Advanced Topics

Memi Lavi
www.memilavi.com

Advanced Topics

- There are a lot of nuances with Event Driven Architecture
- We talked about the mainstream implementations
- Covers ~80% of the cases
- There are advanced topics you should be aware of
- Perhaps will be never used but it’s good to be aware of

Advanced Topics

Mixing EDA with Request / Response

Synchronous EDA

Events as Source of Truth

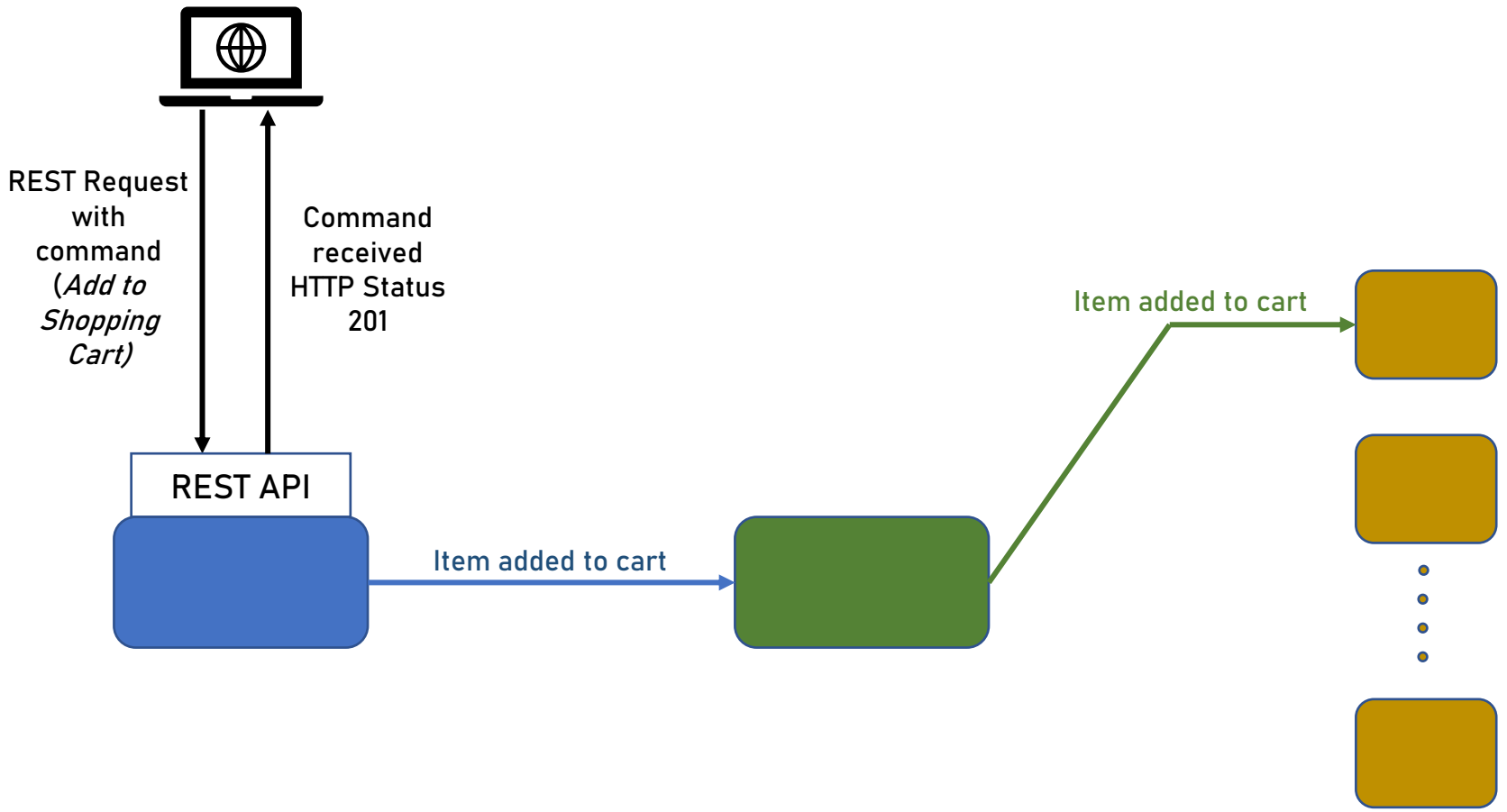
The Saga Pattern

EDA on the Front End

Mixing EDA with Request / Response

- Most EDA systems are not pure EDA
- Main reason:
  - UI Clients need responsiveness and use Web API to call the backend
- If client only asks for data, EDA will probably won't work

Mixing EDA with Request / Response



Synchronous EDA

- EDA is asynchronous by nature
- Normally the producer does not wait for a response to the event
- Sometimes it does, as a separate event
- Quite difficult to implement

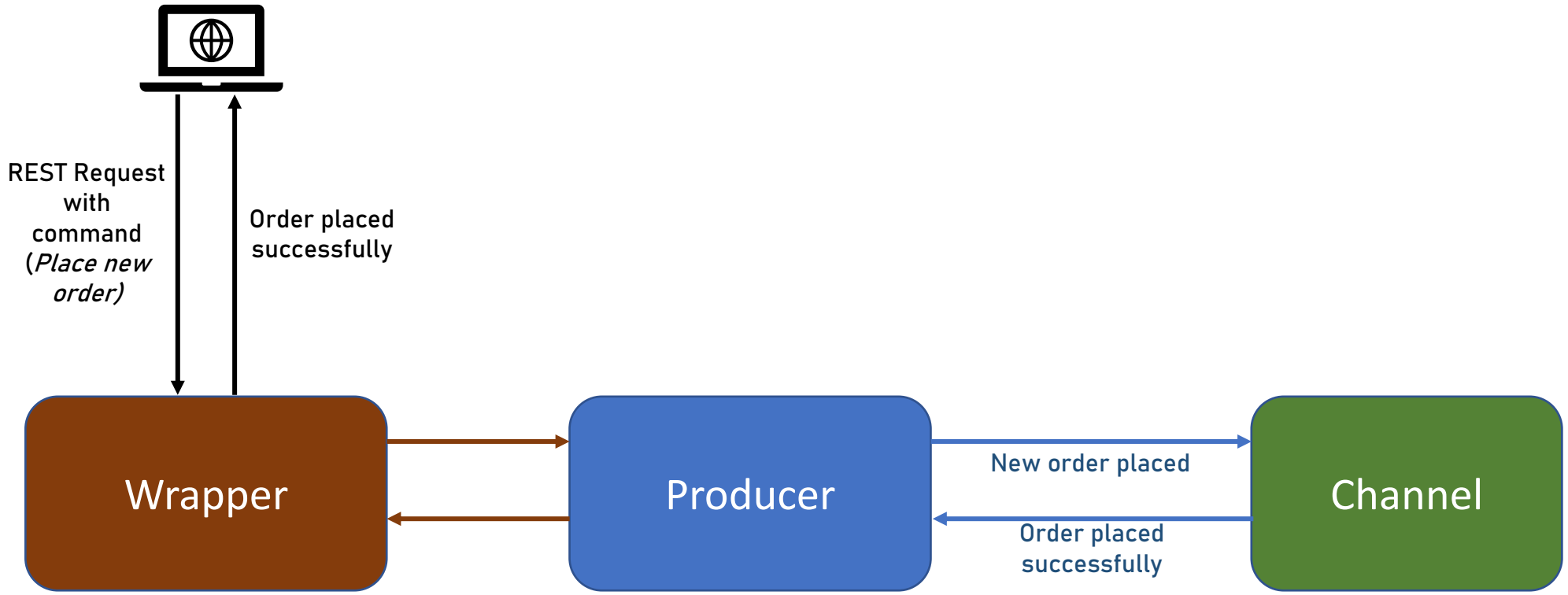
EDA With Response

1. Send the event
2. Wait for the response event…
3. Wait some more…
4. Handle the return event in a separate code segment

Synchronous EDA

- To make the process easier you can create a wrapper around the producer
- The wrapper exposes a synchronous Web API
- Calls the producer and performs all the dirty work of sending -> waiting -> handling response

Synchronous EDA

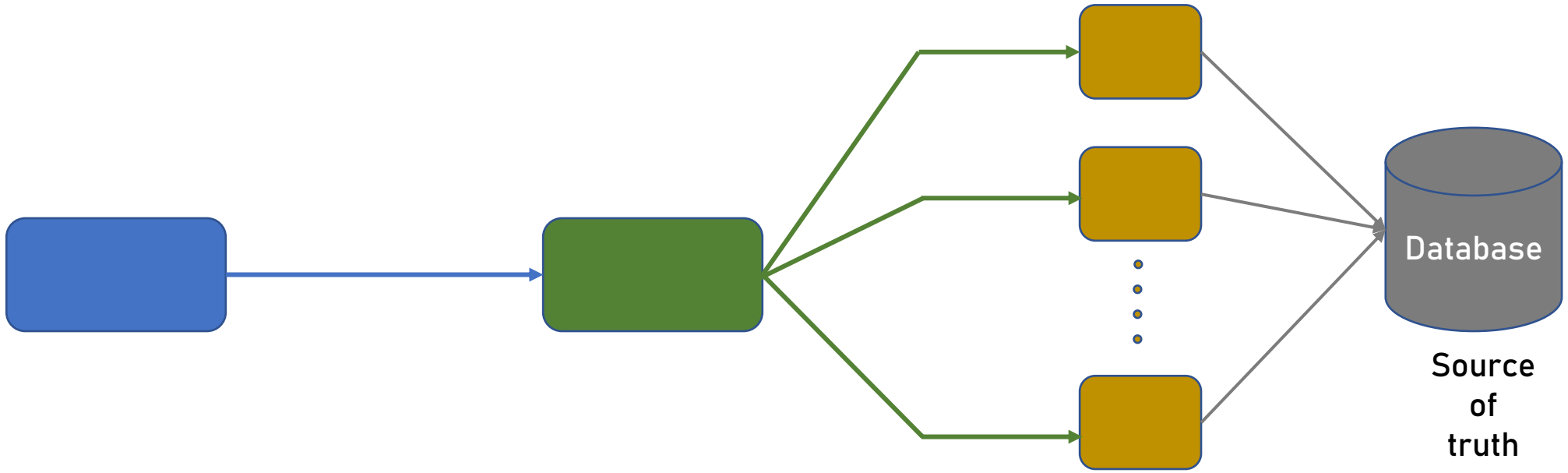


Synchronous EDA

- Not easy to implement
- Do it only if you absolutely need a response from the event
- Some channels (e.g. RabbitMQ) have built-in support for this

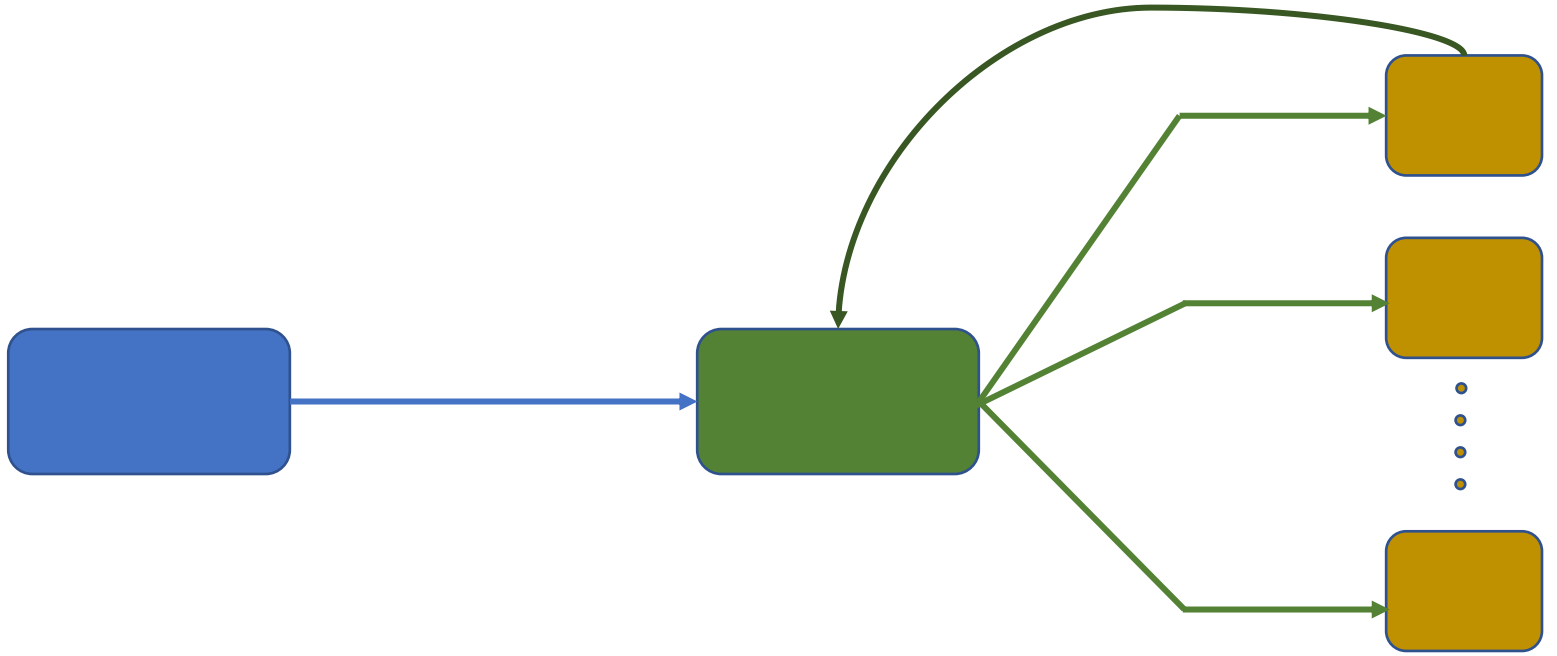
Events as Source of Truth

- With traditional systems, the database holds the operational data and the events trigger actions in the system



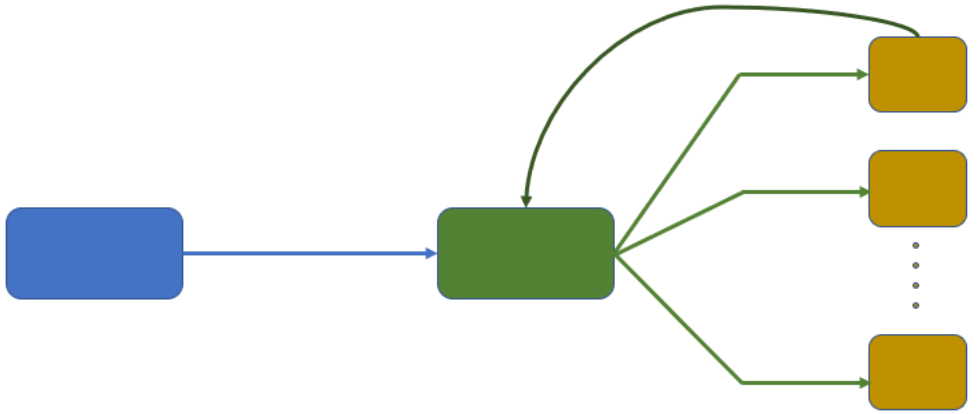
Events as Source of Truth

- With channels that retain events, the channel can function as the source of truth

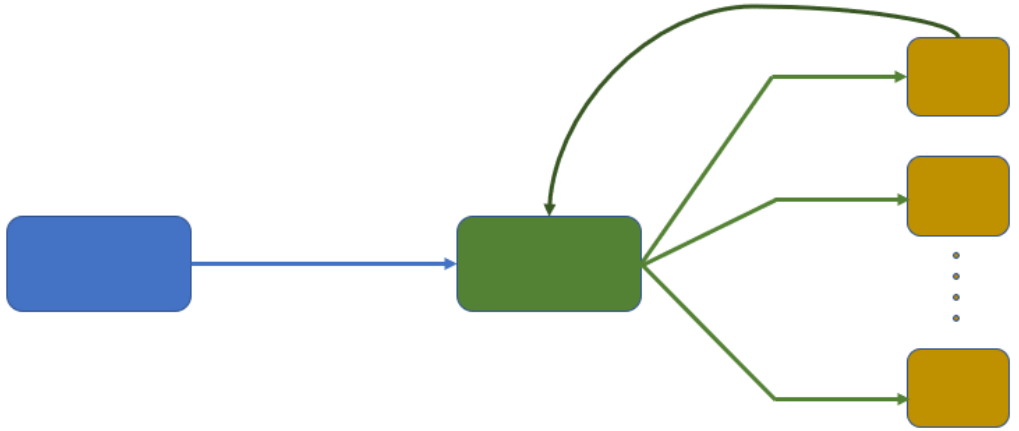


Events as Source of Truth

- Relevant only if:
  - Events are retained
  - The channel has some kind of query language
  - Main functionality of the system is around event streaming, no complex interactions



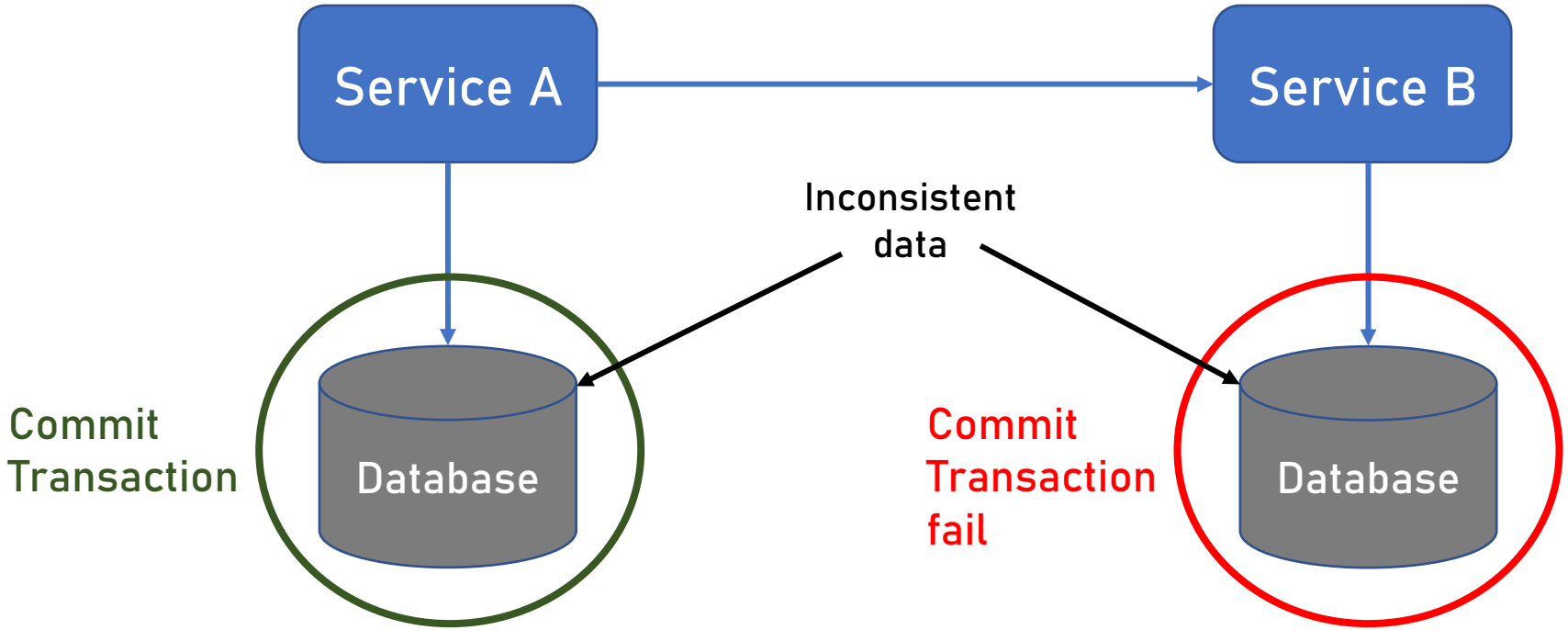
Implementing Events as Source of Truth



- Events are retained
- Has the KQL query language
- Designed for event streaming

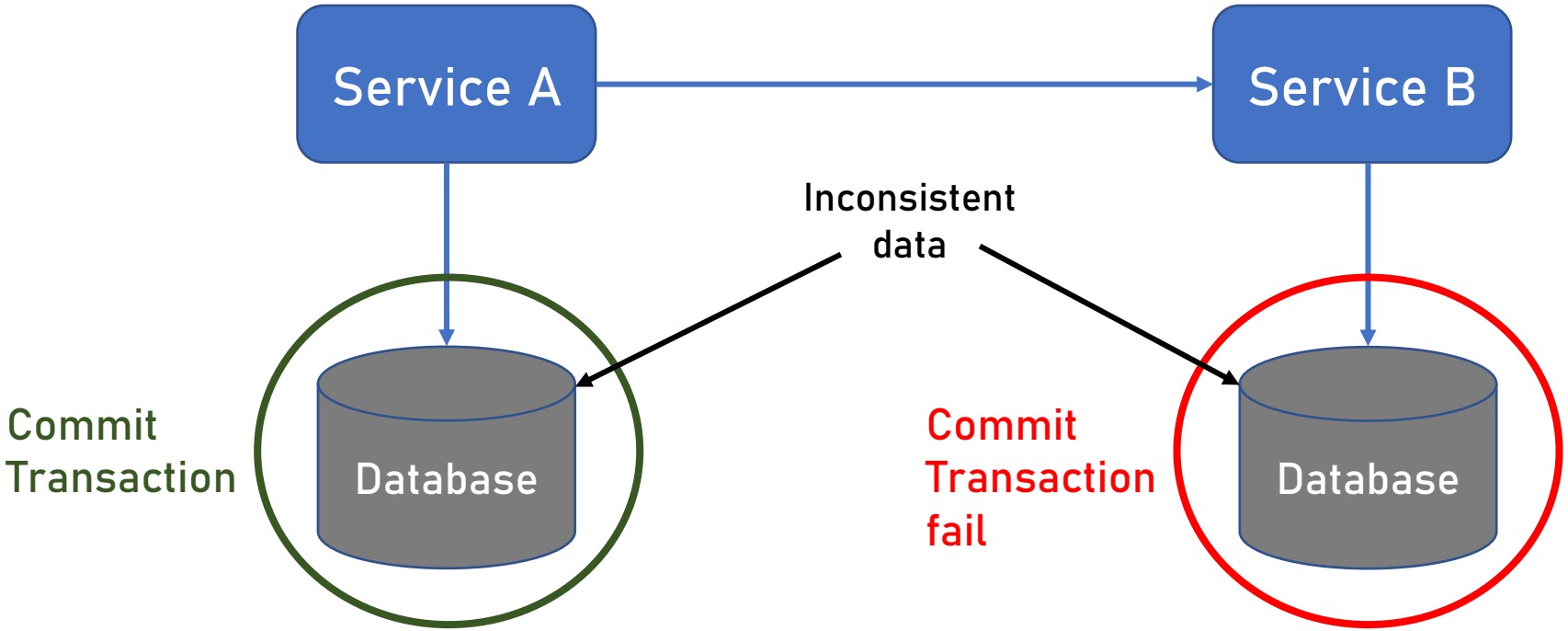
The Saga Pattern

- Transaction management in distributed system is difficult



The Saga Pattern

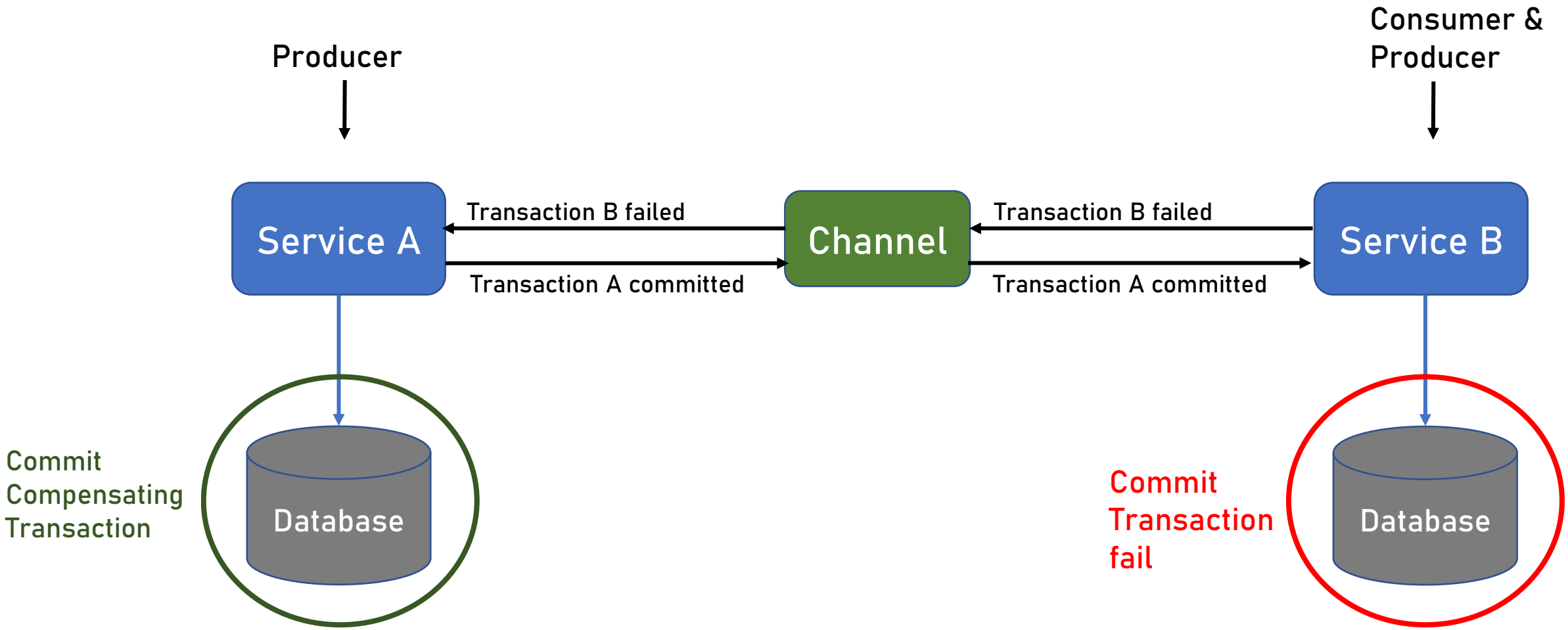
- No accepted, widely used solution
- 2-phase commit protocol exists, but difficult to implement



The Saga Pattern

- The Saga pattern strive to solve this problem
- A sequence of service-scoped transaction, triggered by events
- When a transaction fails, a compensating transaction is triggered

The Saga Pattern



The Saga Pattern

- Not easy to implement
- Consistency is not ensured – compensating transaction can fail
- Quite difficult to debug
- Monitoring is important

The Saga Pattern

- When to use?
  - No need to strongly-coupled transactions
  - Compensating transactions can be defined

EDA on the Front End

- So far we discussed Event Driven Architecture in the back end
- It can also be implemented on the front end
- We won't deep-dive into it, but raise awareness to it

EDA on the Front End

- Two main aspects of EDA on the front end:

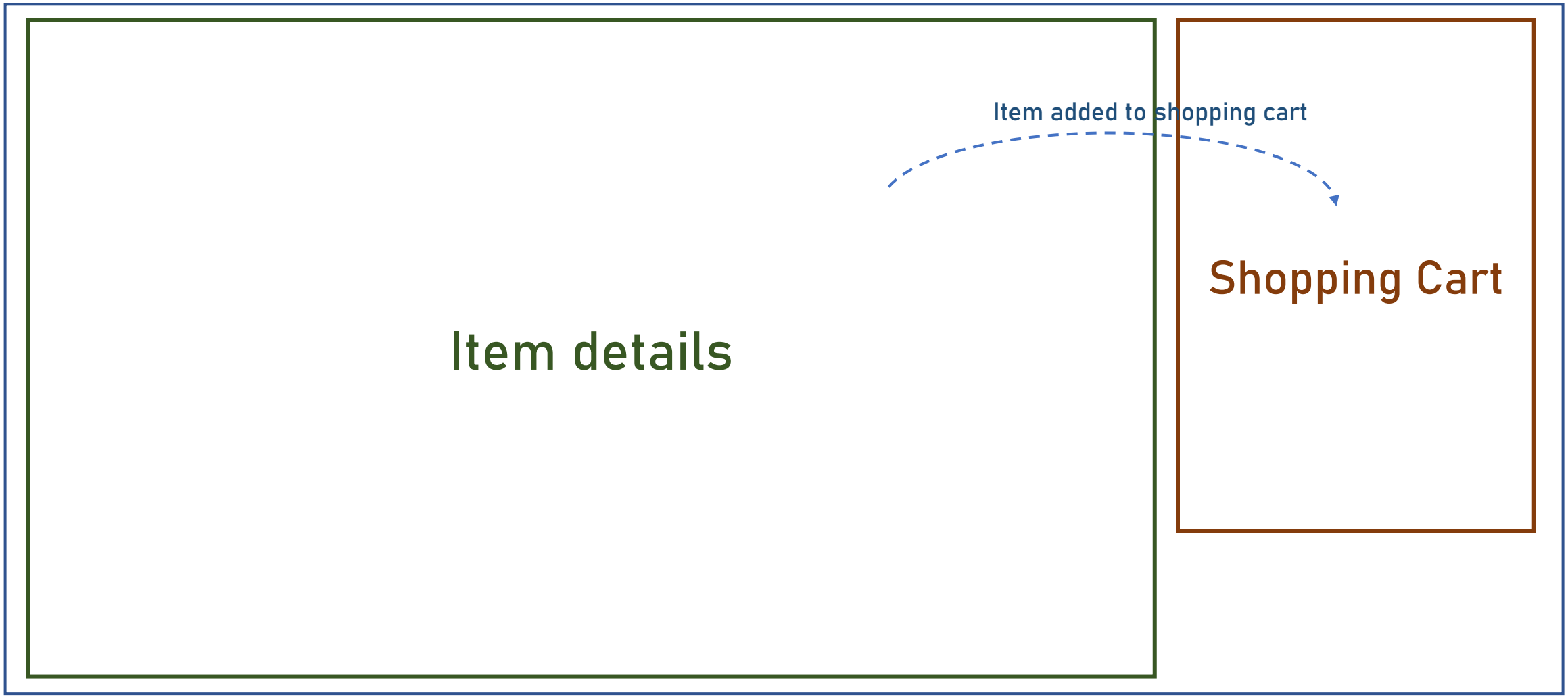
Micro Frontends

Push Notifications

Micro Frontends

- Brings the microservices concept to the front end
- The page is divided to separate, independent UI component
- Each component has its own UI, functionality, backend etc.
- Components communicate with each other using events

Micro Frontends



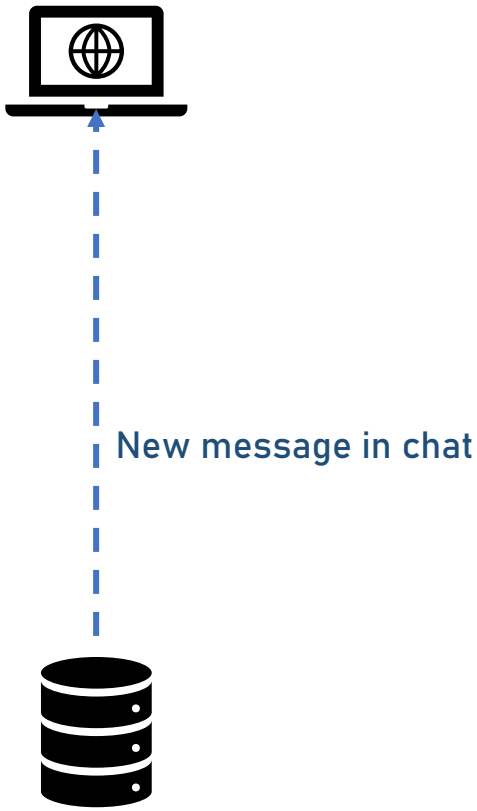
Implementing Micro Frontends

- The components themselves are platform agnostic and can use any frontend framework (or vanilla JS)
- The inter-component communication is done using Browser Events and Custom Elements, supported by all major browsers

Push Notifications

- Events that are sent to the browser from the server
- As opposed to the more traditional Request\Response model
- Not exactly EDA but involves events in the clients
- Very useful for apps require streaming info from the server
  - E.g. Chat

Push Notifications



Implementing Push Notifications

- Quite a lot of libraries and frameworks:
  - SignalR
  - Socket.IO
  - gRPC
  - And more…