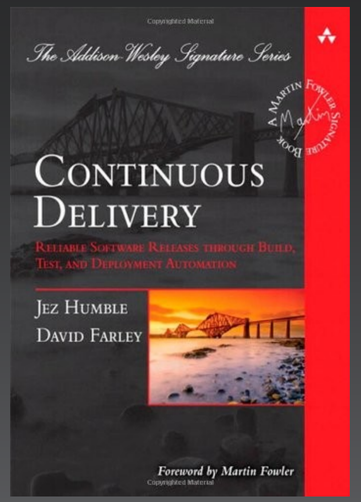
Reactive Systems

21st Century Architecture for 21st Century Problems

Dave Farley
http://www.davefarley.net
@davefarley77

Continuous Delivery ltd

http://www.continuous-delivery.co.uk



Our World Is Changing

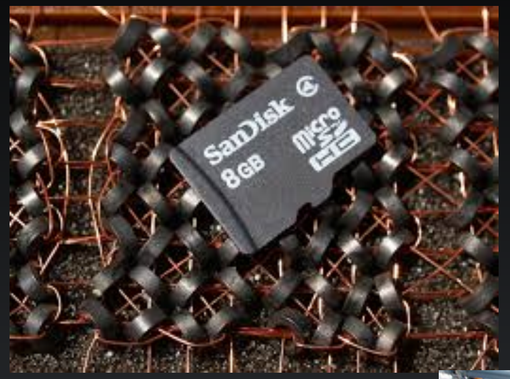
Large Applications circa 2005:

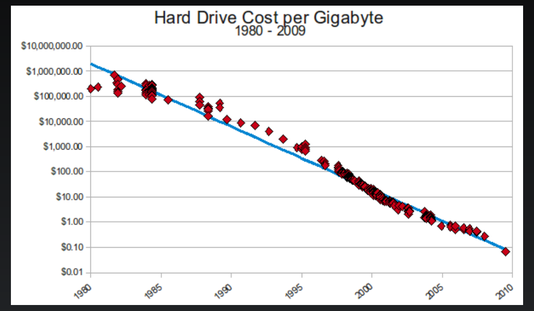
- 10's of Servers
- Seconds of Response Time
- Hours of Offline Maintenance
- Gigabytes of Data

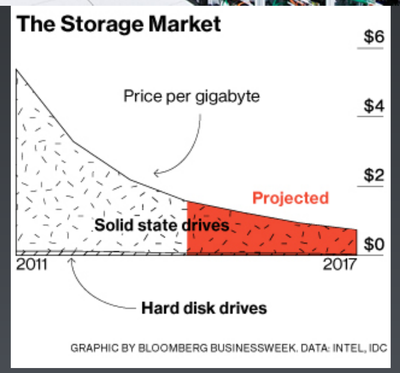
Large Applications Now:

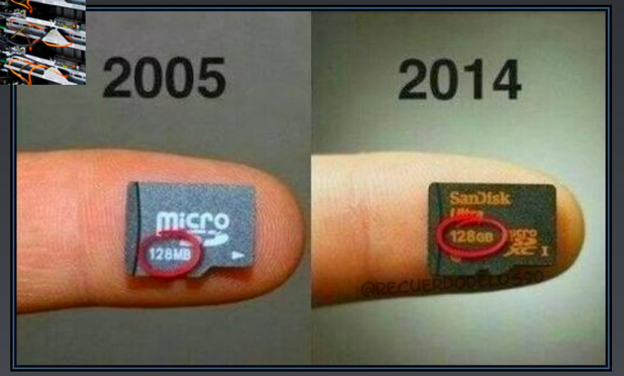
- Handheld Devices to 1000's of multi-core processors
- Millisecond Response Time
- 100% Uptime
- Petabytes of Data

Our World Is Changing









Continuous Delivery ltd

Mechanical Sympathy

What if 1 CPU cycle took 1 second?

Continuous Delivery ltd

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***
Solid-state disk I/O	*100 us*	

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***
Solid-state disk I/O	*100 us*	***4 days***

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***
Solid-state disk I/O	*100 us*	***4 days***
Spinning Rust I/O	*5 ms*	

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***
Solid-state disk I/O	*100 us*	***4 days***
Spinning Rust I/O	*5 ms*	***6 months***

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***
Solid-state disk I/O	*100 us*	***4 days***
Spinning Rust I/O	*5 ms*	***6 months***
Internet London to Australia	*180 ms*	

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***
Solid-state disk I/O	*100 us*	***4 days***
Spinning Rust I/O	*5 ms*	***6 months***
Internet London to Australia	*180 ms*	***19 years***

Mechanical Sympathy

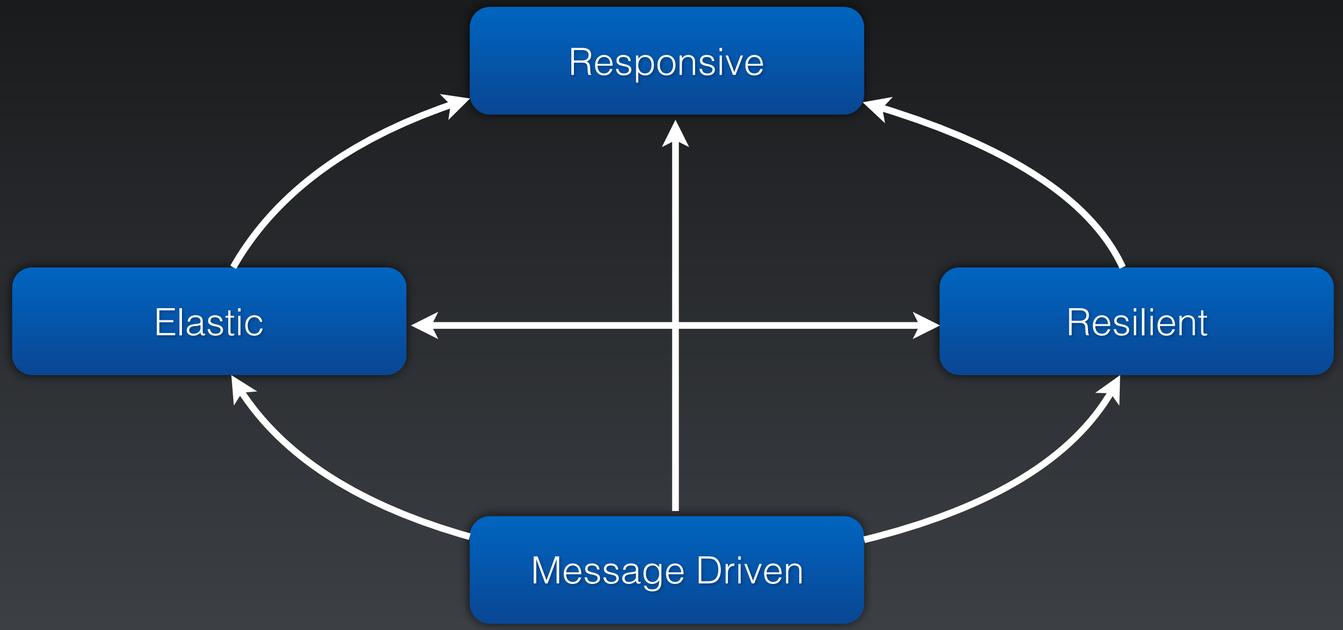
1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***
Solid-state disk I/O	*100 us*	***4 days***
Spinning Rust I/O	*5 ms*	***6 months***
Internet London to Australia	*180 ms*	***19 years***
Computer Reboot	*5 min*	

Mechanical Sympathy

1 CPU cycle	*0.3 ns*	***1 s***
Level 1 Cache Hit	*0.9 ns*	***3 s***
Level 2 Cache Hit	*2.8 ns*	***9 s***
Level 3 Cache Hit	*12.9 ns*	***43 s***
Main Memory Access	*120 ns*	***6 min***
Computer to Compter over 10m fibre	*20.05 us*	***18 h***
Solid-state disk I/O	*100 us*	***4 days***
Spinning Rust I/O	*5 ms*	***6 months***
Internet London to Australia	*180 ms*	***19 years***
Computer Reboot	*5 min*	***31,000 years***

The Reactive Manifesto

"21st Century Problems are not best solved with 20th Century Software Architectures"



The Evolution of modern hardware has changed many of the common assumptions of software development

Source: www.reactivemanifesto.org

Reactive Systems Are:

Responsive:

- Responds in a Timely Manner
- Cornerstone of Usability
- Also Quick to Detect Problems

Reactive Systems Are:

Resilient:

- Remains Responsive in the Face of Failure
- Resilience Depends on - Replication, Containment, Isolation and Delegation

Reactive Systems Are:

Elastic:

- Remains Responsive Under Varying Workload
- Responds to Change in the Input Rate By Increasing or Decreasing Resources that Service the Input
- Decentralised Architecture, No Contention Points, No Central Bottlenecks

Reactive Systems Are:

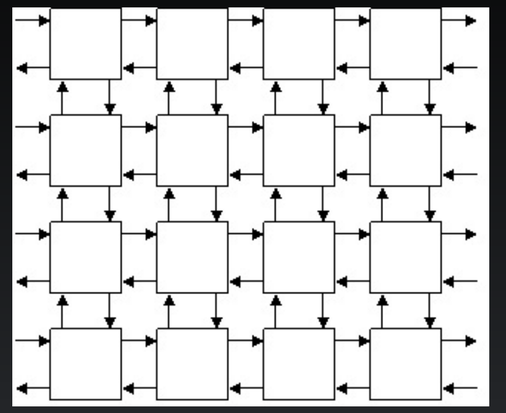
Message Driven:

- Asynchronous Message Passing is the foundation for all of these properties
- Loose-Coupling, Isolation, Location Transparency
- Ability to Delegate Errors

Properties of Reactive Systems

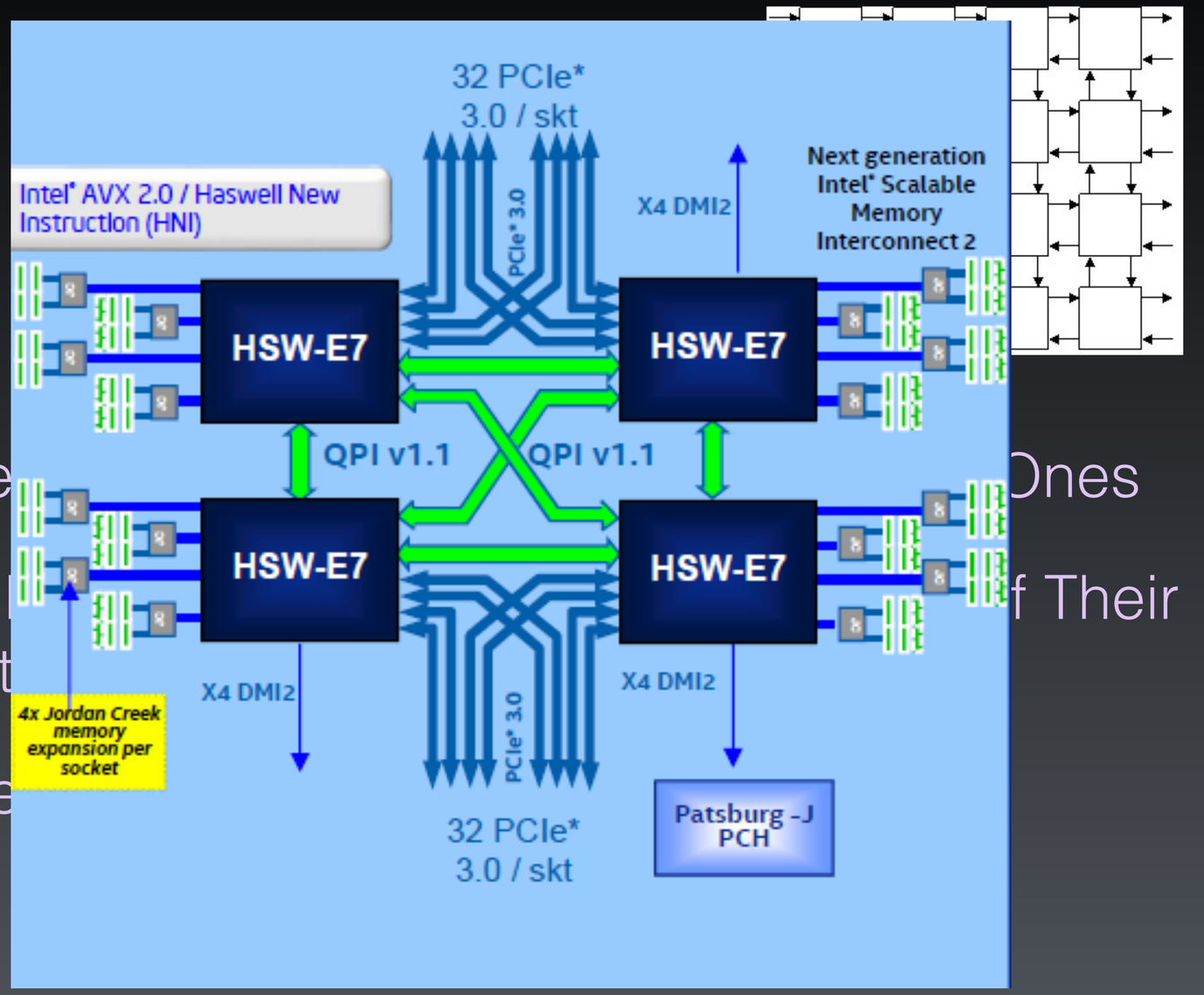
- Flexible
- Loosely-Coupled
- Scalable
- Easier to Develop
- More Tolerant of Failure
- Respond to Failure Gracefully
- Responsive to Users

Fractal Architecture



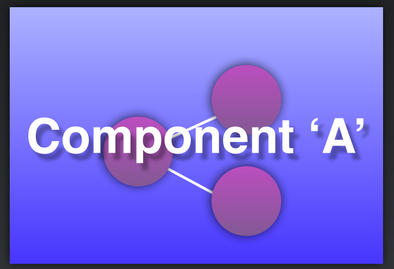
- Large Systems Are Composed of Smaller Ones
- They Depend on the Reactive Properties of Their Constituents
- These Benefits Operate At All Scales
- Such Systems are Composable

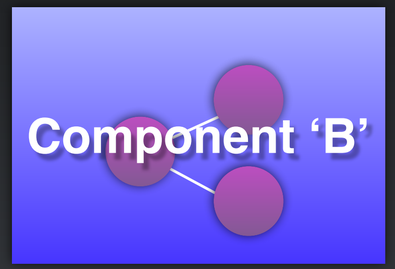
Fractal Architecture



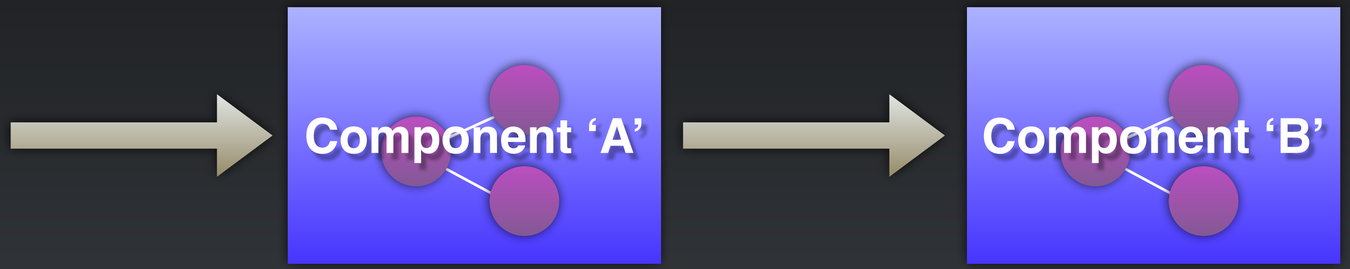
- Large Ones
- They f Their Const
- These
- Such

Failure Modes in Synchronous Messaging

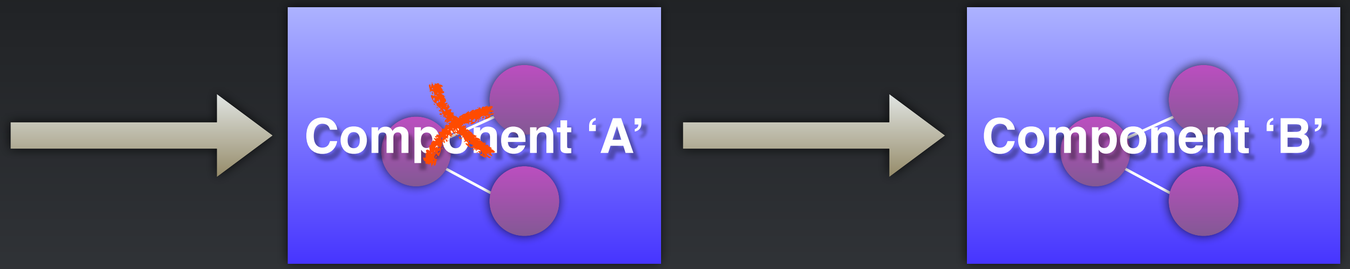




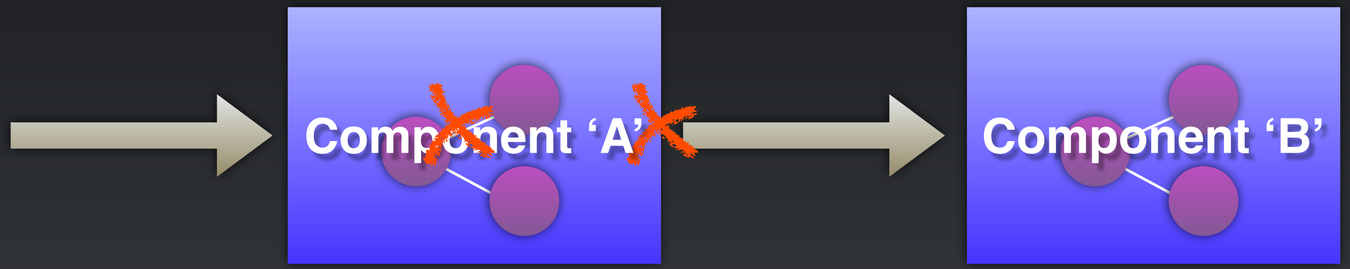
Failure Modes in Synchronous Messaging



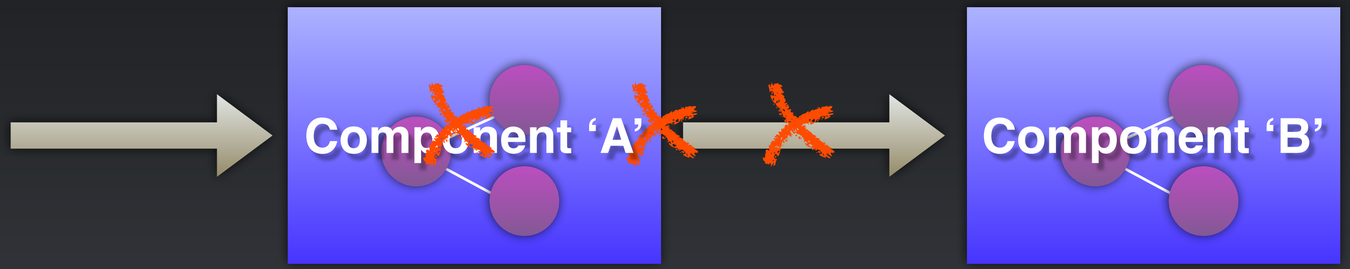
Failure Modes in Synchronous Messaging



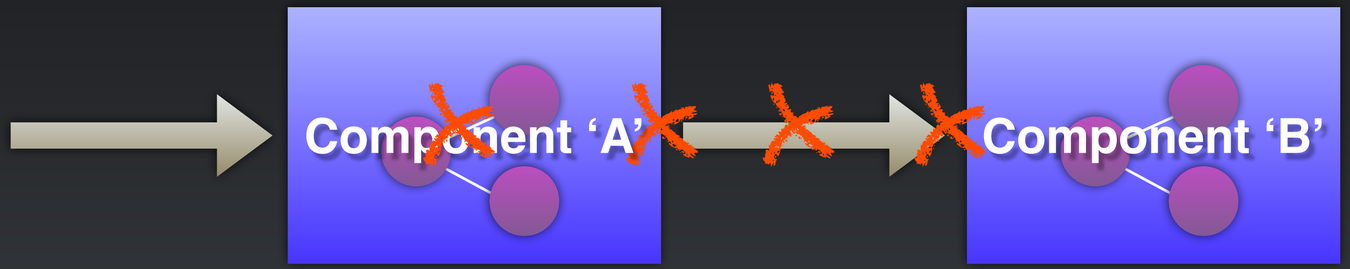
Failure Modes in Synchronous Messaging



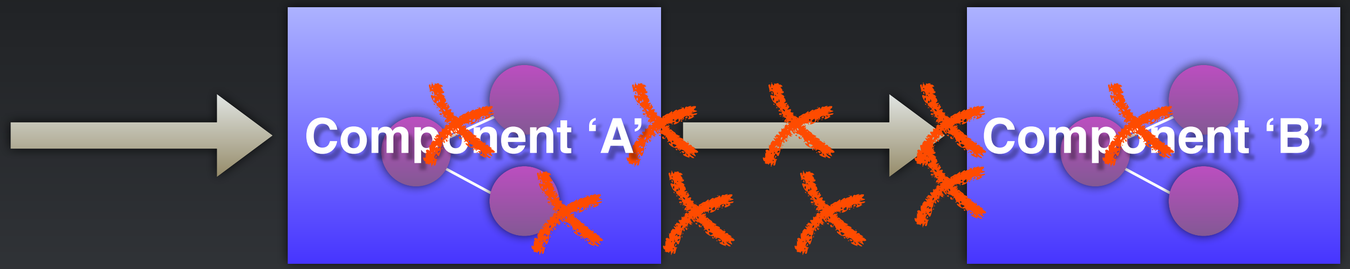
Failure Modes in Synchronous Messaging



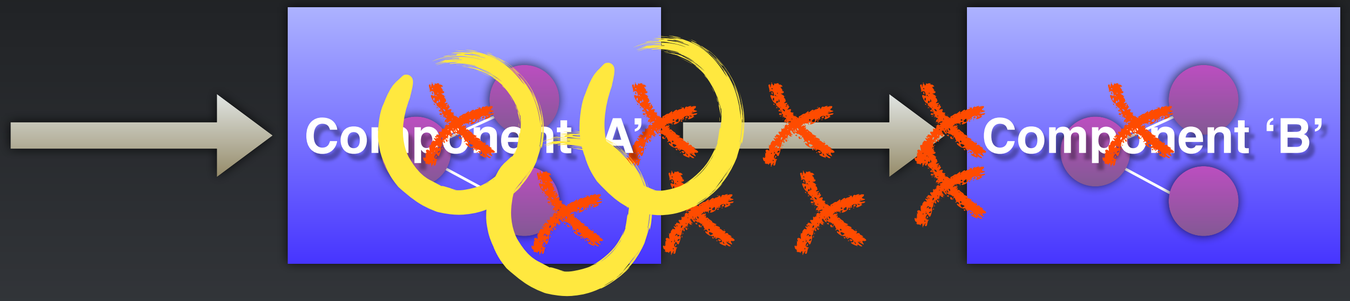
Failure Modes in Synchronous Messaging



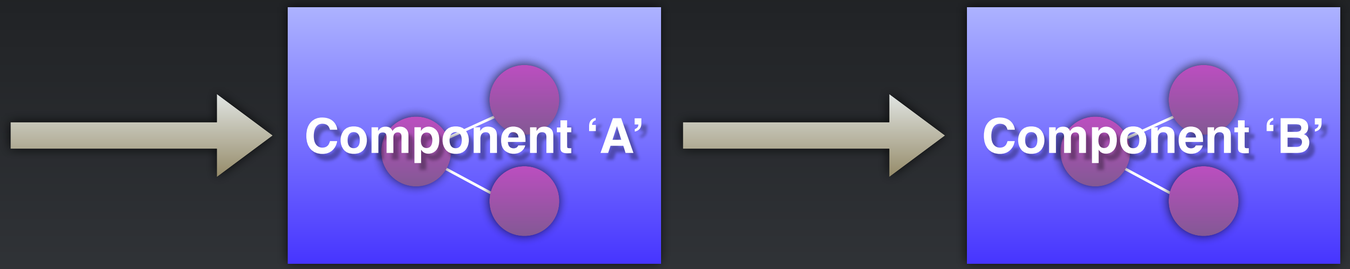
Failure Modes in Synchronous Messaging



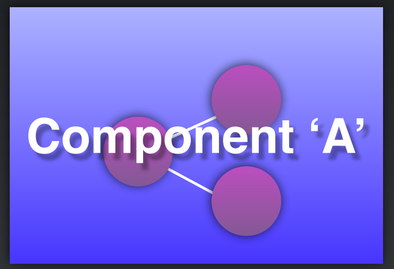
Failure Modes in Synchronous Messaging

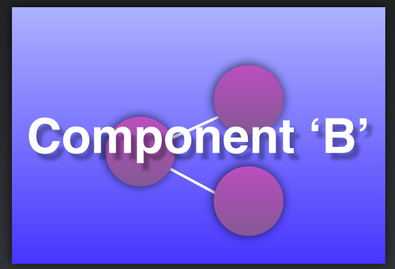


Failure Modes in Synchronous Messaging



Synch Messaging Breeds Complexity

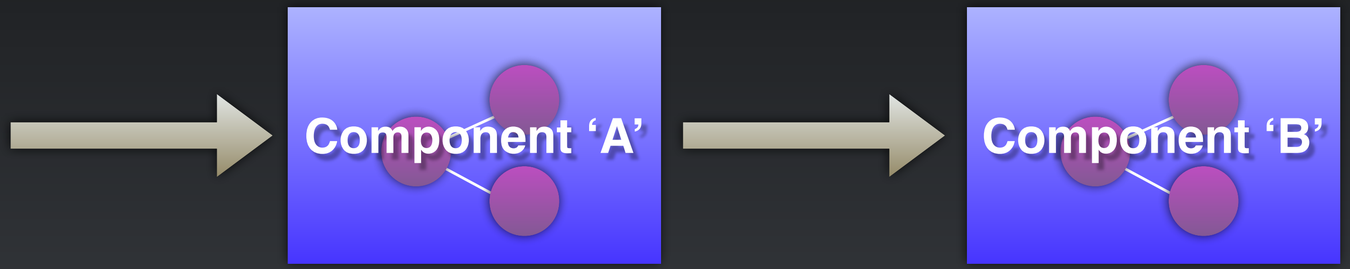




Synch Messaging Breeds Complexity

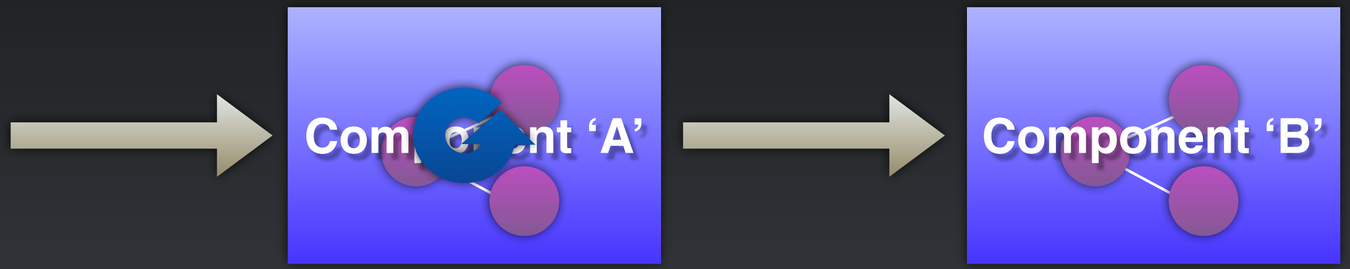
Synchronous Comms Increases Coupling in Location and Time

Synch Messaging Breeds Complexity



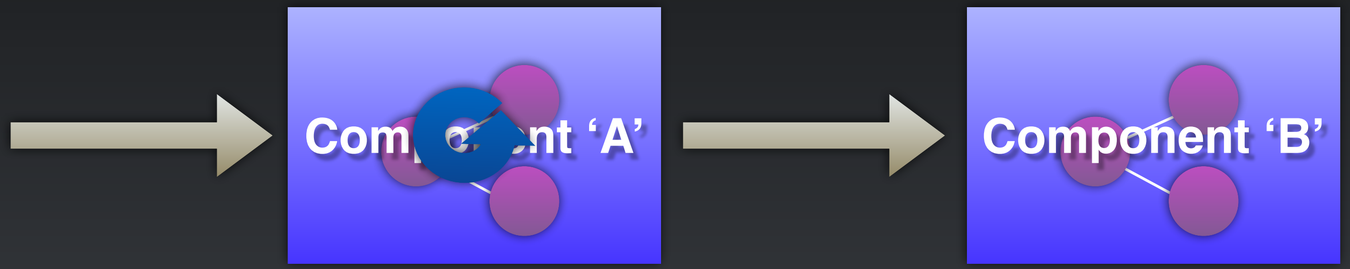
Synchronous Comms Increases Coupling in Location and Time

Synch Messaging Breeds Complexity



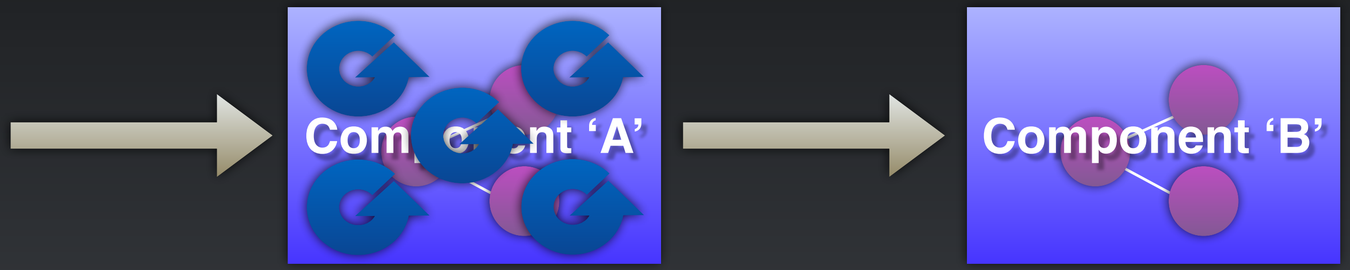
Synchronous Comms Increases Coupling in Location and Time

Synch Messaging Breeds Complexity



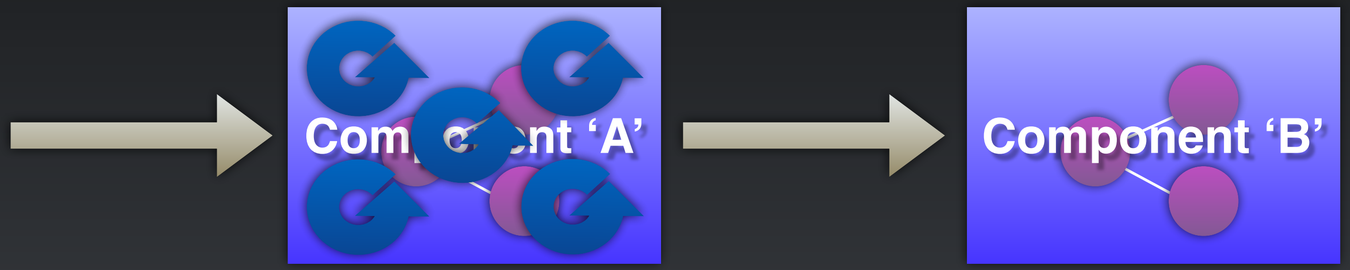
Synchronous Comms Increases Coupling in Location and Time

Synch Messaging Breeds Complexity



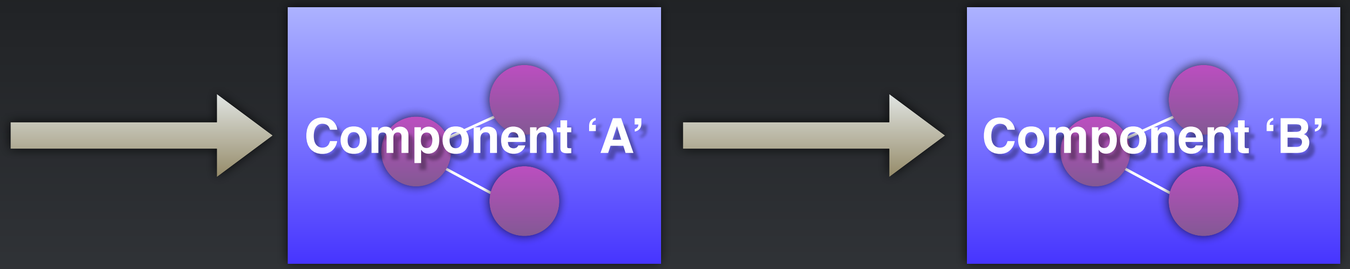
Synchronous Comms Increases Coupling in Location and Time

Synch Messaging Breeds Complexity

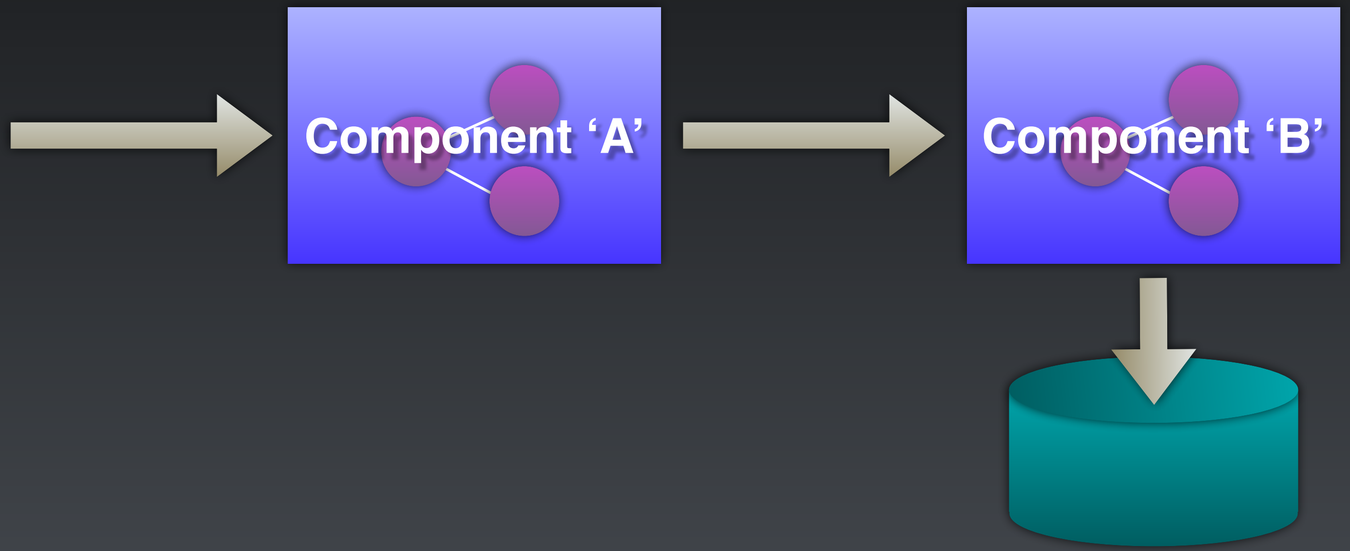


Synchronous Comms Increases Coupling in Location and Time

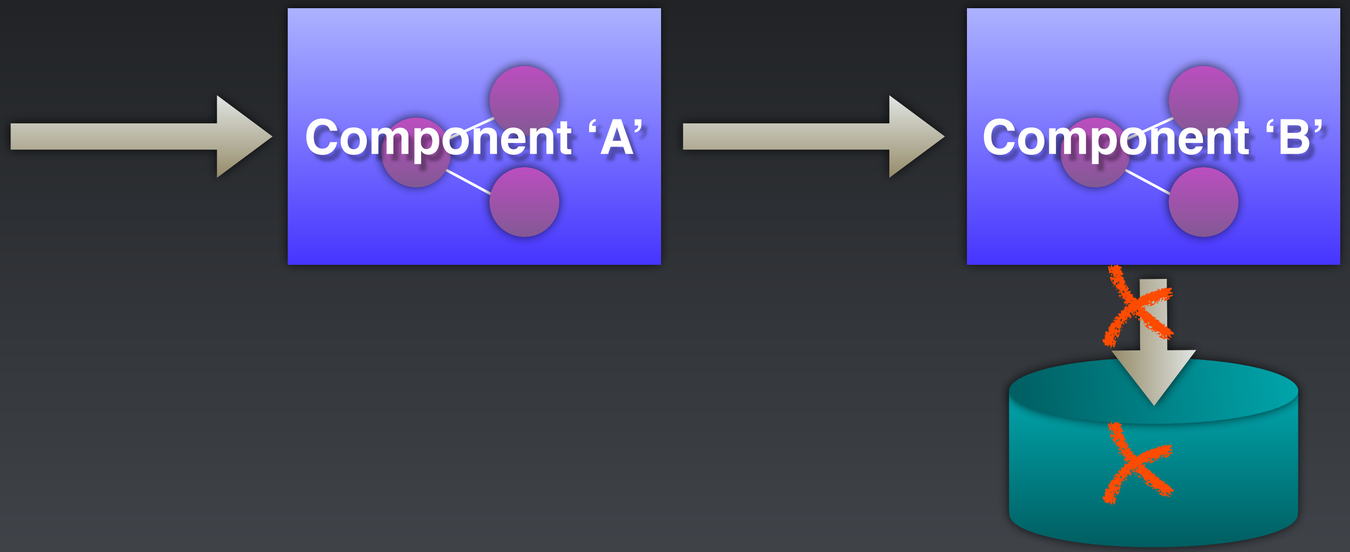
Synch Messaging Breeds Complexity



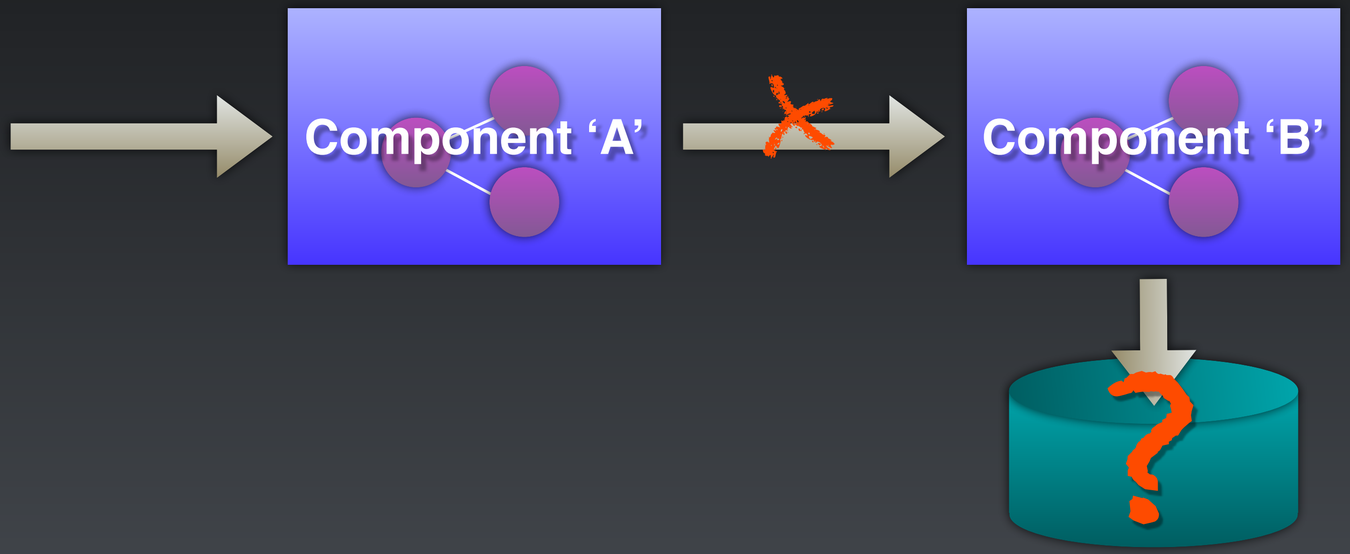
Synch Messaging Breeds Complexity



Synch Messaging Breeds Complexity

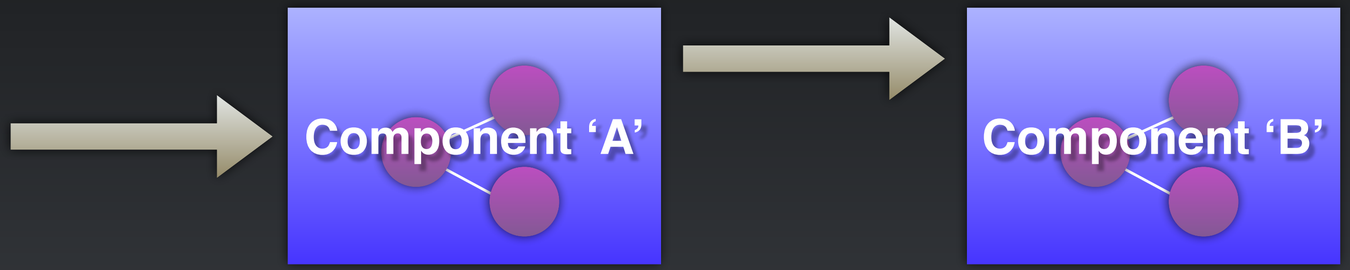


Synch Messaging Breeds Complexity



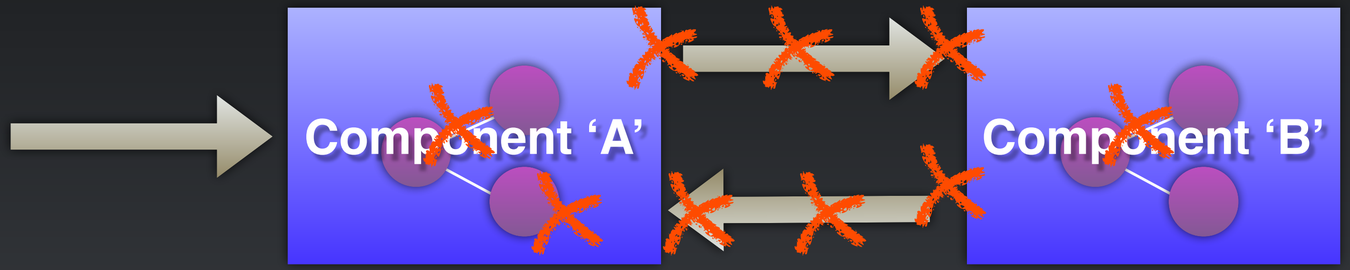
The Benefits of Asynchrony

The Benefits of Asynchrony



The Benefits of Asynchrony

The Benefits of Asynchrony



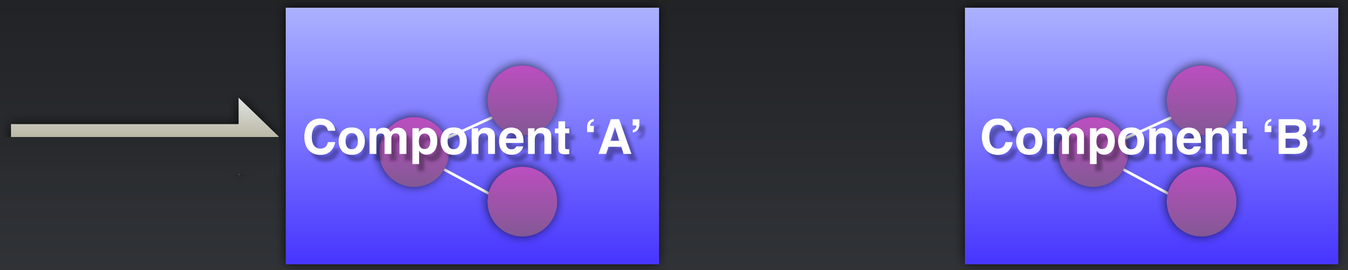
The Benefits of Asynchrony

The Benefits of Asynchrony

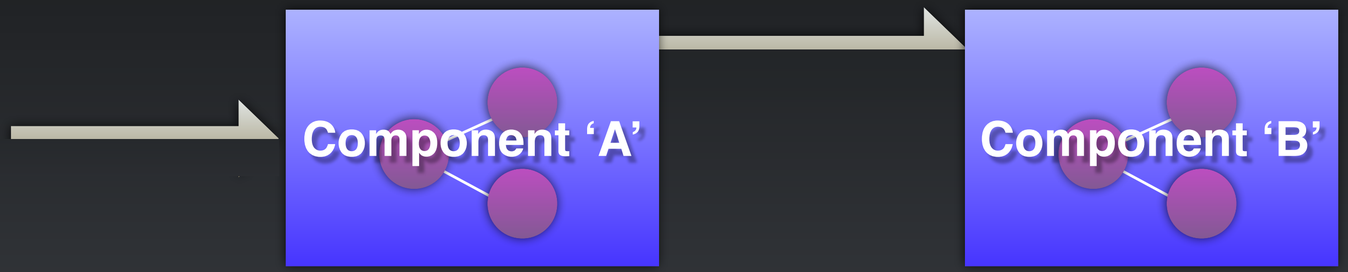
The Benefits of Asynchrony

The Benefits of Asynchrony

The Benefits of Asynchrony

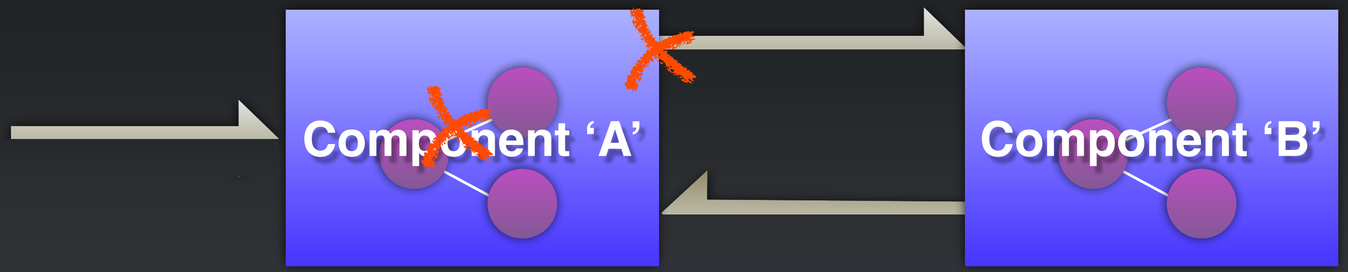


The Benefits of Asynchrony

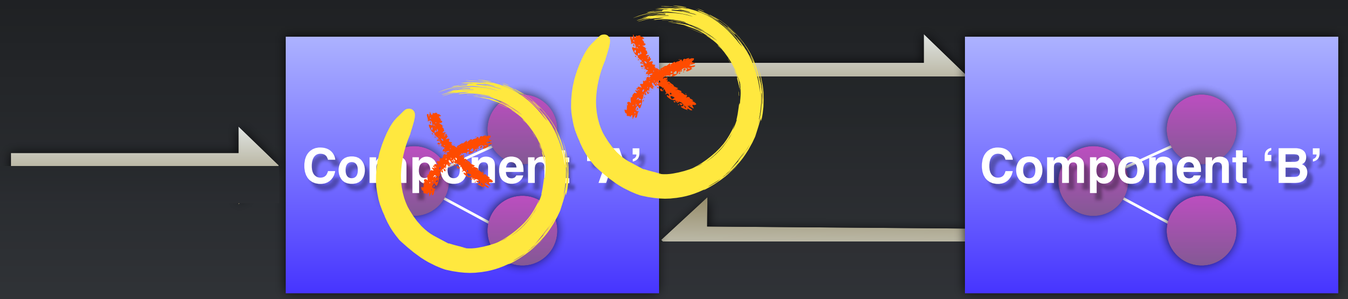


The Benefits of Asynchrony

The Benefits of Asynchrony



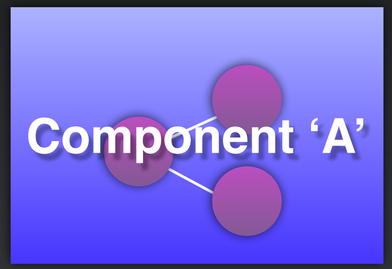
The Benefits of Asynchrony



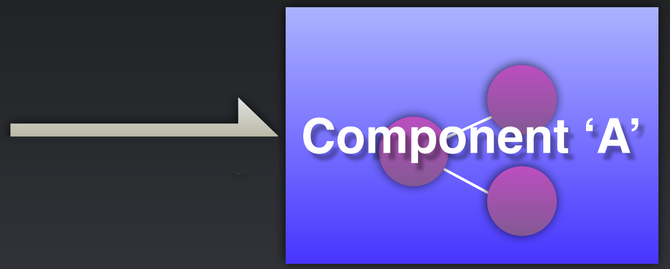
The Benefits of Asynchrony

The Benefits of Asynchrony

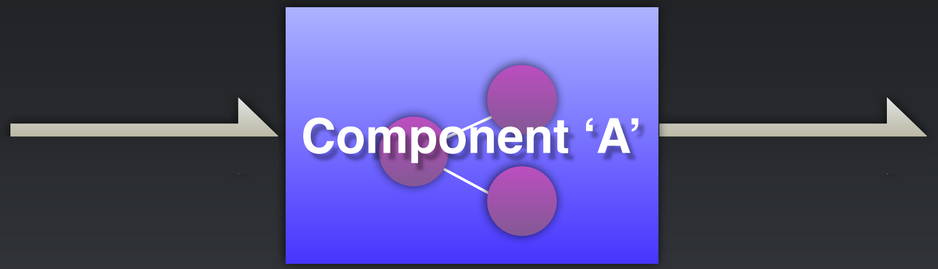
The Benefits of Asynchrony



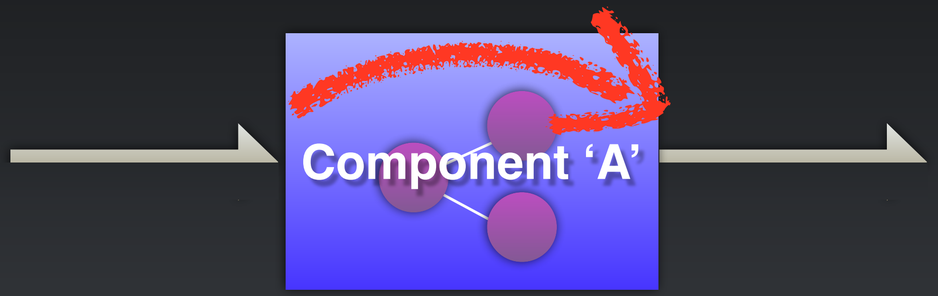
The Benefits of Asynchrony



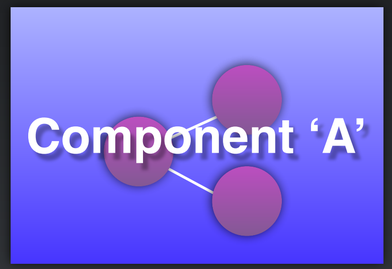
The Benefits of Asynchrony



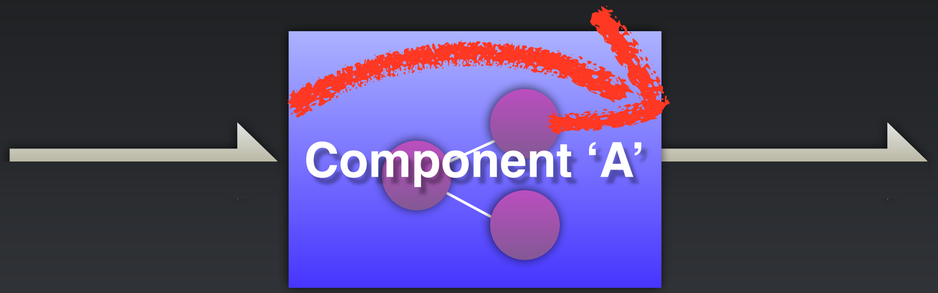
The Benefits of Asynchrony



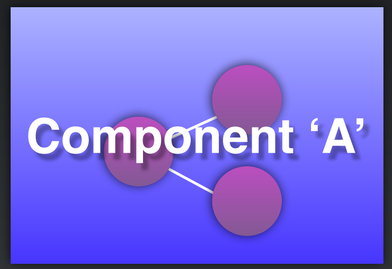
The Benefits of Asynchrony



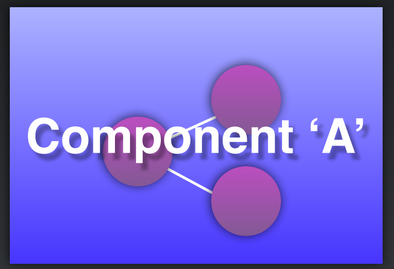
The Benefits of Asynchrony

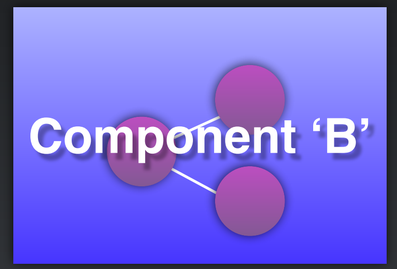


The Benefits of Asynchrony



An Example

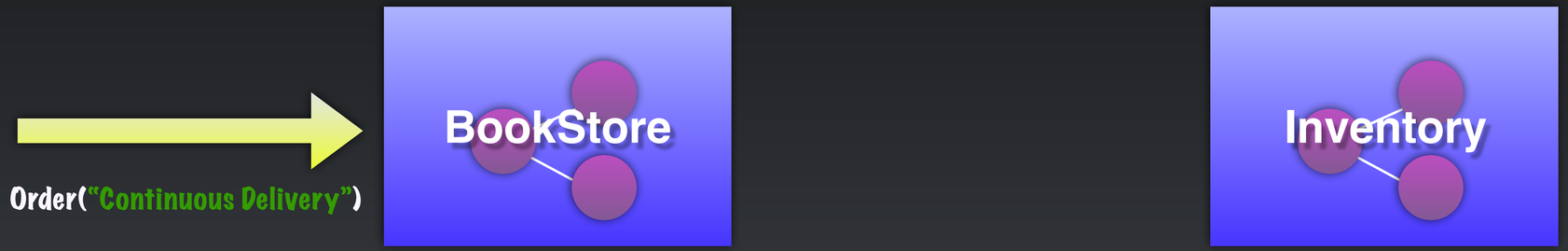




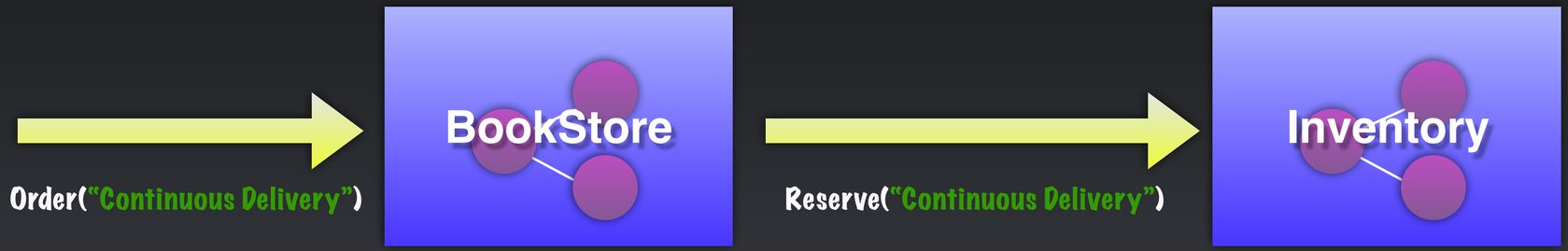
Continuous Delivery ltd

An Example

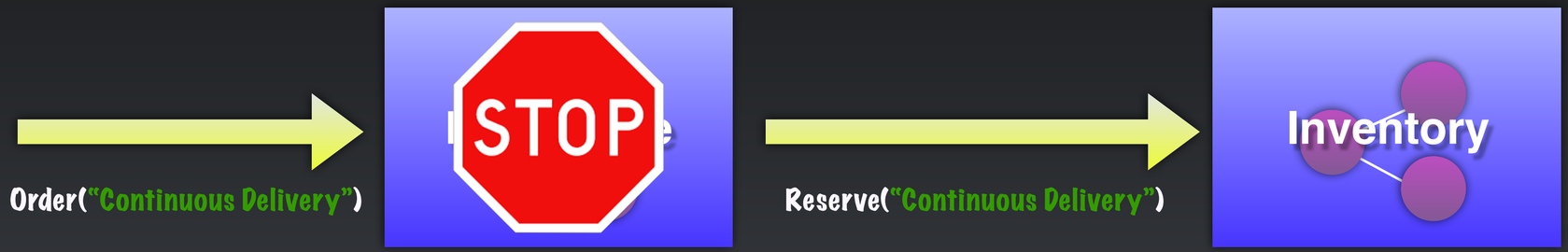
An Example



An Example

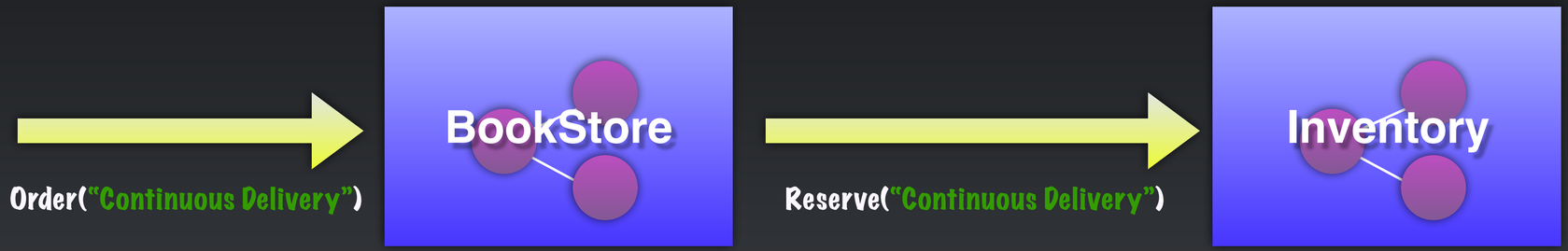


An Example



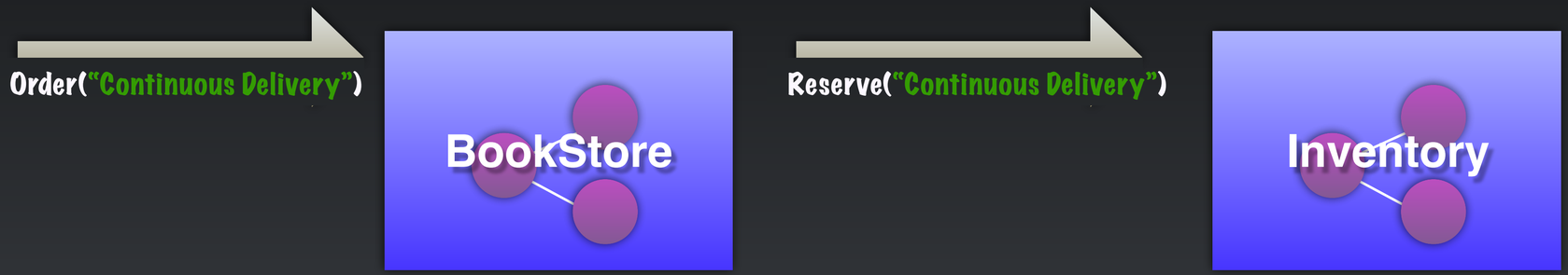
Continuous Delivery ltd

An Example

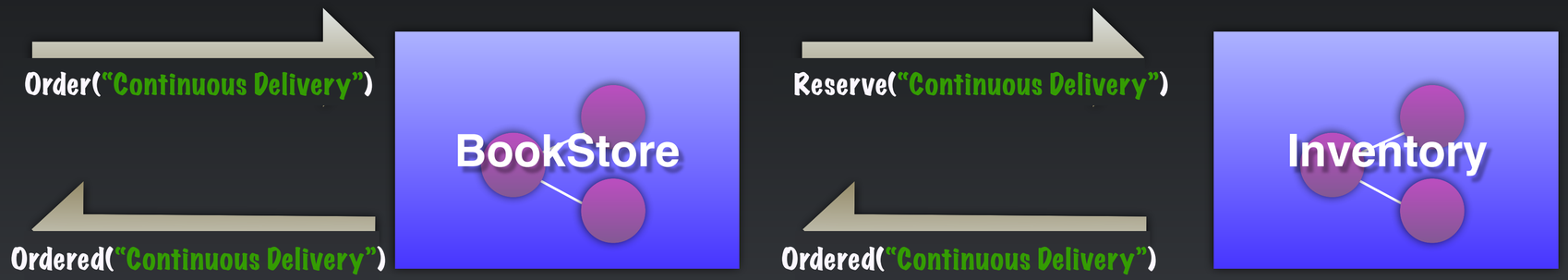


An Example

An Example

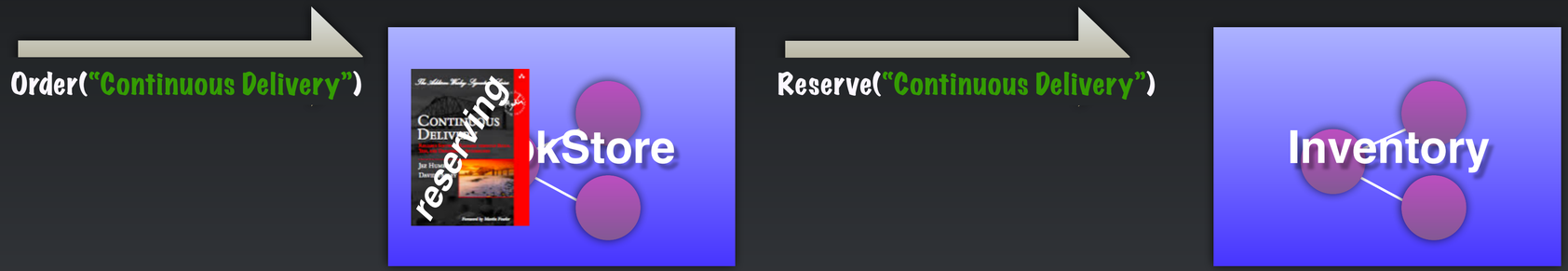


An Example

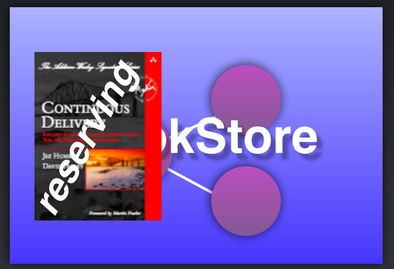


An Example

An Example

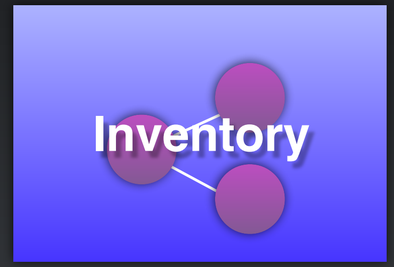


An Example

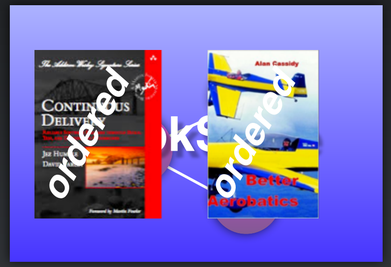


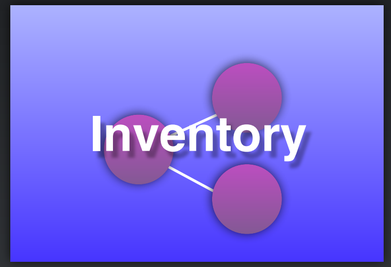
An Example

An Example



An Example



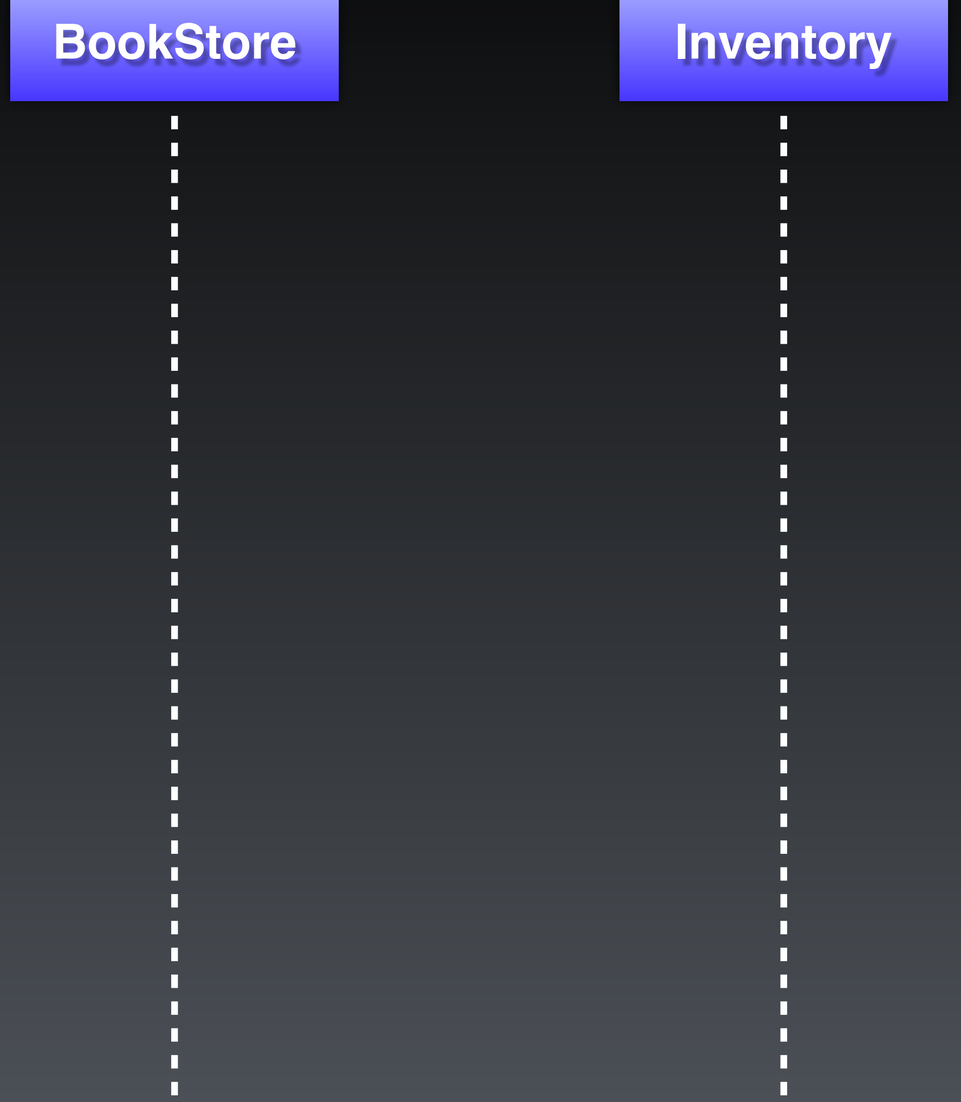


Continuous Delivery ltd

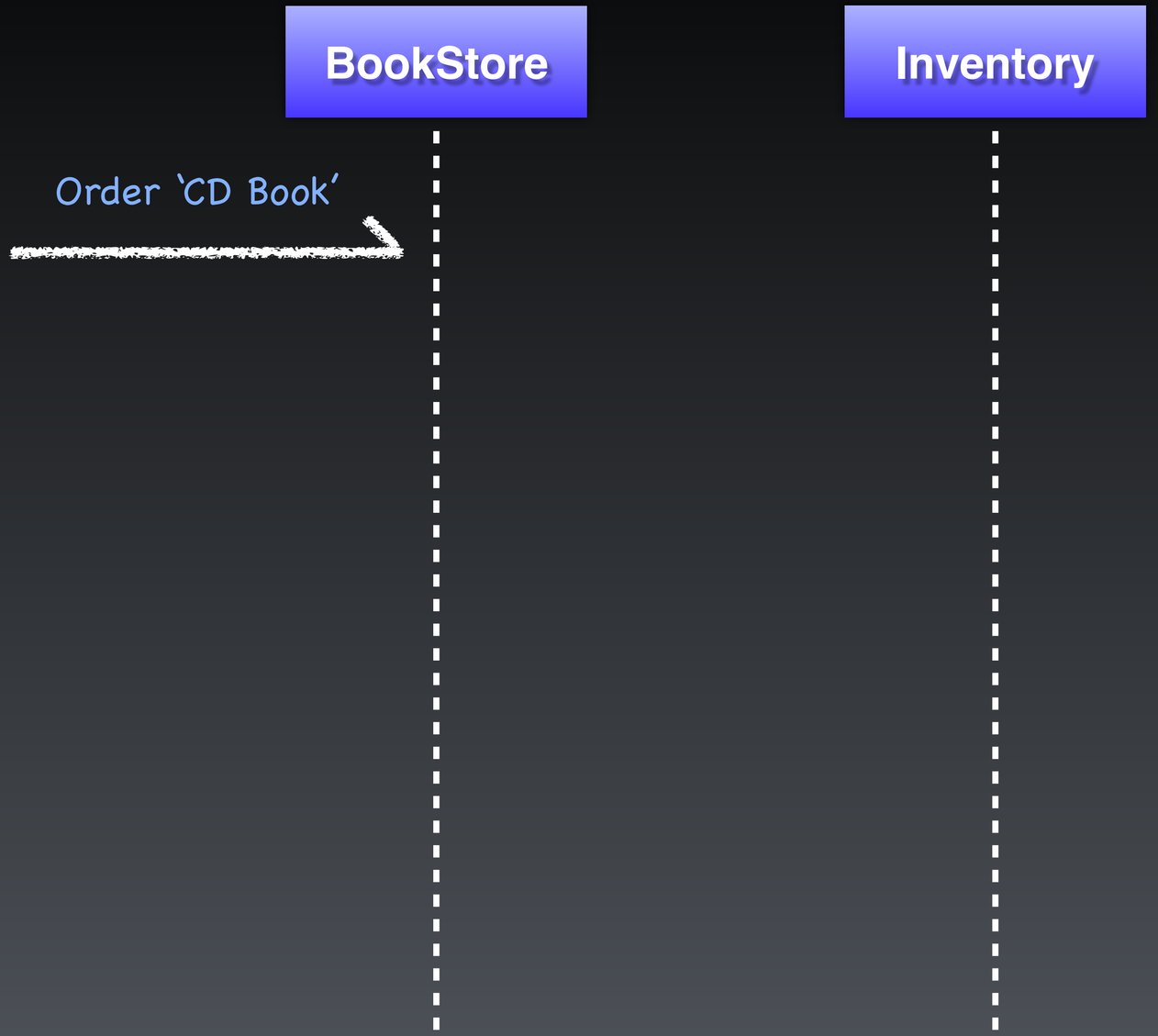
Services as State Machines

- Extremely useful pattern
- Very simple…
  - Use Domain-level message semantics
  - Migrate state of domain model based on message input
  - Generate events on state change
  - *Run business logic on a single thread*

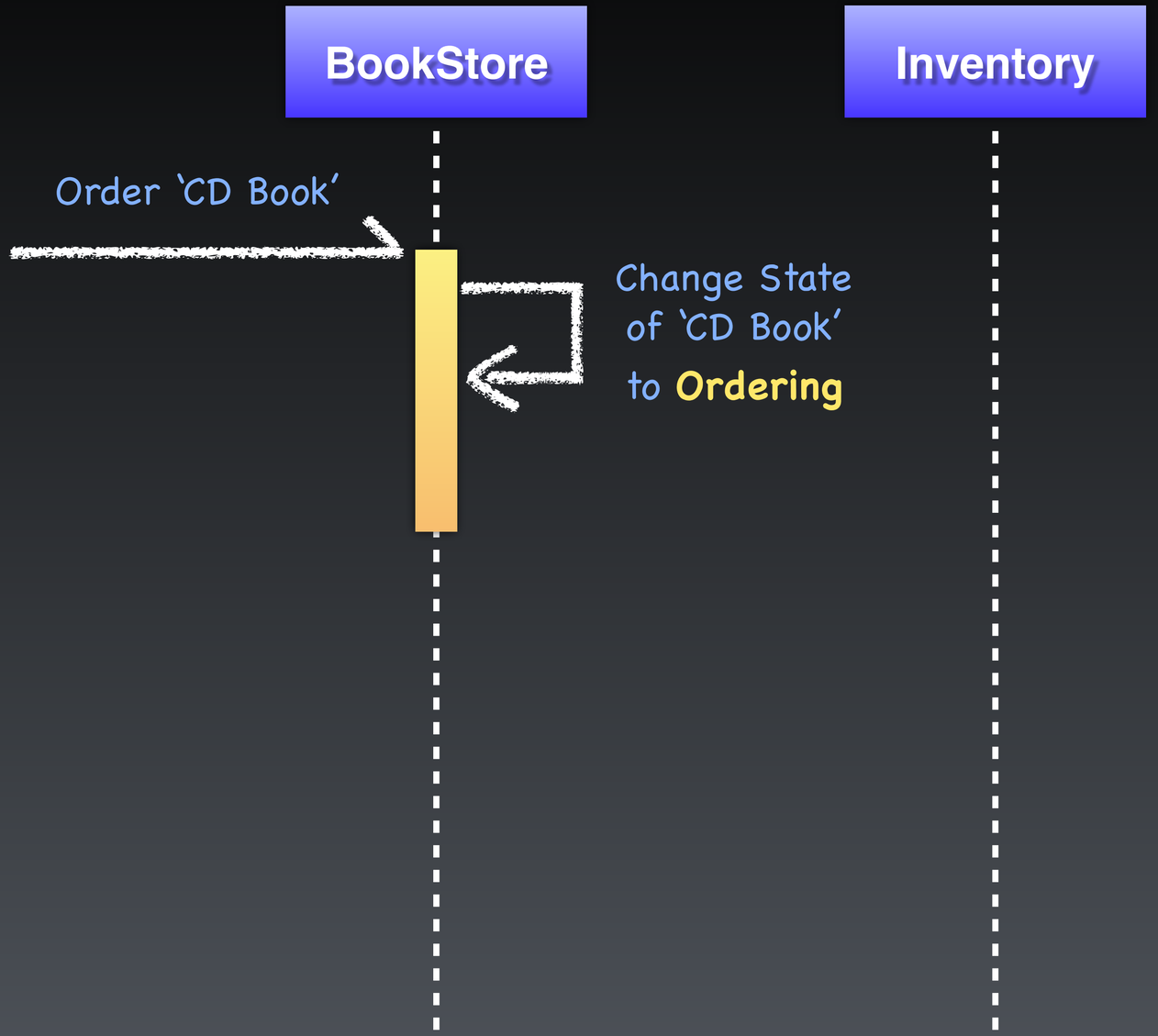
Services As State Machines



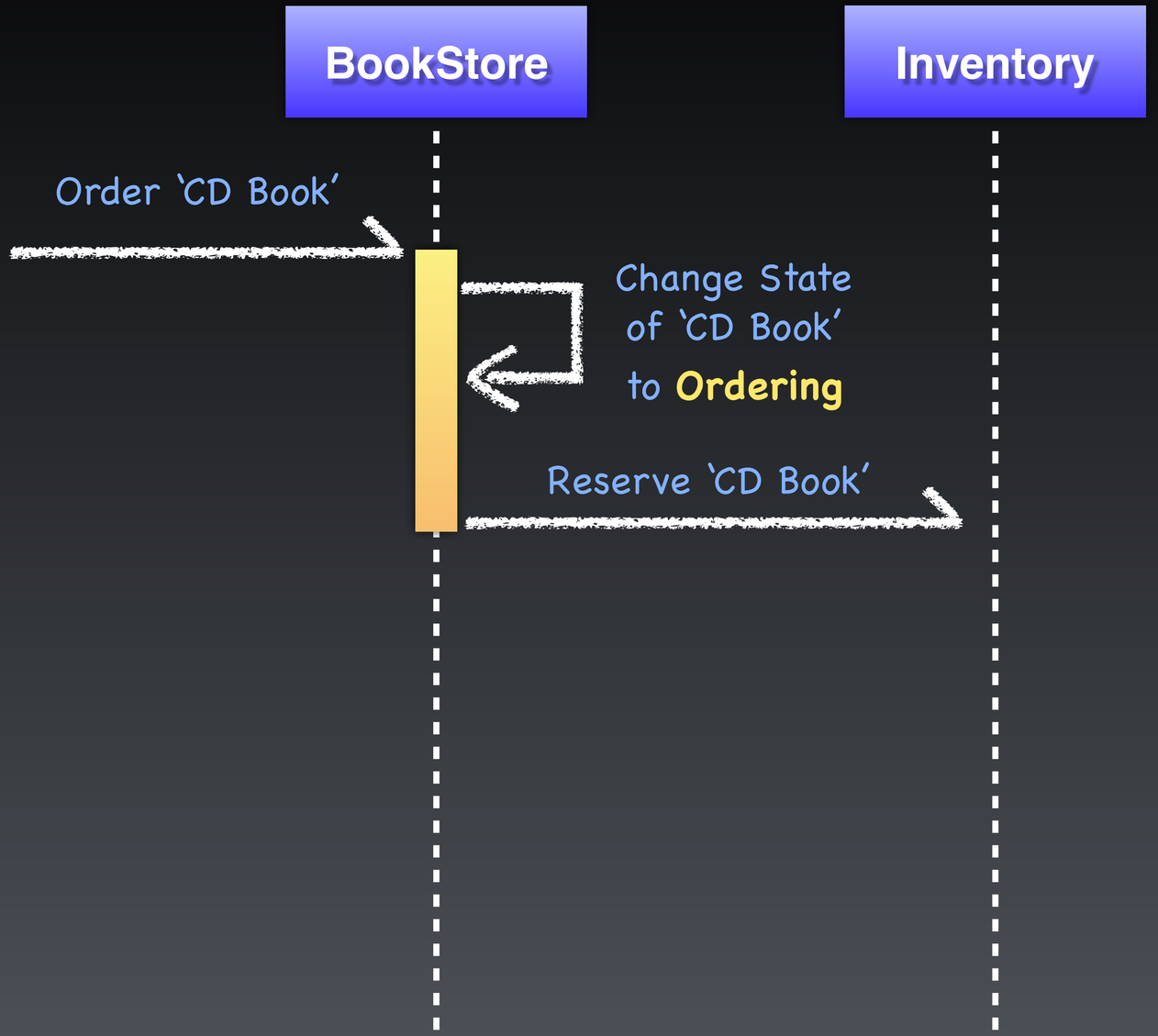
Services As State Machines



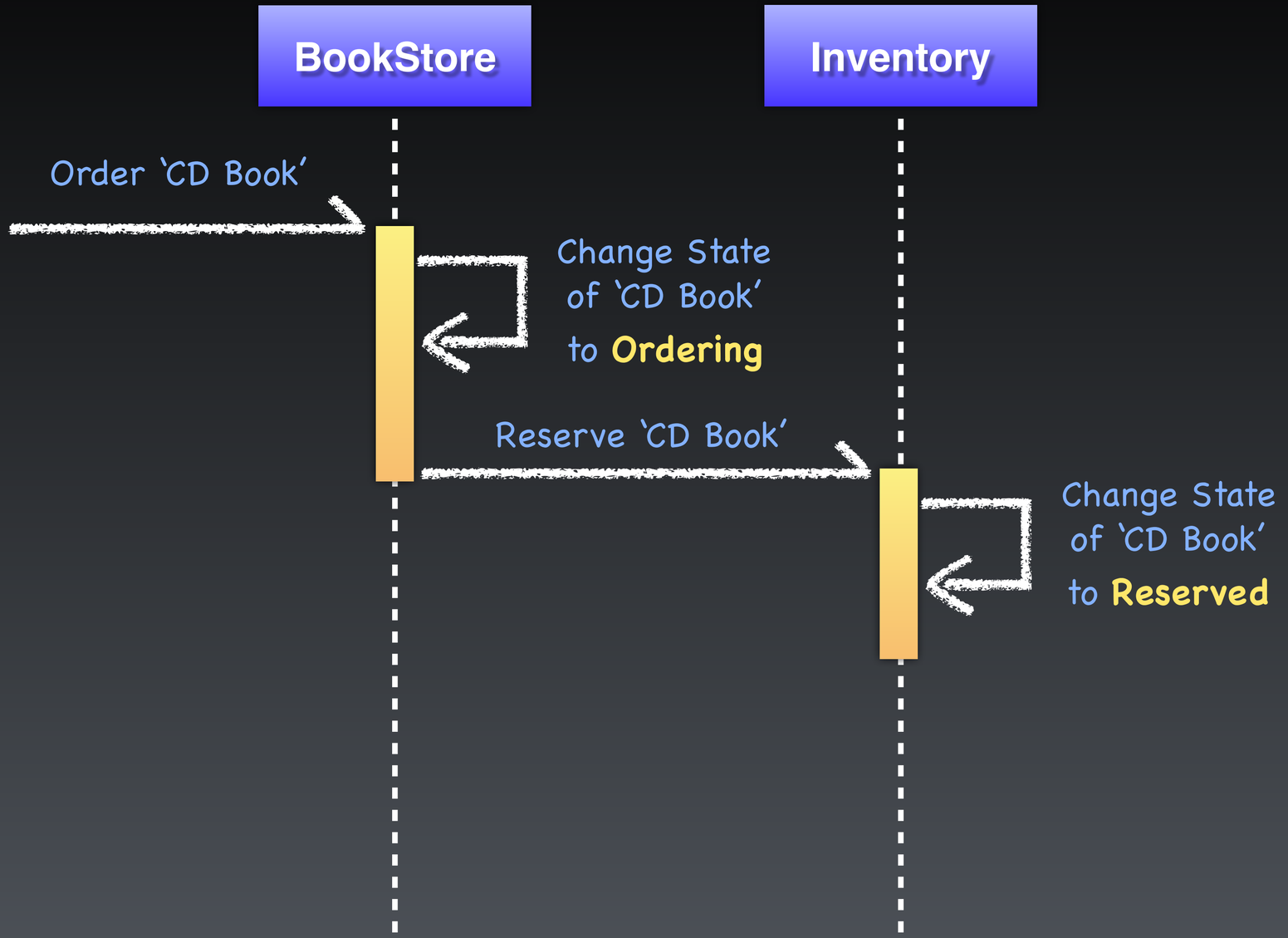
Services As State Machines



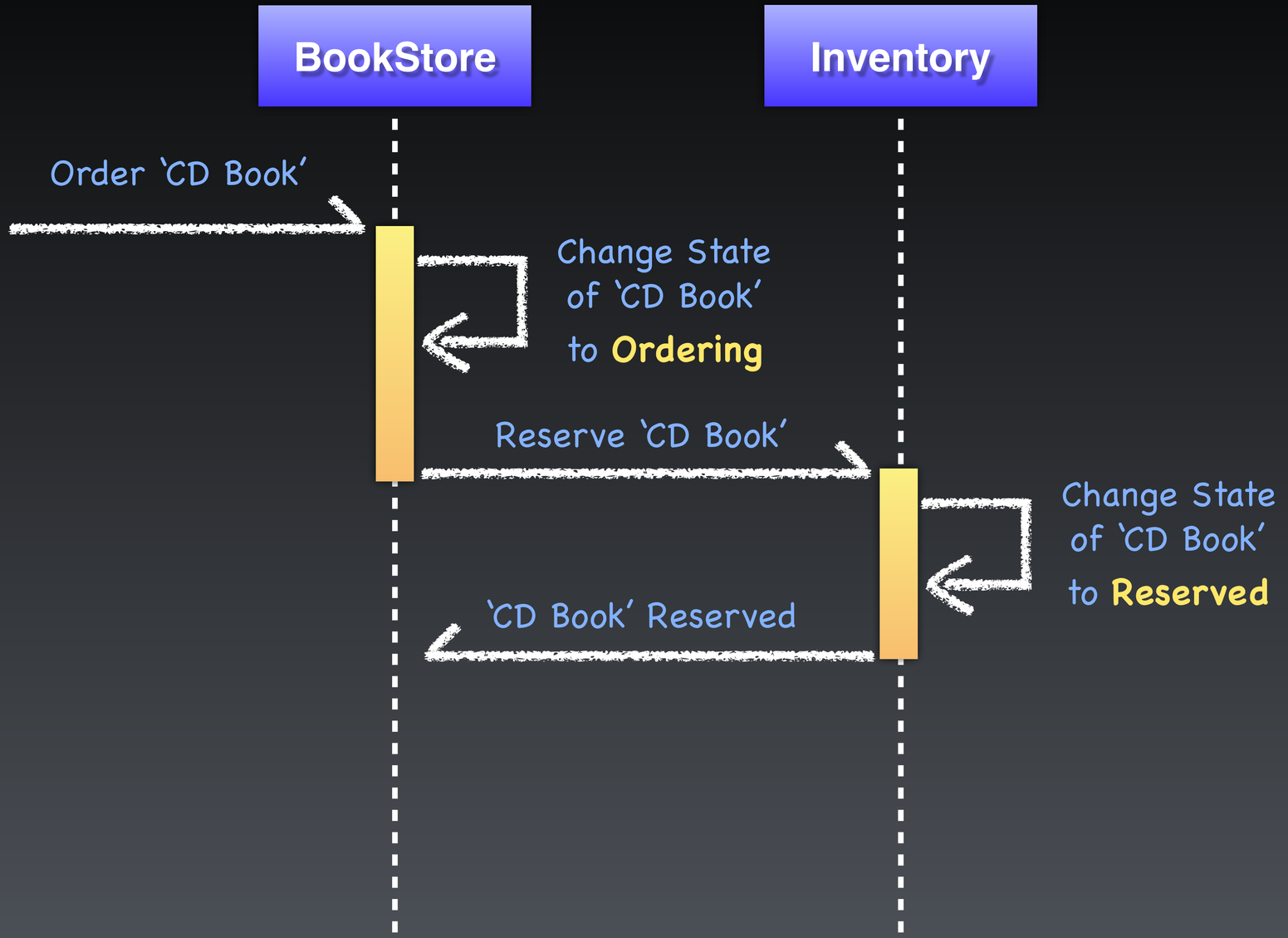
Services As State Machines



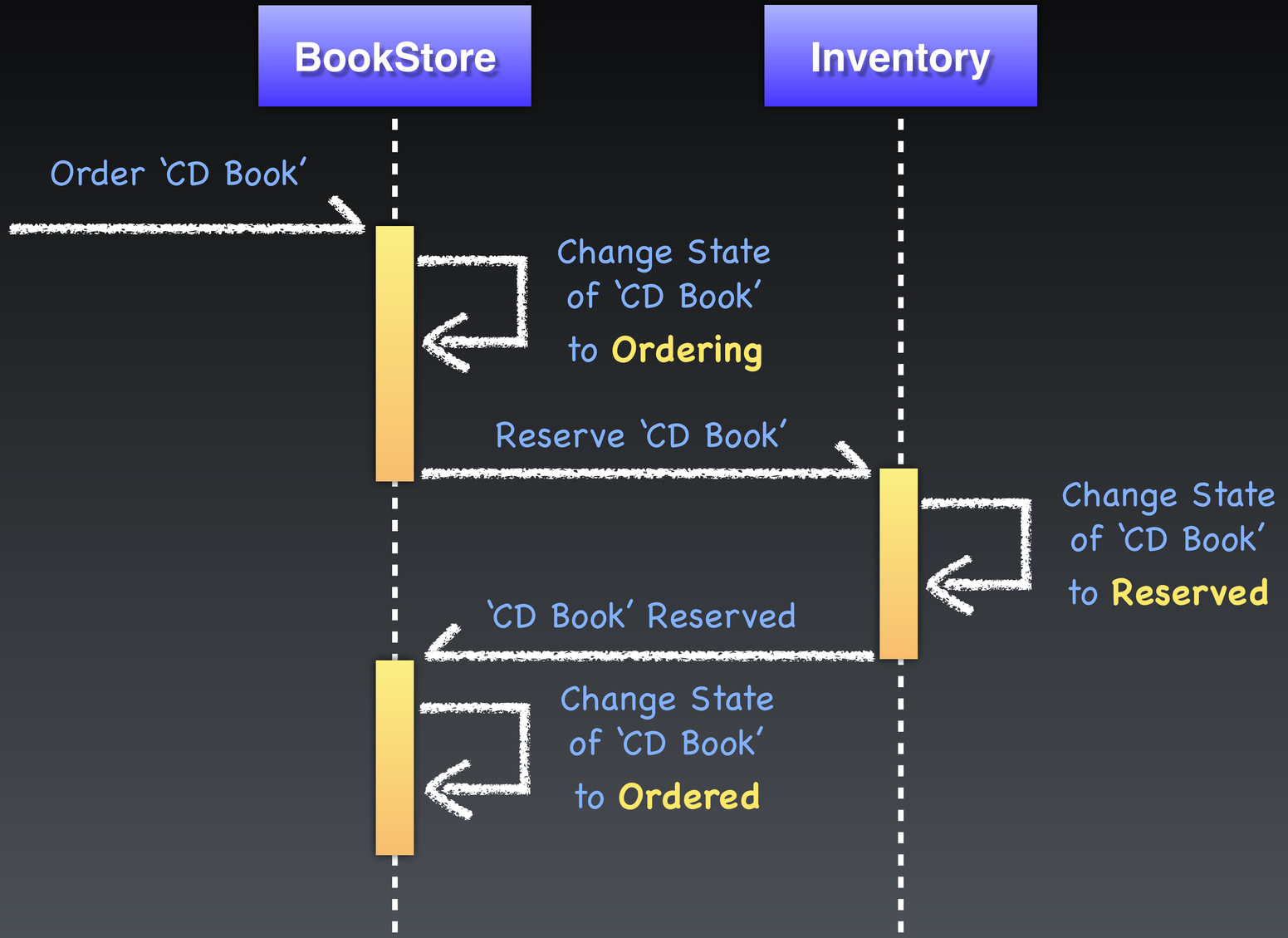
Services As State Machines



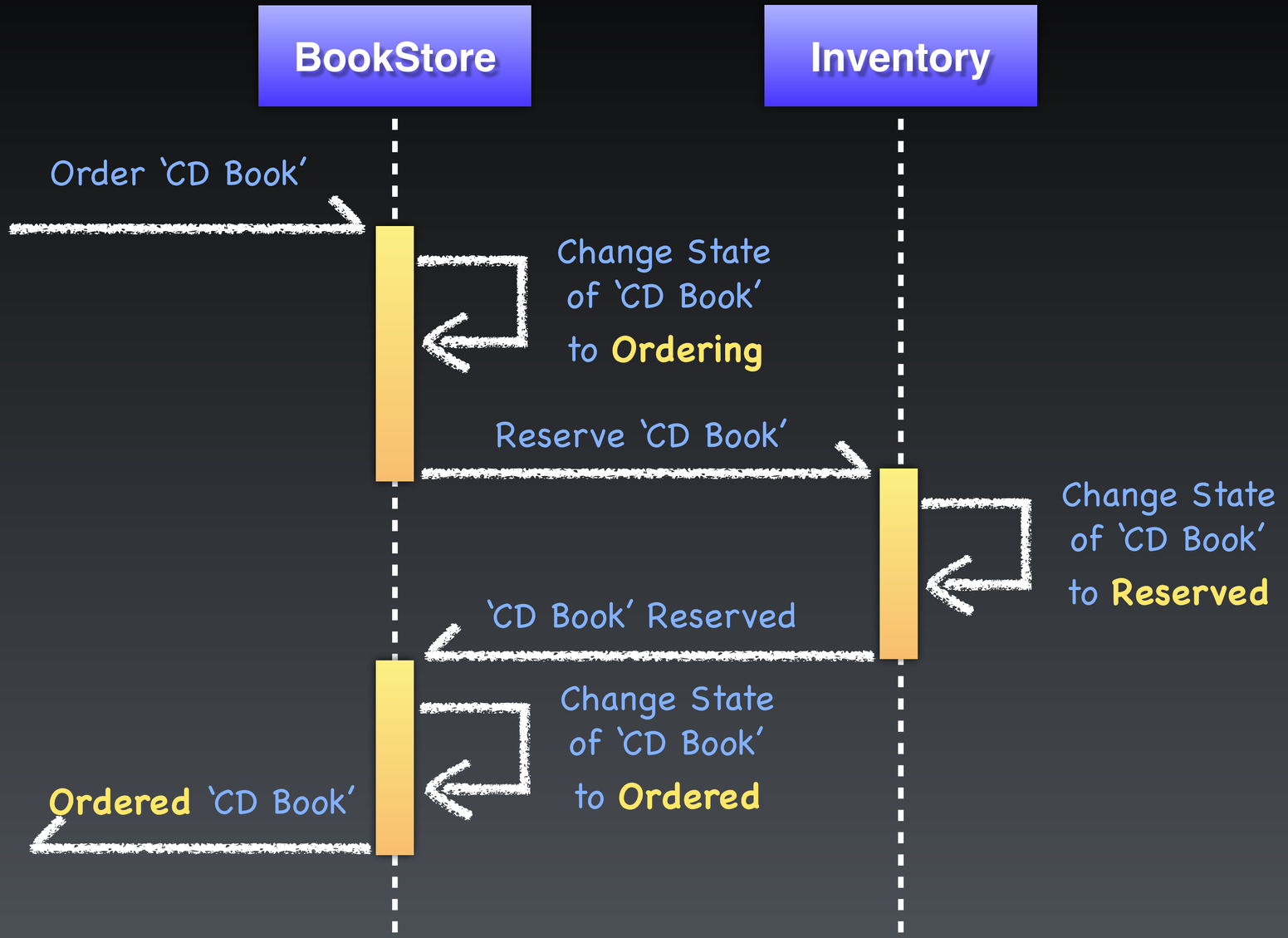
Services As State Machines



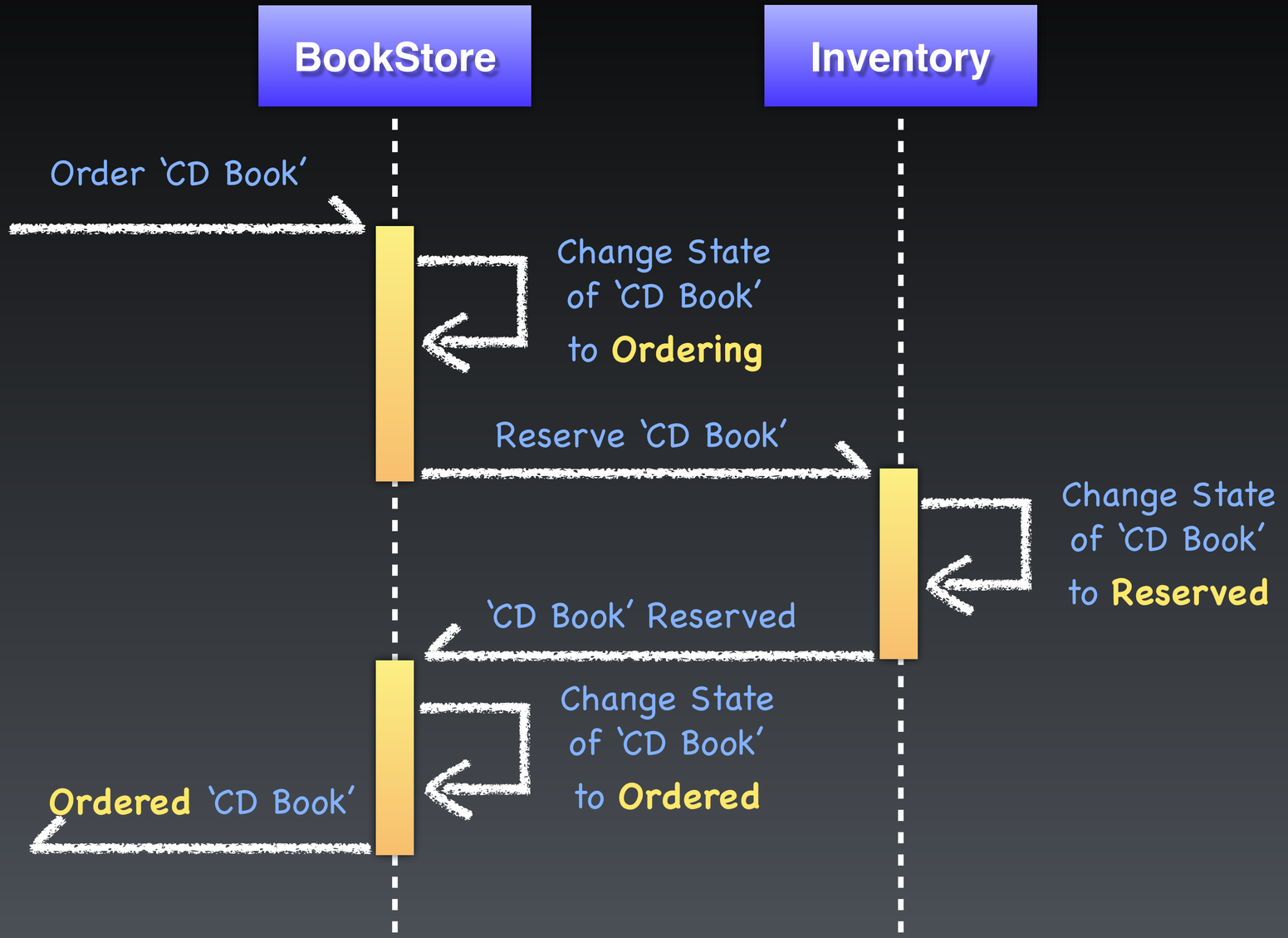
Services As State Machines



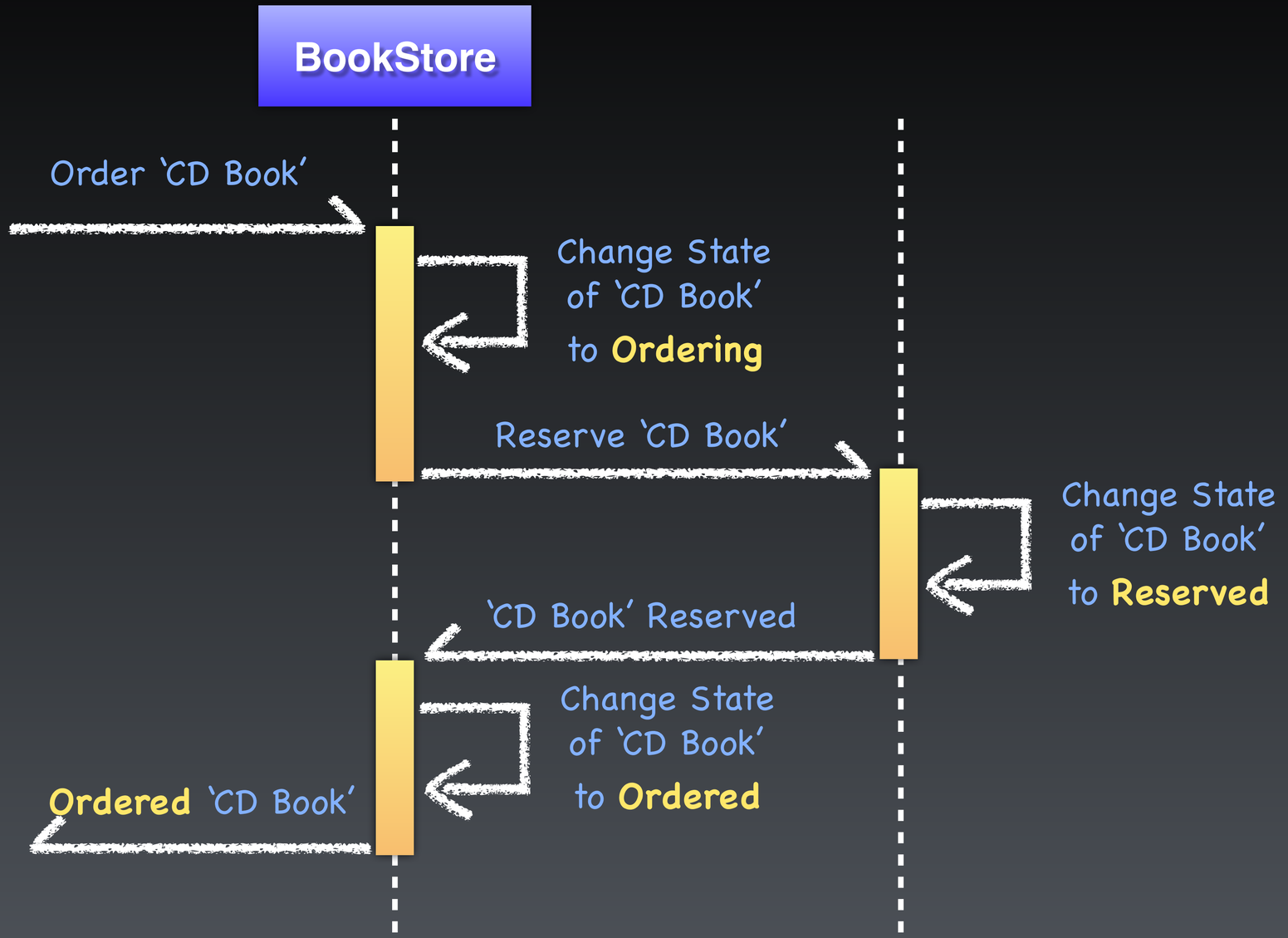
Services As State Machines



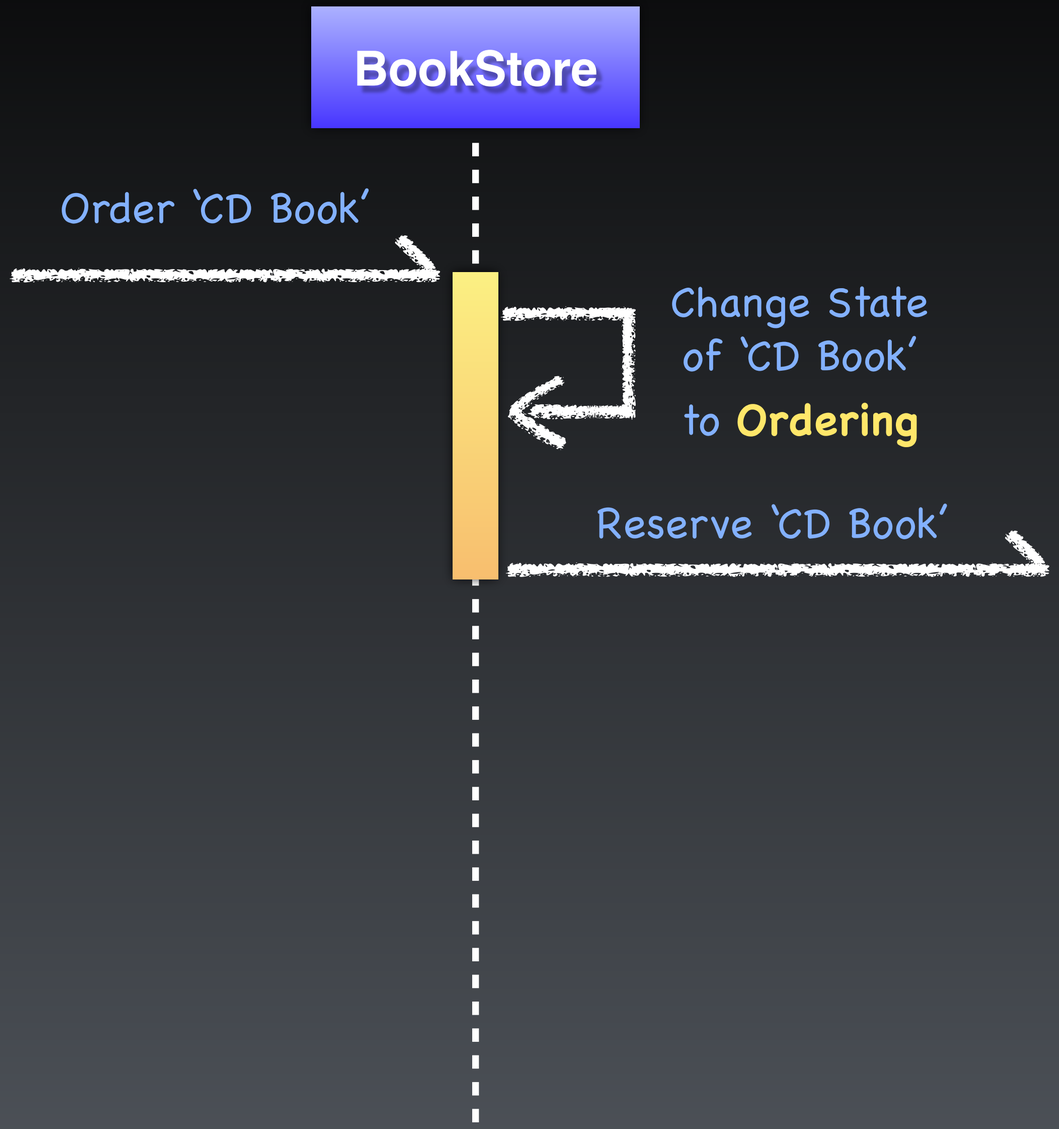
Services As State Machines



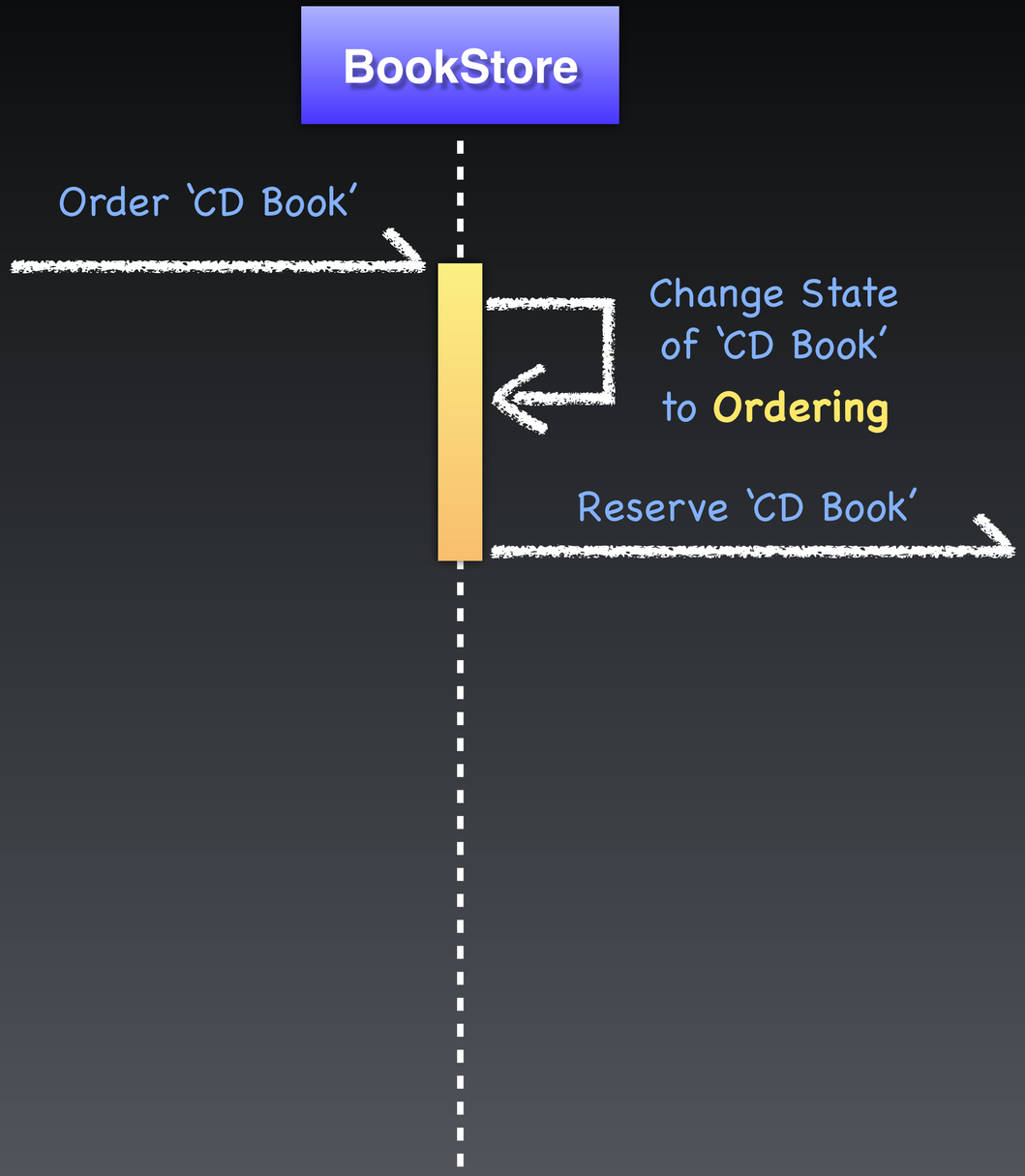
Services As State Machines



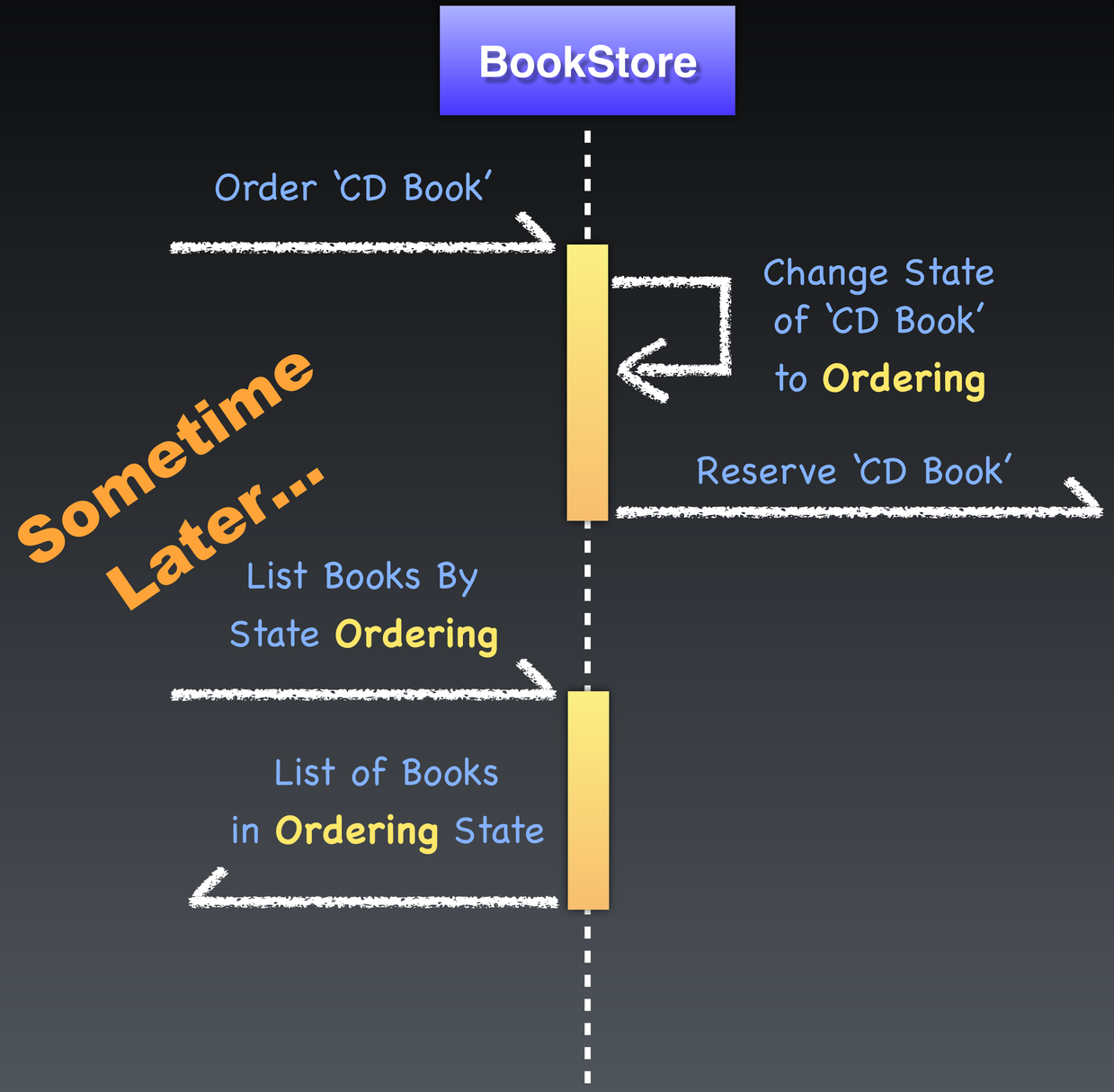
Services As State Machines



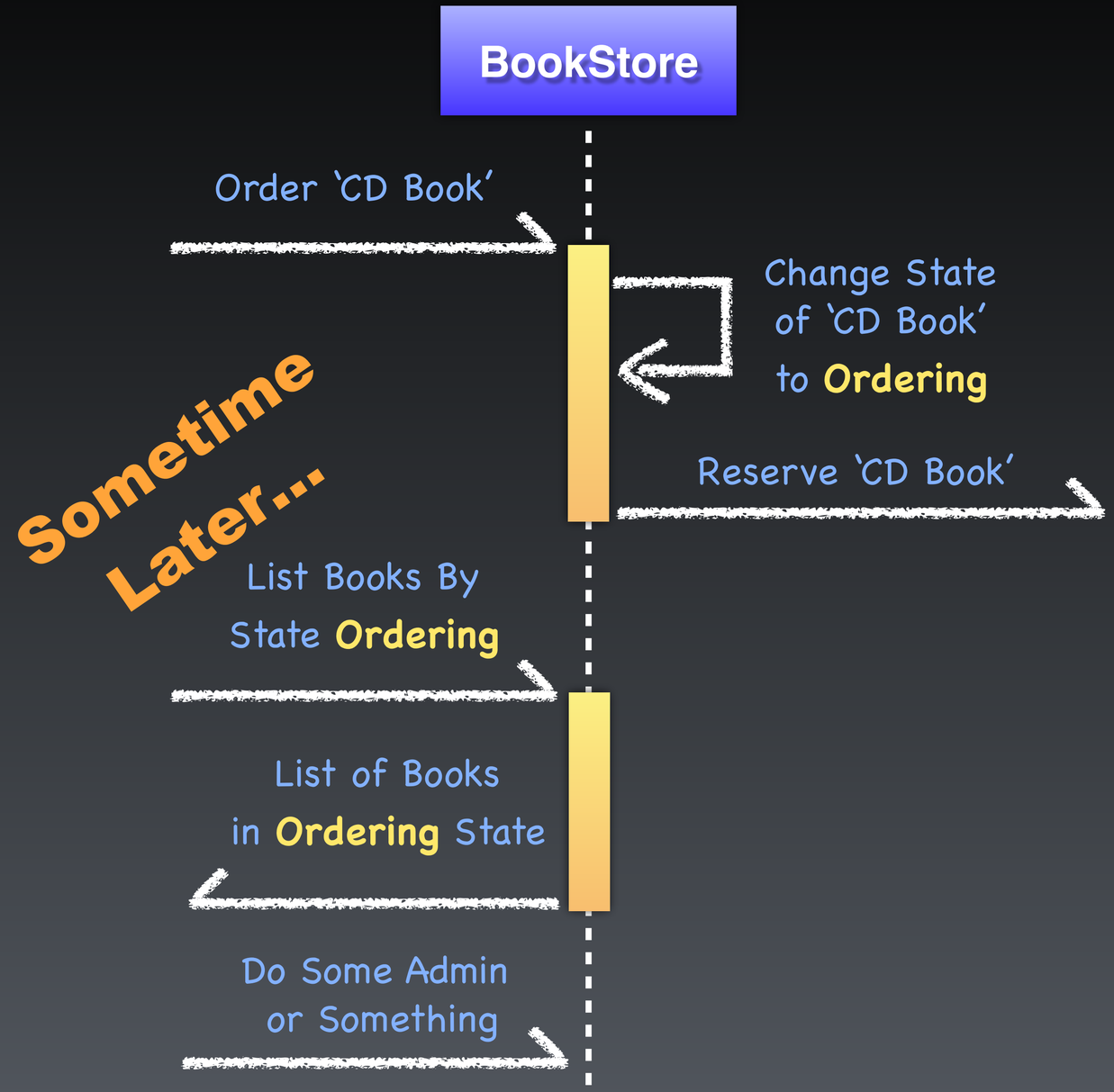
Services As State Machines



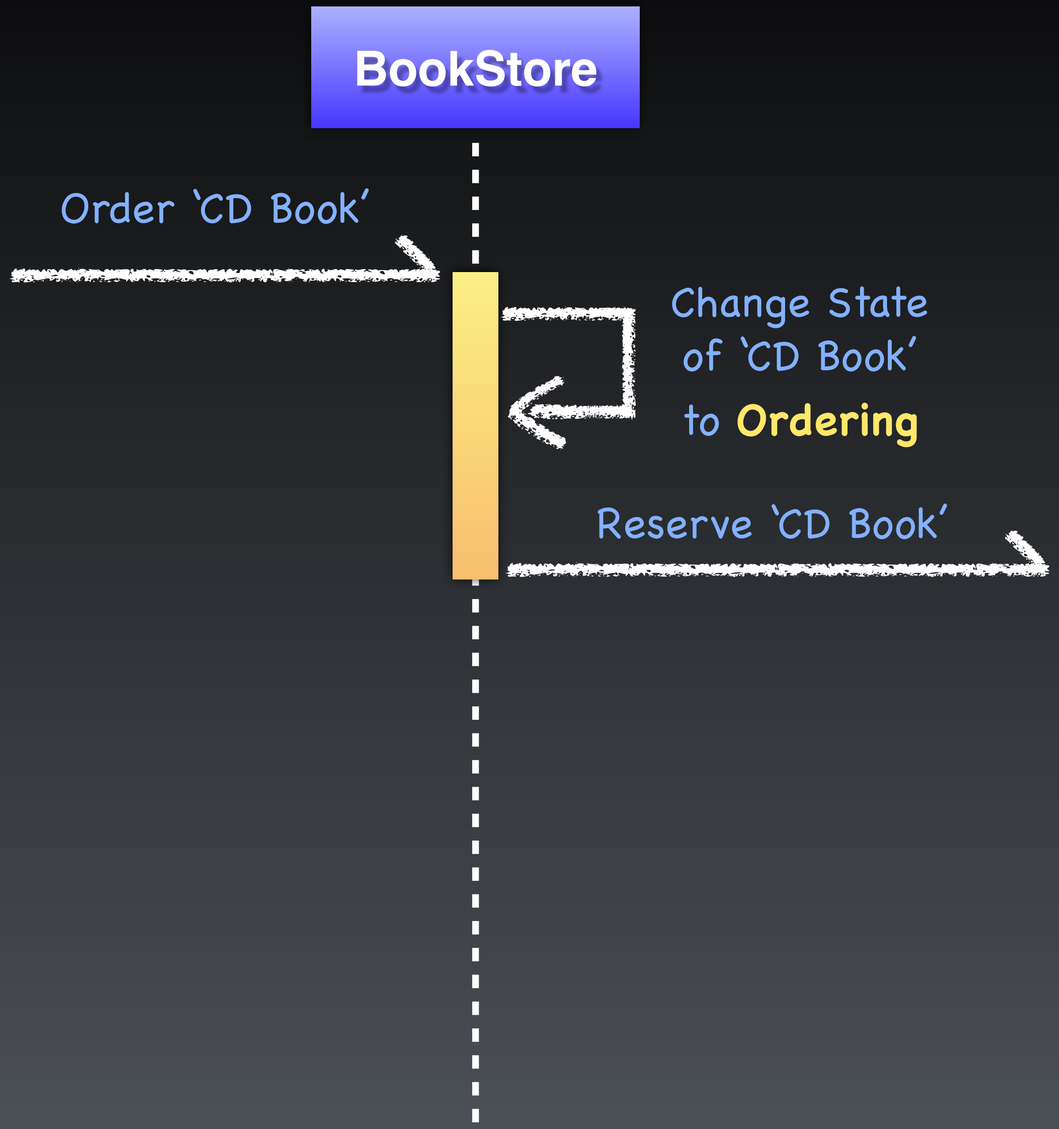
Services As State Machines



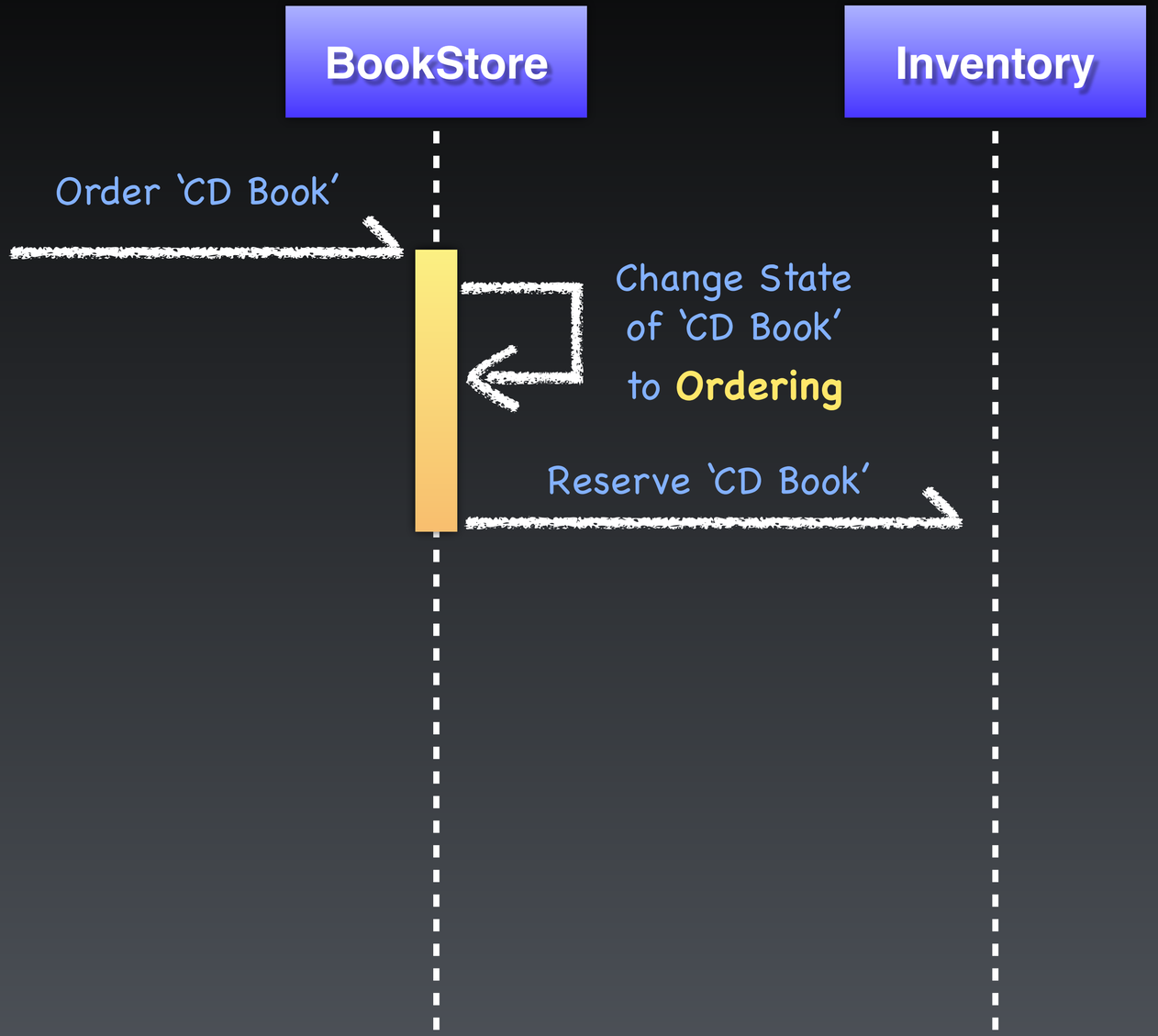
Services As State Machines



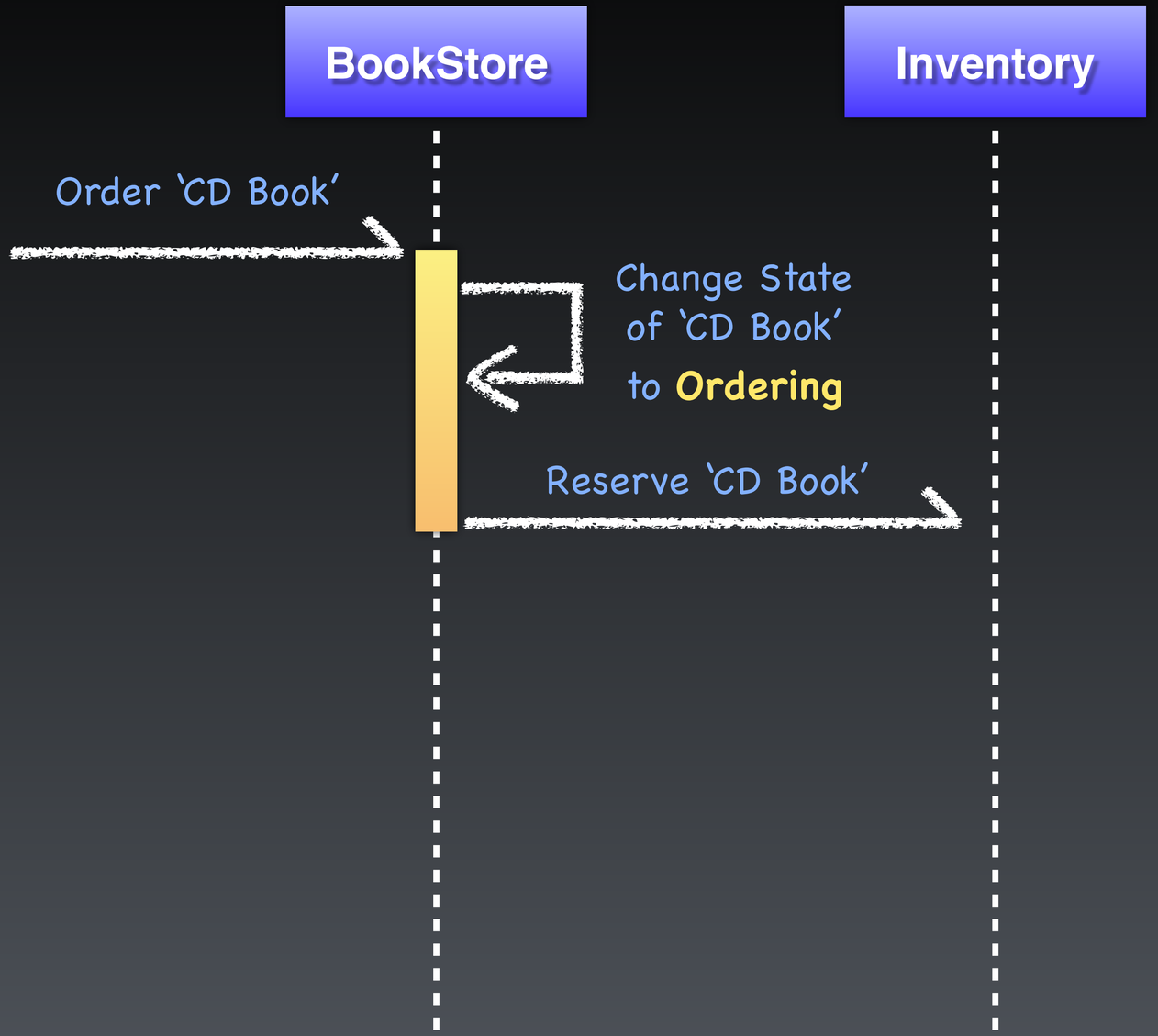
Services As State Machines



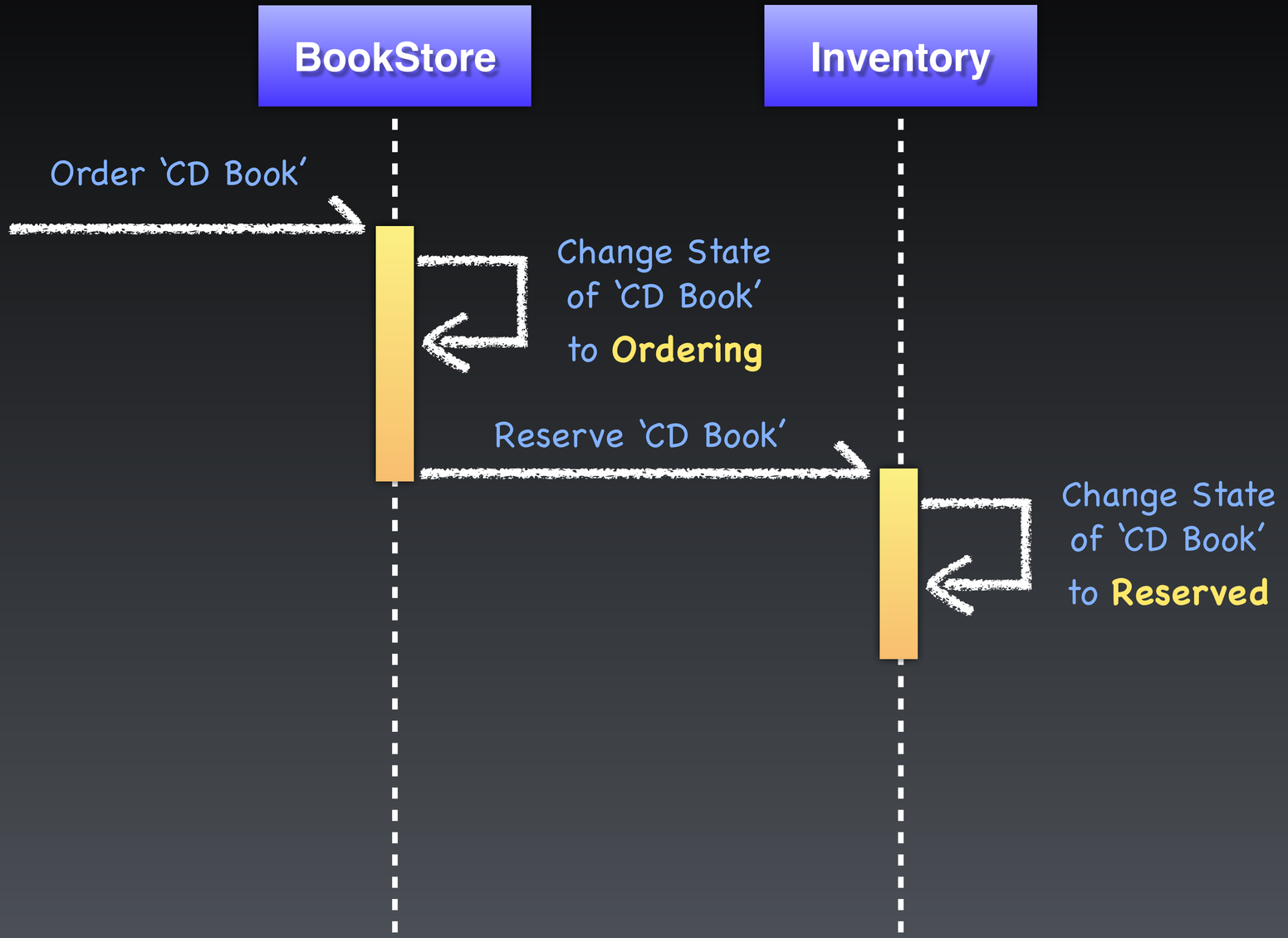
Services As State Machines



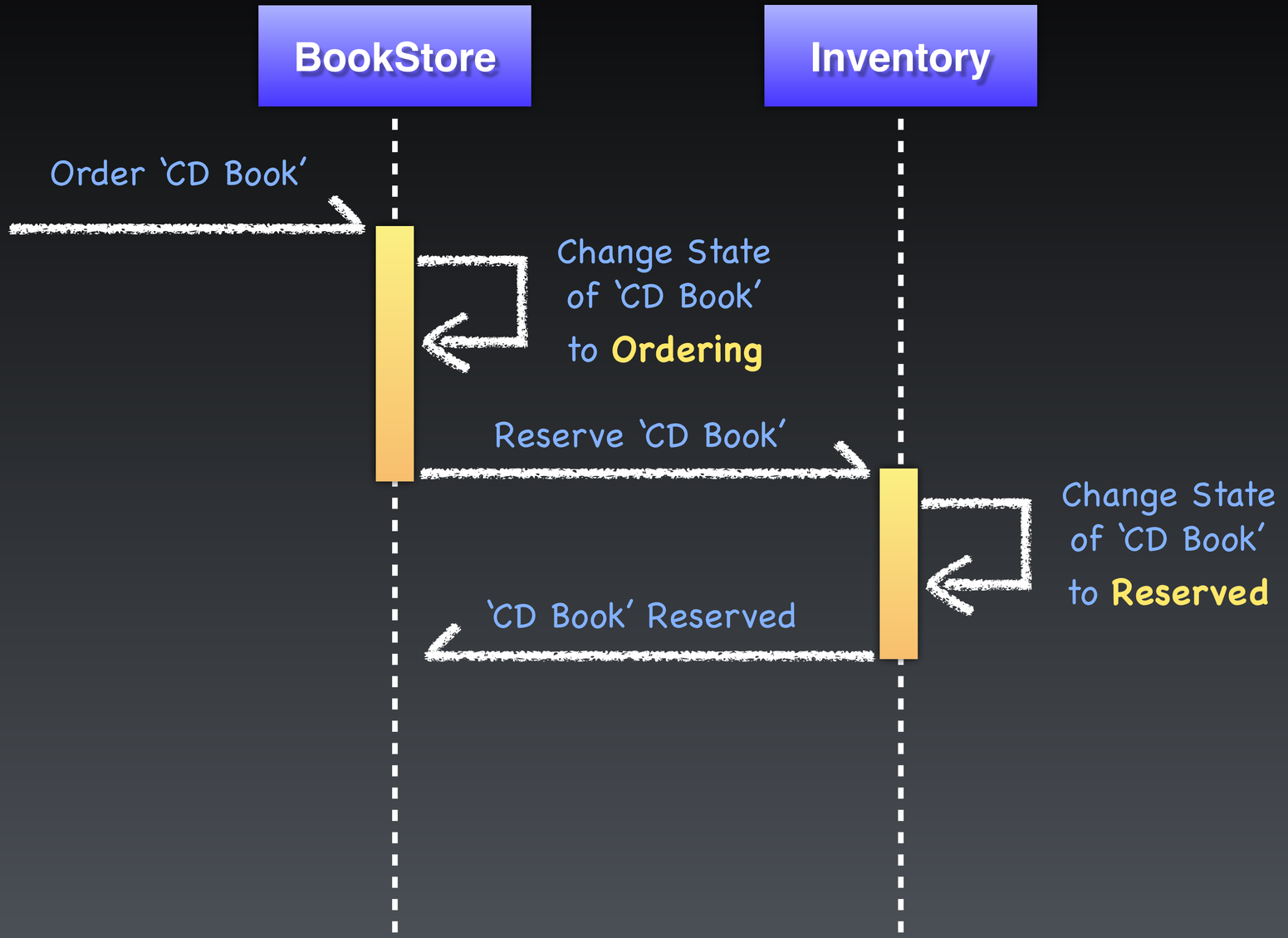
Services As State Machines



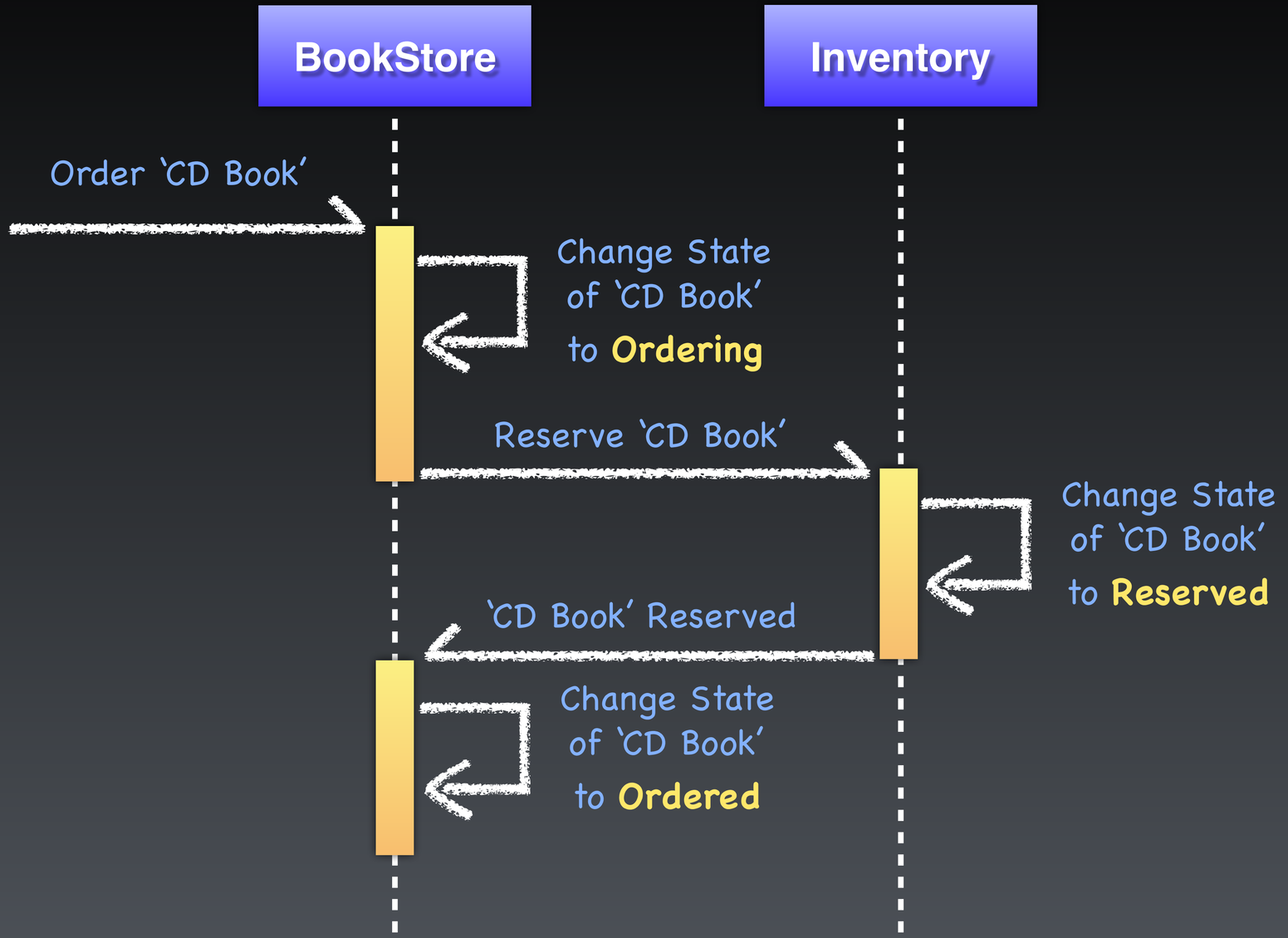
Services As State Machines



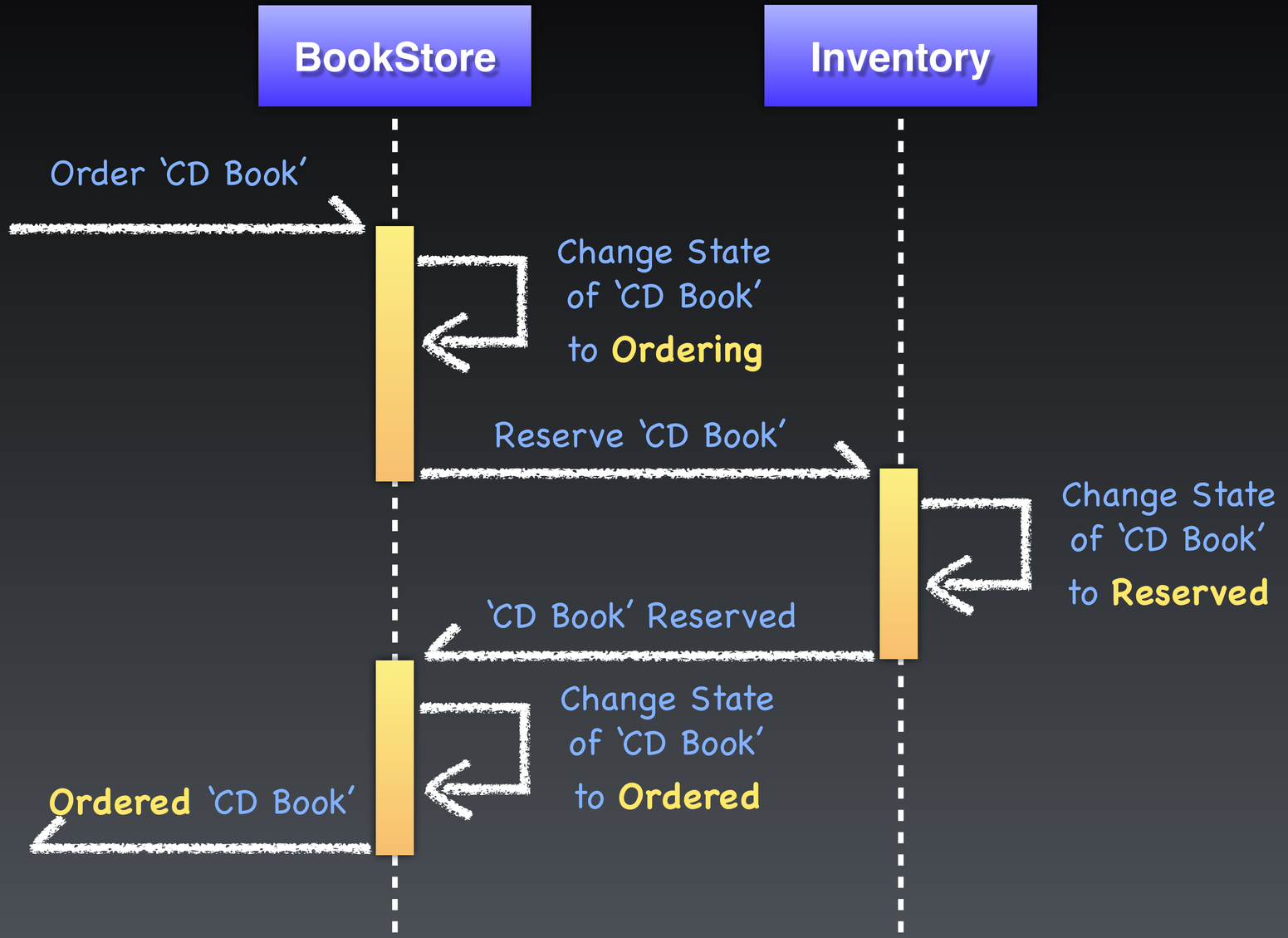
Services As State Machines



Services As State Machines



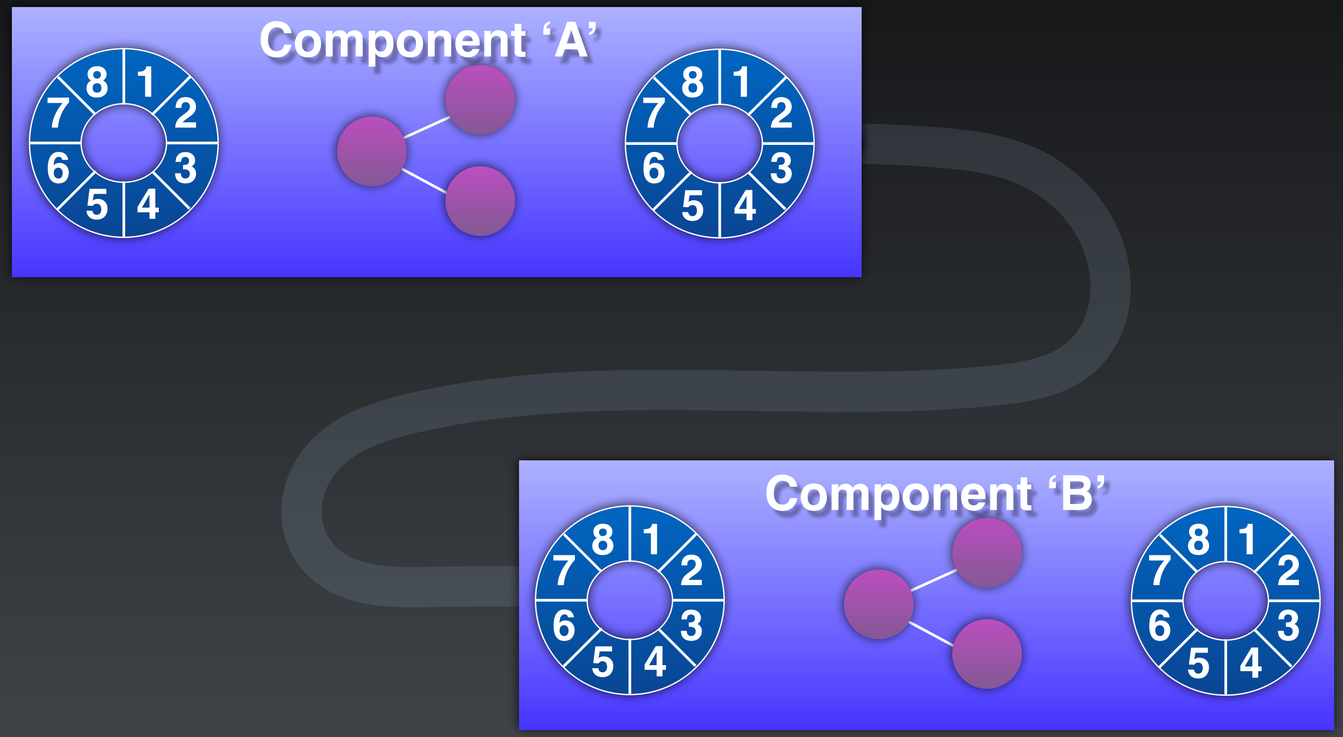
Services As State Machines



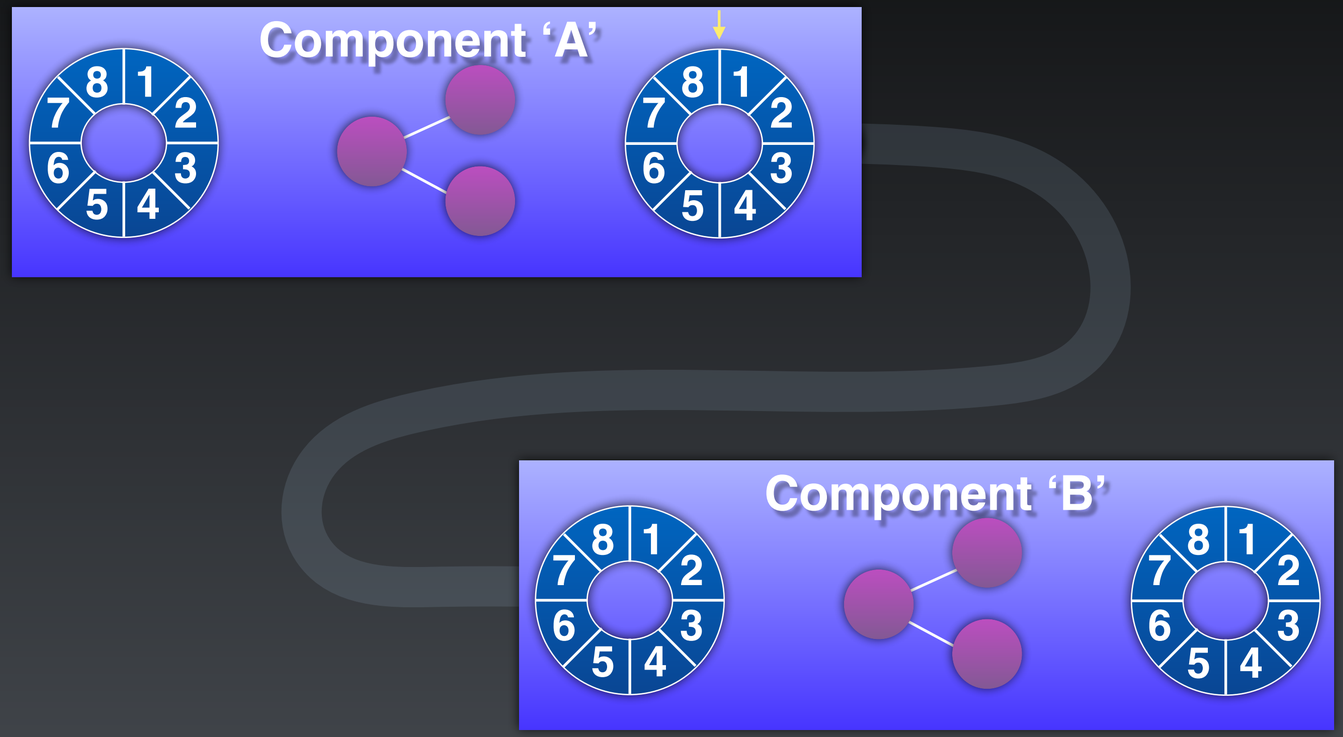
Importance of Idempotence

- This Model works really well!
- But, we need be confident in our messaging…
  - Order
  - Deterministic State
  - Durability

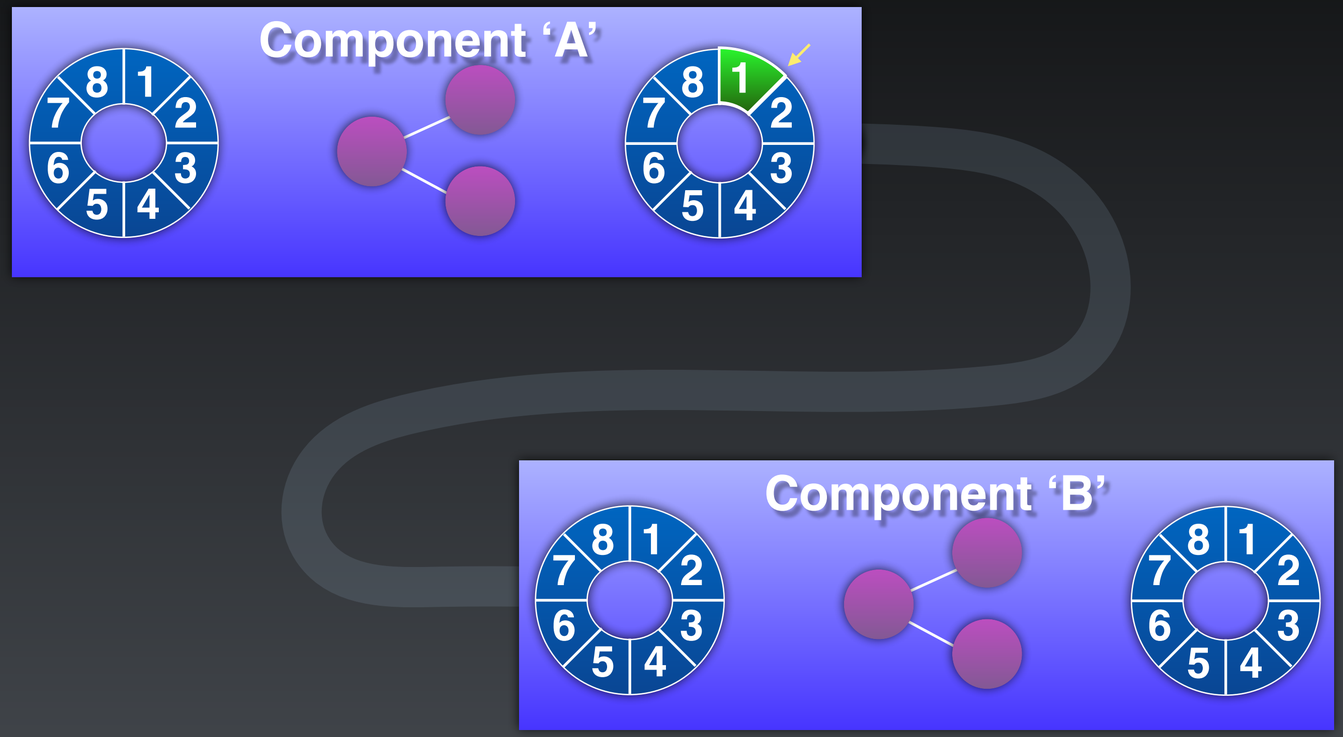
An Example of Idempotence



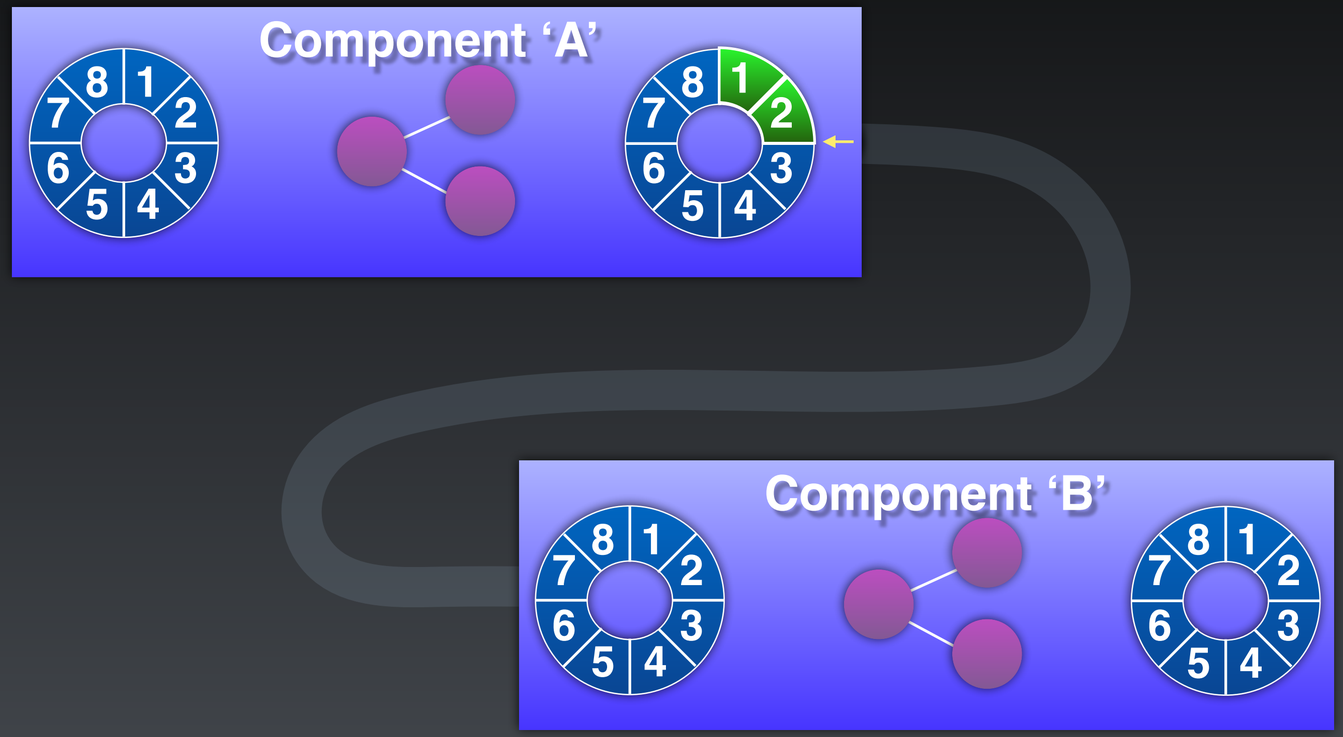
An Example of Idempotence



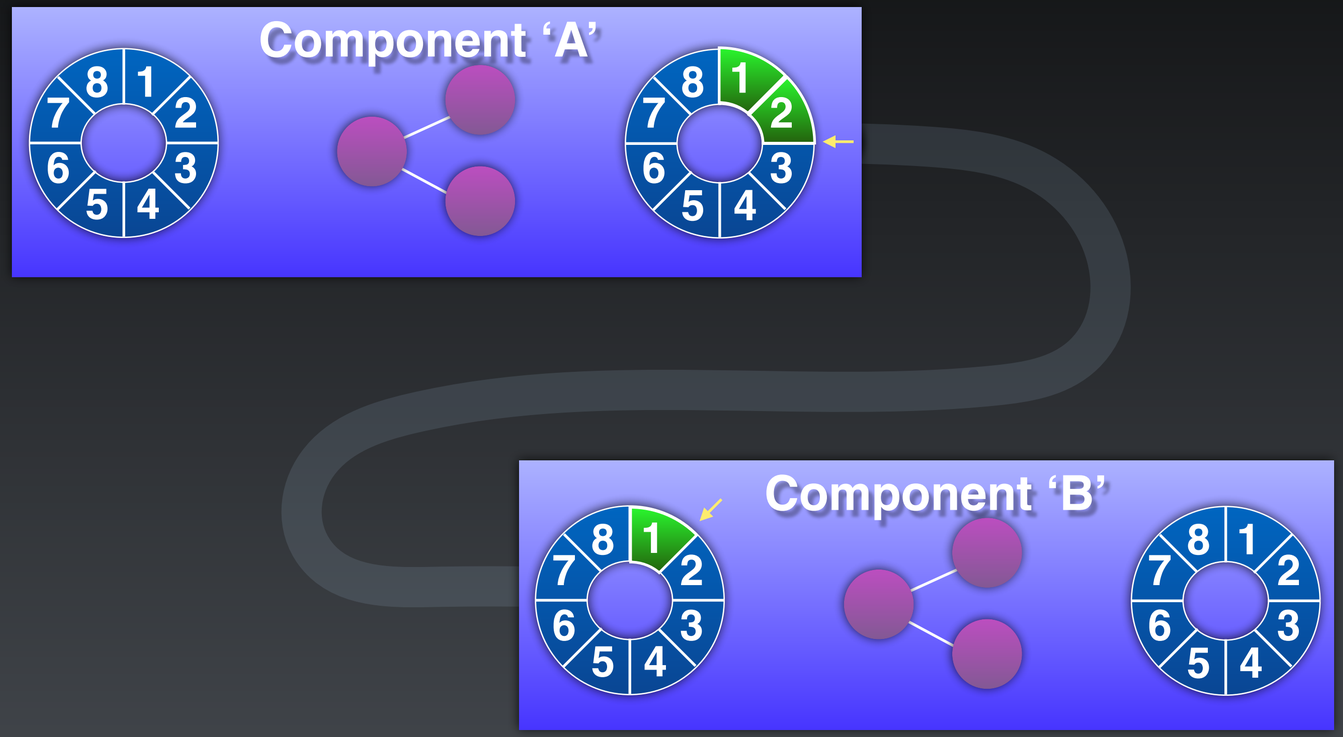
An Example of Idempotence



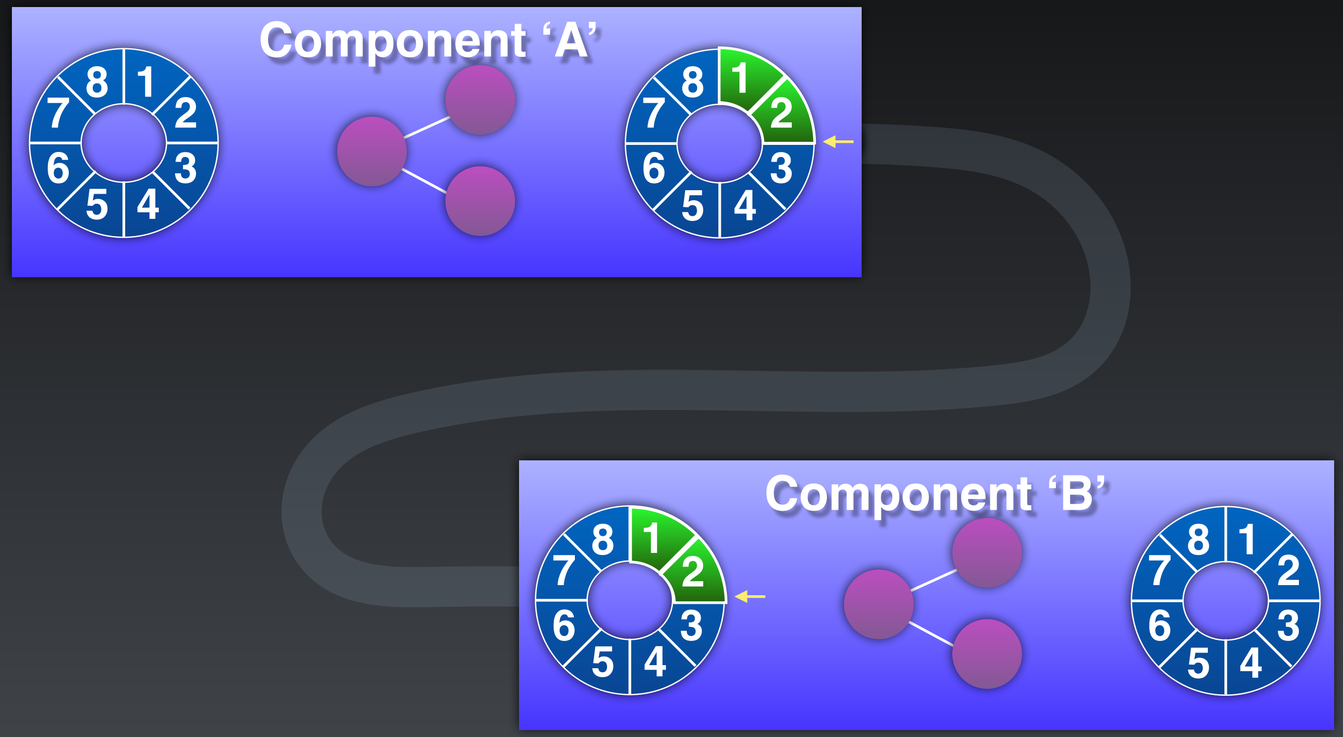
An Example of Idempotence



An Example of Idempotence

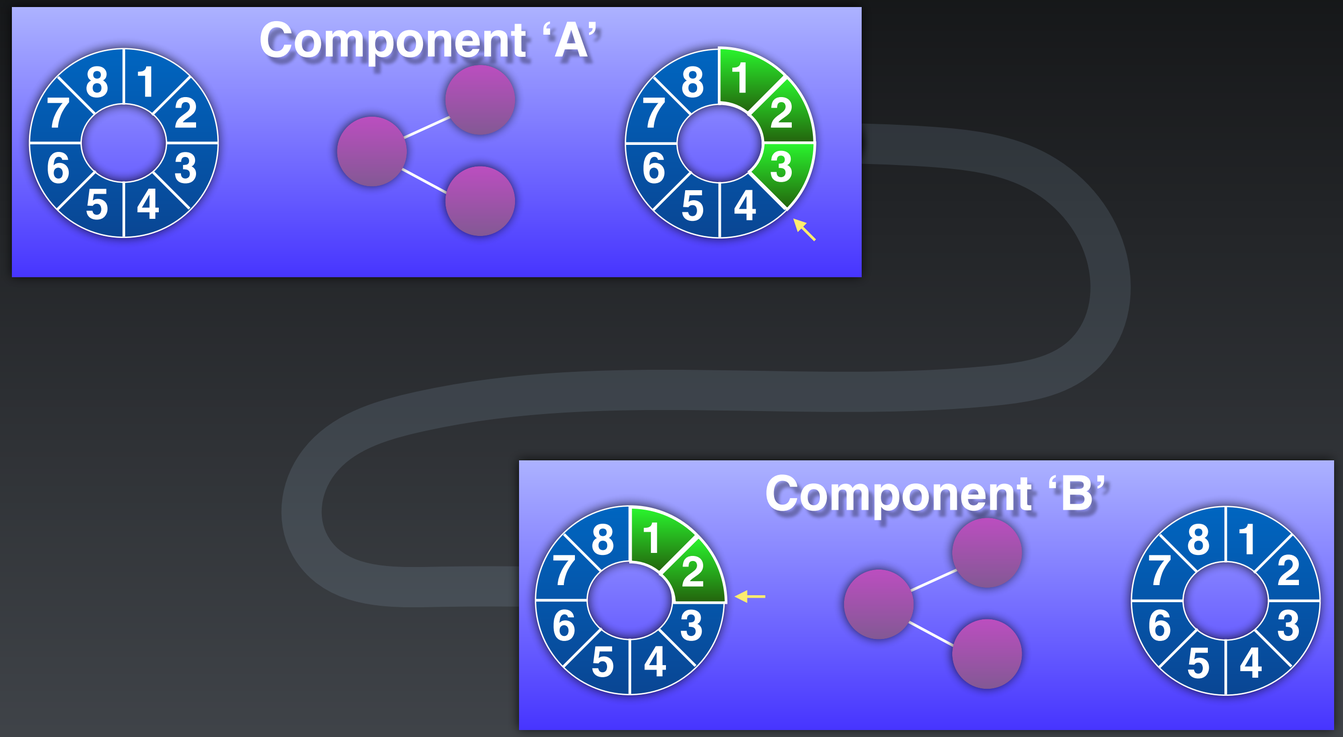


An Example of Idempotence

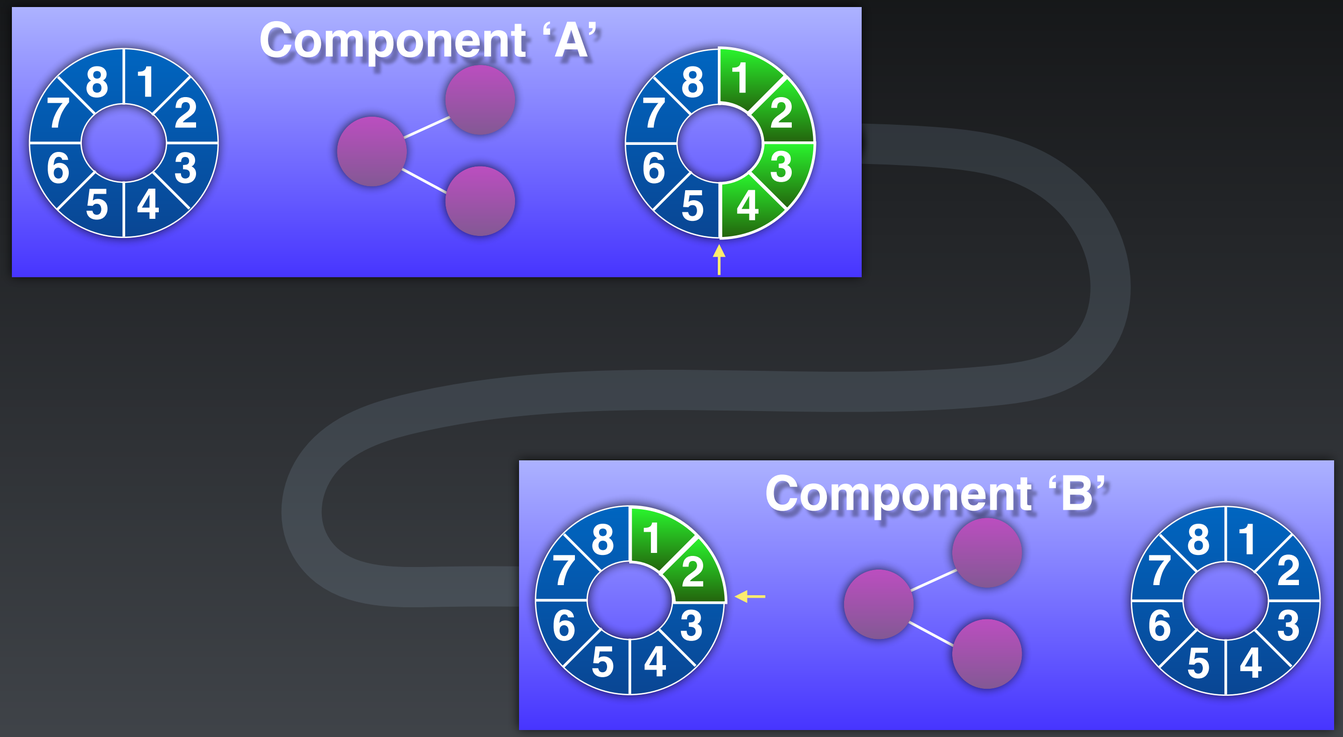


Continuous Delivery ltd

An Example of Idempotence

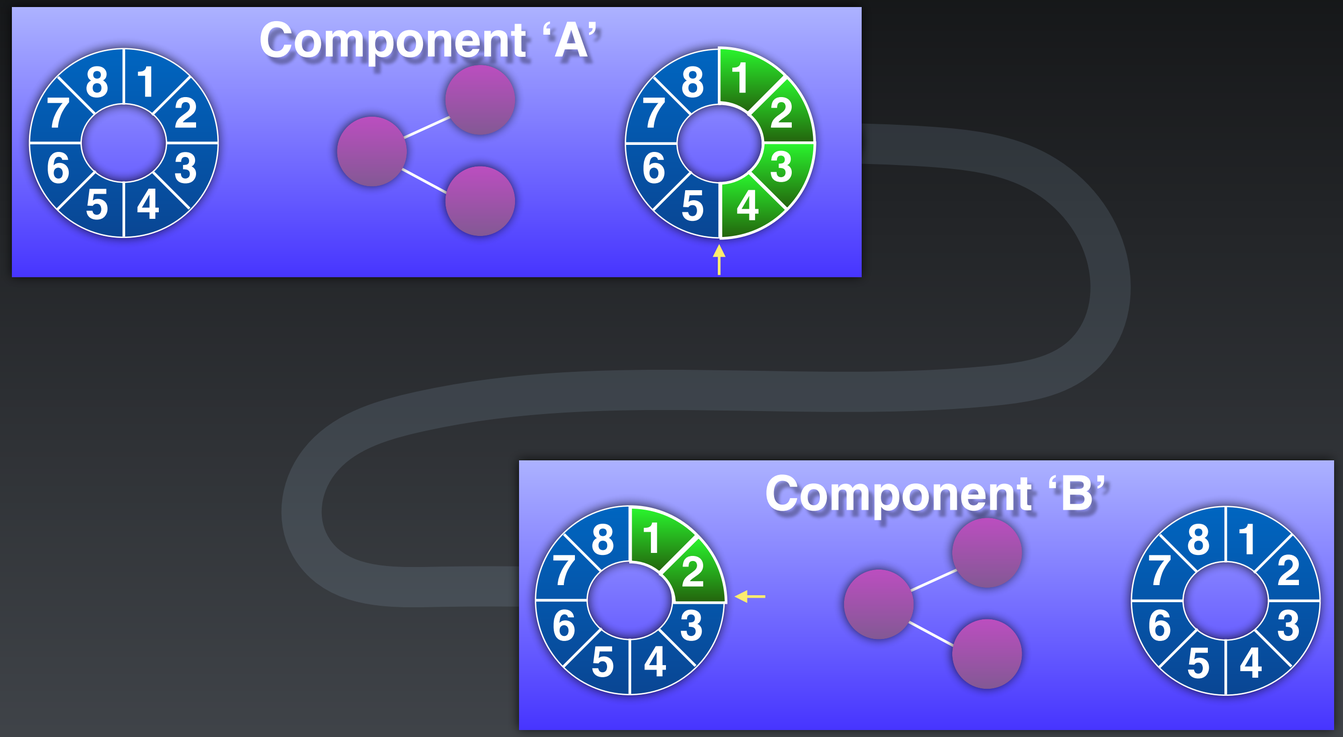


An Example of Idempotence



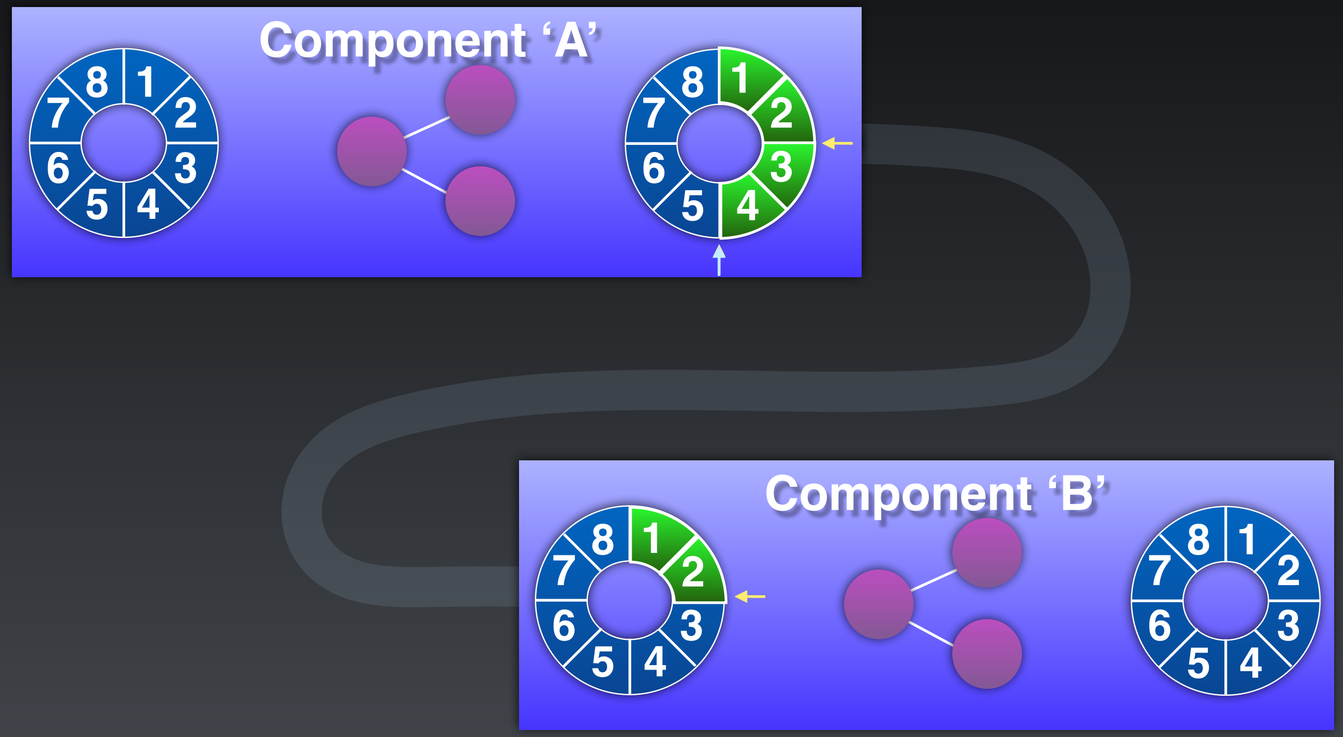
Continuous Delivery ltd

An Example of Idempotence



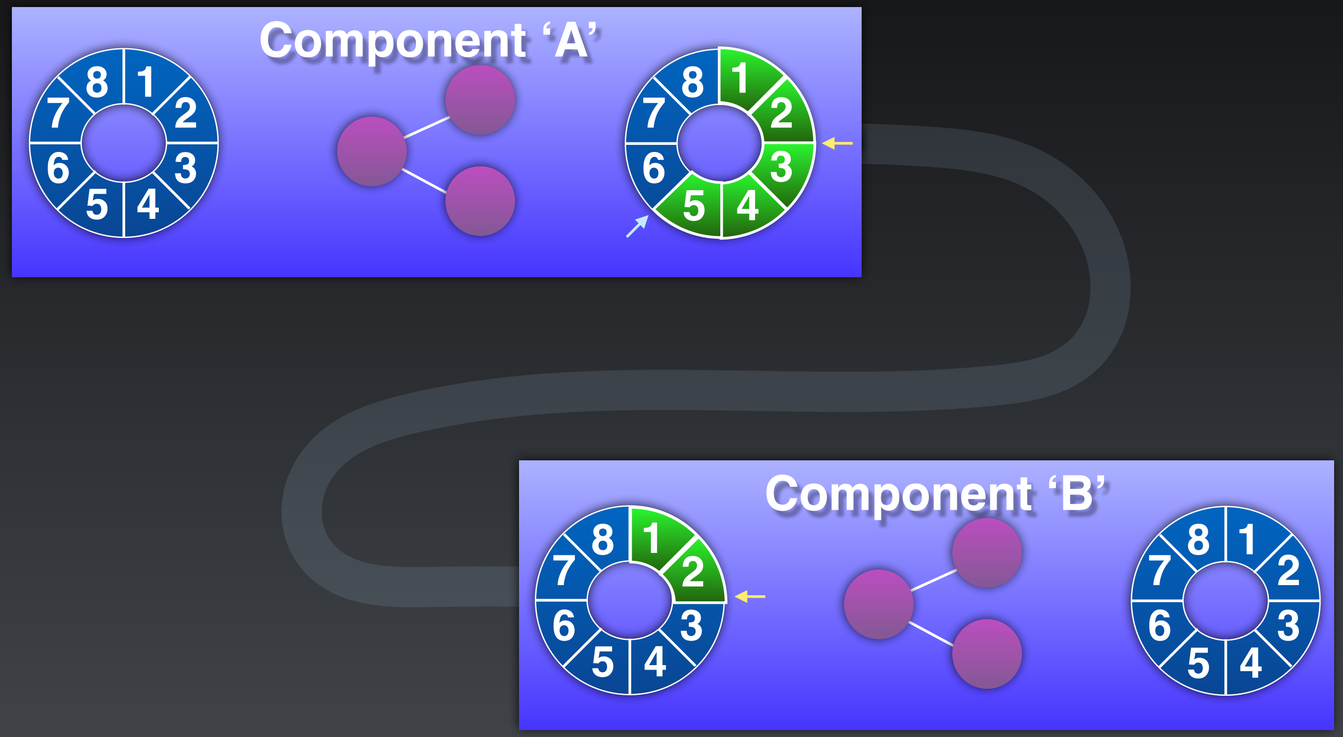
Continuous Delivery ltd

An Example of Idempotence

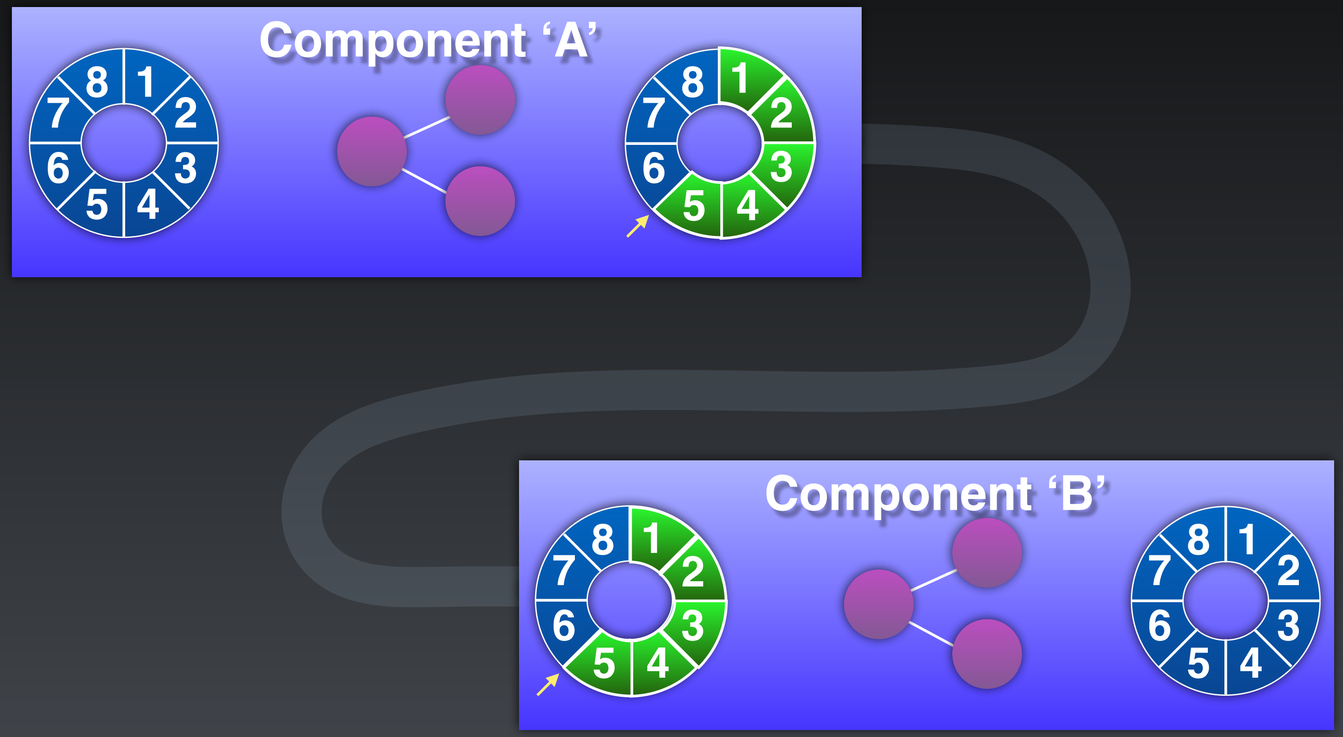


Continuous Delivery ltd

An Example of Idempotence



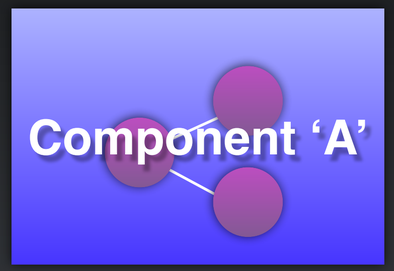
An Example of Idempotence

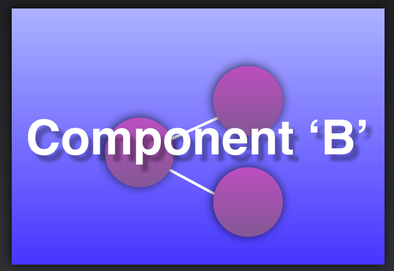


Isolation

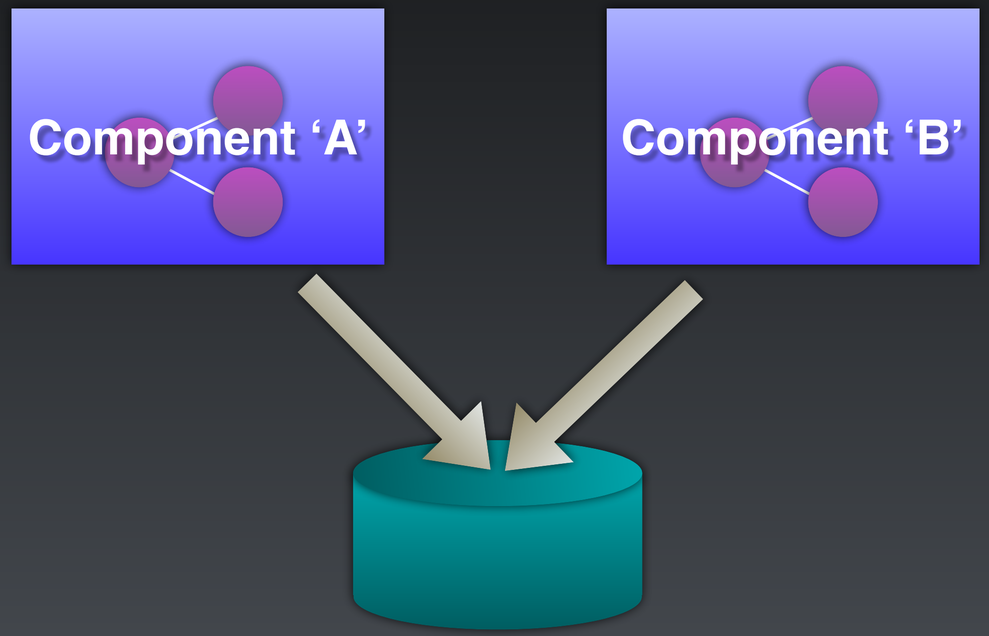
- Decoupling in Time and Space
  - Time - Sender and Receiver have independent lifecycle
  - Space - Location Transparency
- Share Nothing!
- Built on Inter-Component Communication over Well Defined Protocols

Isolation

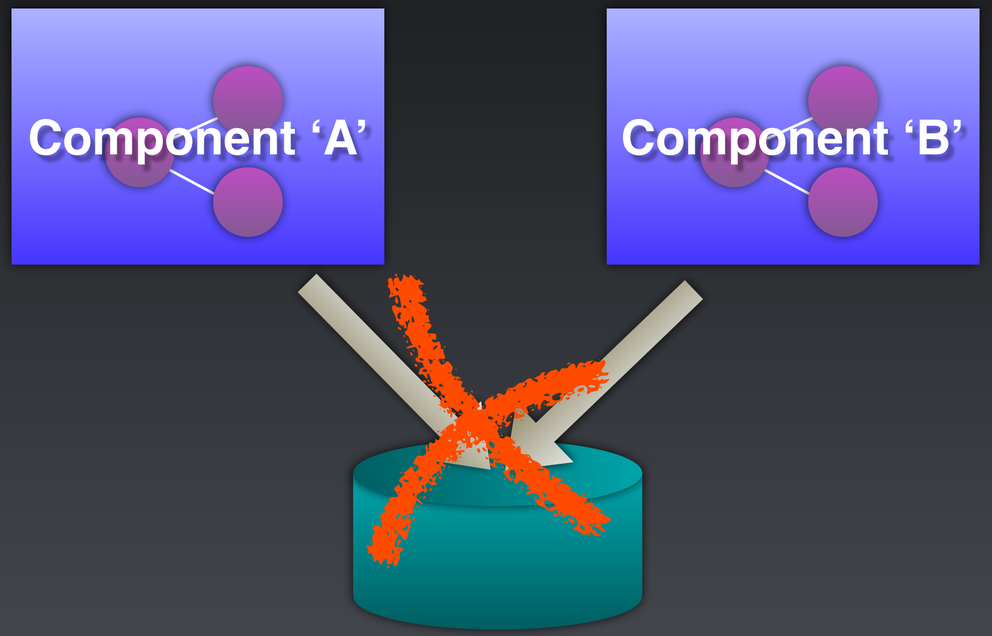




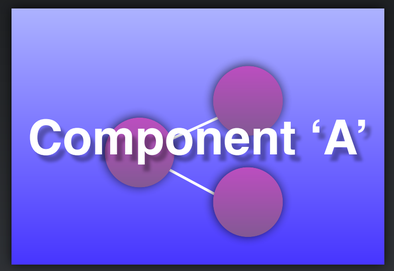
Isolation

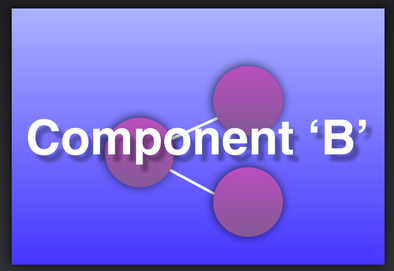


Isolation

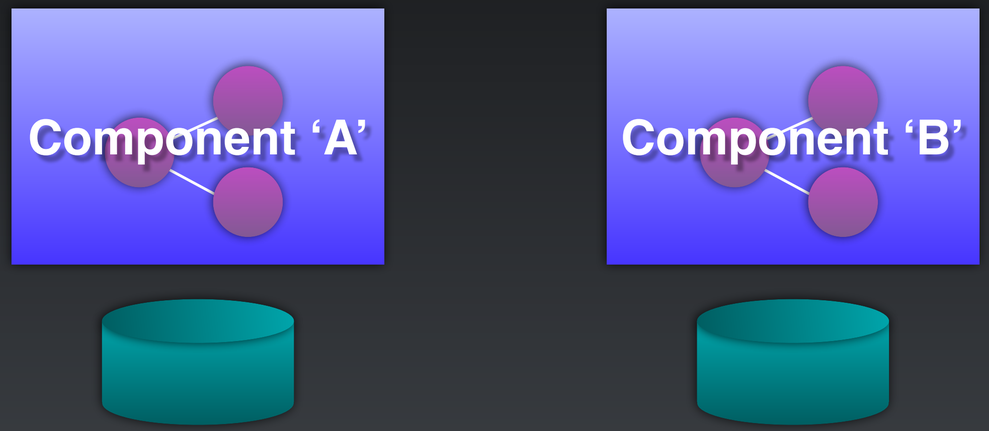


Isolation

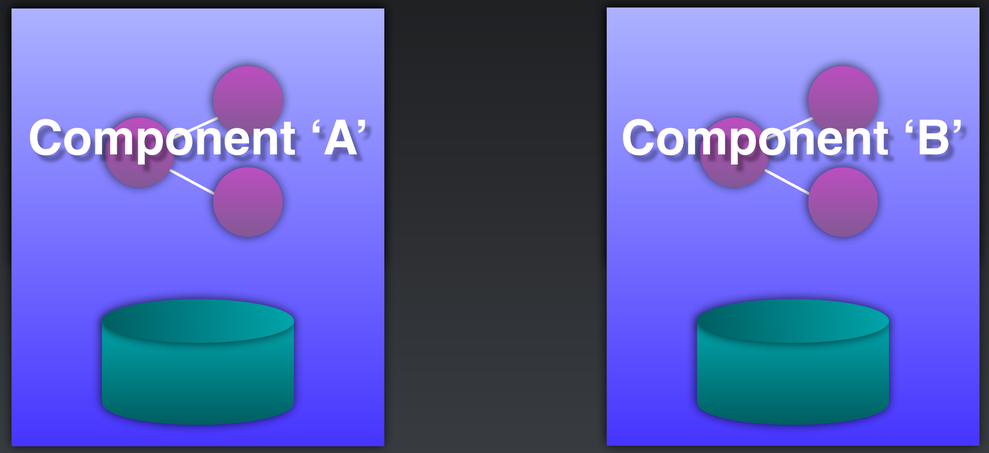




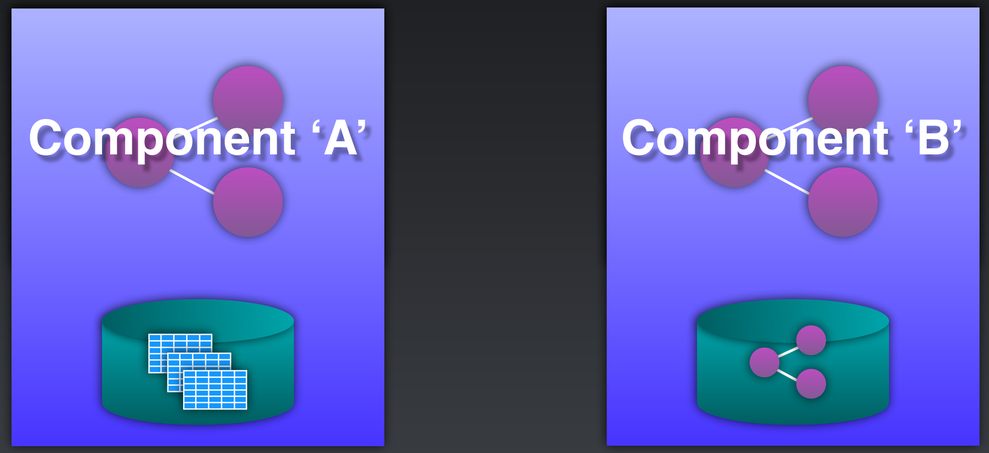
Isolation



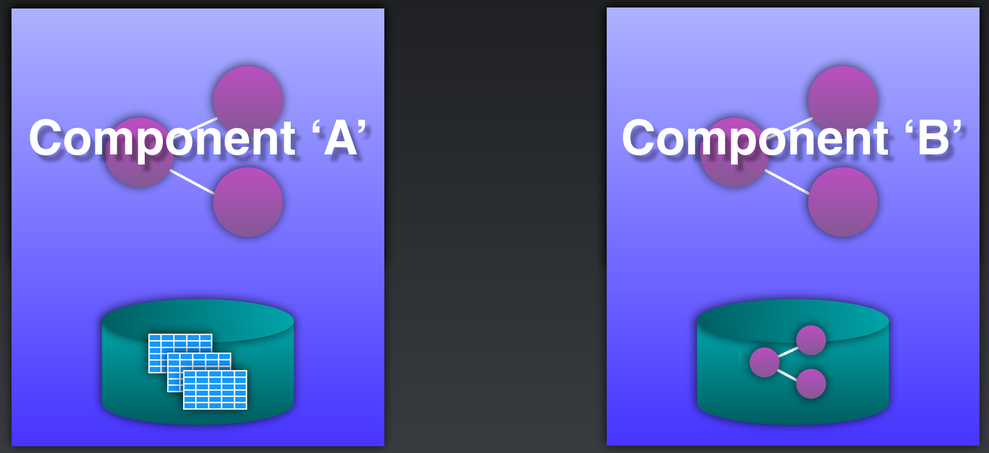
Isolation



Isolation

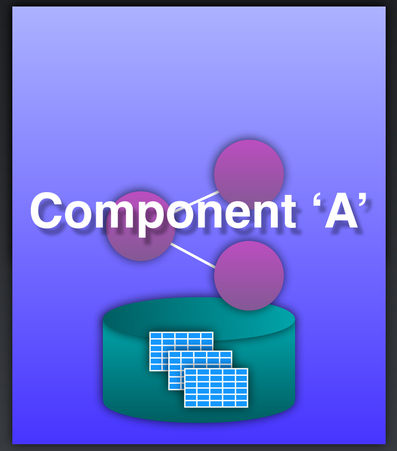


Share Nothing

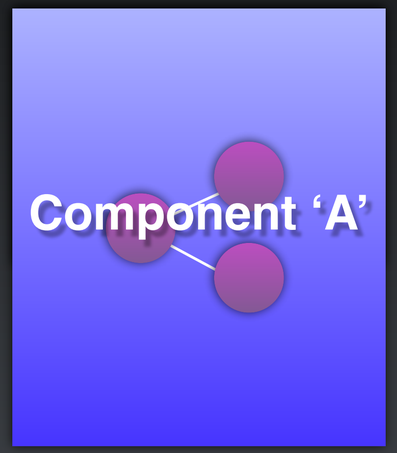


Share Nothing

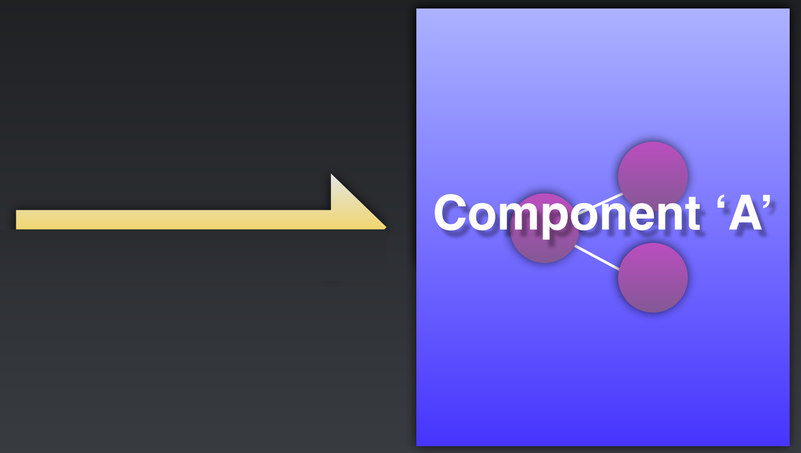
Simple Programming Model



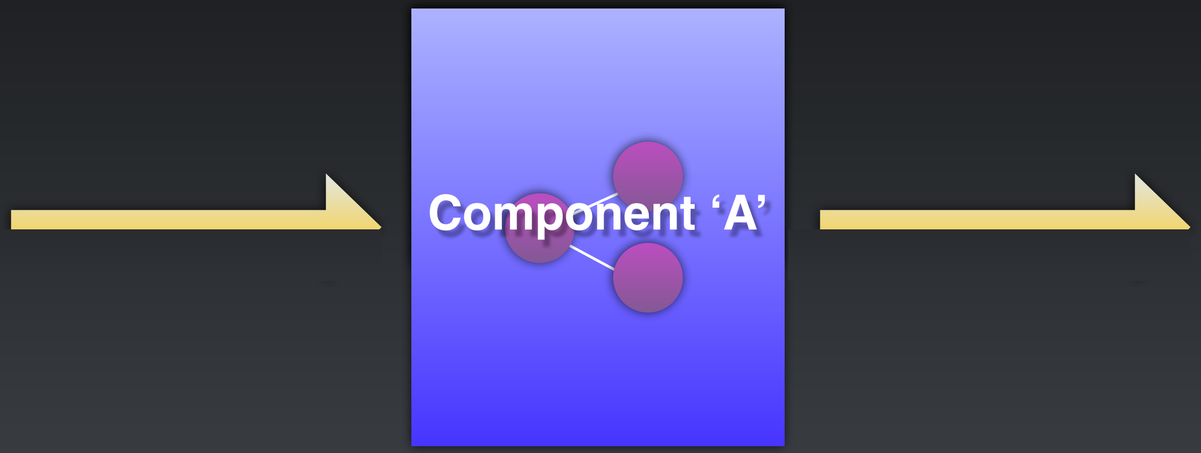
Simple Programming Model



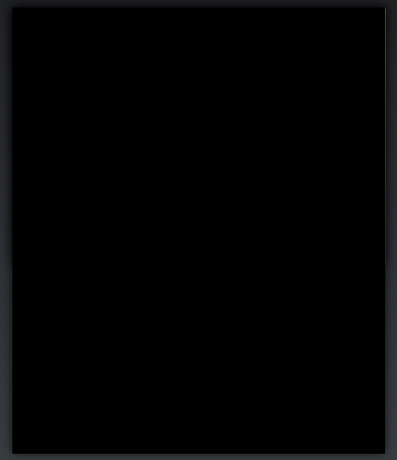
Simple Programming Model



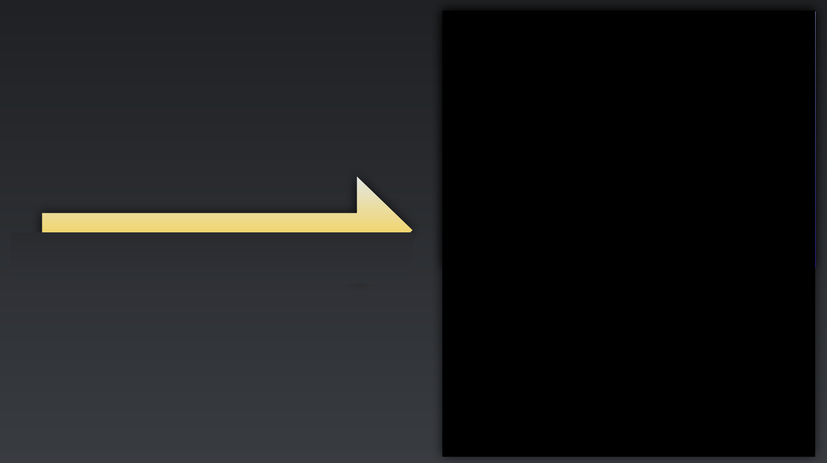
Simple Programming Model



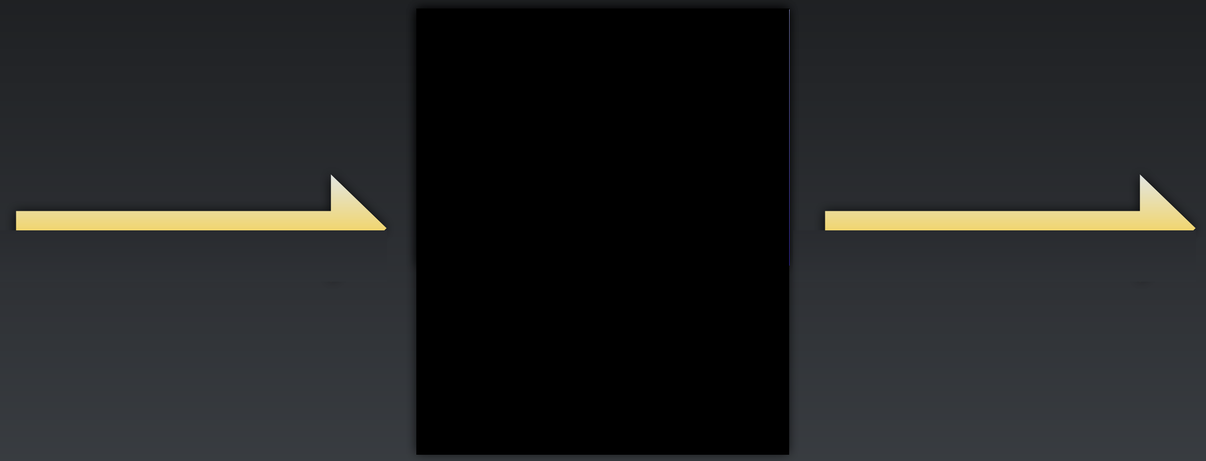
Simple Programming Model



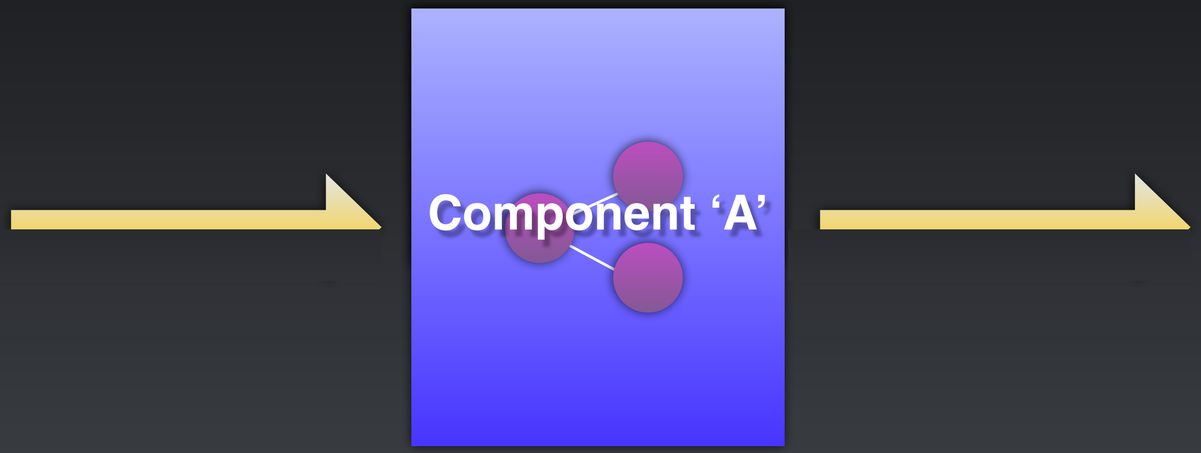
Simple Programming Model



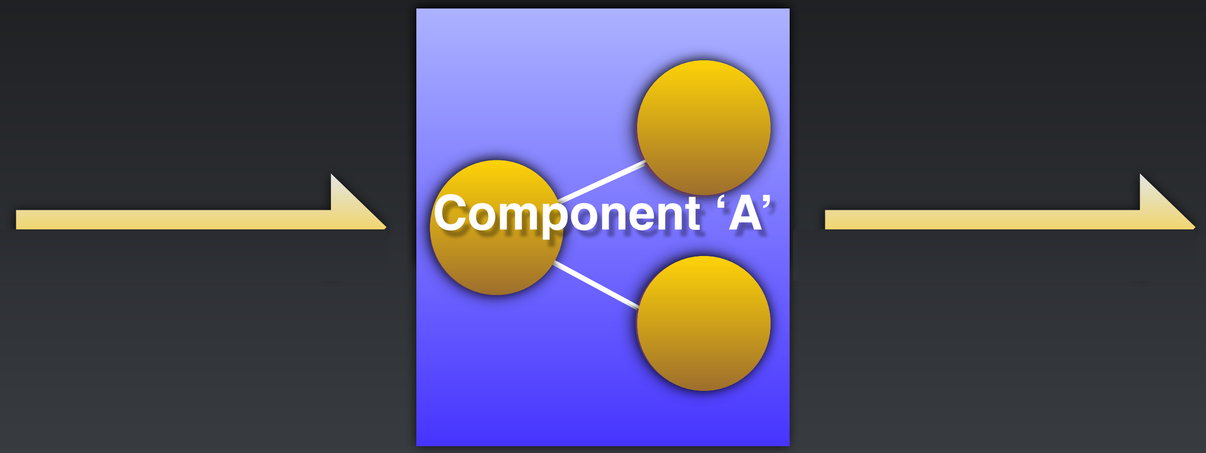
Simple Programming Model



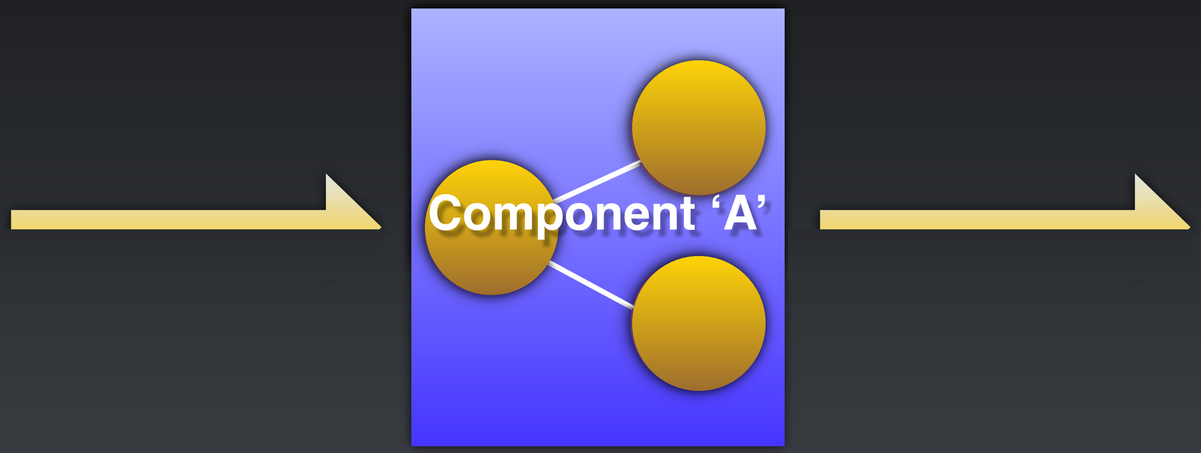
Simple Programming Model



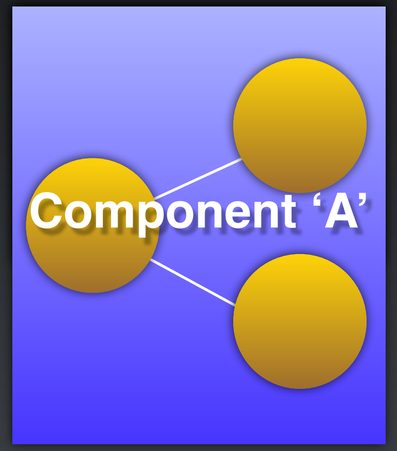
Simple Programming Model



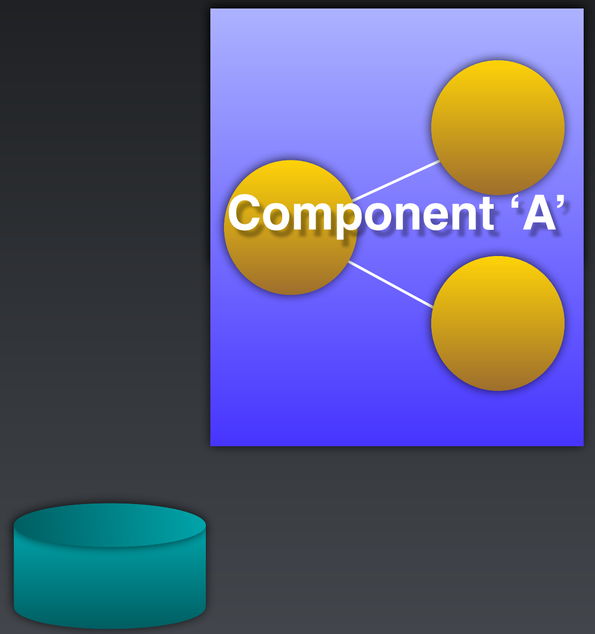
Simple Domain Model



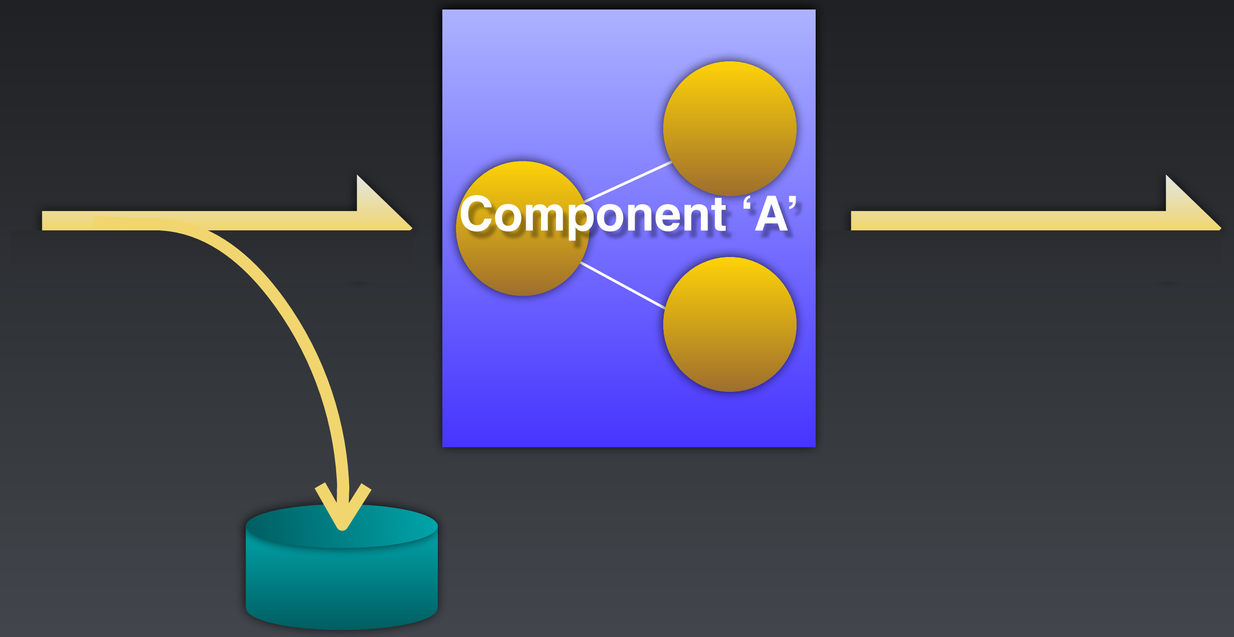
Message Based Persistence



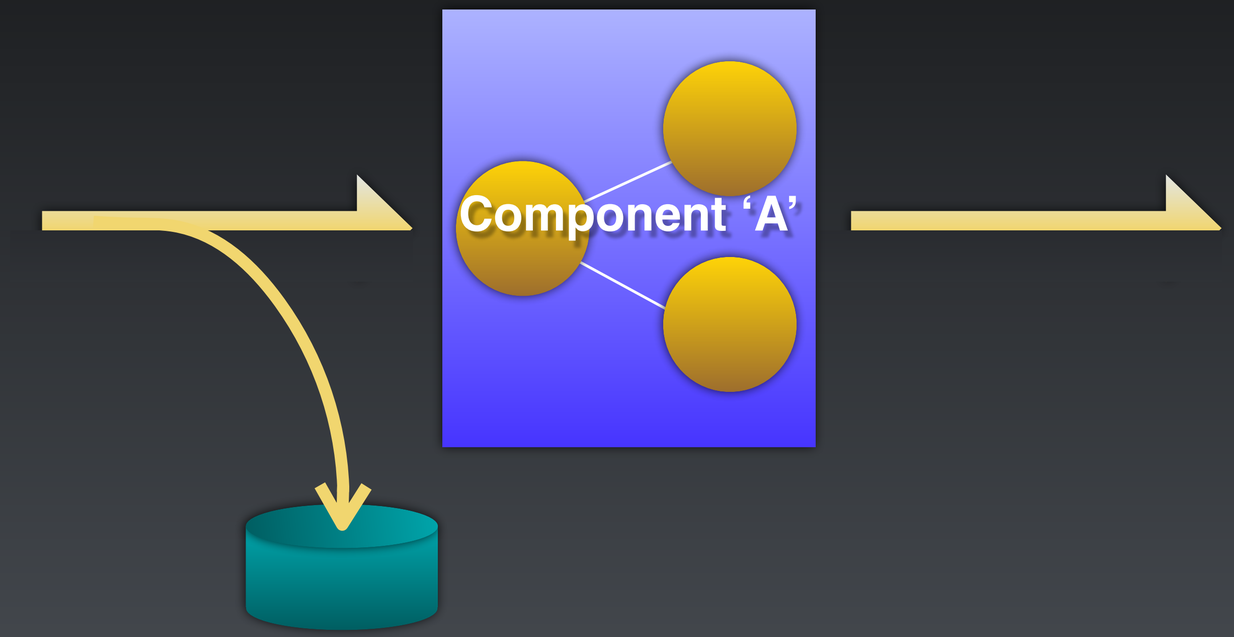
Message Based Persistence



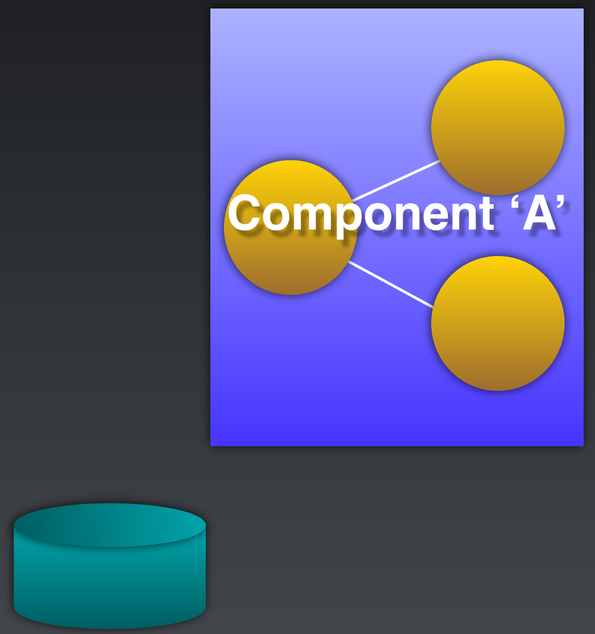
Message Based Persistence



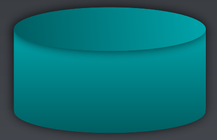
Message Based Persistence



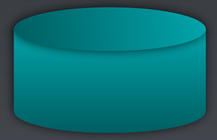
Message Based Persistence



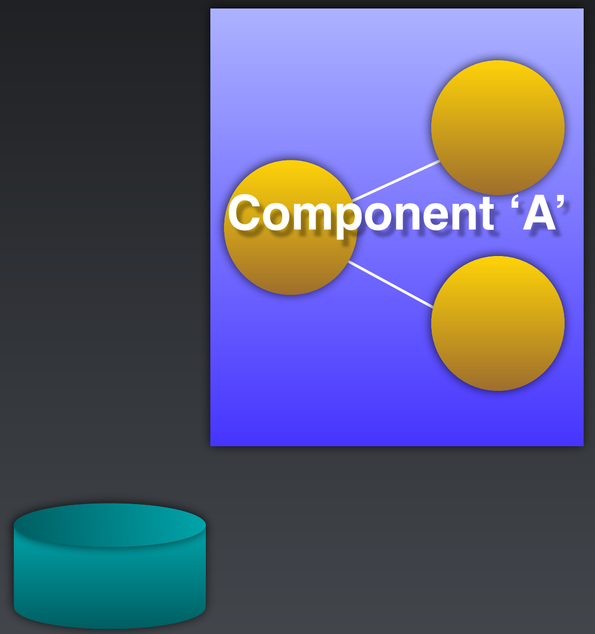
Message Based Persistence



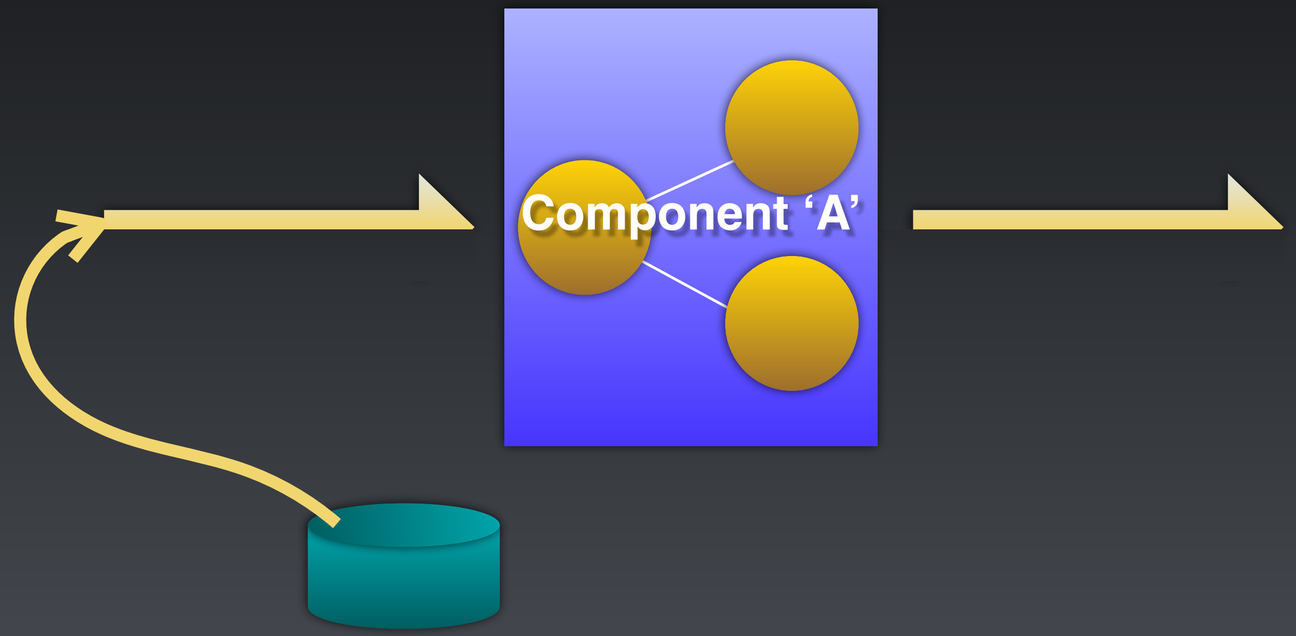
Message Based Persistence



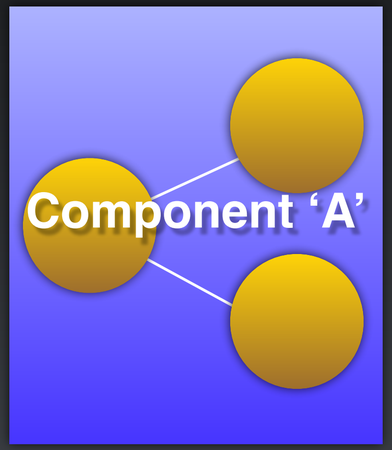
Message Based Persistence



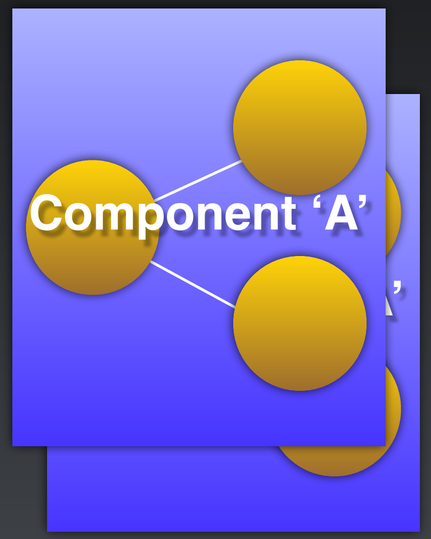
Message Based Persistence



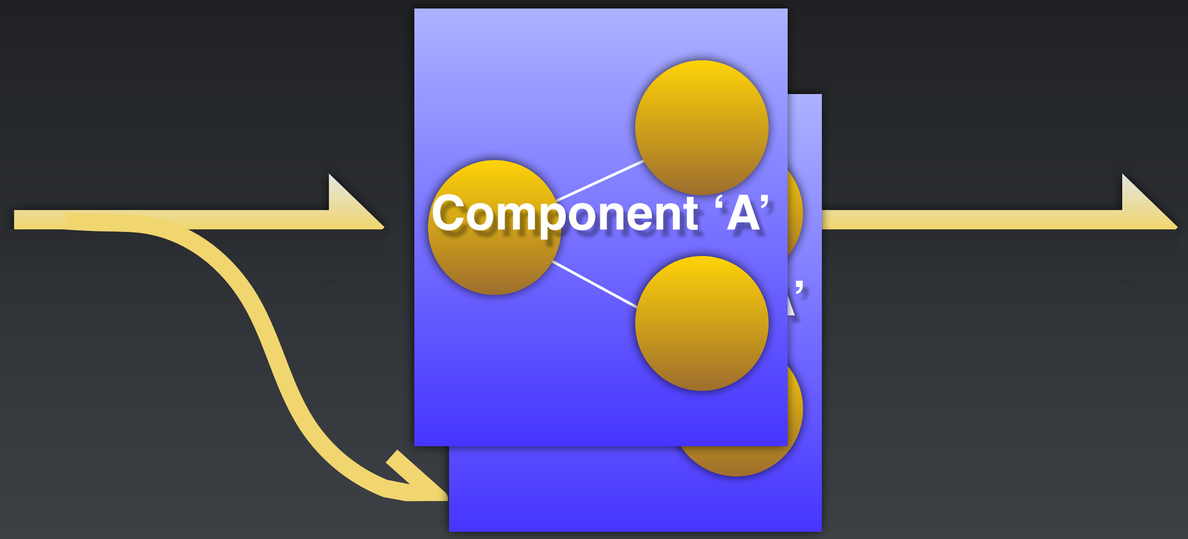
Cluster Based Persistence



Cluster Based Persistence



Cluster Based Persistence



Back-Pressure

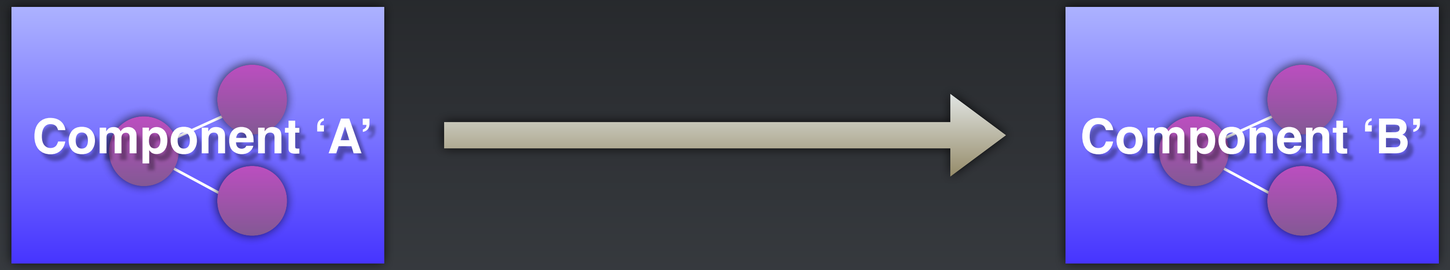
- You Can’t Isolate Stress
- The System as a Whole Needs to Respond Sensibly
- Unacceptable For a Stressed Component to Fail Catastrophically or Loose Messages
- Queues Represent An Unstable State - Load
- Components Under Stress Need to Reflect This By Applying Back-Pressure, Slowing Upstream Inputs

Queues Represent an Unstable State

Queues are always full or always empty.
Anything else is transitional, on its way to full or empty.

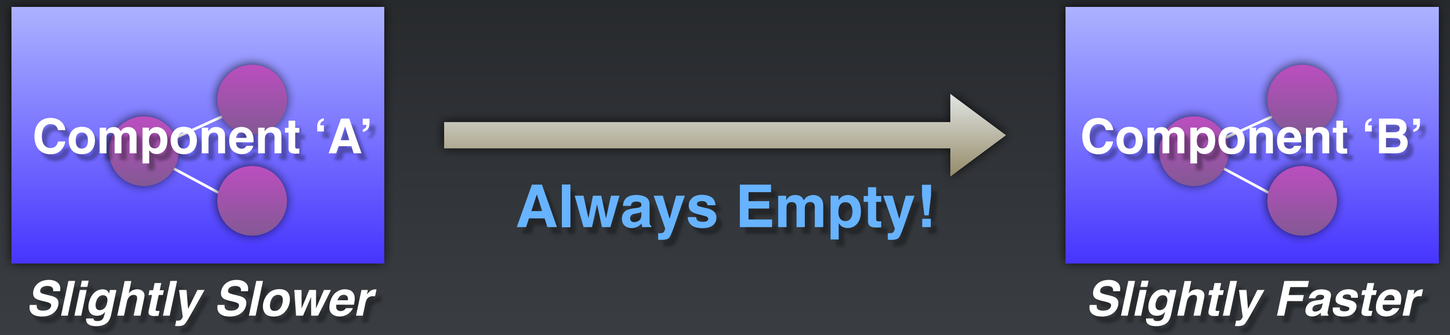
Queues Represent an Unstable State

Queues are always full or always empty.
Anything else is transitional, on its way to full or empty.



Queues Represent an Unstable State

Queues are always full or always empty.
Anything else is transitional, on its way to full or empty.

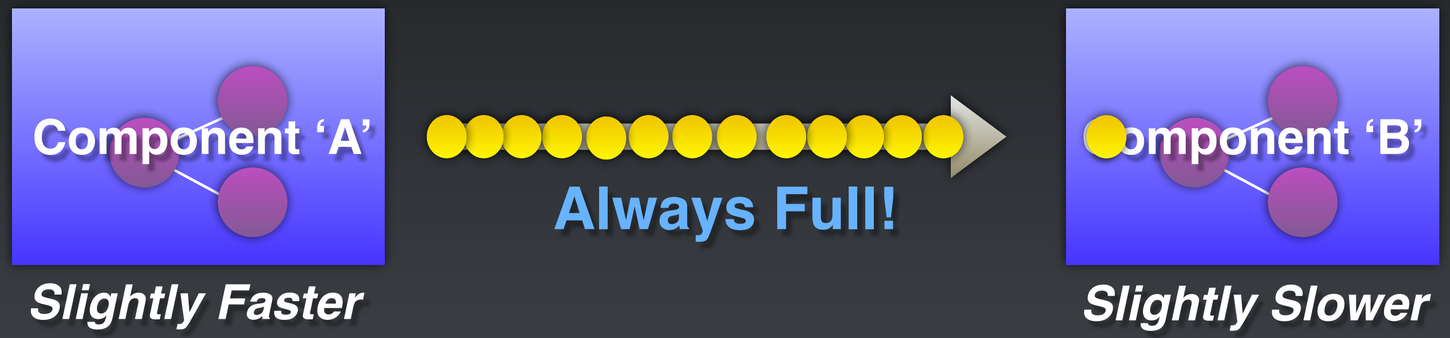


Queues Represent an Unstable State

Queues are always full or always empty.
Anything else is transitional, on its way to full or empty.

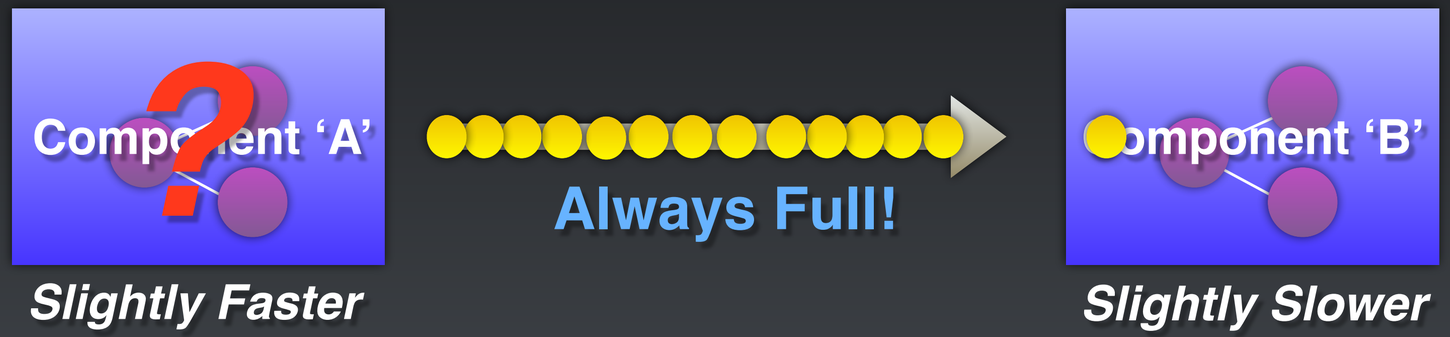
Queues Represent an Unstable State

Queues are always full or always empty.
Anything else is transitional, on its way to full or empty.



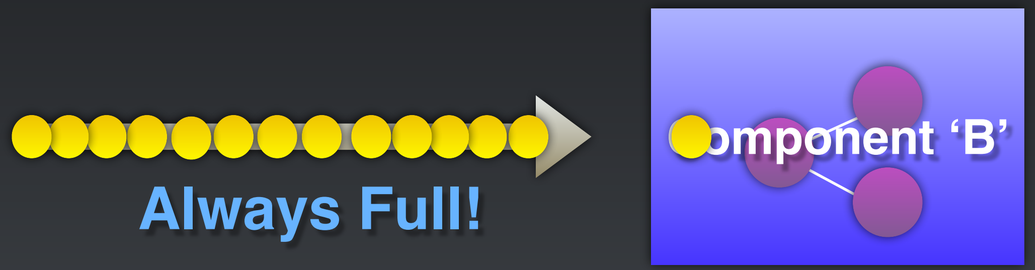
Queues Represent an Unstable State

Queues are always full or always empty.
Anything else is transitional, on its way to full or empty.



Queues Represent an Unstable State

Queues are always full or always empty.
Anything else is transitional, on its way to full or empty.



Slightly Faster

Slightly Slower

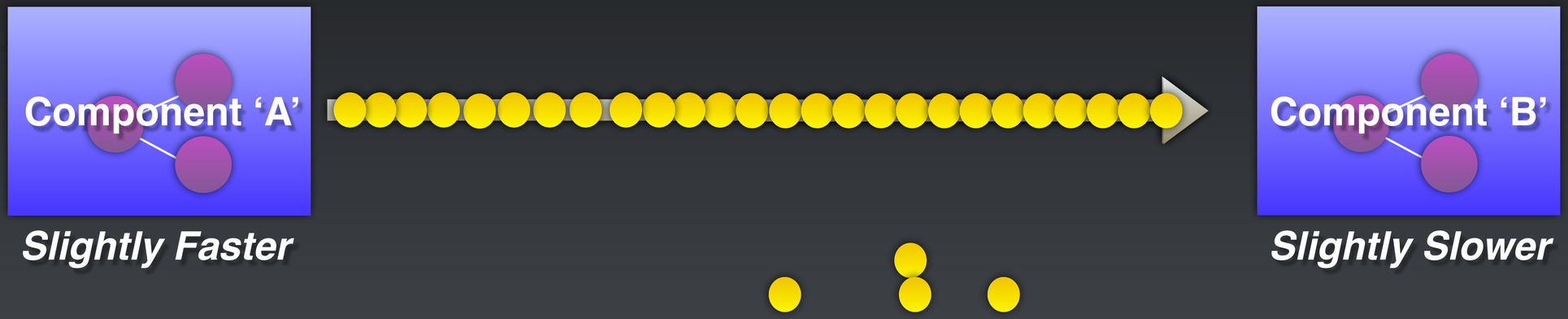
Never Use Unbounded Queues!

“I know let’s allow our queues to grow”

Continuous Delivery ltd

Never Use Unbounded Queues!

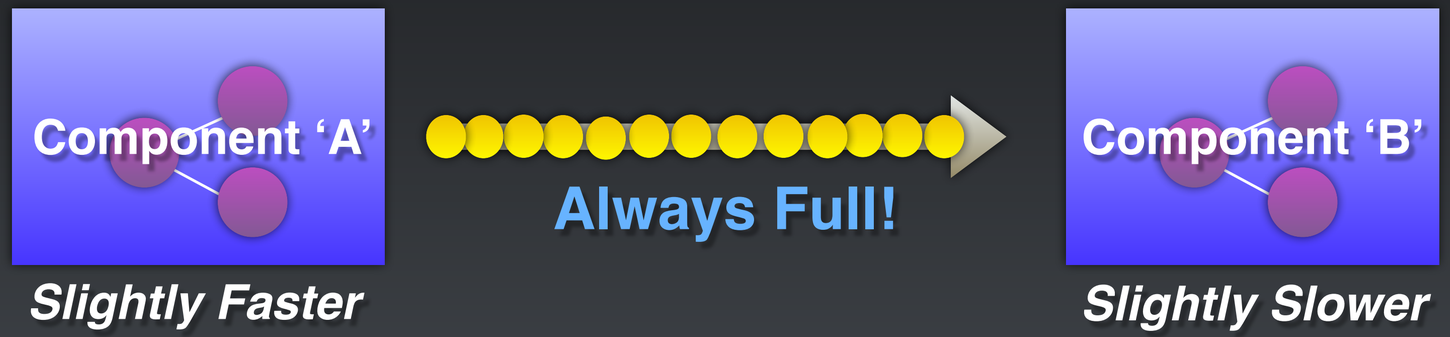
“I know let’s allow our queues to grow”



Never Use Unbounded Queues!

Continuous Delivery ltd

Back-Pressure



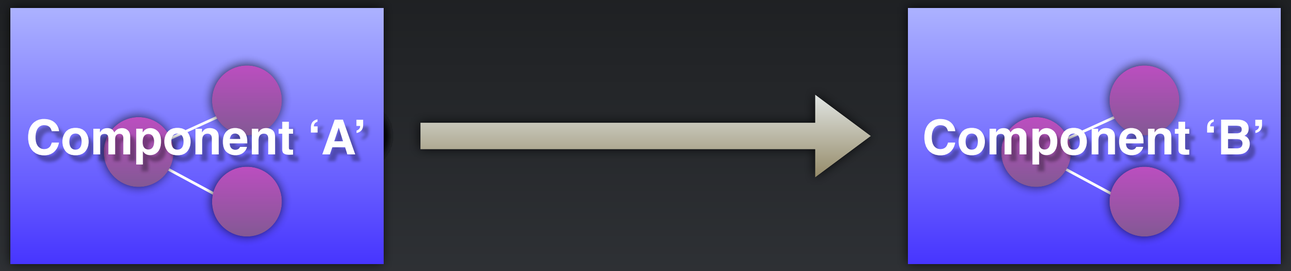
Back-Pressure

Back-Pressure

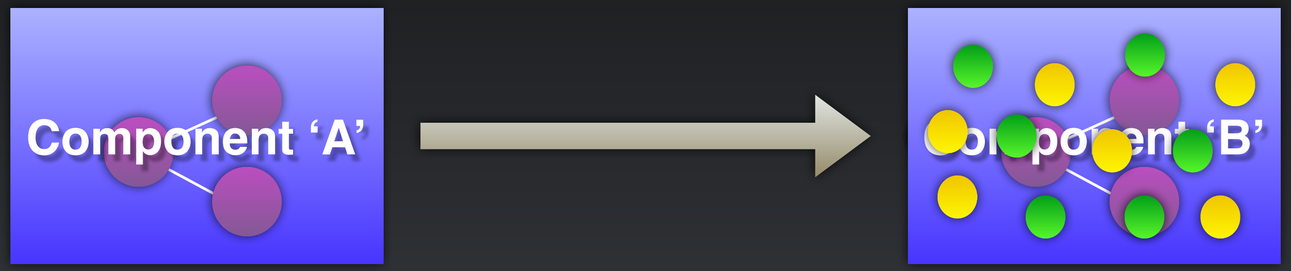
Location Transparency

- Elastic Systems Need To React To Changes In Demand
- We Are All Doing Distributed Computing
- Embracing This Means There Is No Difference Between Horizontal (Cluster) and Vertical (Multicore) Scalability
- Components Should Be Mobile
- One Pattern For Communications
  - Local Communications Is An Optimisation

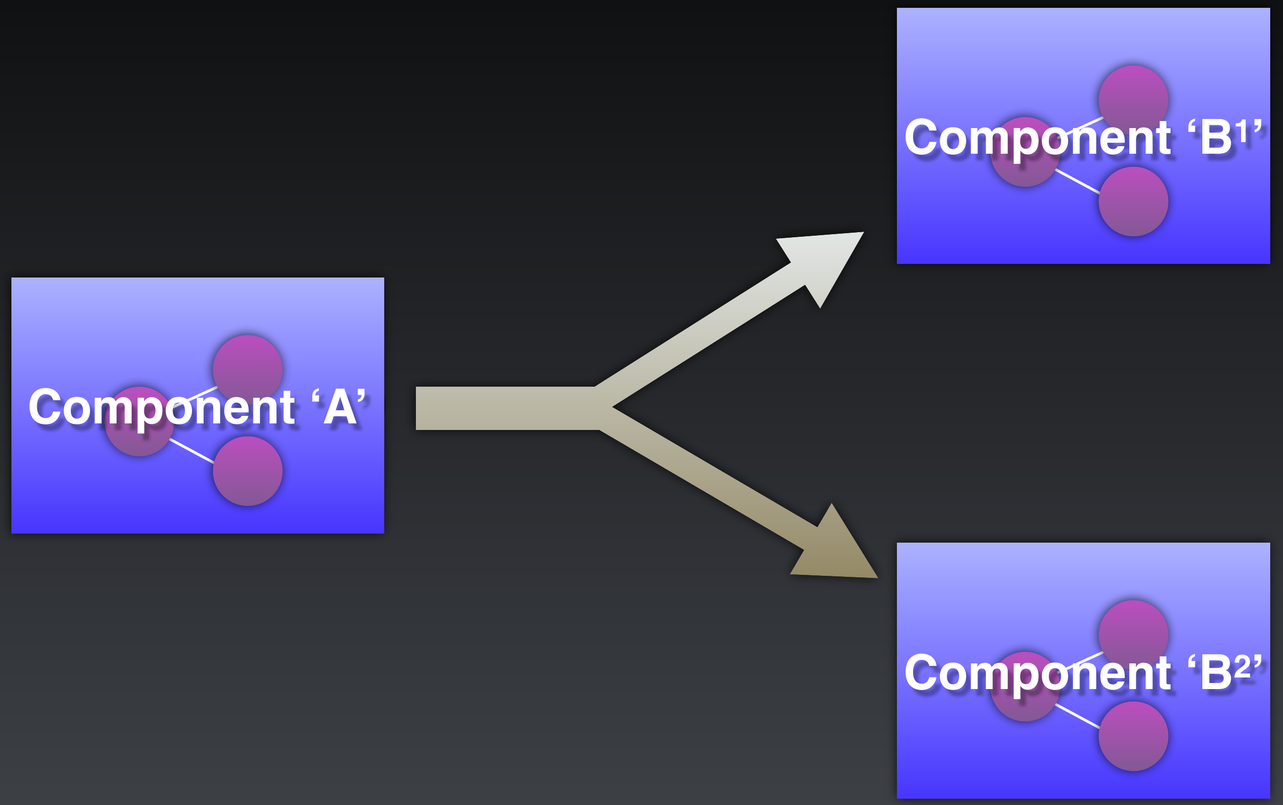
Linear Scalability Through Sharding



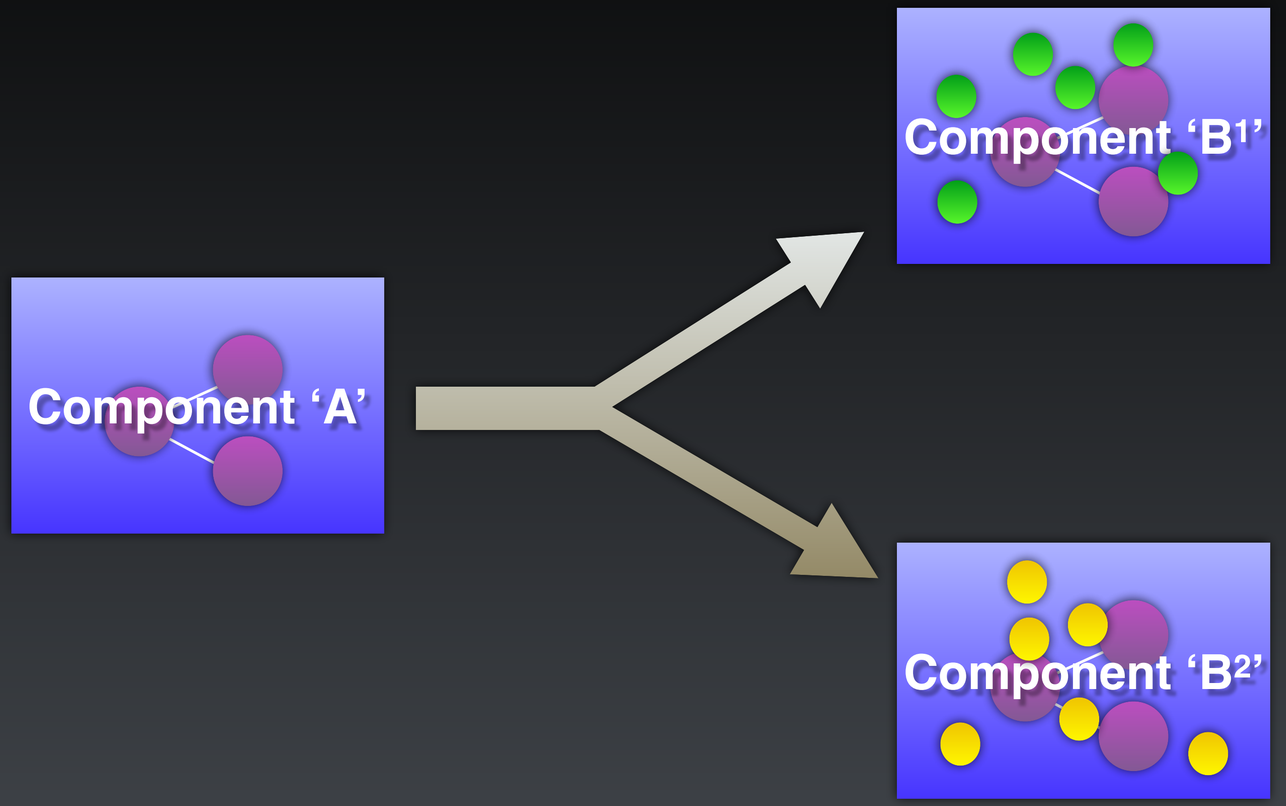
Linear Scalability Through Sharding



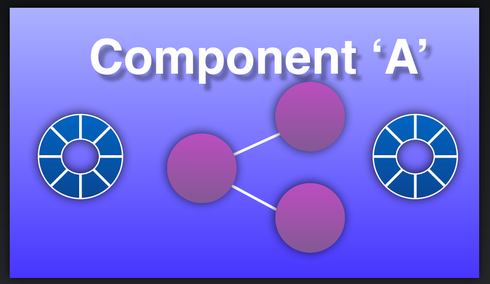
Linear Scalability Through Sharding

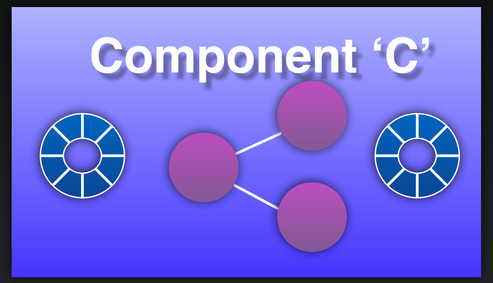


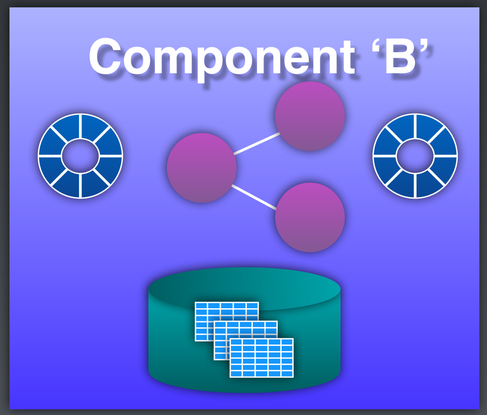
Linear Scalability Through Sharding

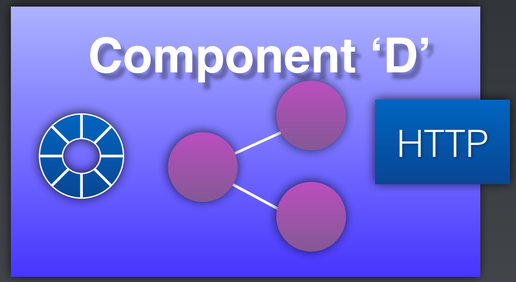


So What does this all look like?



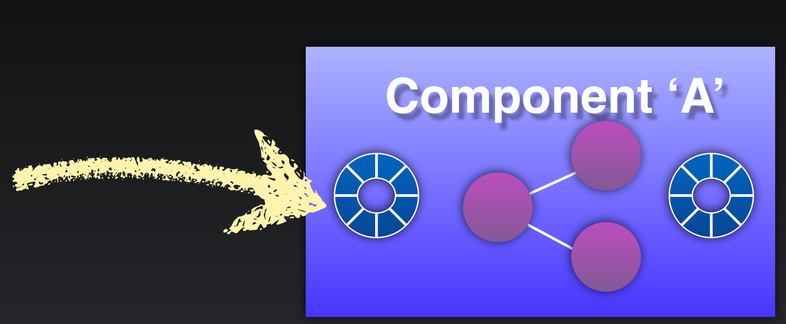


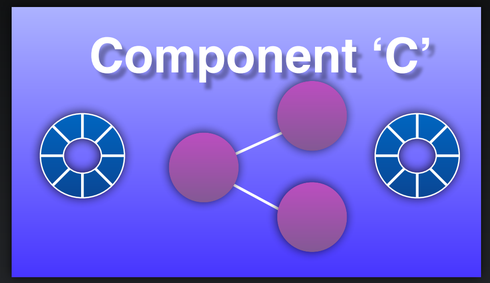


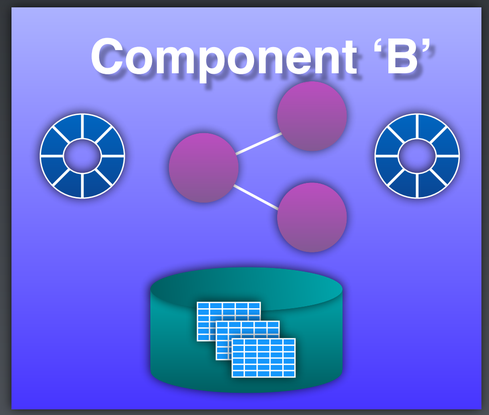


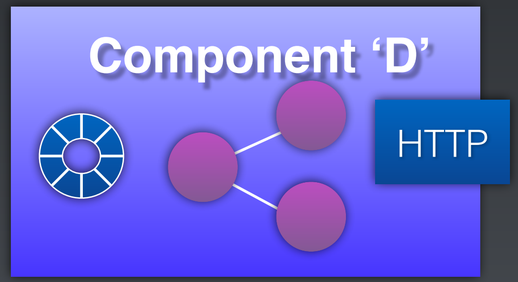
Continuous Delivery ltd

So What does this all look like?



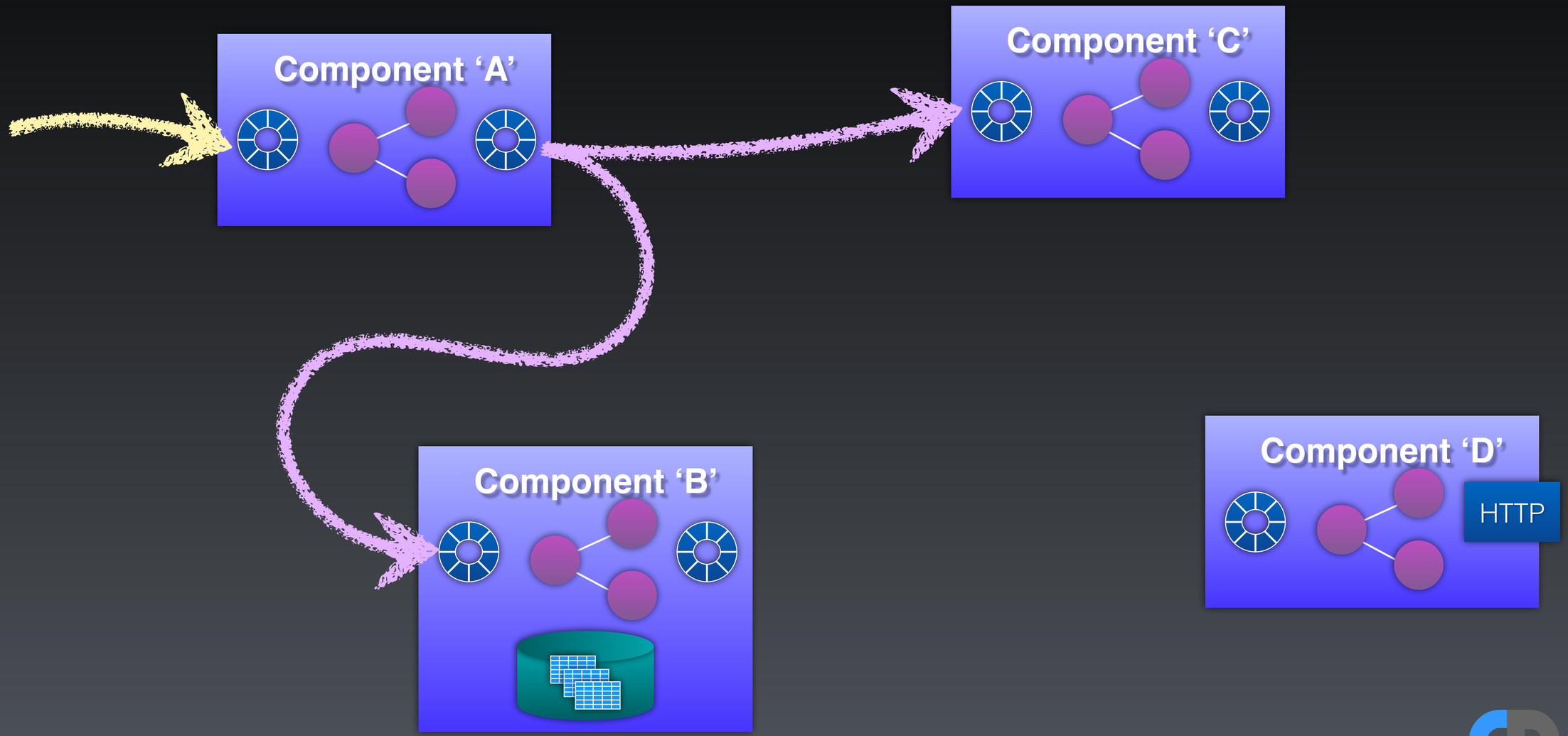




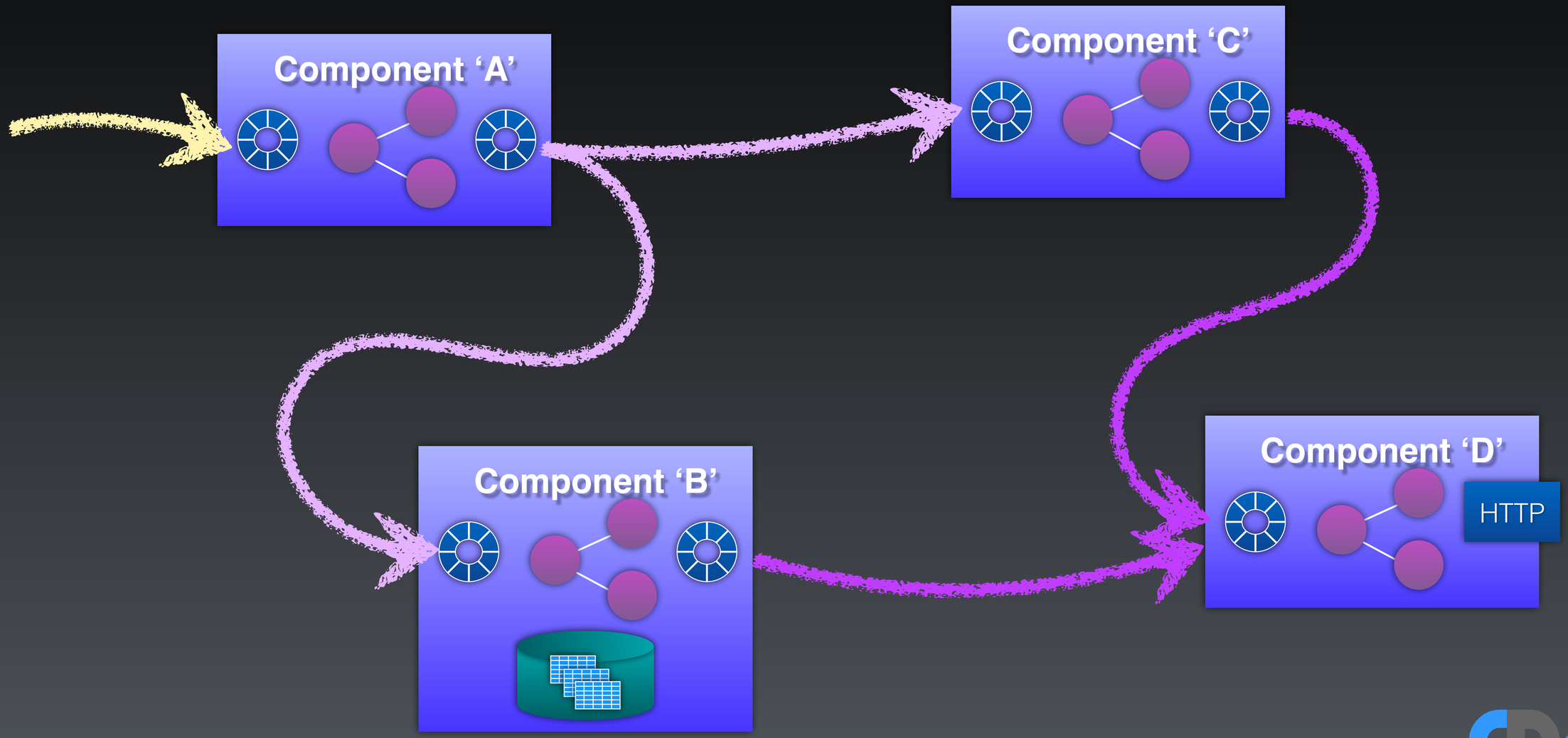


Continuous Delivery ltd

So What does this all look like?

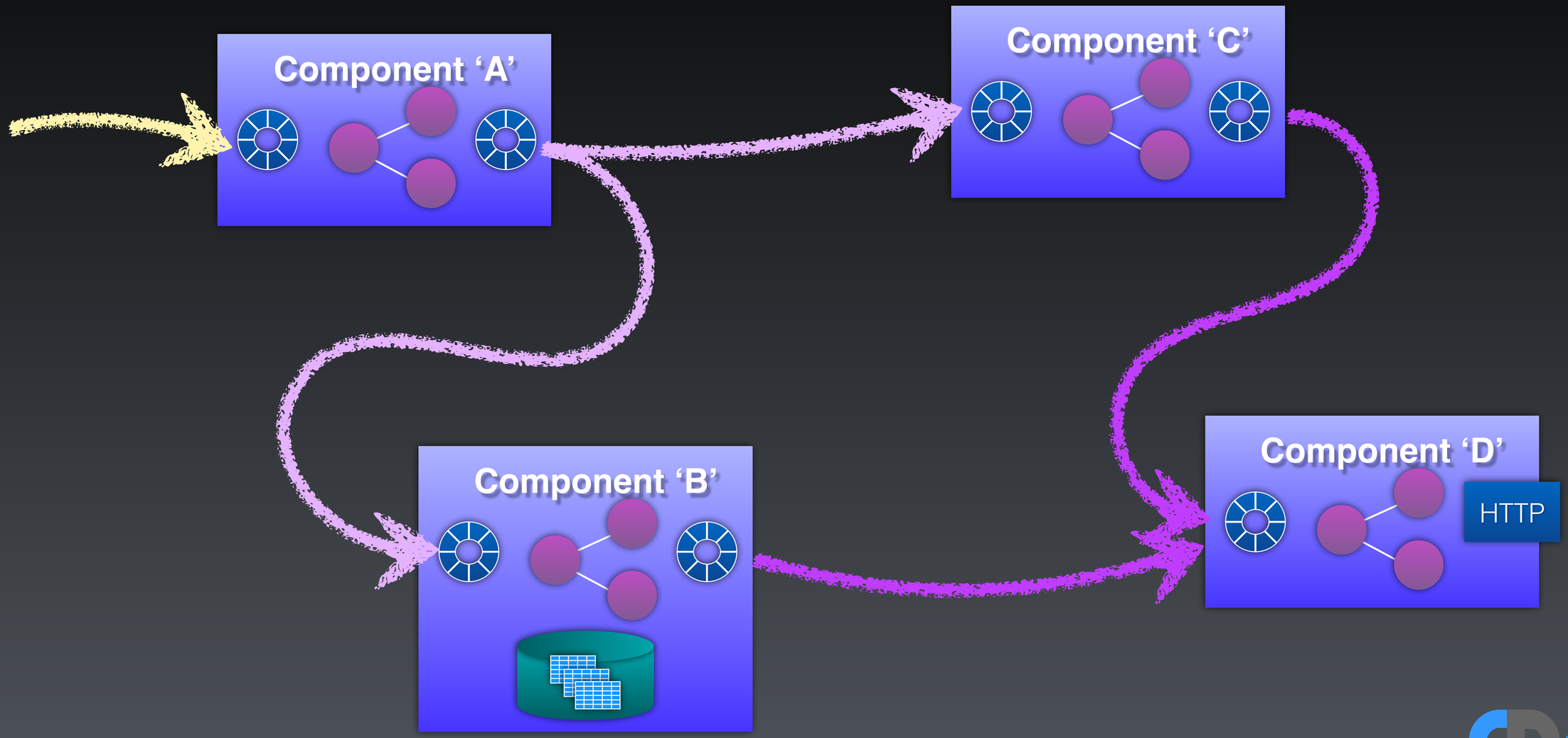


So What does this all look like?



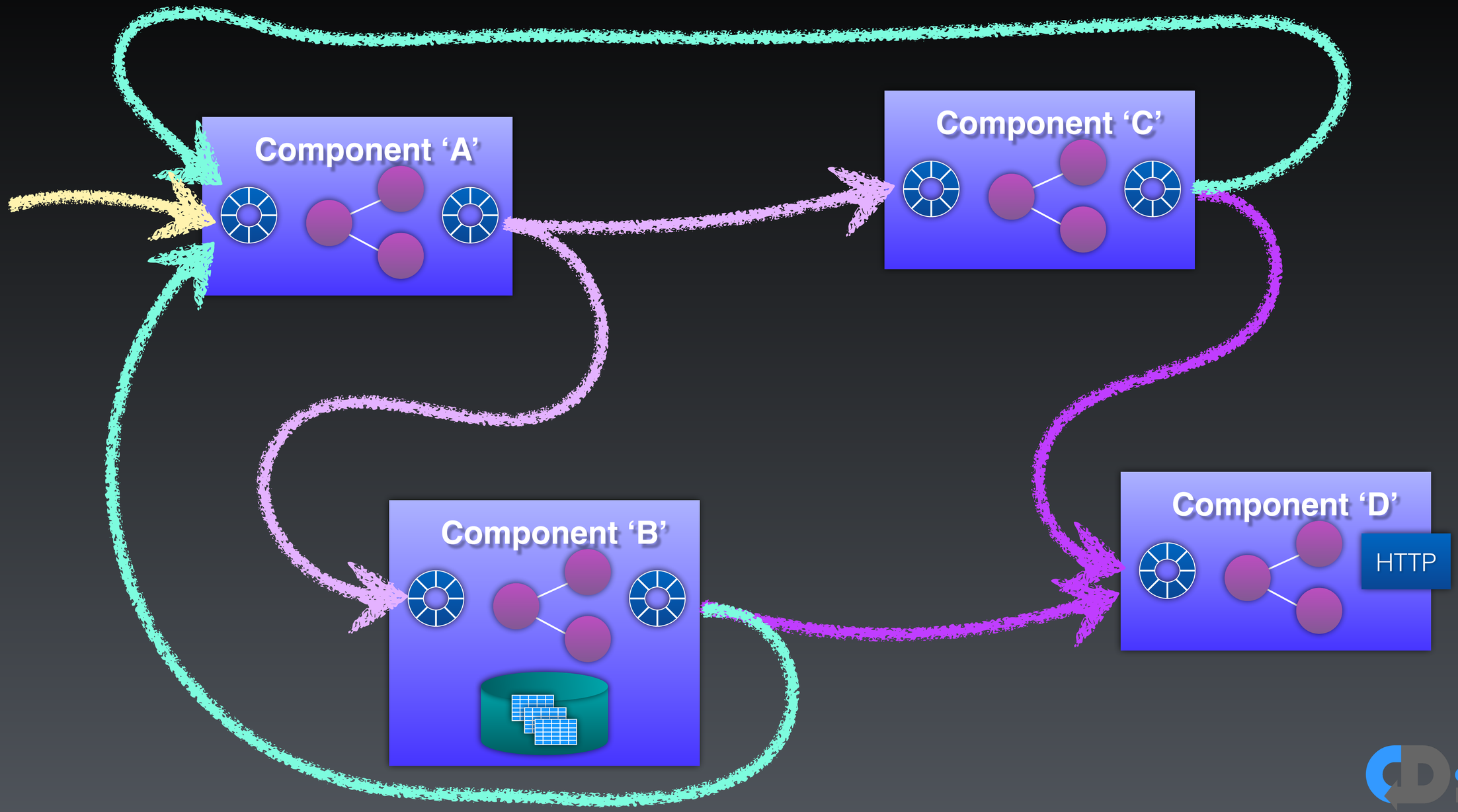
Continuous Delivery ltd

So What does this all look like?



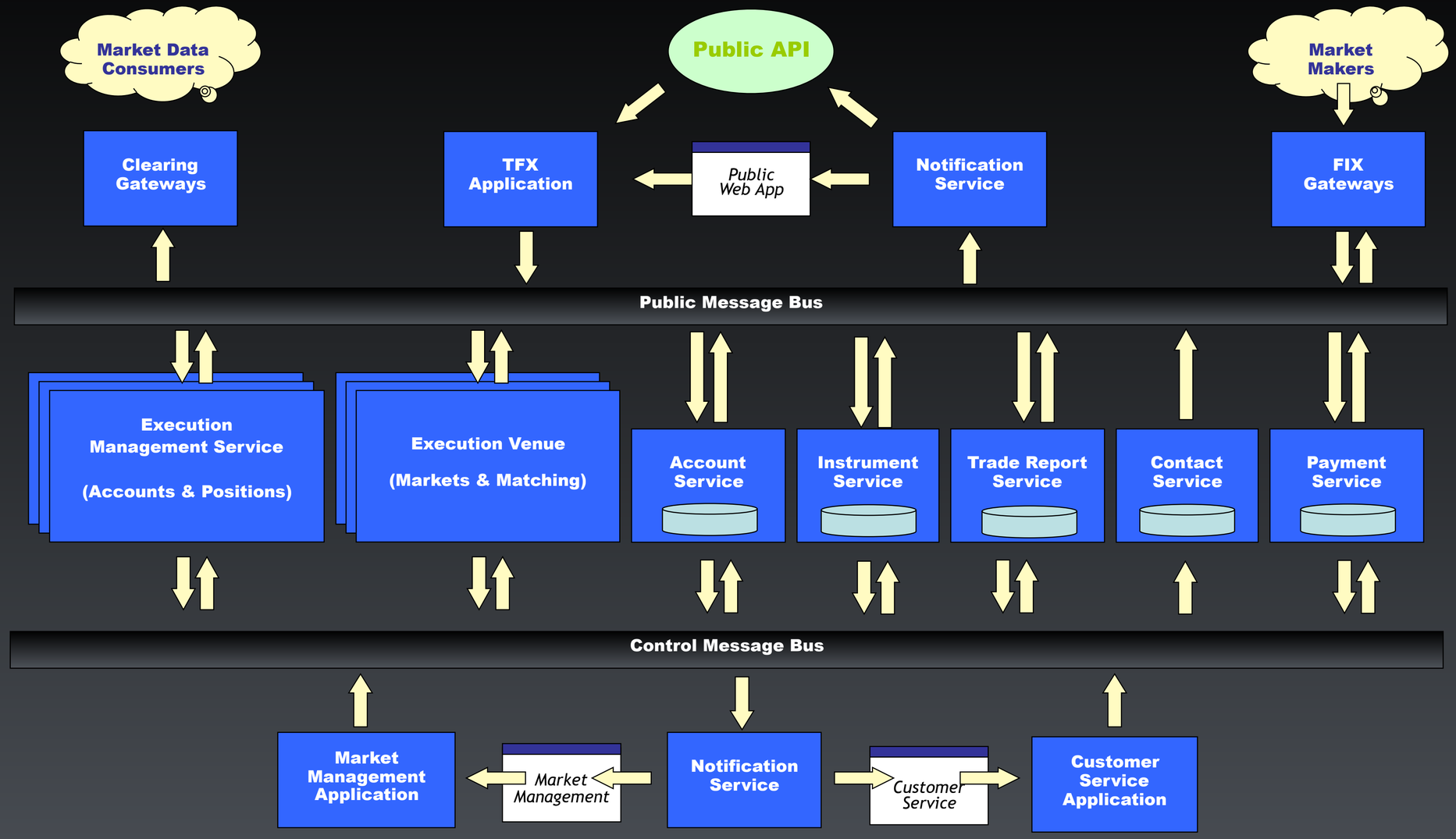
Continuous Delivery ltd

So What does this all look like?

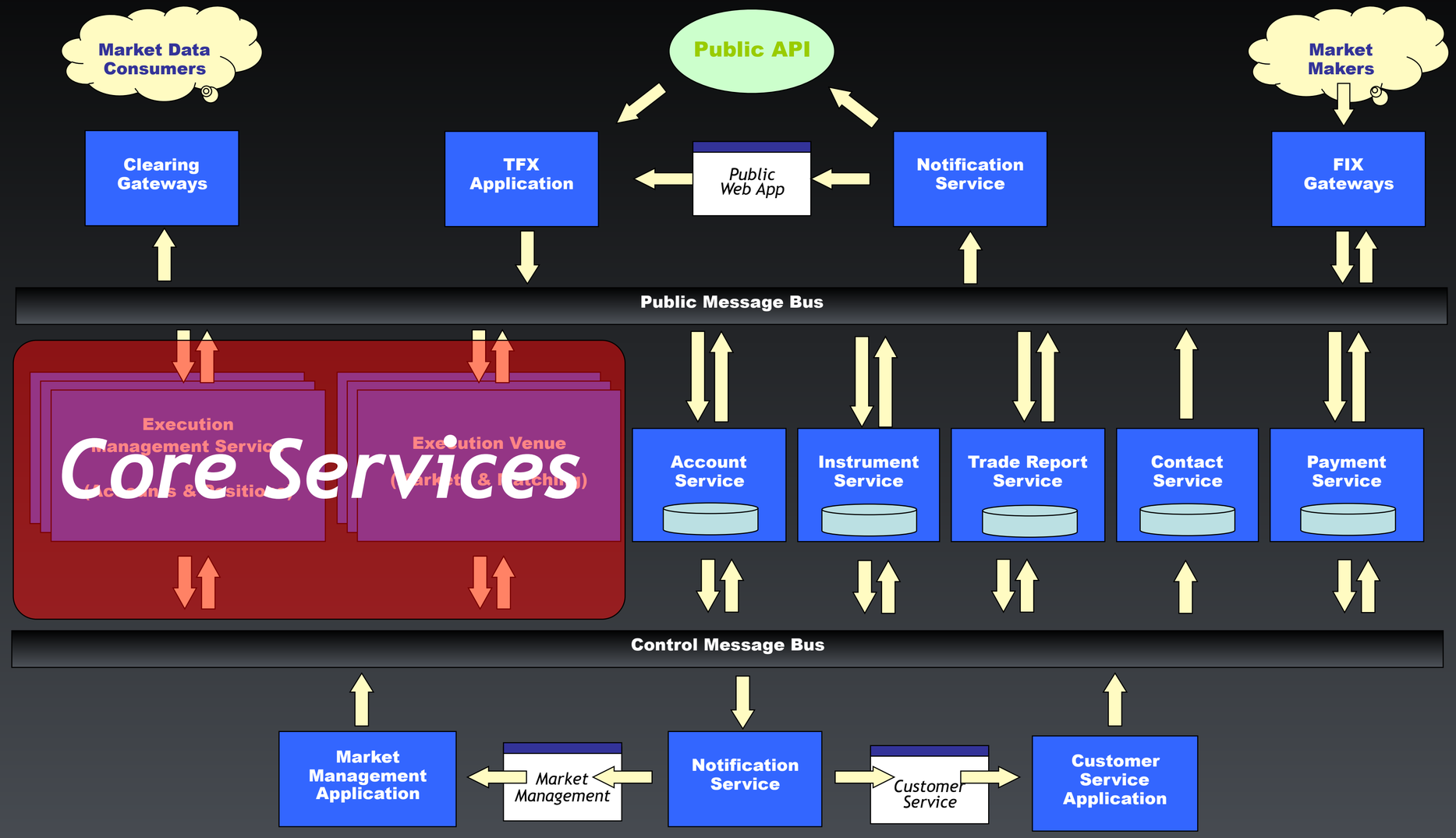


Continuous Delivery ltd

Example Reactive, MicroService architecture

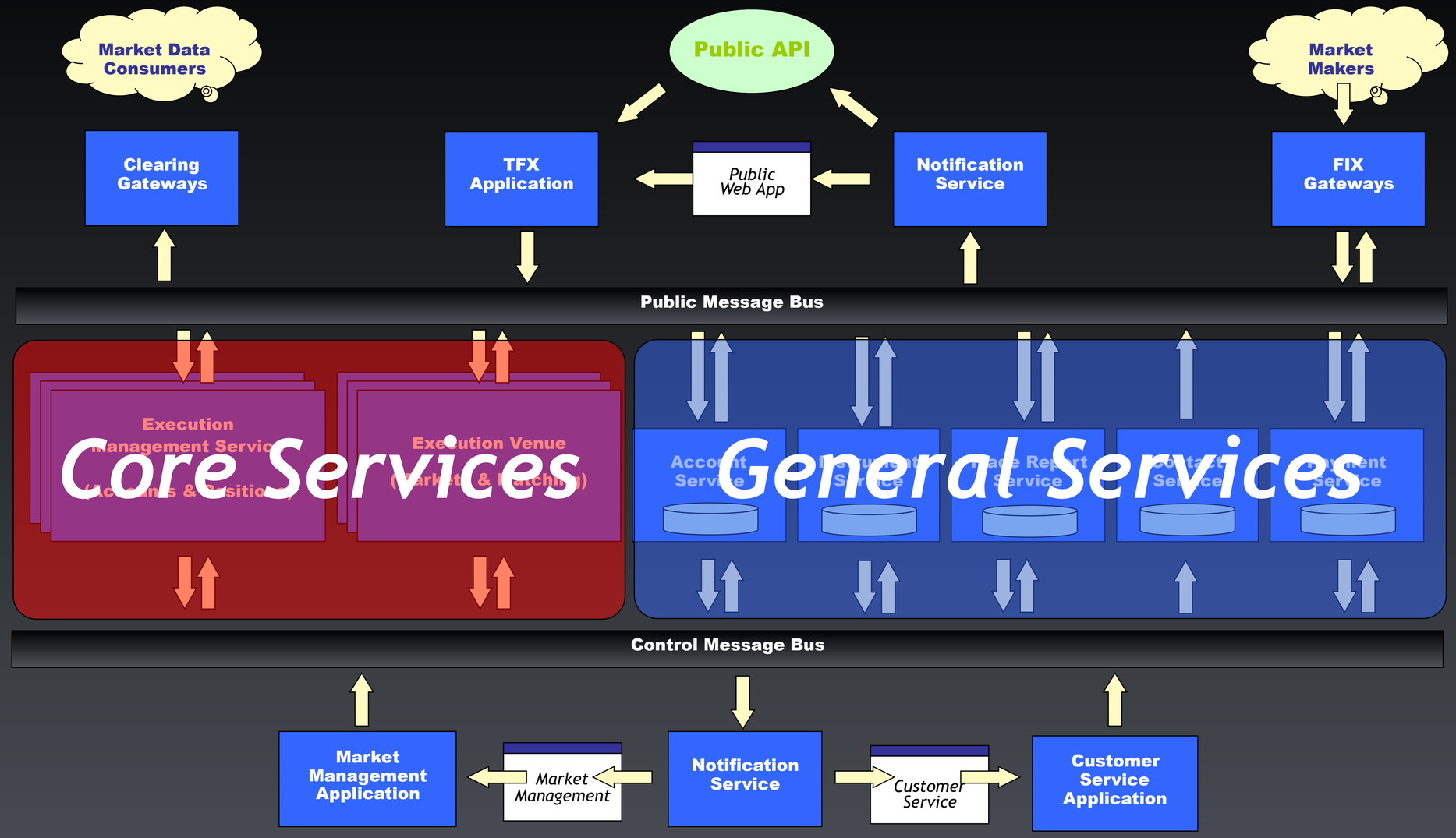


Example Reactive, MicroService architecture

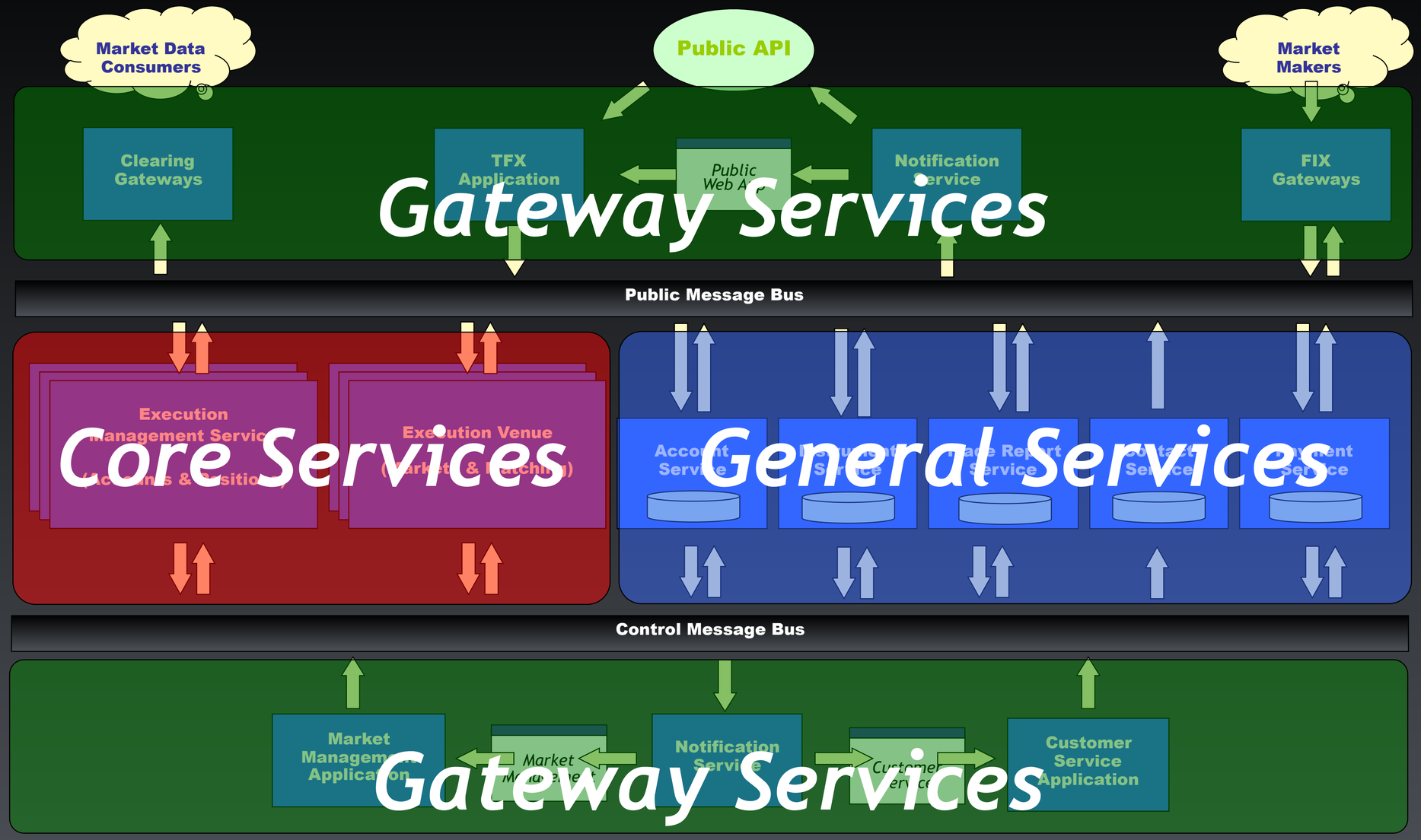


Continuous Delivery ltd

Example Reactive, MicroService architecture

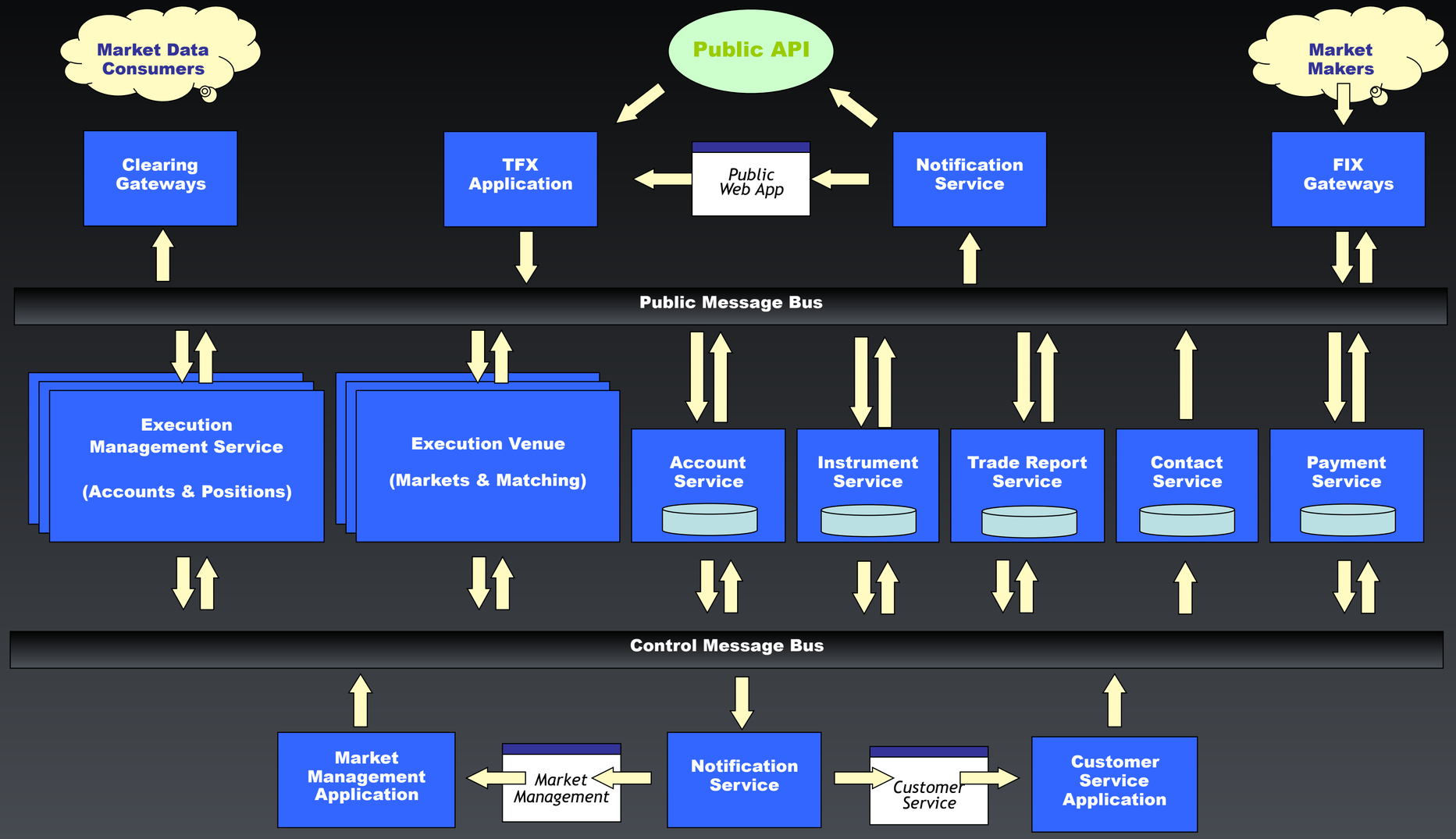


Example Reactive, MicroService architecture



Continuous Delivery ltd

Example Reactive, MicroService architecture



Where to start?

Aeron

Something to keep an eye on:

Statefull Serverless

Q&A

http://www.continuous-delivery.co.uk

Dave Farley

http://www.davefarley.net

@davefarley77

Continuous
Delivery ltd