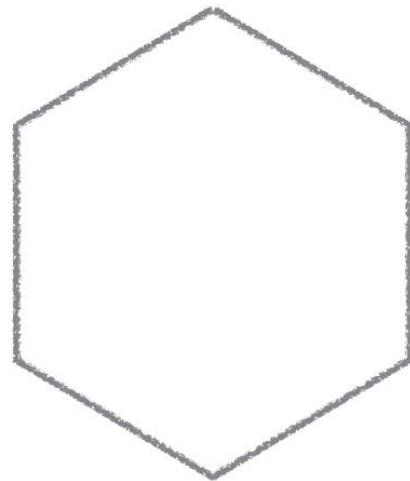
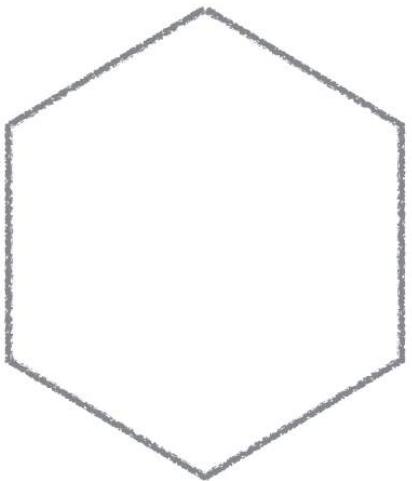
SERVICE MESHES

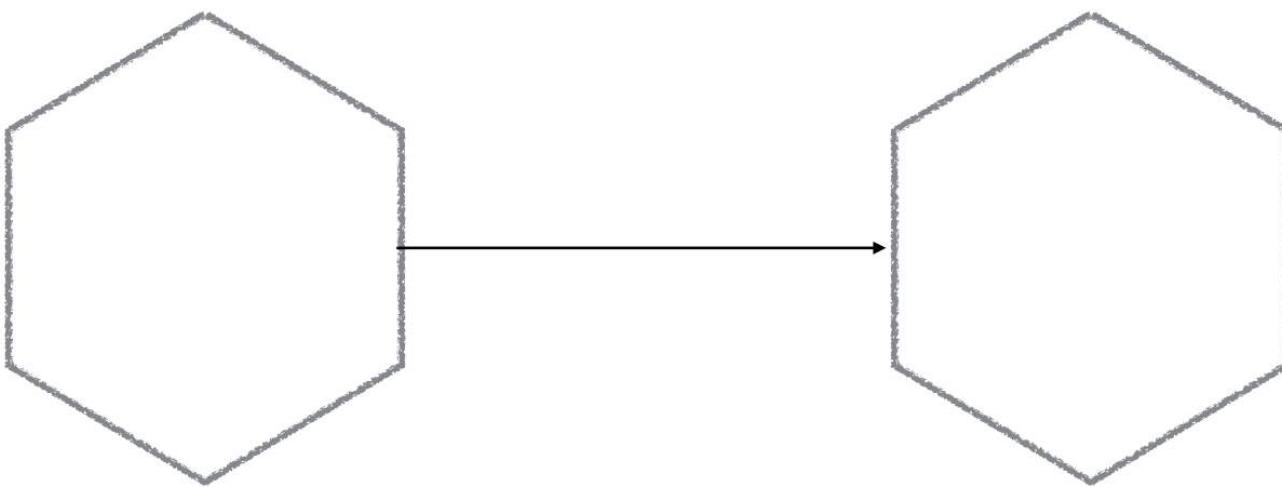


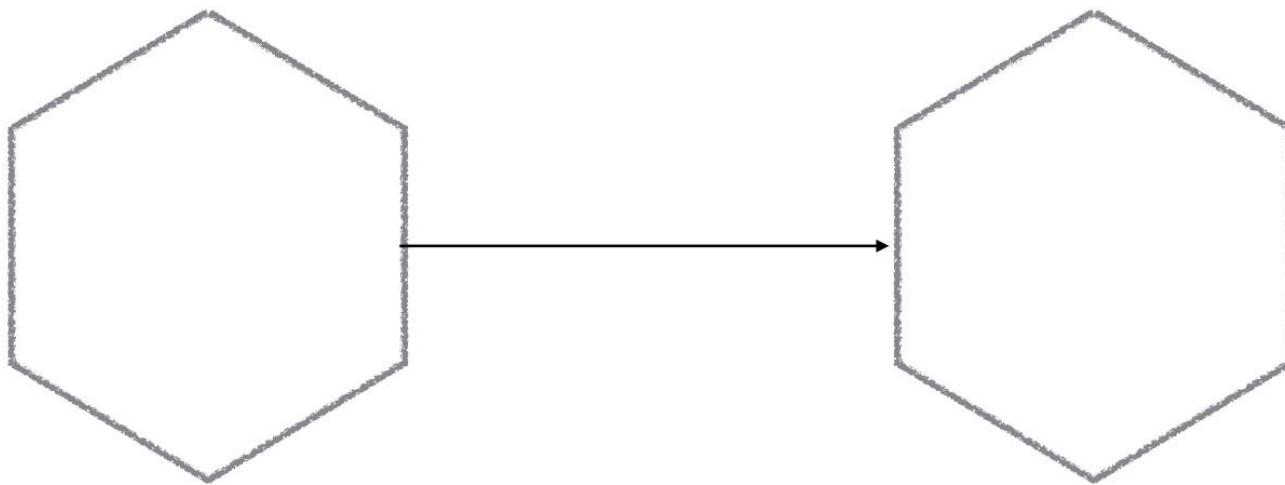
Linkerd



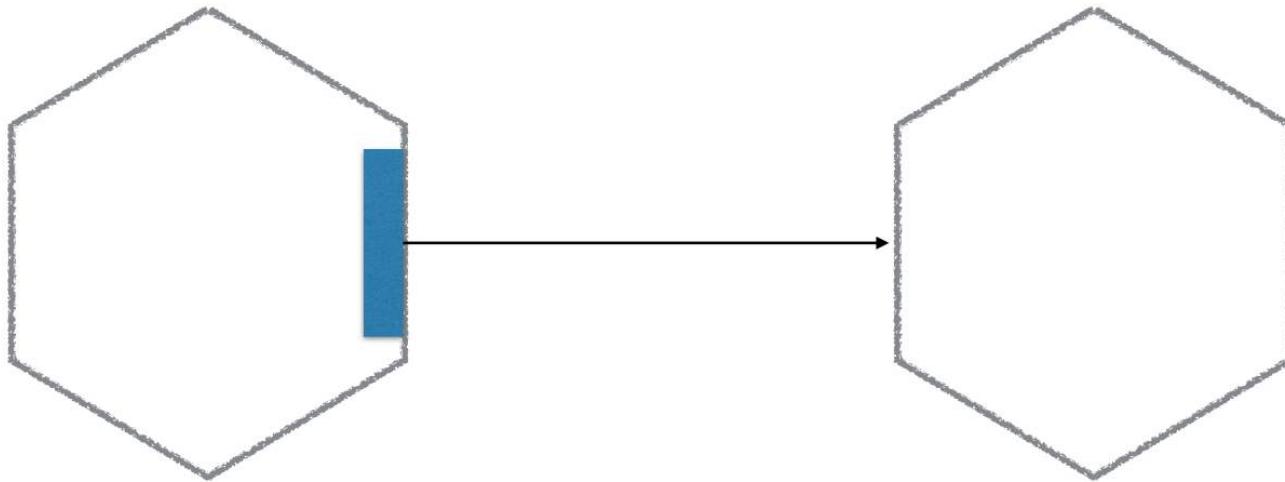
Istio



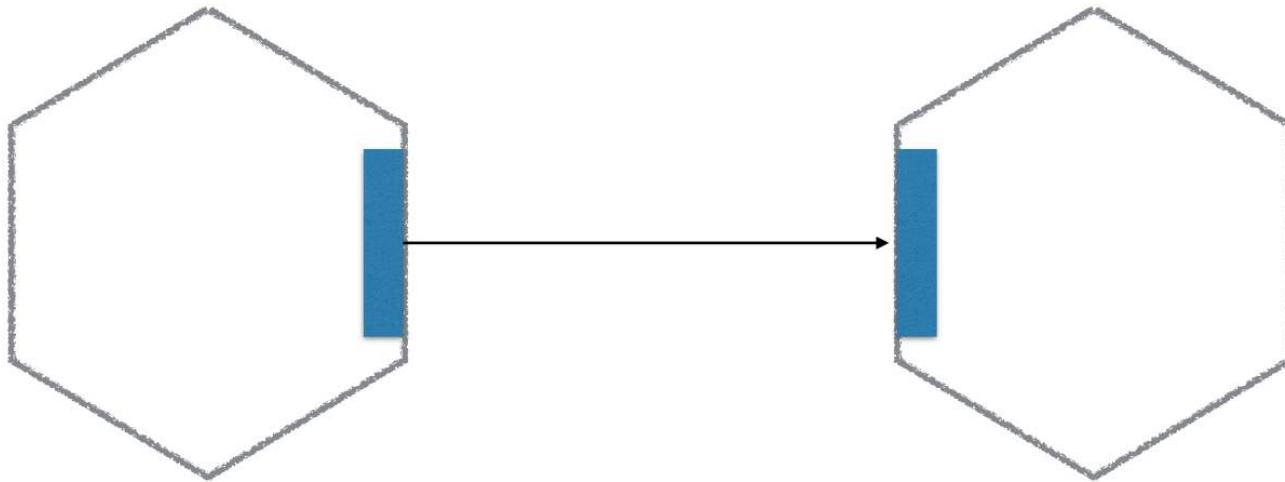




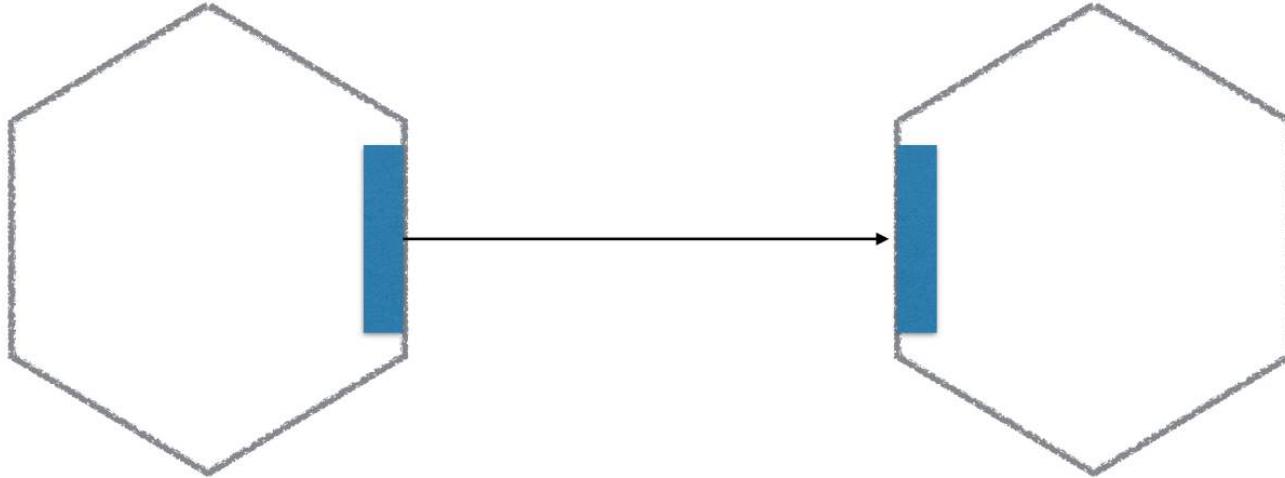
Tracing



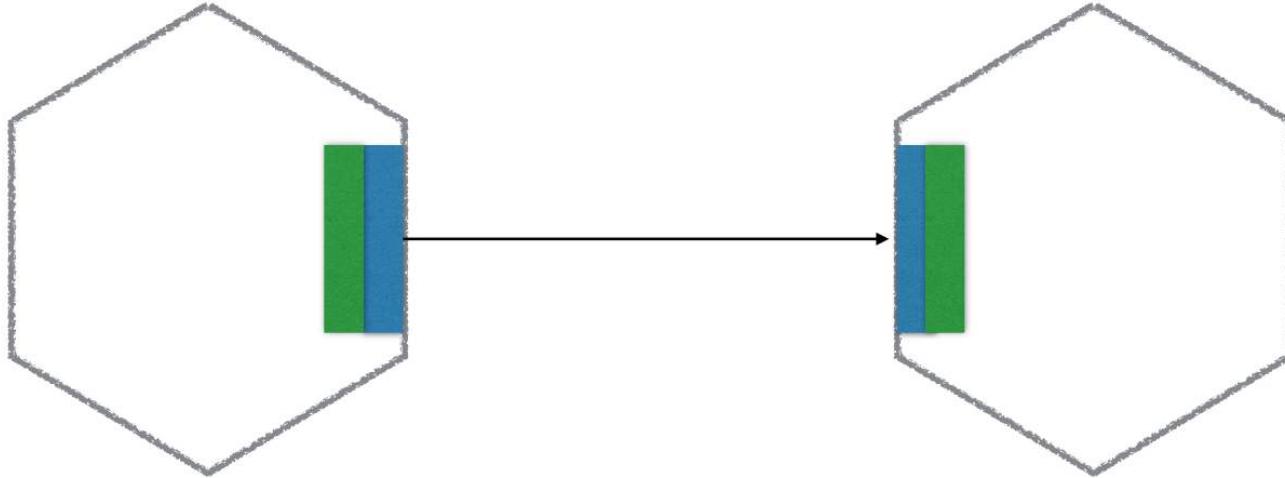
Tracing



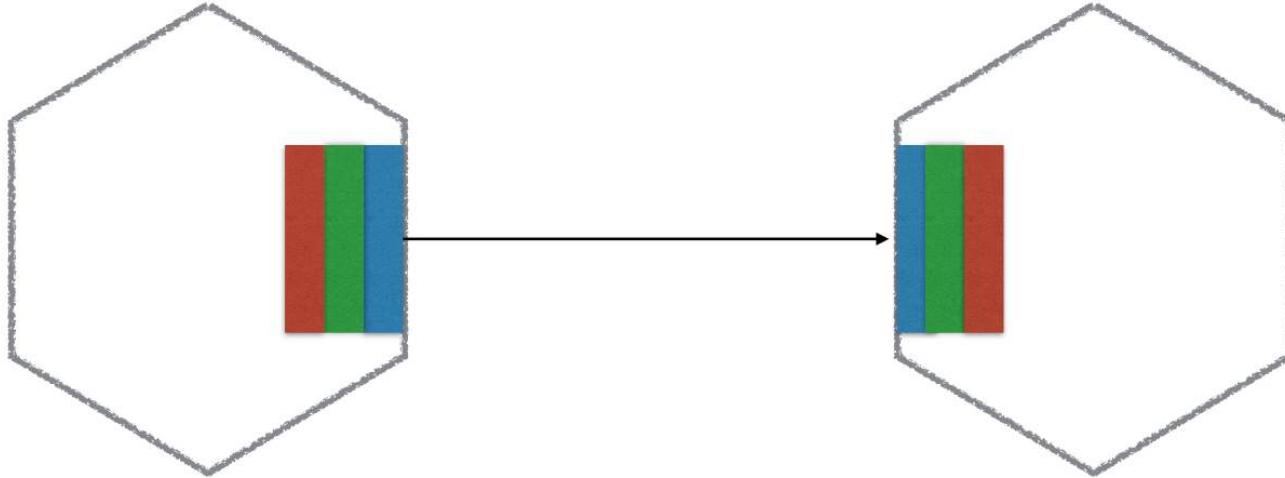
Tracing



Tracing Load Balancing & Service Discovery

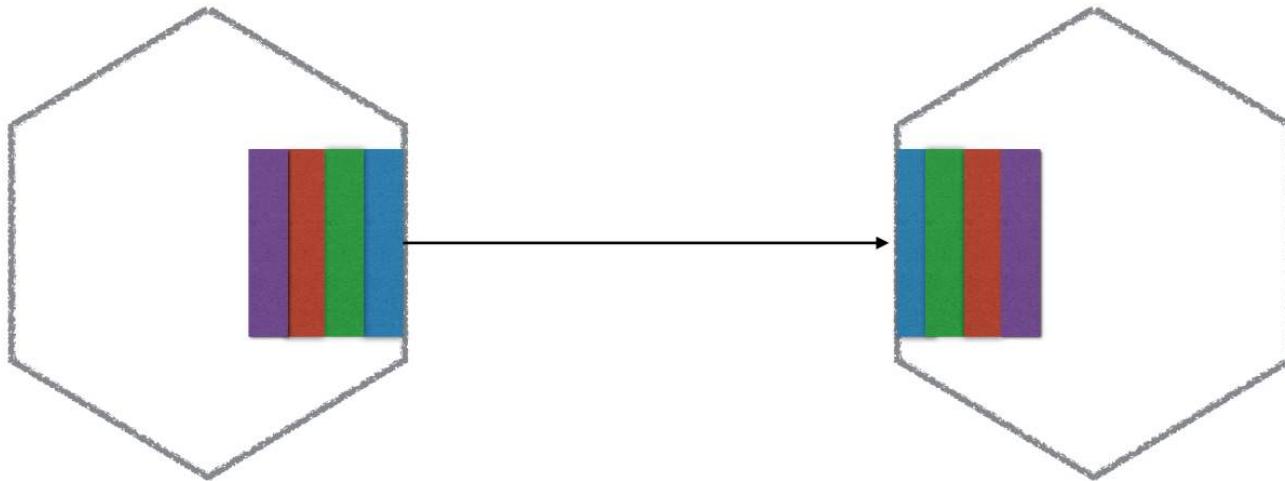


Tracing Load Balancing & Service Discovery



Tracing Load Balancing & Service Discovery

Auth

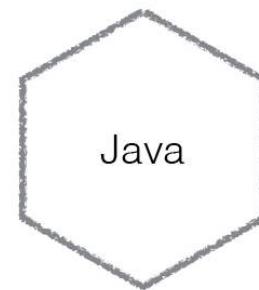
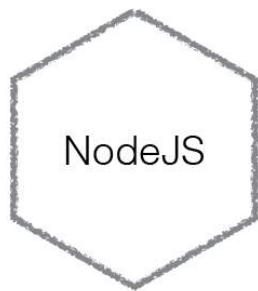
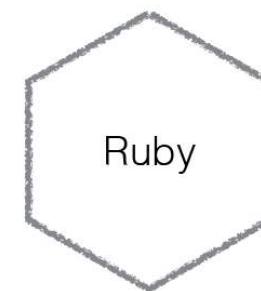
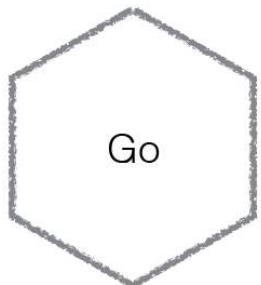


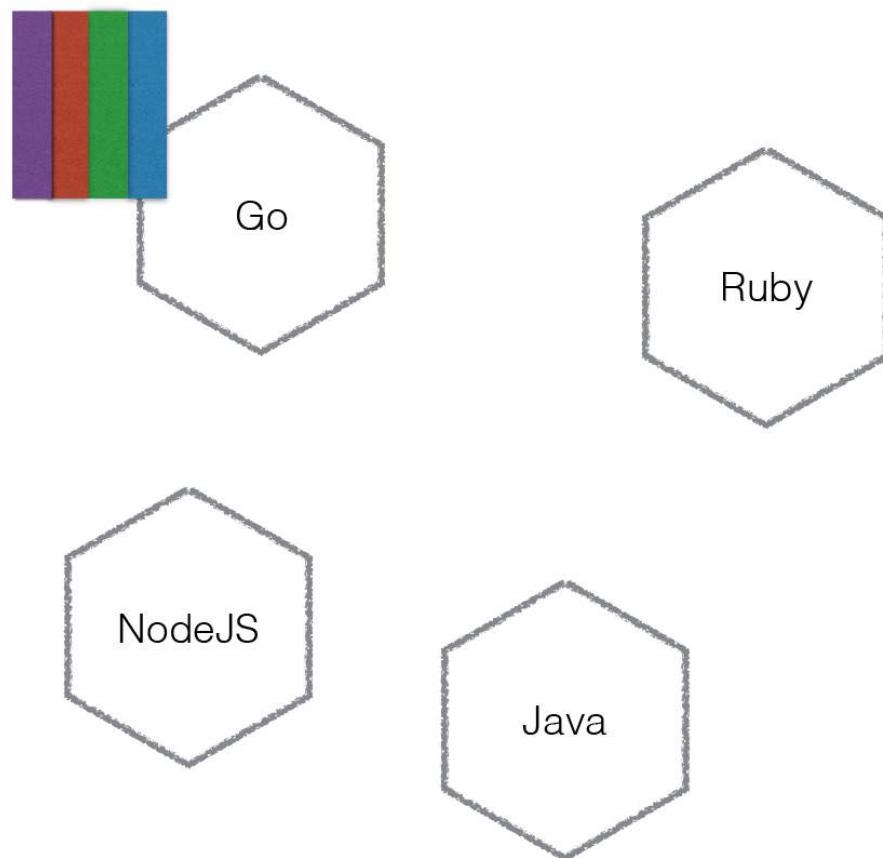
Tracing

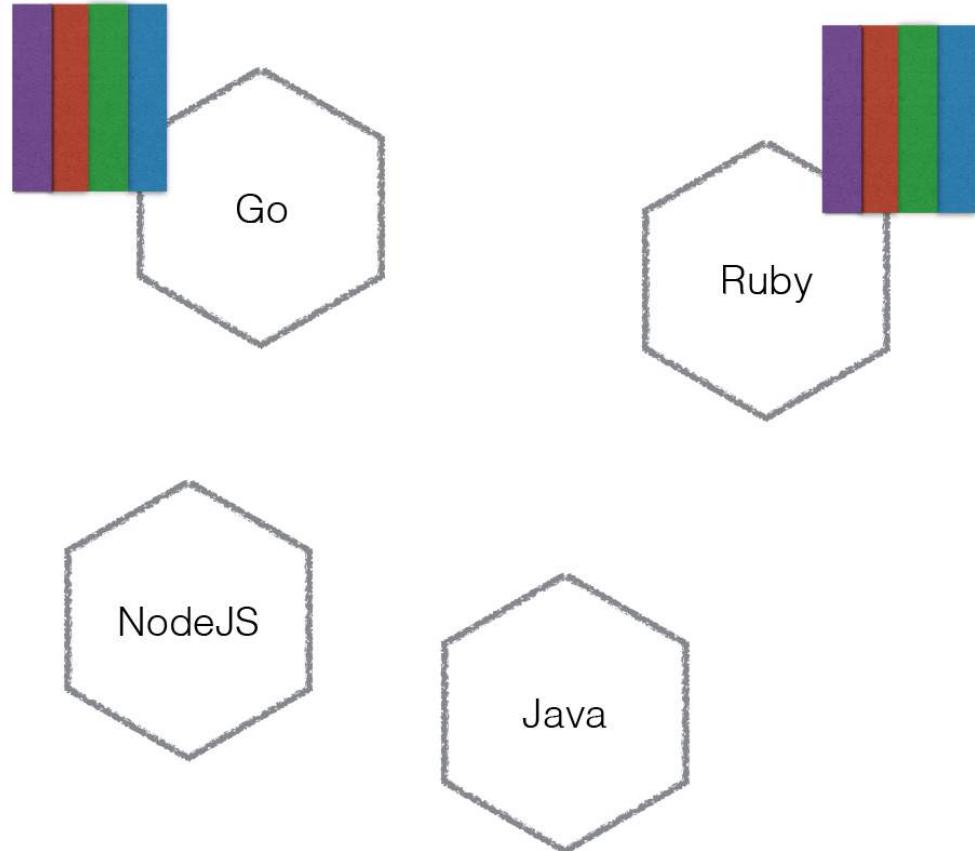
Load Balancing & Service Discovery

Auth

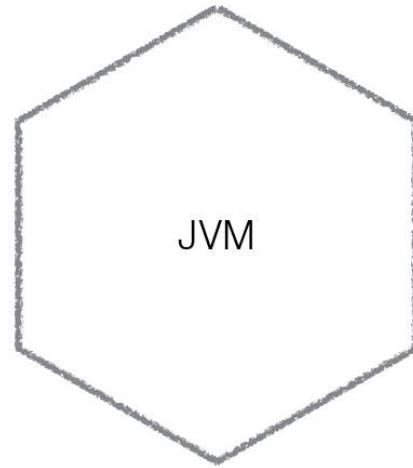
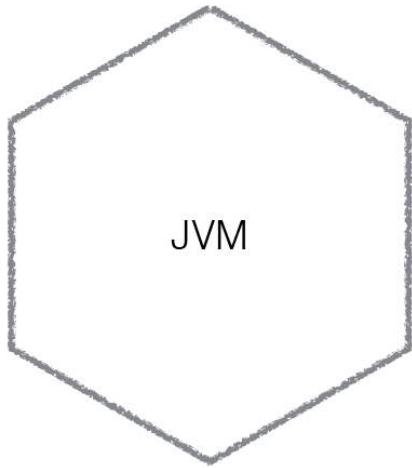
Connection Resilience & Retry



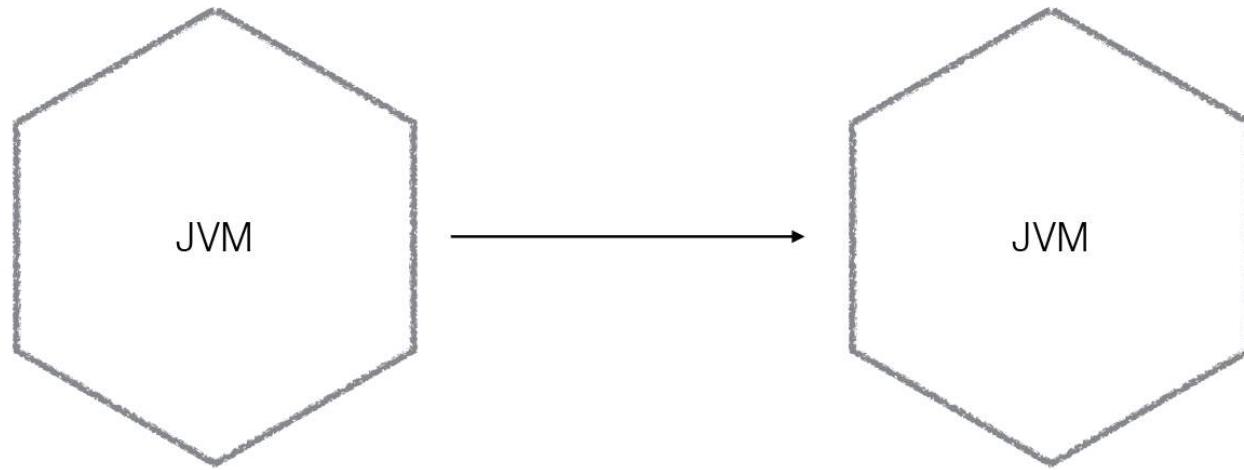




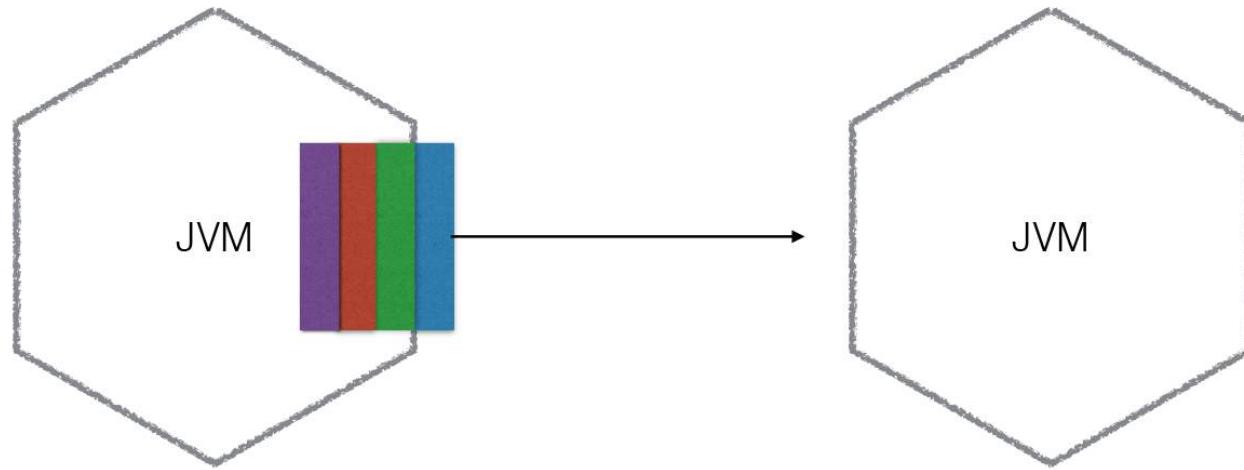
SIDECAR



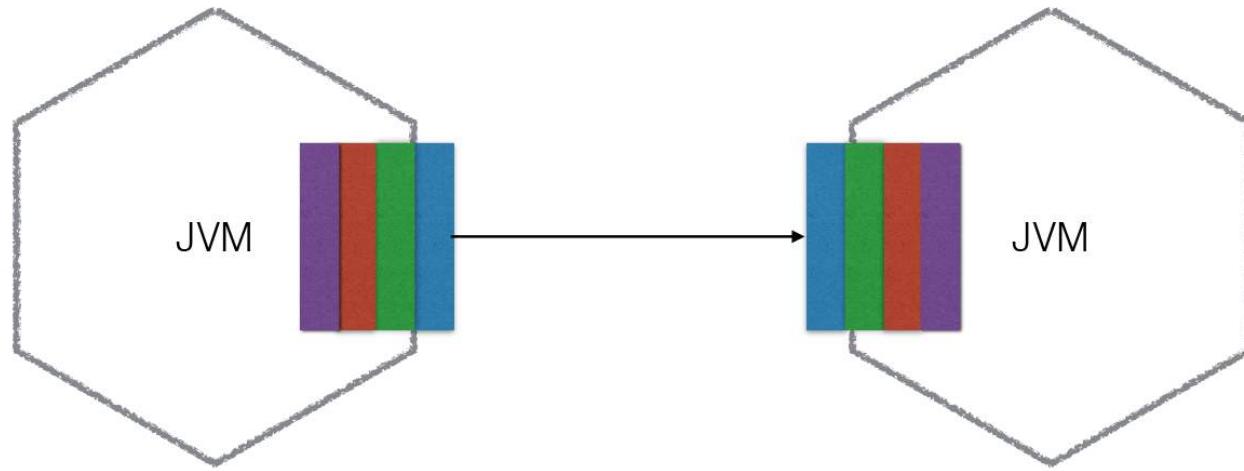
SIDECAR



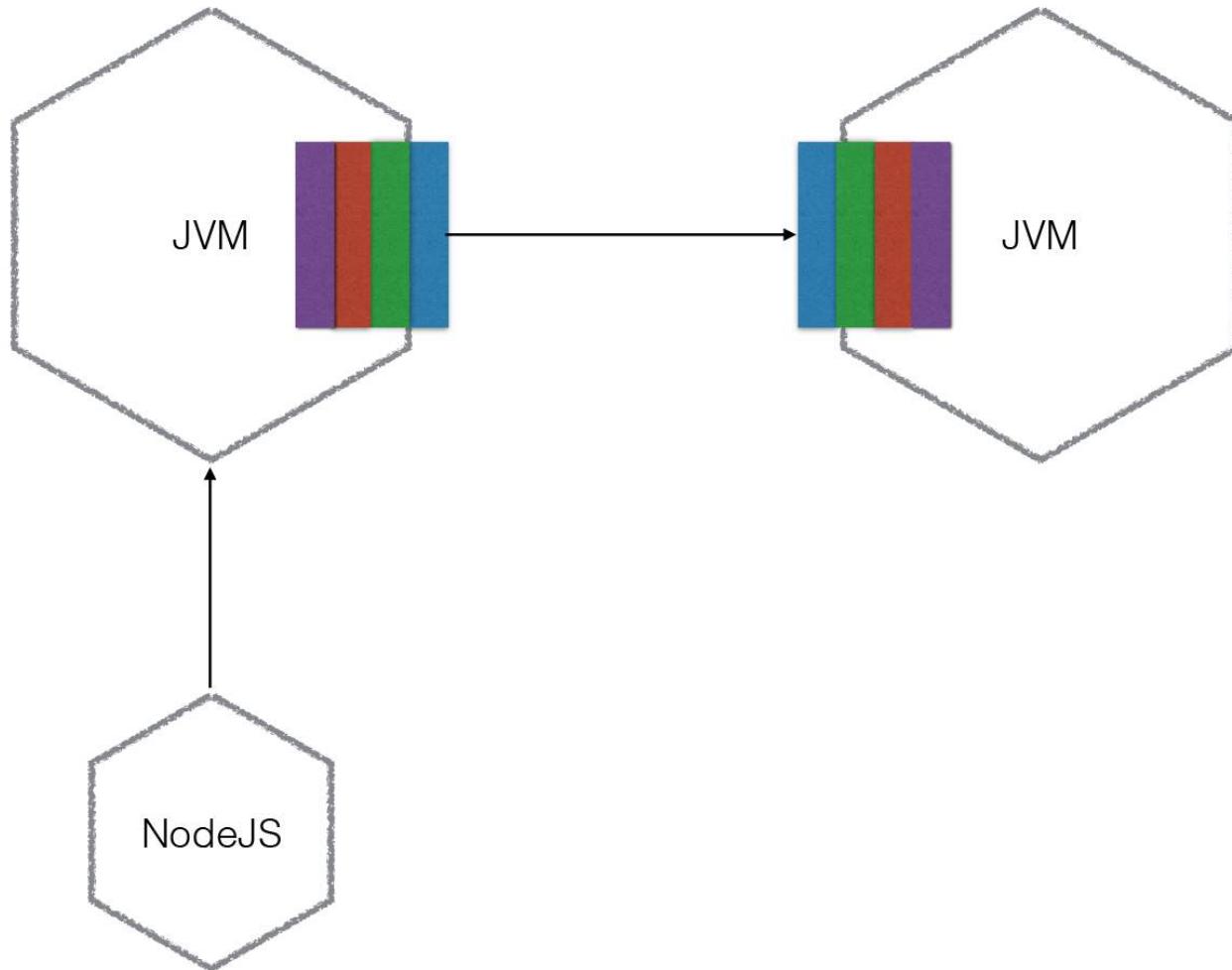
SIDECAR



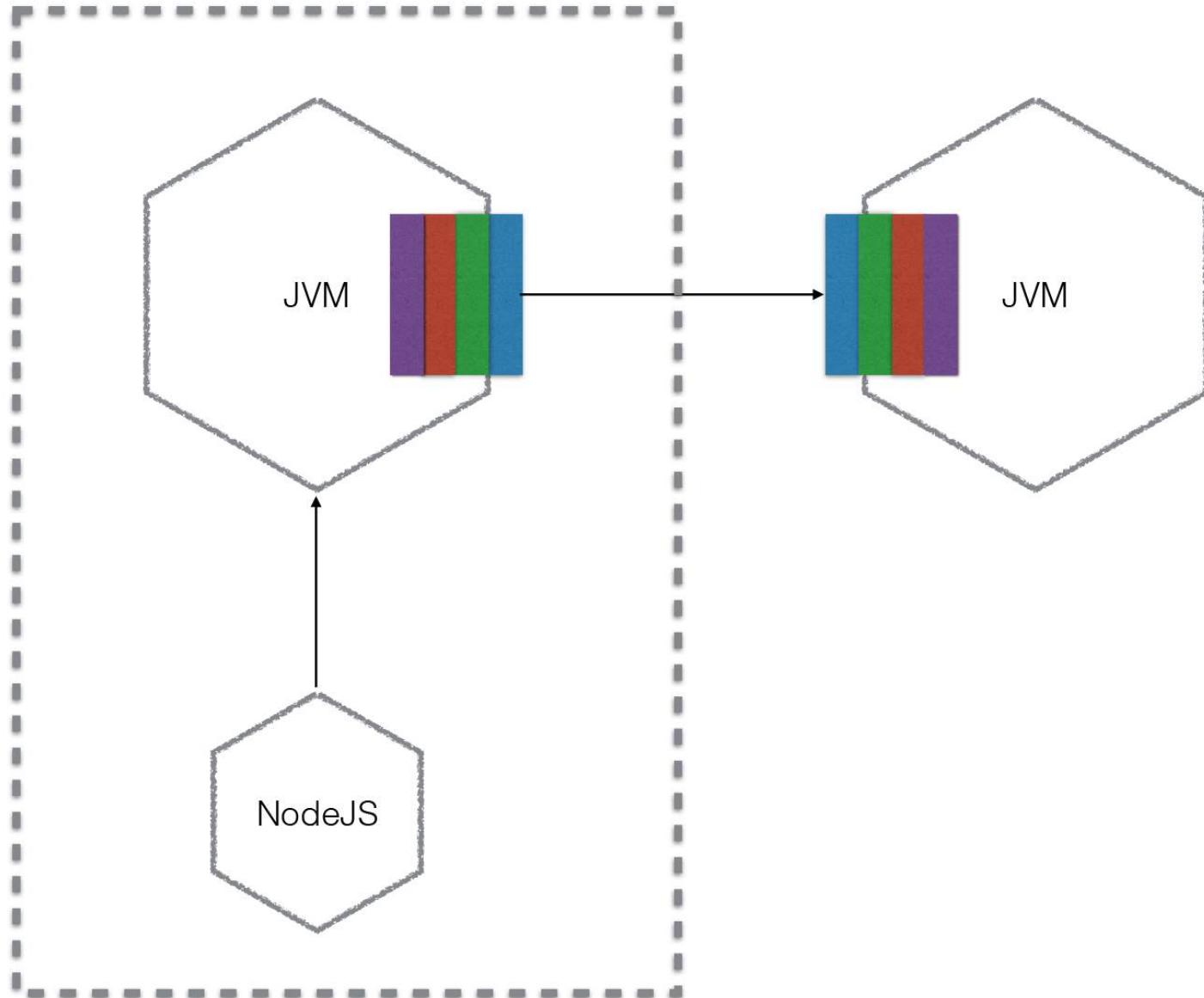
SIDECAR

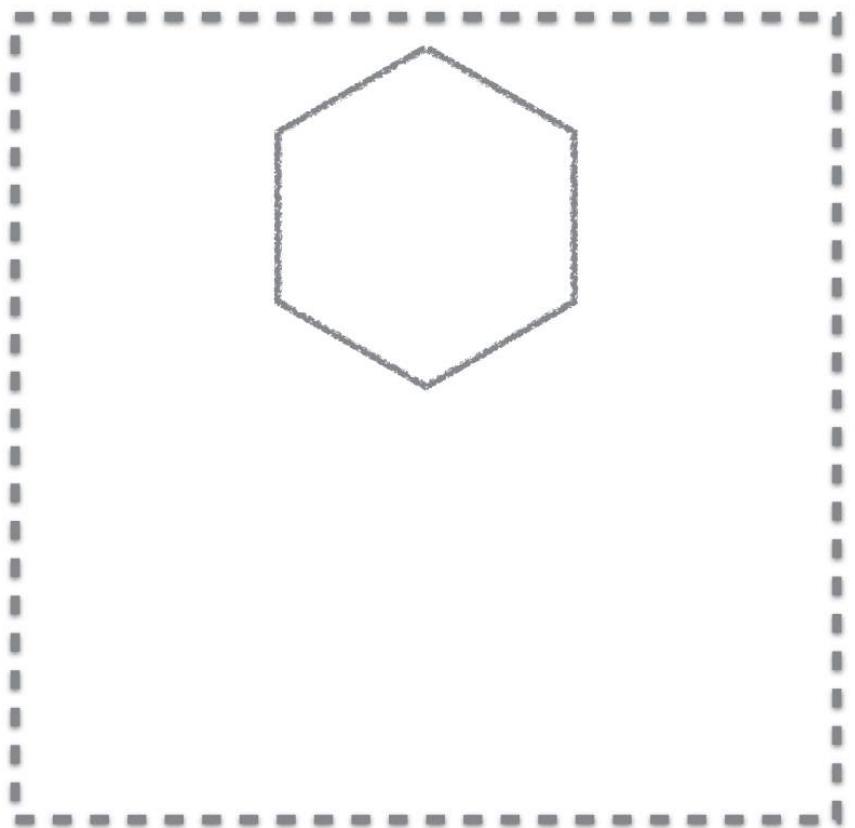


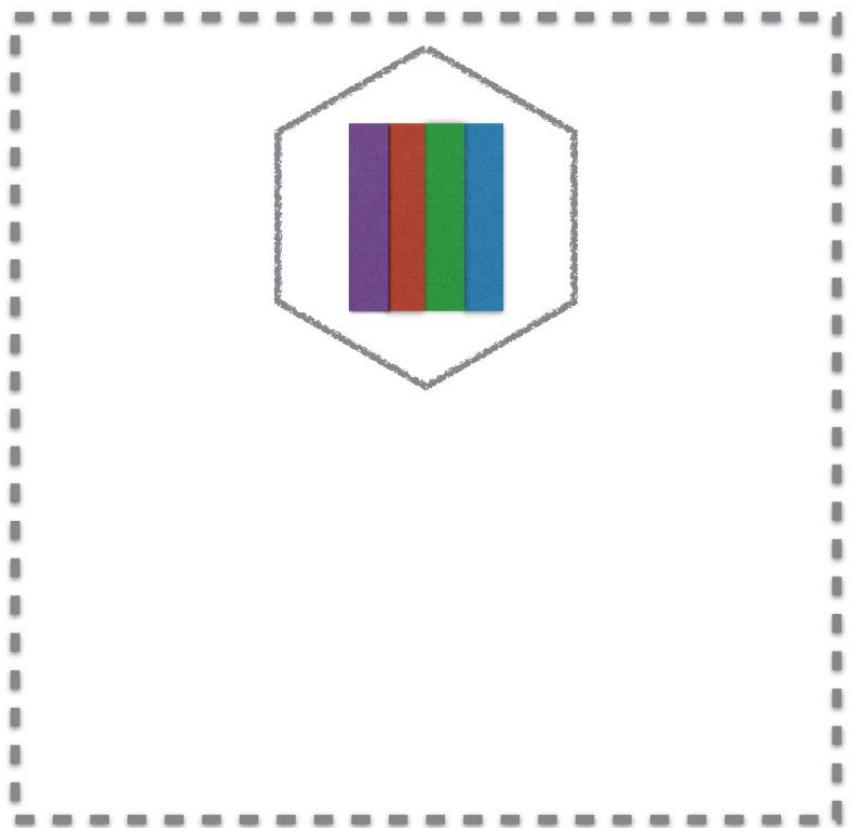
SIDECAR

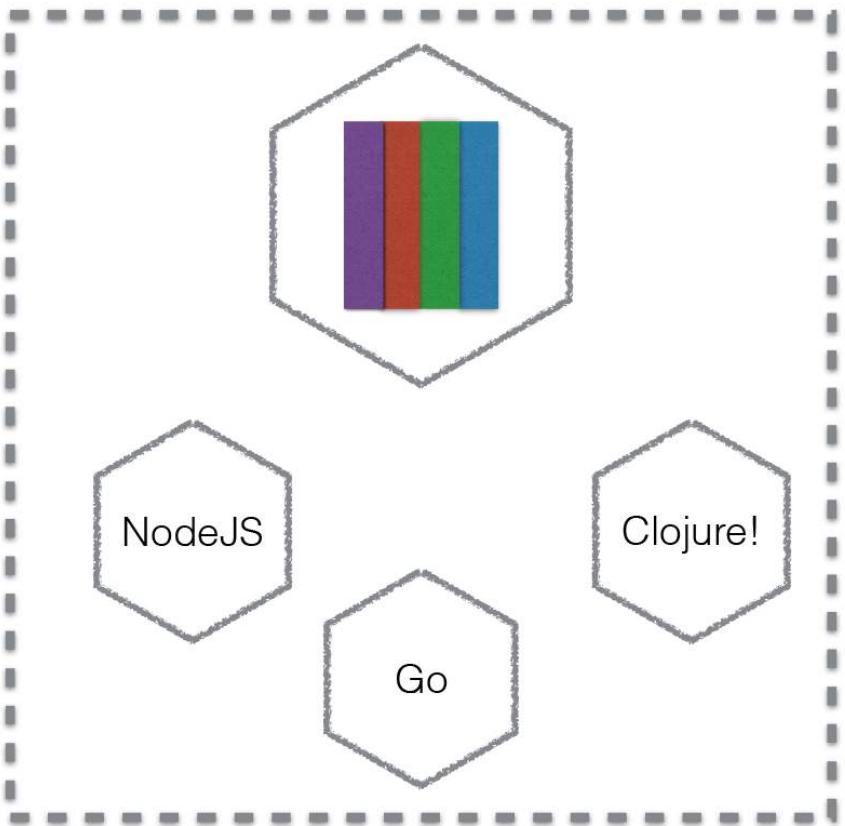


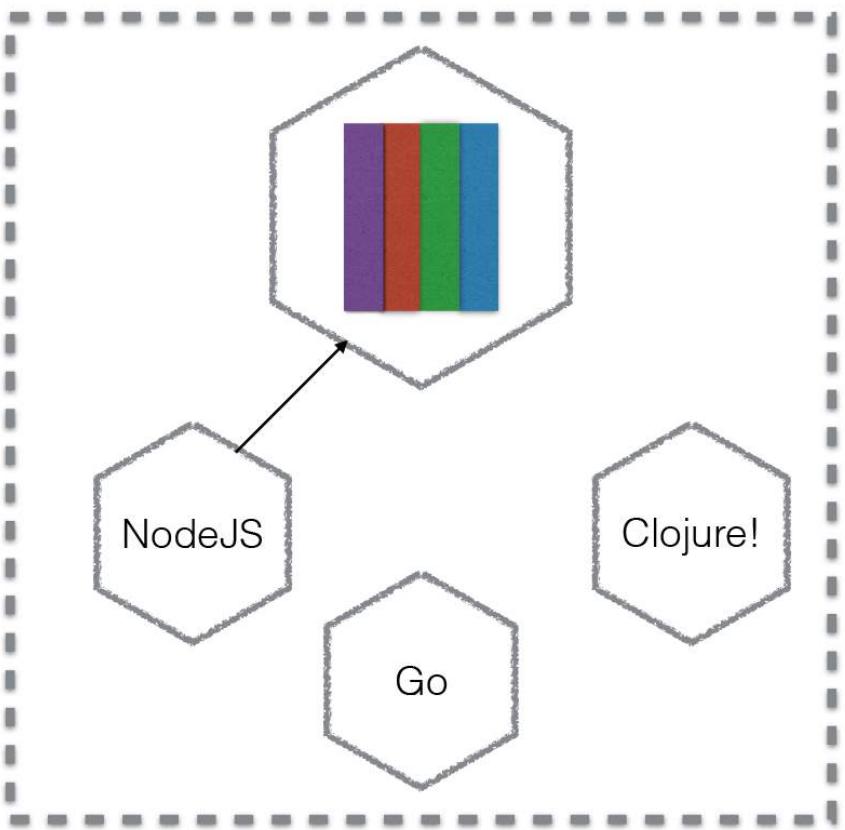
SIDECAR

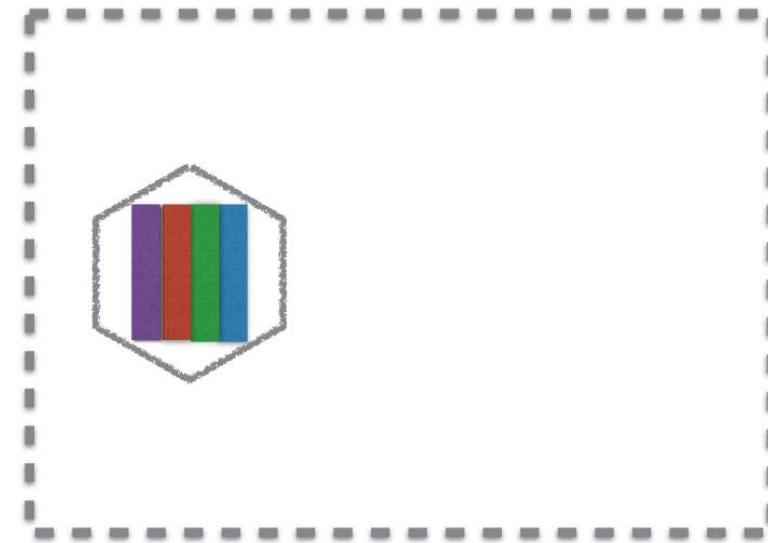
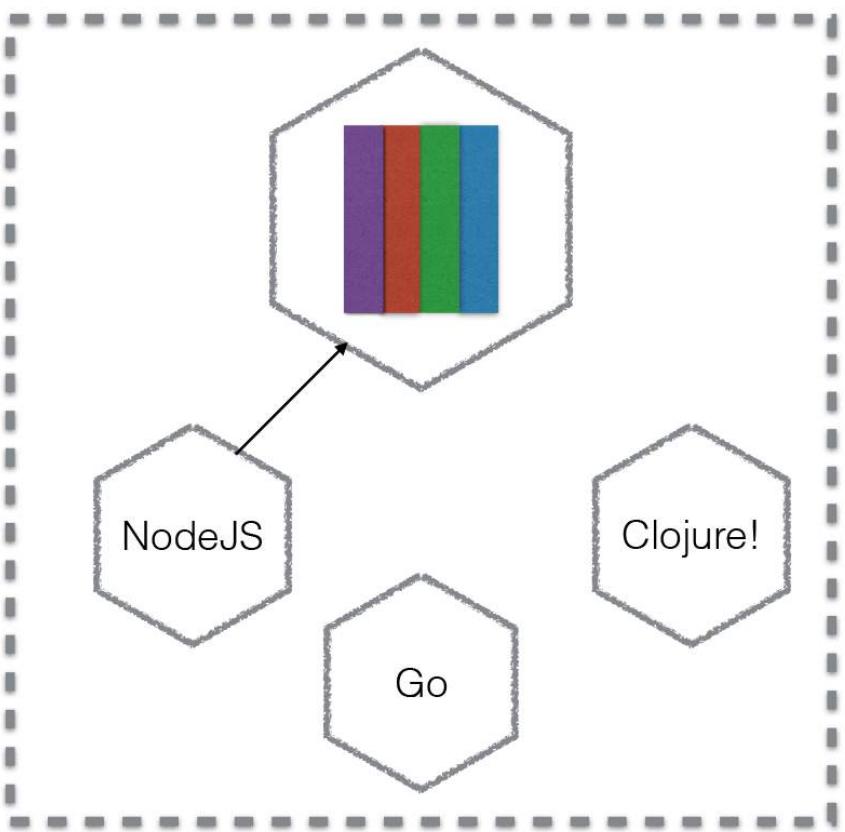


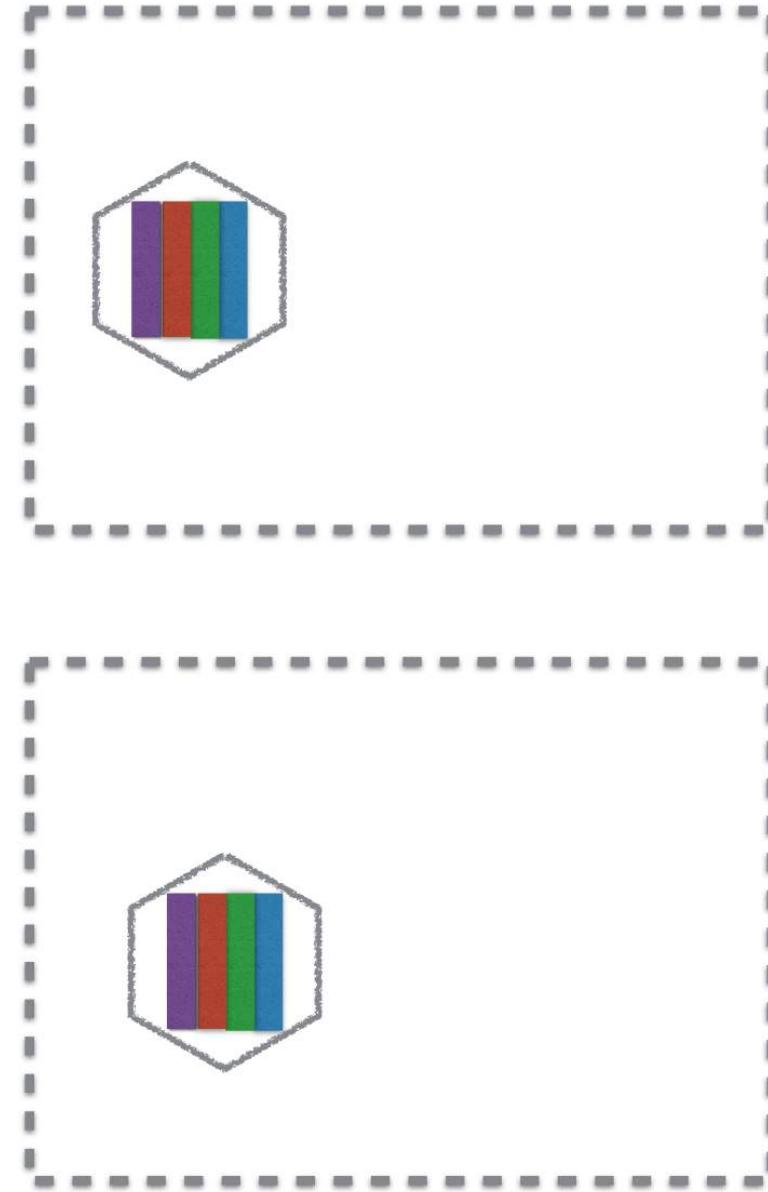
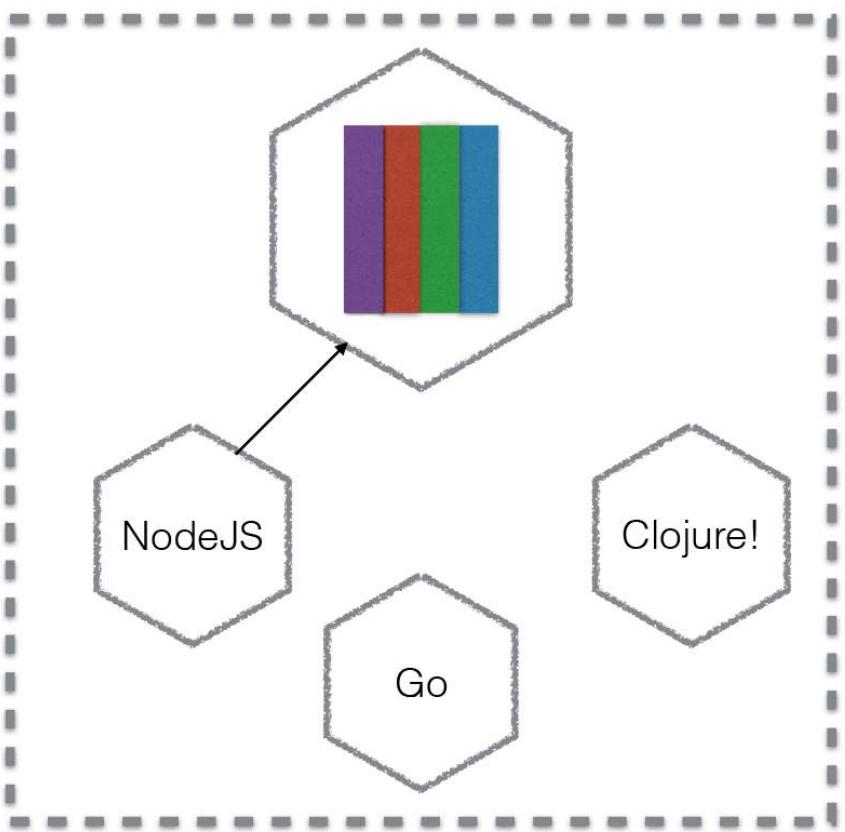


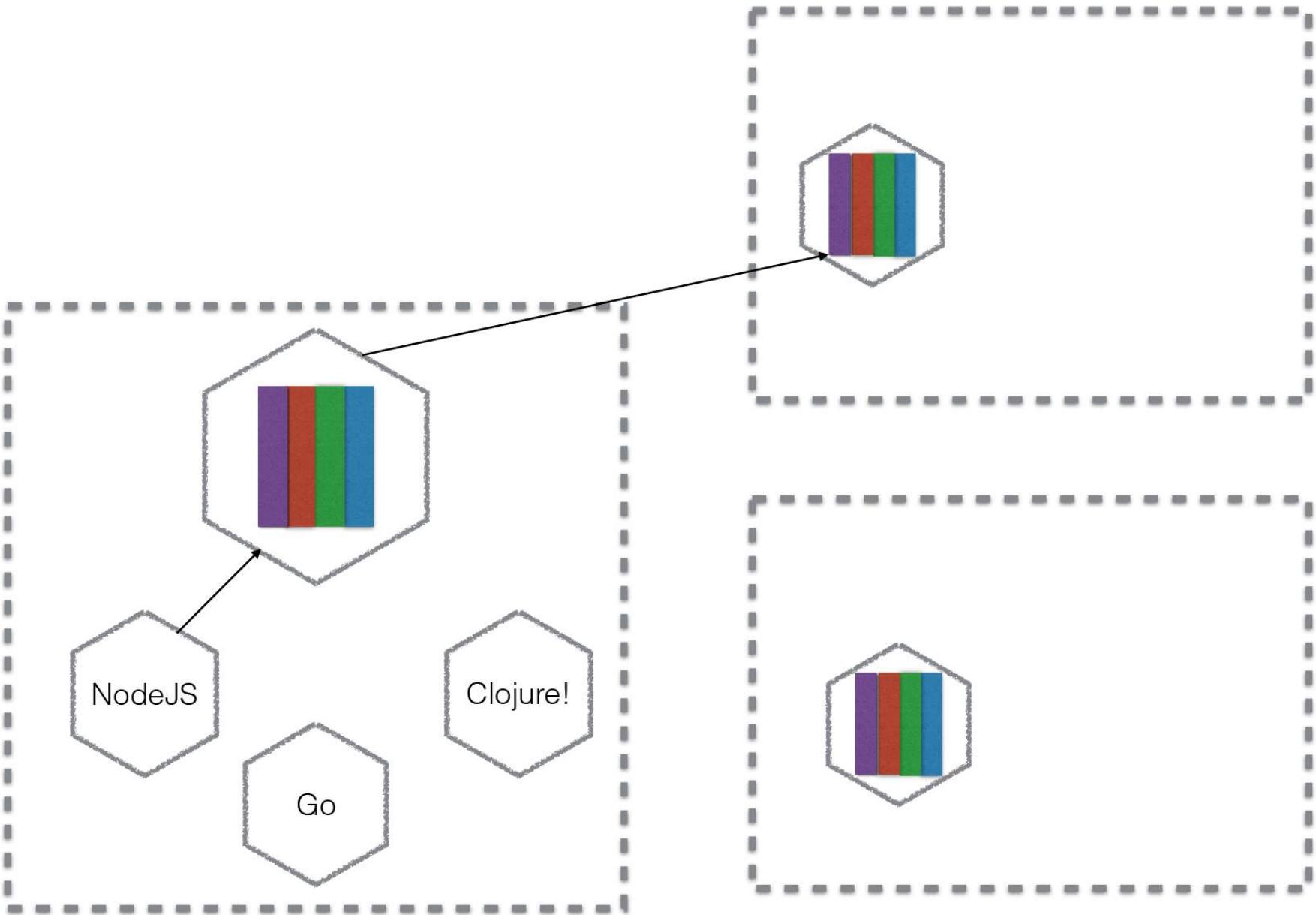


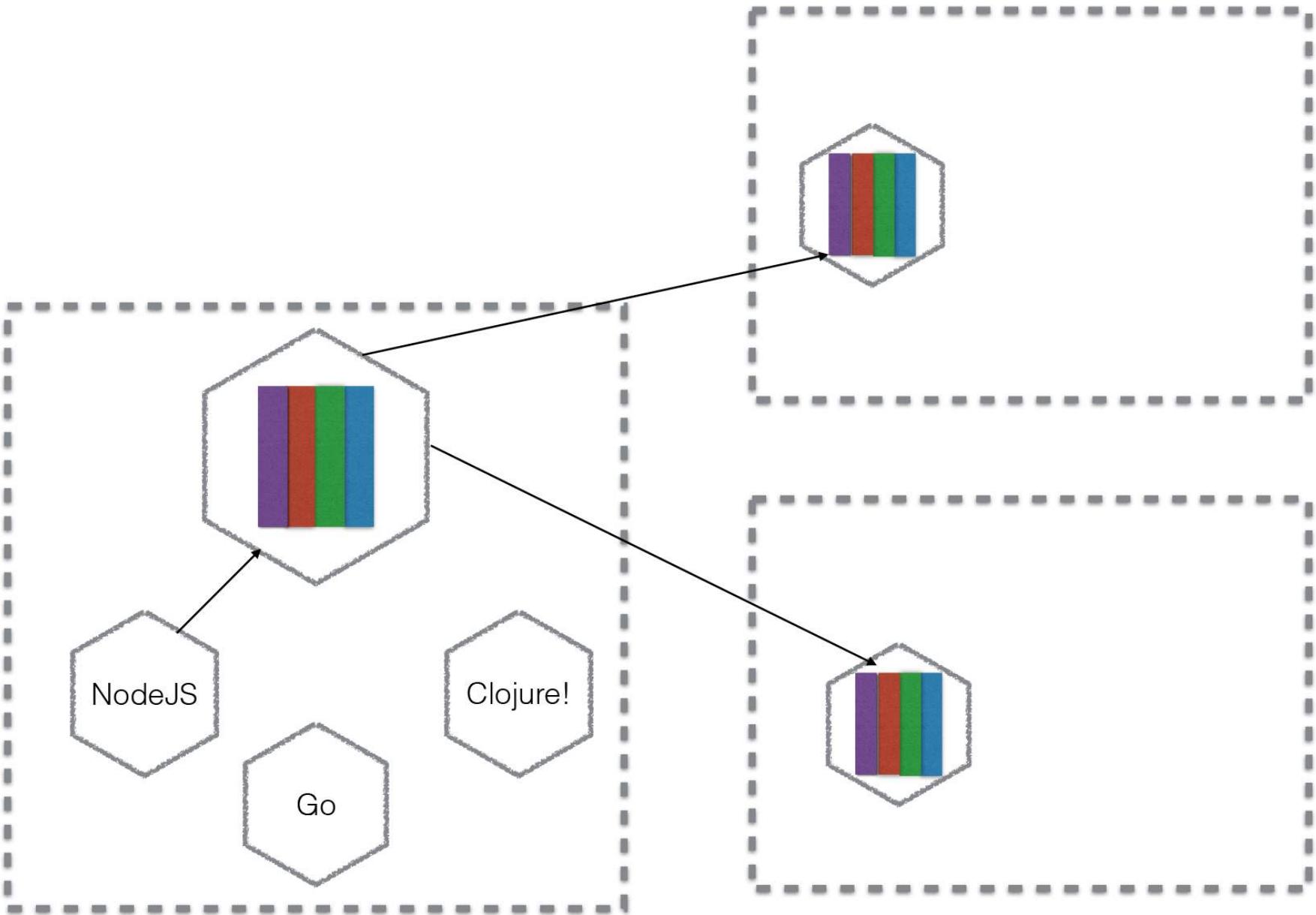


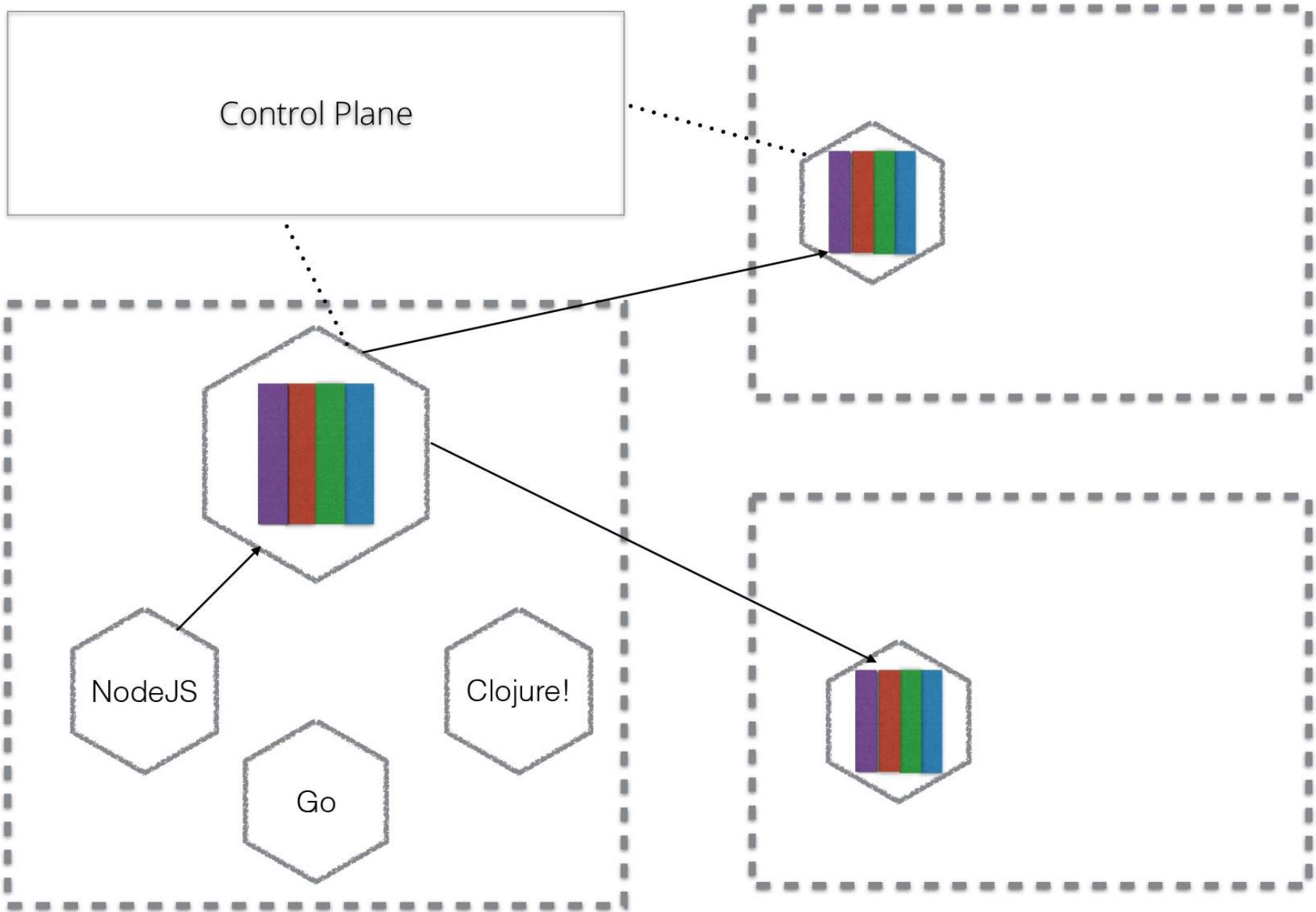












ALTERNATIVE MODELS

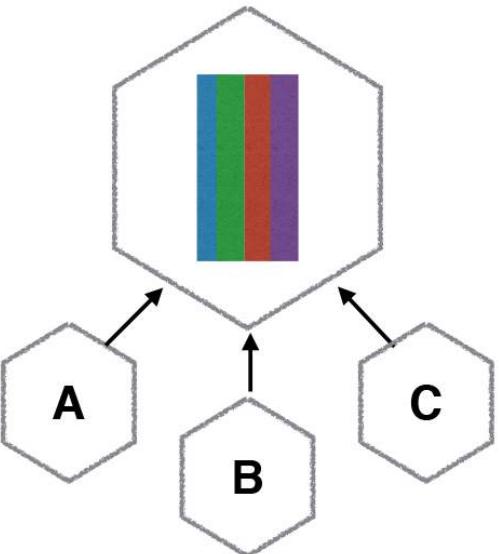
Local Proxy

Sidecar



ALTERNATIVE MODELS

Local Proxy

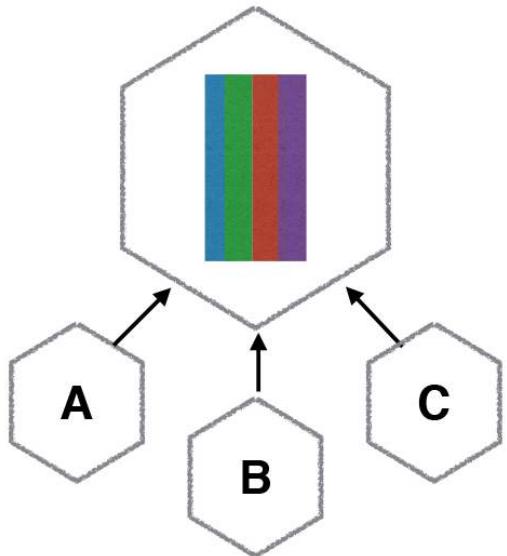


Sidecar



ALTERNATIVE MODELS

Local Proxy



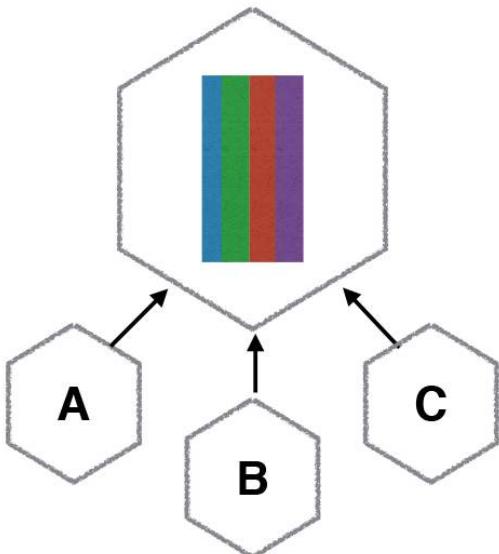
Linkerd

Sidecar

⋮

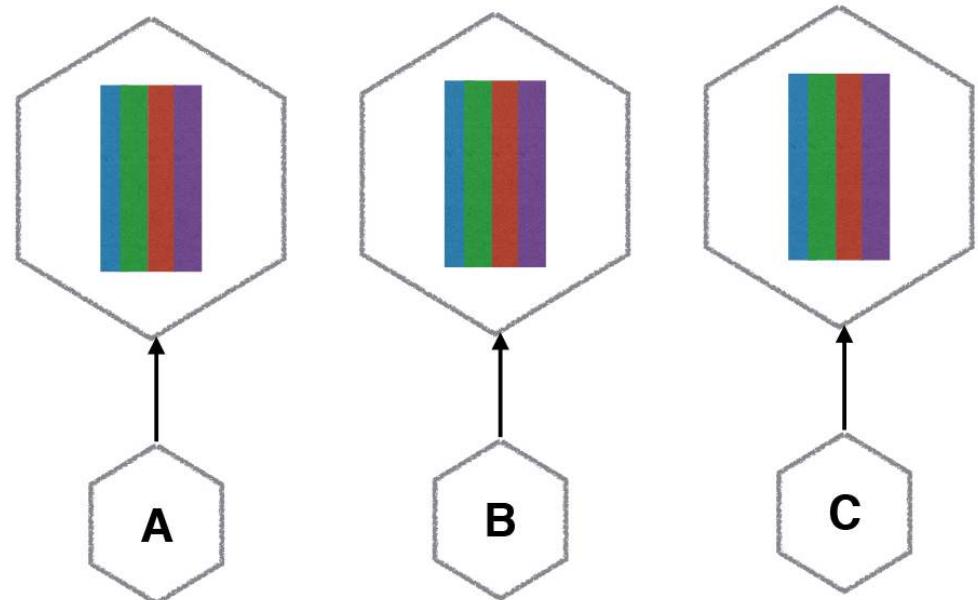
ALTERNATIVE MODELS

Local Proxy



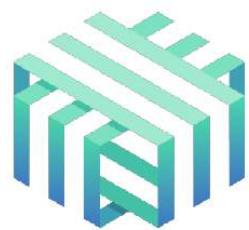
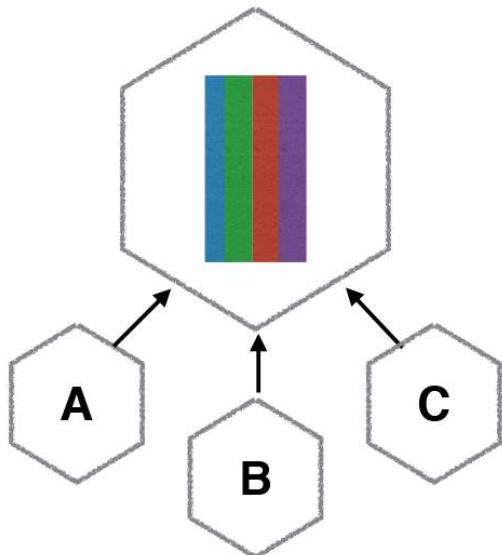
Linkerd

Sidecar



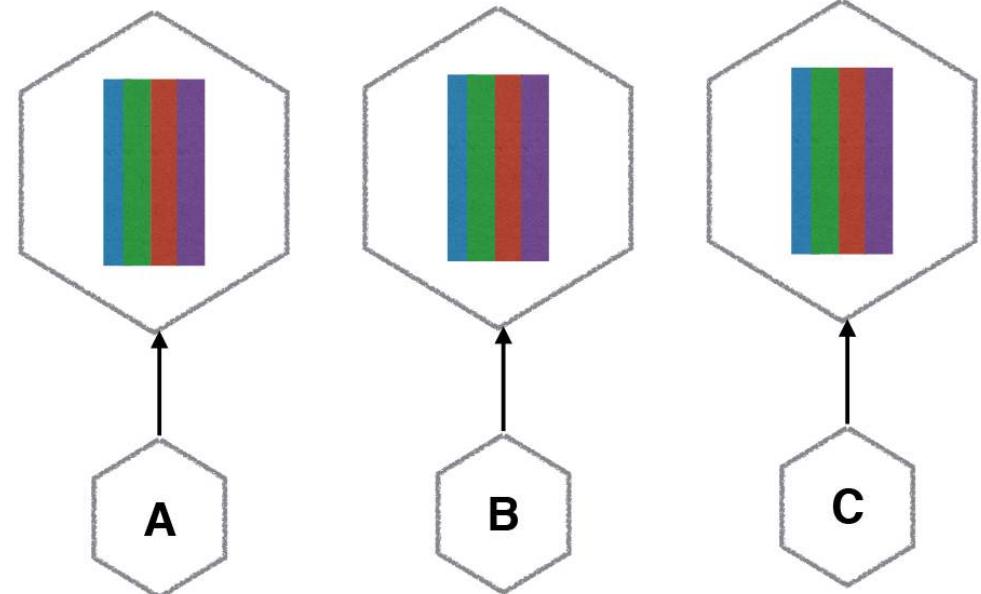
ALTERNATIVE MODELS

Local Proxy



Linkerd

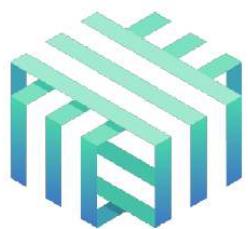
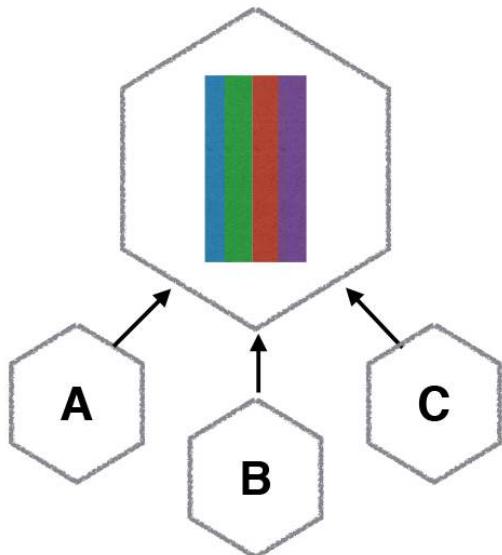
Sidecar



Istio

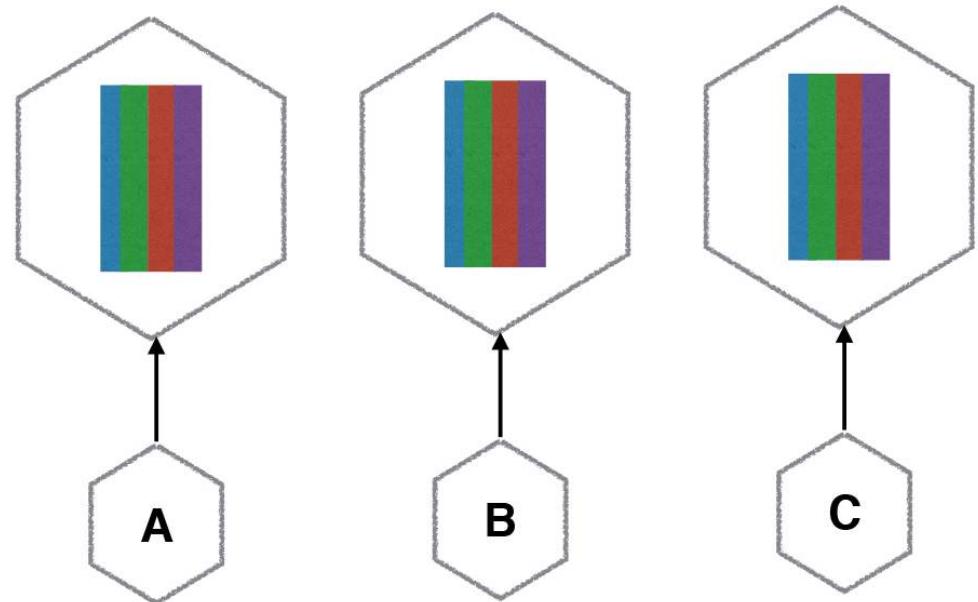
ALTERNATIVE MODELS

Local Proxy



Linkerd

Sidecar



Istio



CONDUIT

Or do you just want a message broker?

THANKS!

Sam Newman.

Home About **Talks** Podcast Writing Contact

Talks & Workshops.

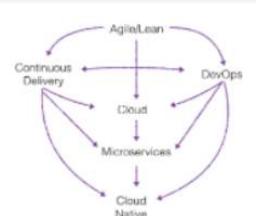
Here are a list of the talks I am currently presenting. On request, I can present different topics or even my older talks. If you want me to present these topics at your conference or company, then please [contact me](#).

You can also see where I'll be speaking next on my [events page](#).

What Is This Cloud Native Thing Anyway? 45min Talk

A talk exploring what the hell Cloud Native means

→ [Find Out More](#)



```
graph TD; Agile[Agile/Lean] --> DevOps[DevOps]; DevOps --> CD[Continuous Delivery]; CD --> Cloud[Cloud]; Cloud --> MS[Microservices]; MS --> CN[Cloud Native]; MS --> CD; CN --> MS;
```

Feature Branches And Toggles In A Post-GitHub World.

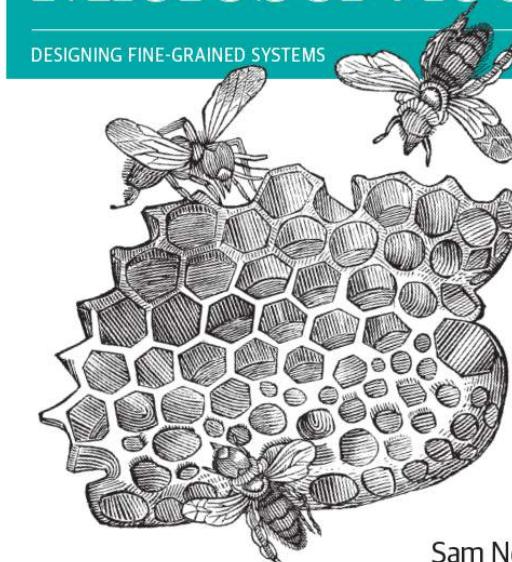


1. Validate the integration
2. When the build breaks, fix it
3. Integrate early

O'REILLY®

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



Sam Newman

<https://samnewman.io/>

@samnewman