ThoughtWorks®

WHEN MICROSERVICES MEET EVENT SOURCING

Vinicius Gomes

VINICIUS GOMES

Software developer at **ThoughtWorks**

vvgomes.com/blog

twitter.com/vvgomes

AGENDA

The Traditional Approach

Introduction to Event Sourcing

CQRS

Demo

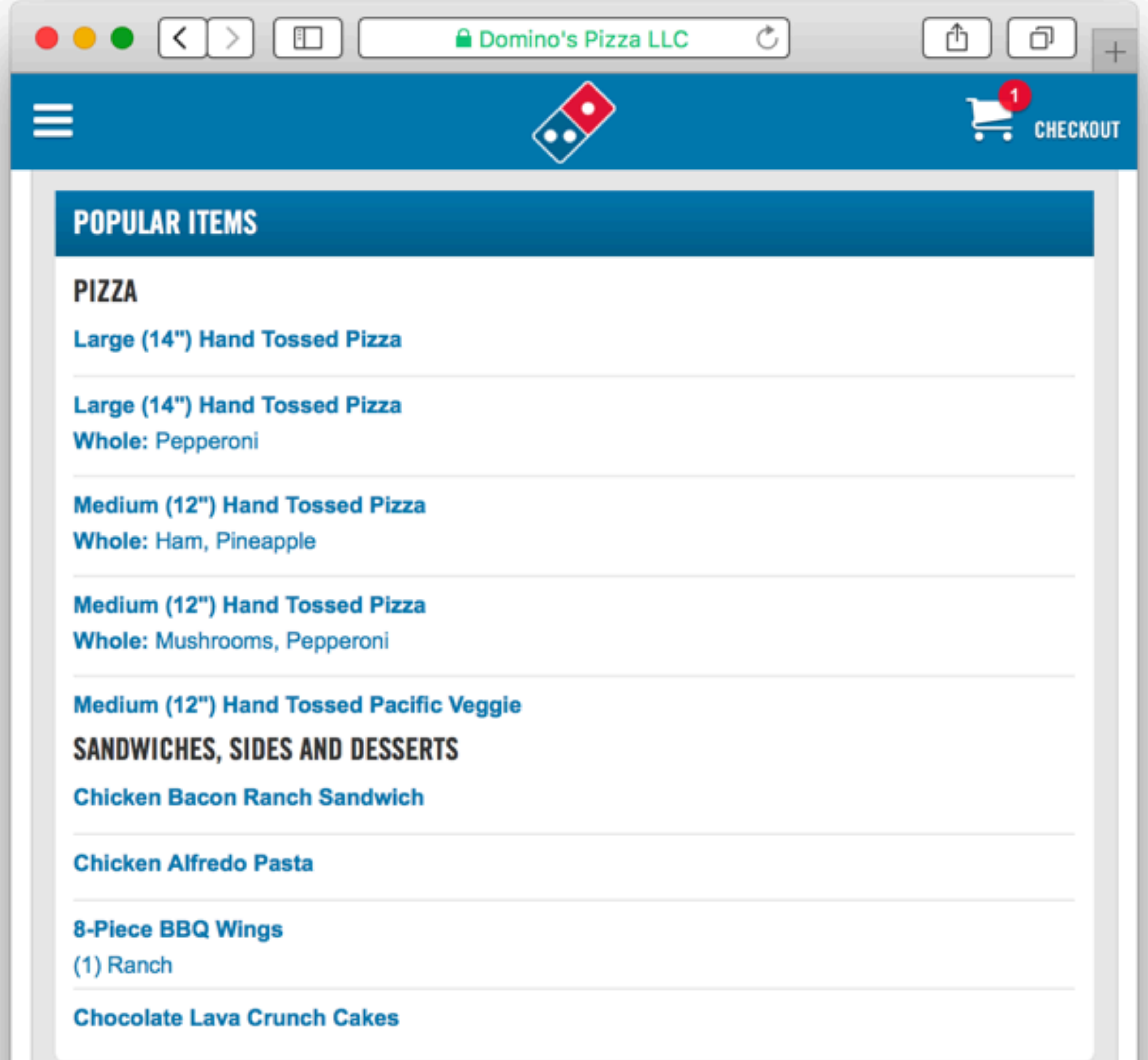
Summary

ThoughtWorks®

THE TRADITIONAL APPROACH

EXAMPLE

Online Restaurant



POPULAR ITEMS

PIZZA

Large (14") Hand Tossed Pizza

Large (14") Hand Tossed Pizza
Whole: Pepperoni

Medium (12") Hand Tossed Pizza
Whole: Ham, Pineapple

Medium (12") Hand Tossed Pizza
Whole: Mushrooms, Pepperoni

Medium (12") Hand Tossed Pacific Veggie

SANDWICHES, SIDES AND DESSERTS

Chicken Bacon Ranch Sandwich

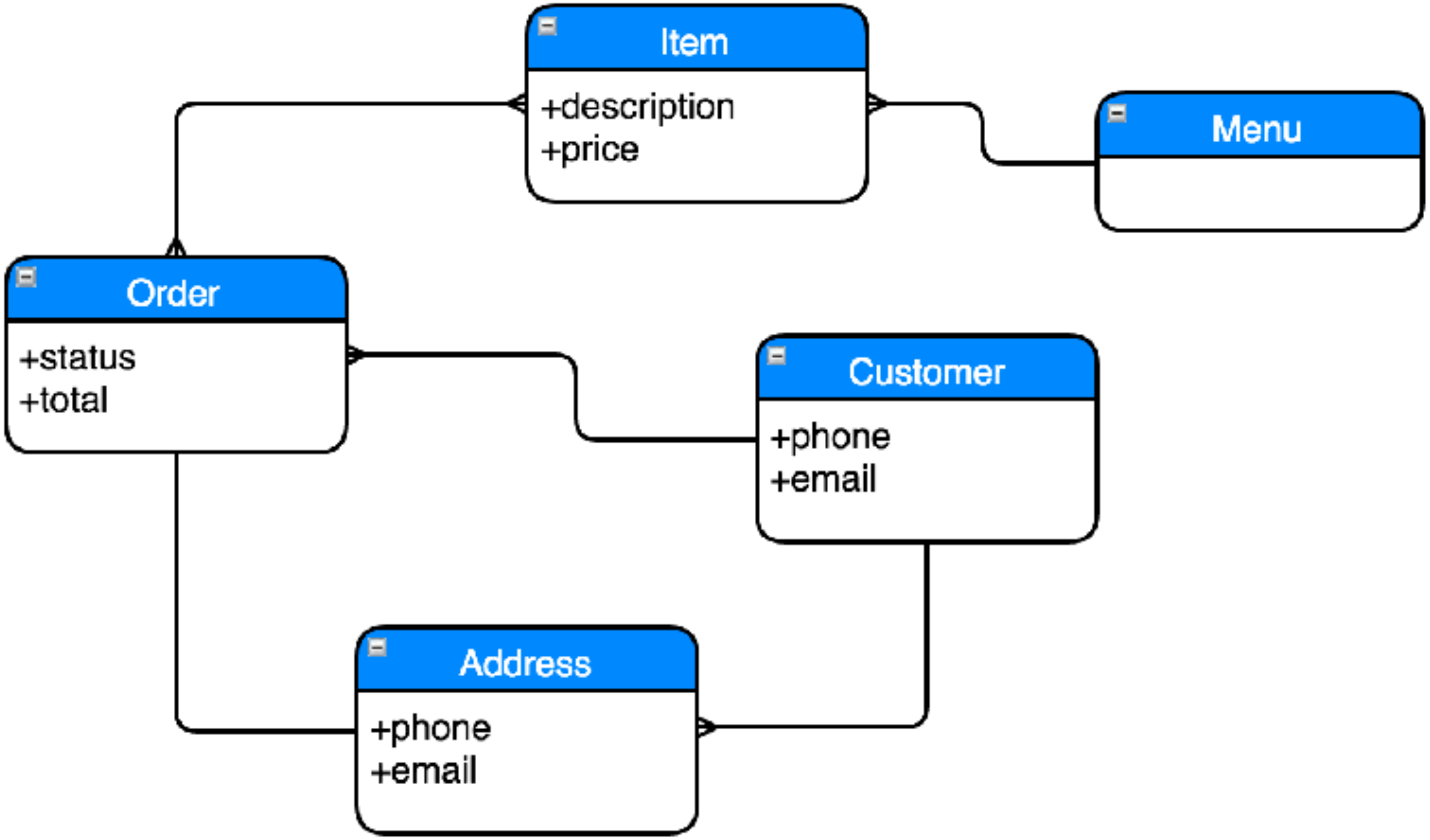
Chicken Alfredo Pasta

8-Piece BBQ Wings
(1) Ranch

Chocolate Lava Crunch Cakes

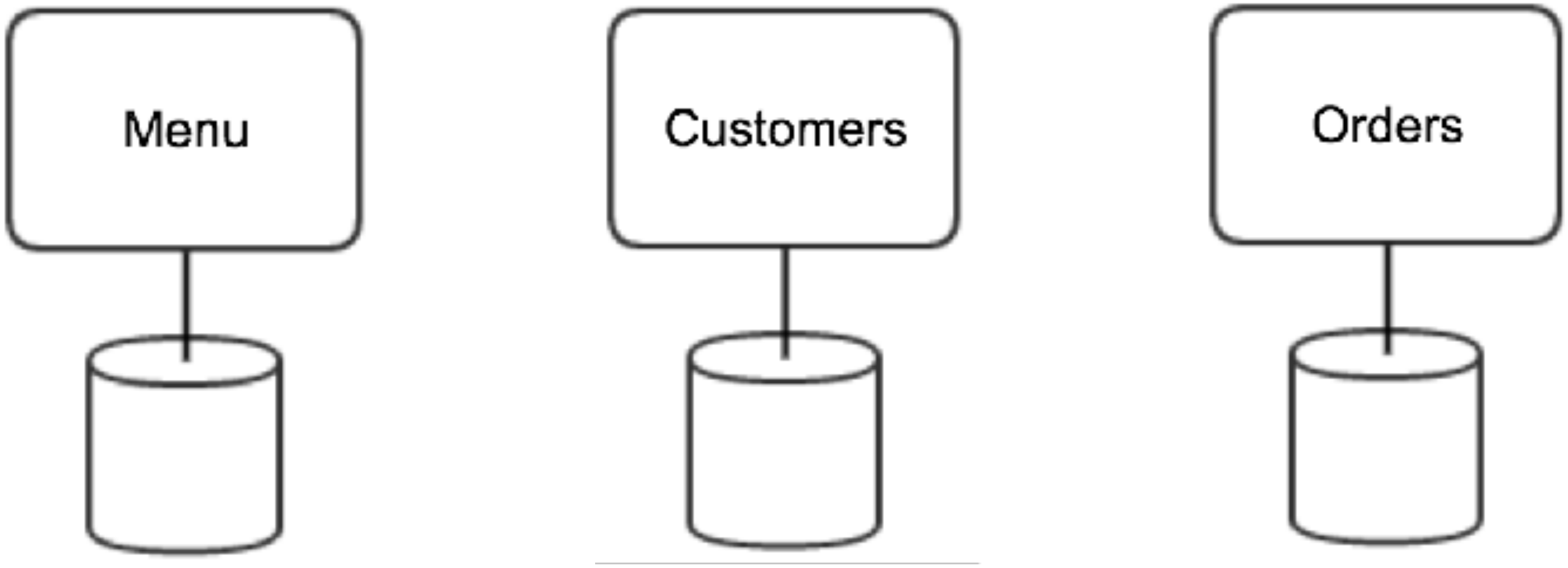
ONLINE RESTAURANT

Data Model

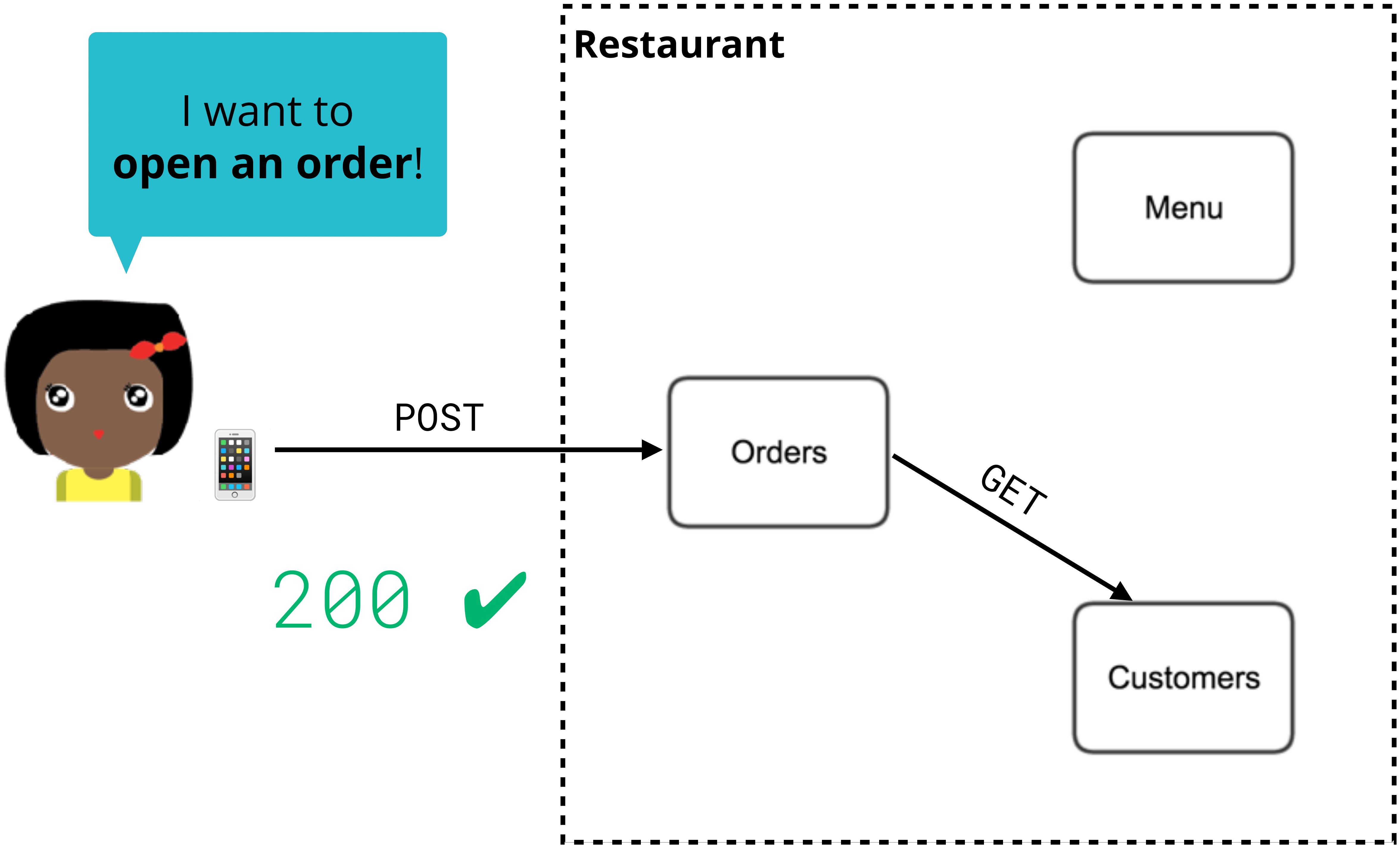


ONLINE RESTAURANT

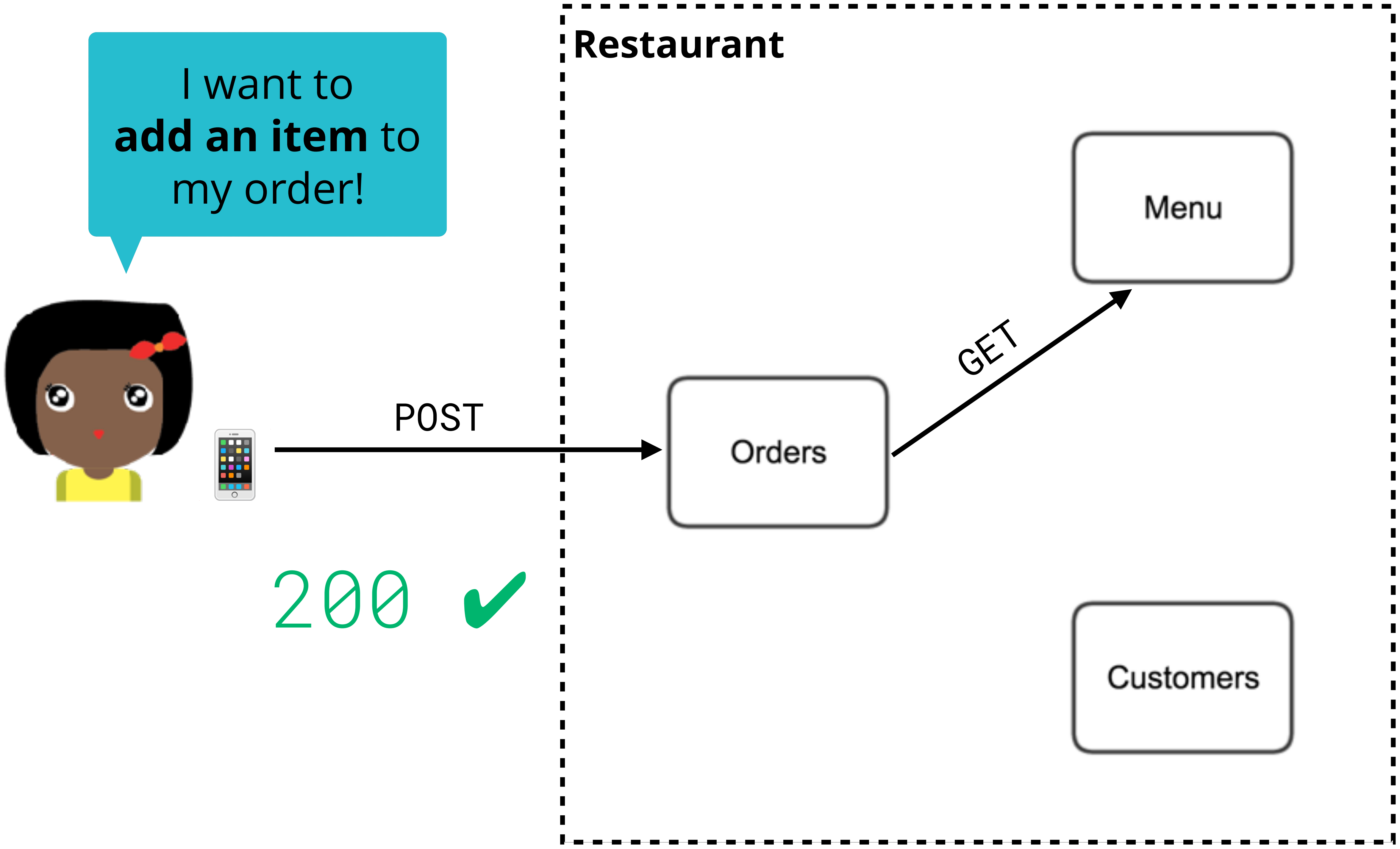
Services



ONLINE RESTAURANT

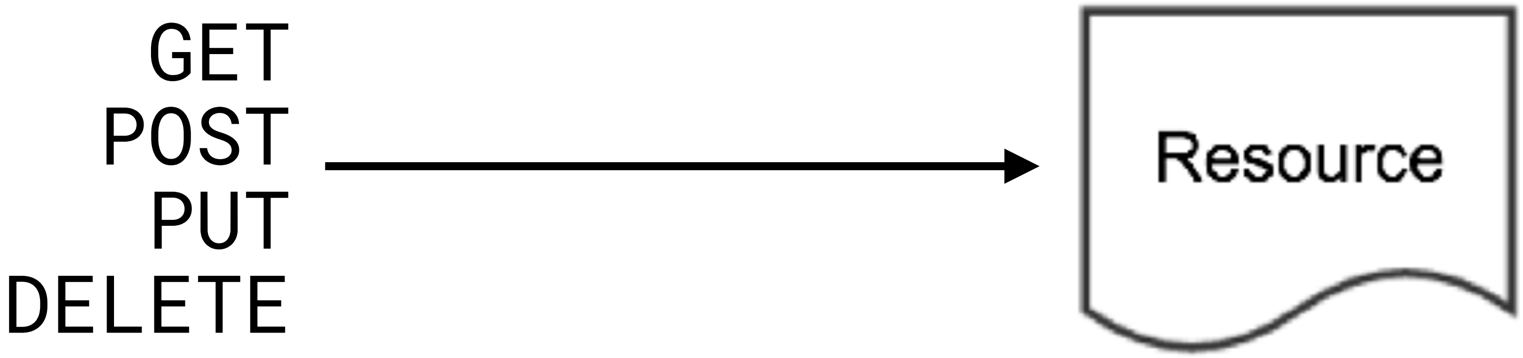


ONLINE RESTAURANT



THE TRADITIONAL APPROACH

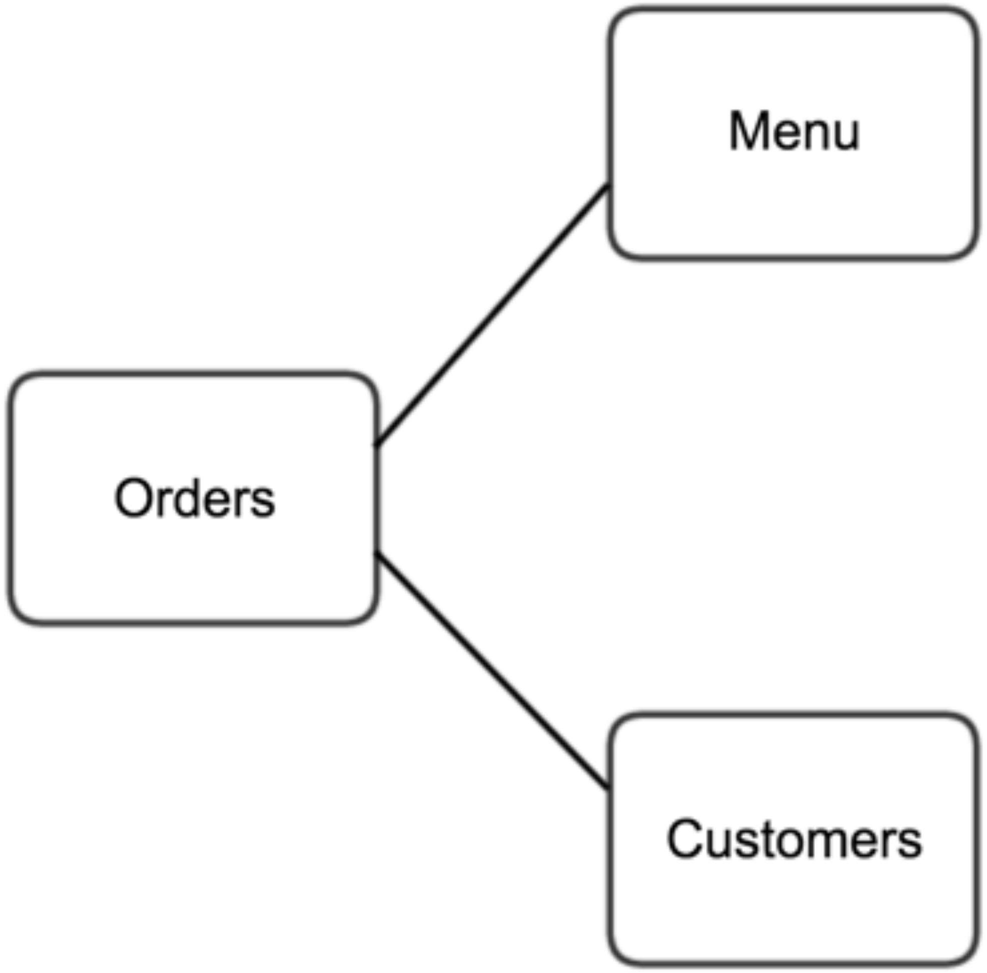
- Simple to implement
- Technology agnostic



CHALLENGES

CHALLENGES

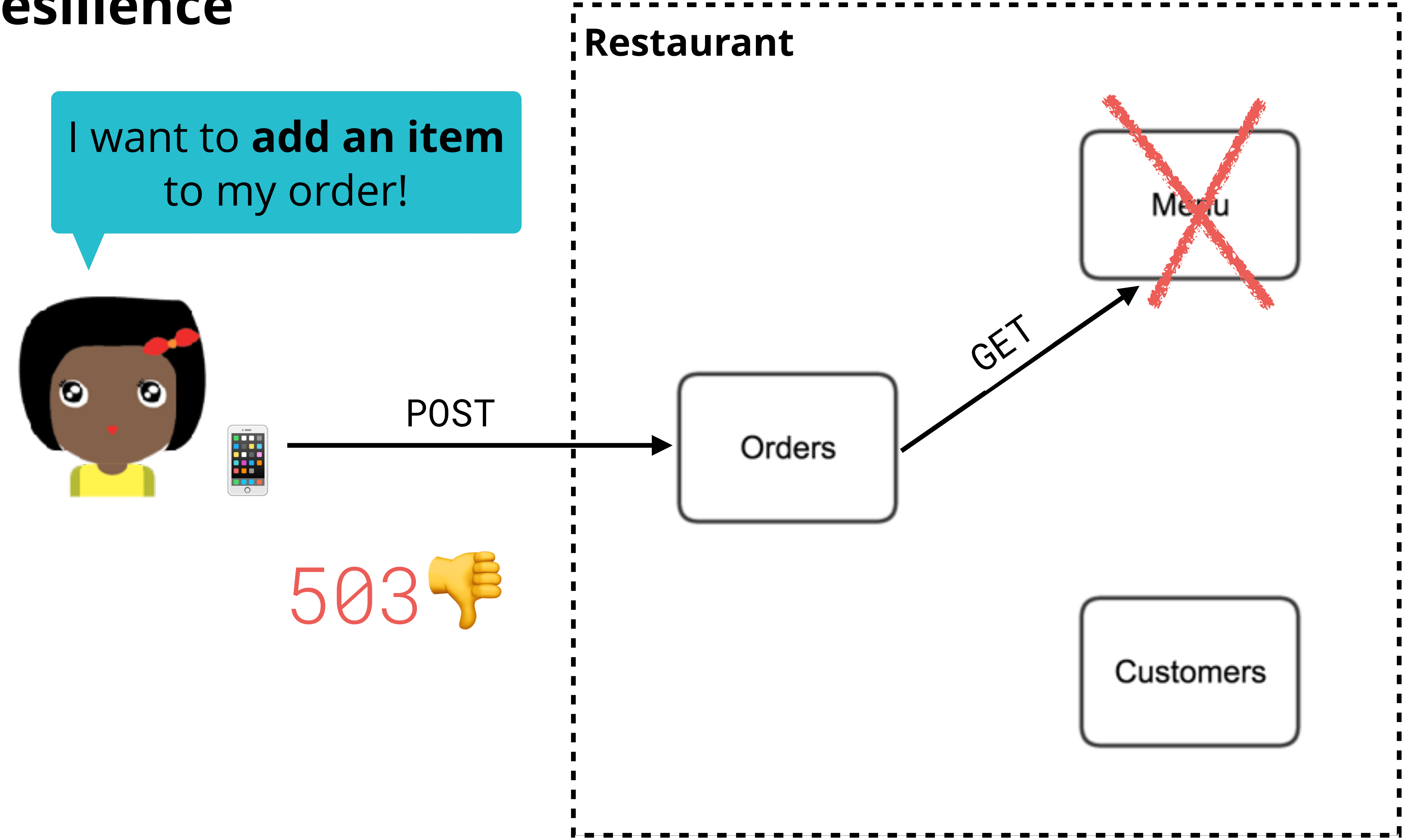
Coupling



CHALLENGES

Coupling

Resilience



I want to **add an item** to my order!

POST

503 👎

Restaurant

Orders

GET

Menu

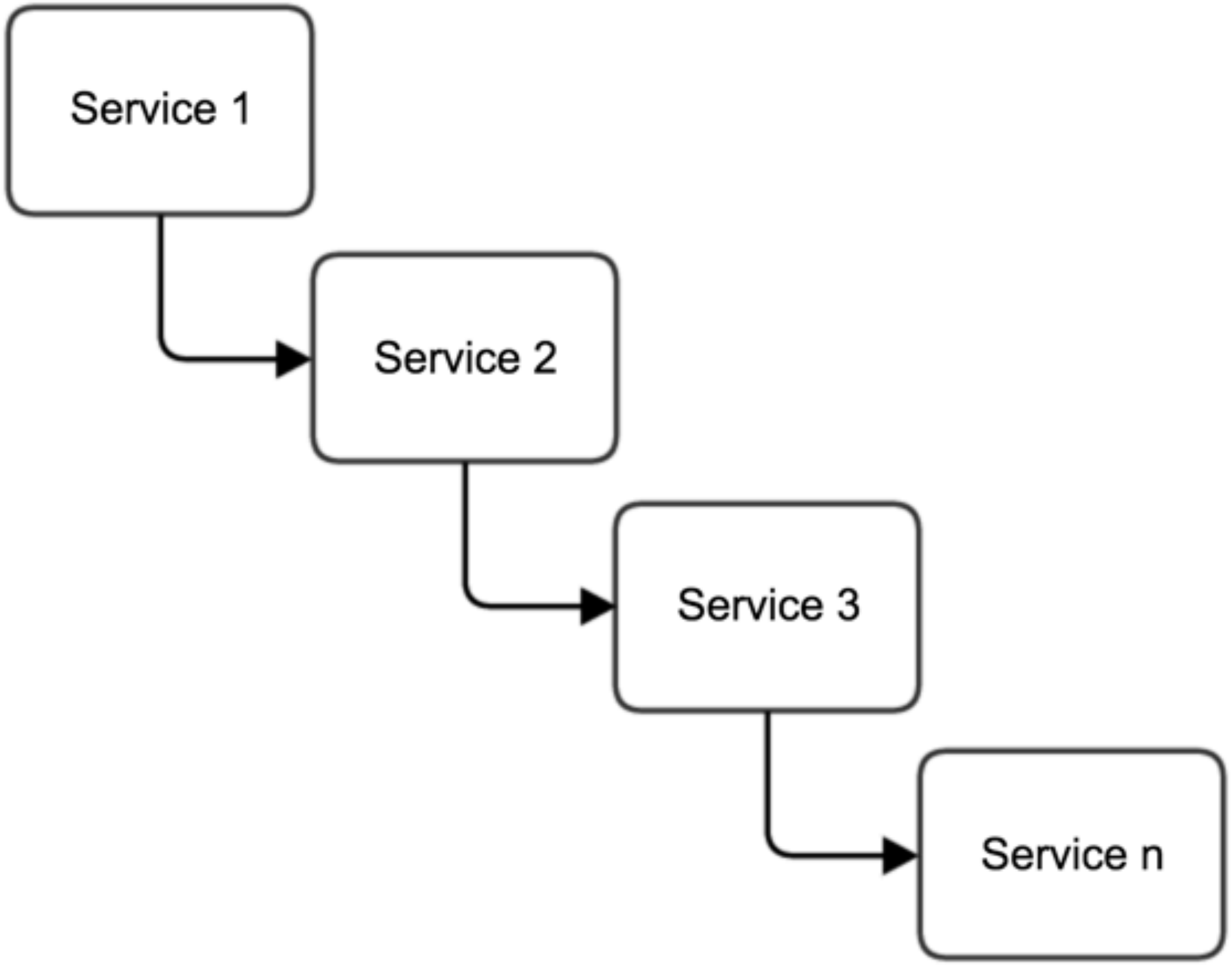
Customers

CHALLENGES

Coupling

Resilience

Response time



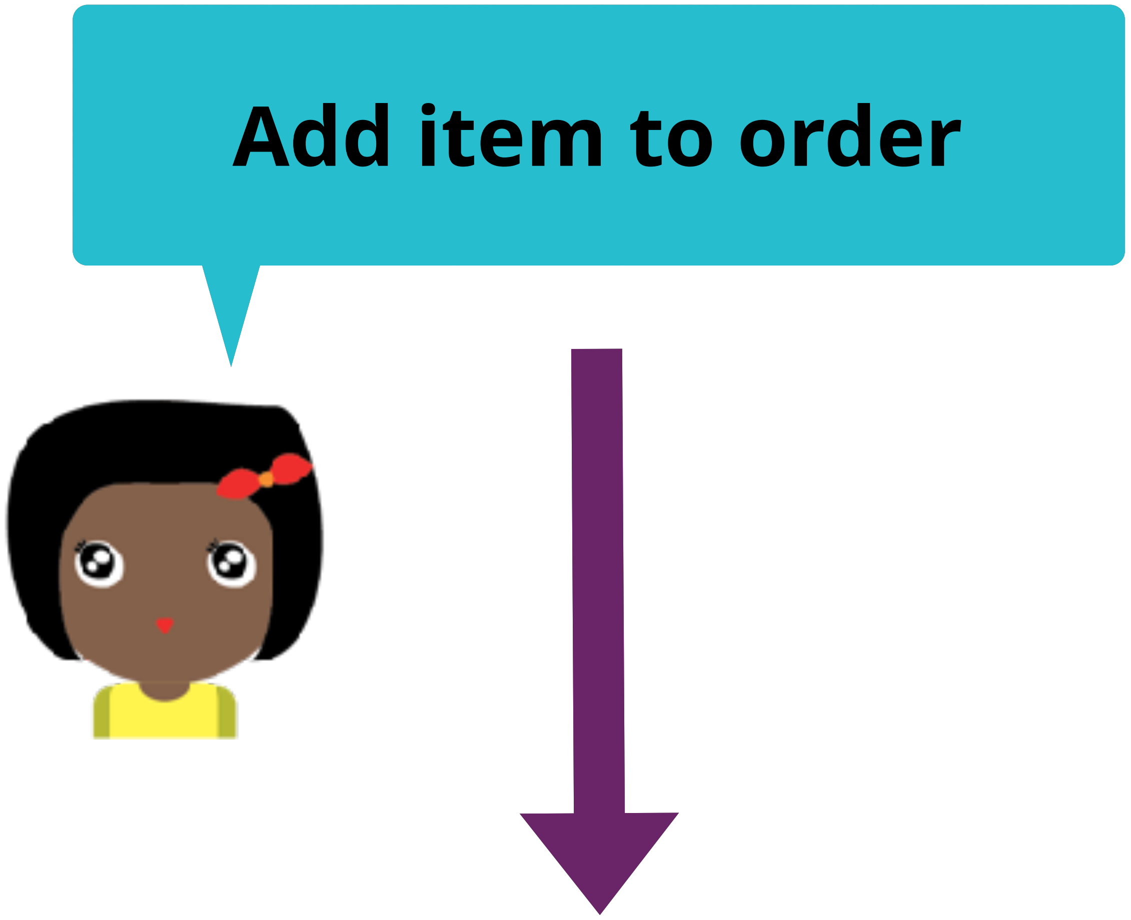
CHALLENGES

Coupling

Resilience

Response time

User intent



Add item to order

```
POST /orders/8659a0d6/items/
```

CHALLENGES

Coupling

Resilience

Response time

User intent

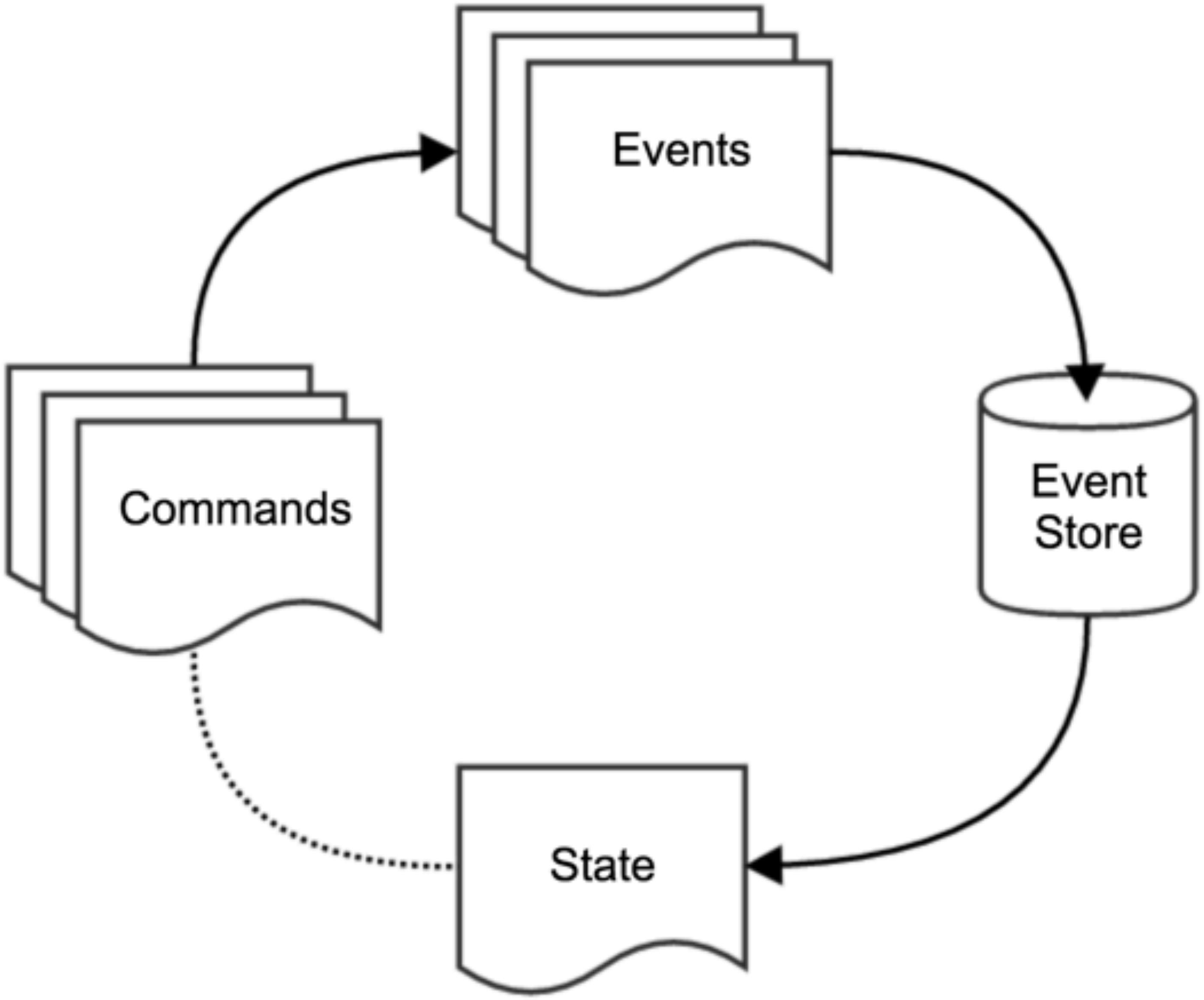
Current / mutable state

```
{
  "id": “6a208c41…”,
  "status": "OPEN",
  "items": [
    {
      "id": “be596c8e…”,
      "quantity": 1
    },
    {
      "id": “5d09509c…”,
      "quantity": 2
    },
    {
      "id": “cc52d1b6…”,
      "quantity": 1
    }
  ]
}
```

ThoughtWorks®

EVENT SOURCING

EVENT SOURCING



DOMAIN EVENTS

Model the past

Immutable

Permanent

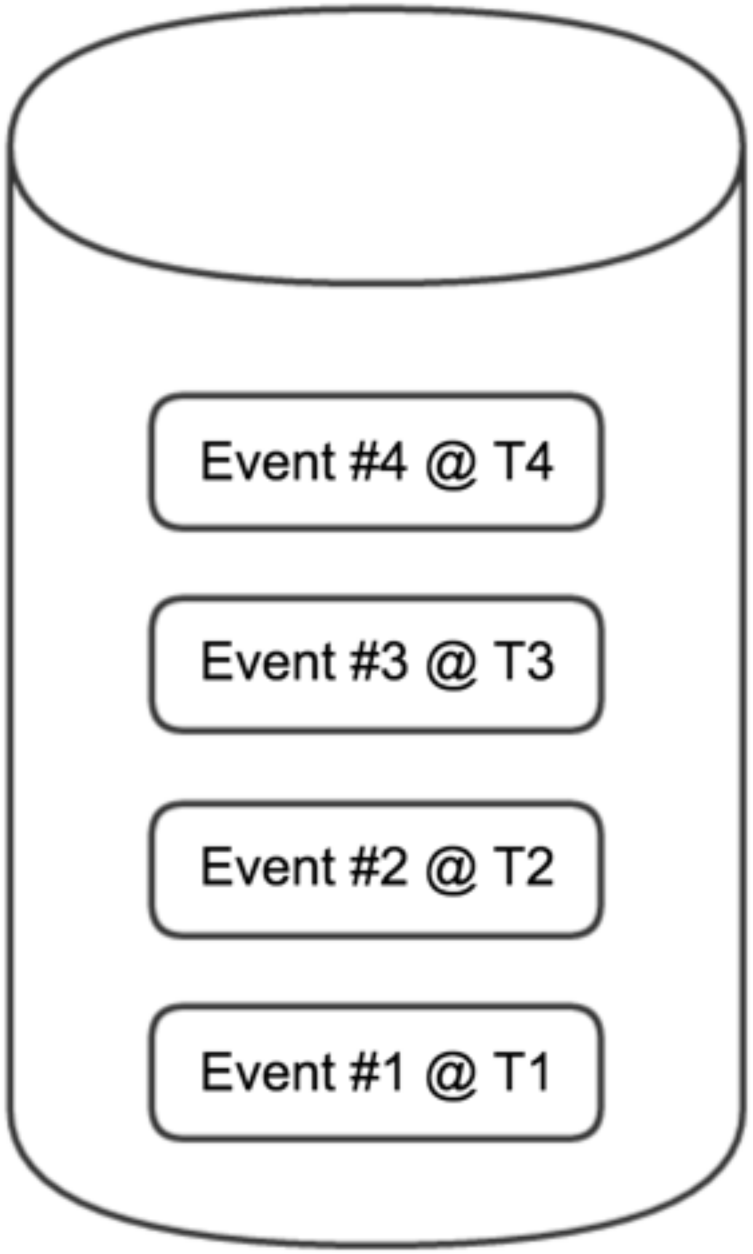
```
{
  "payload": {
    "orderId": “be596c8e…”,
    "itemId": “5d09509c…”,
    "quantity": 1
  },
  "payloadType": “com.restaurant.ItemAddedToOrderEvent”,
  "timestamp": “2017-03-25 08:48:51 -03:00”,
  "revision": “2”,
  "aggregateId": “6a208c41…”
}
```

EVENT STORE

Append only

Sequential

Long-lived



AGGREGATES

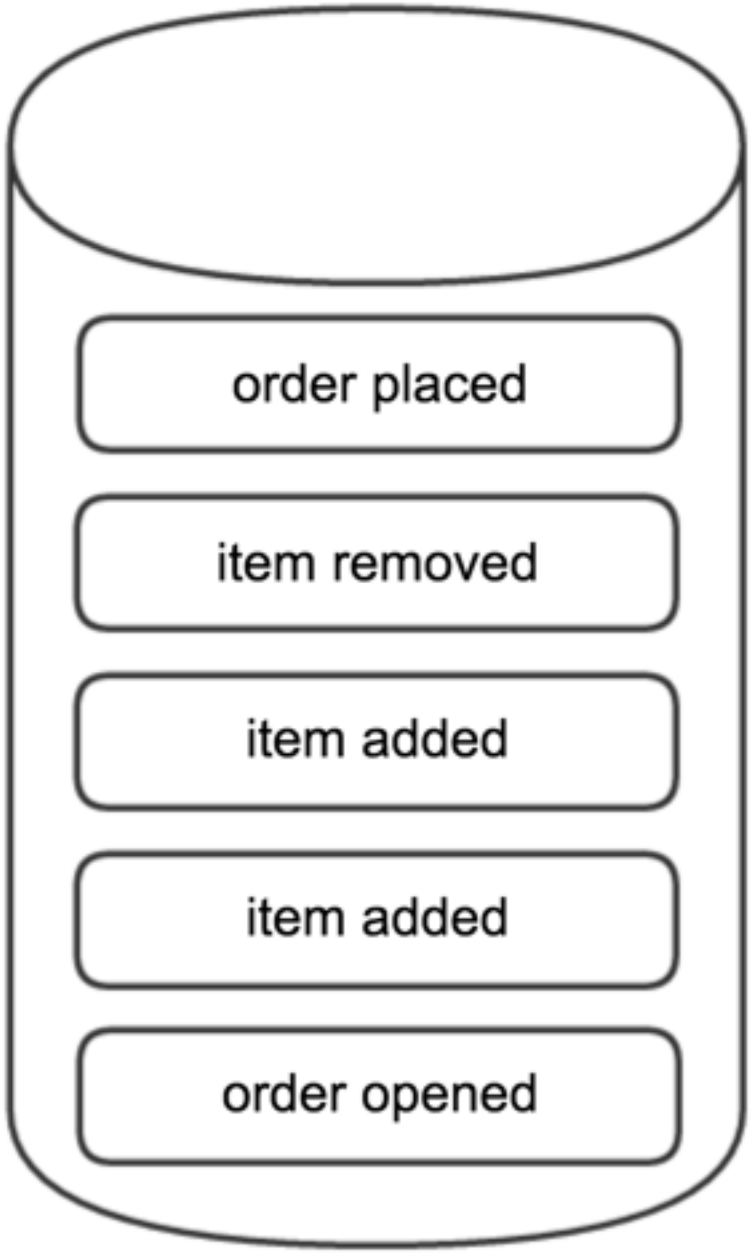
Reducing events into state

```
state = reduce(handle, {}, events)
```

Reducer function

AGGREGATES

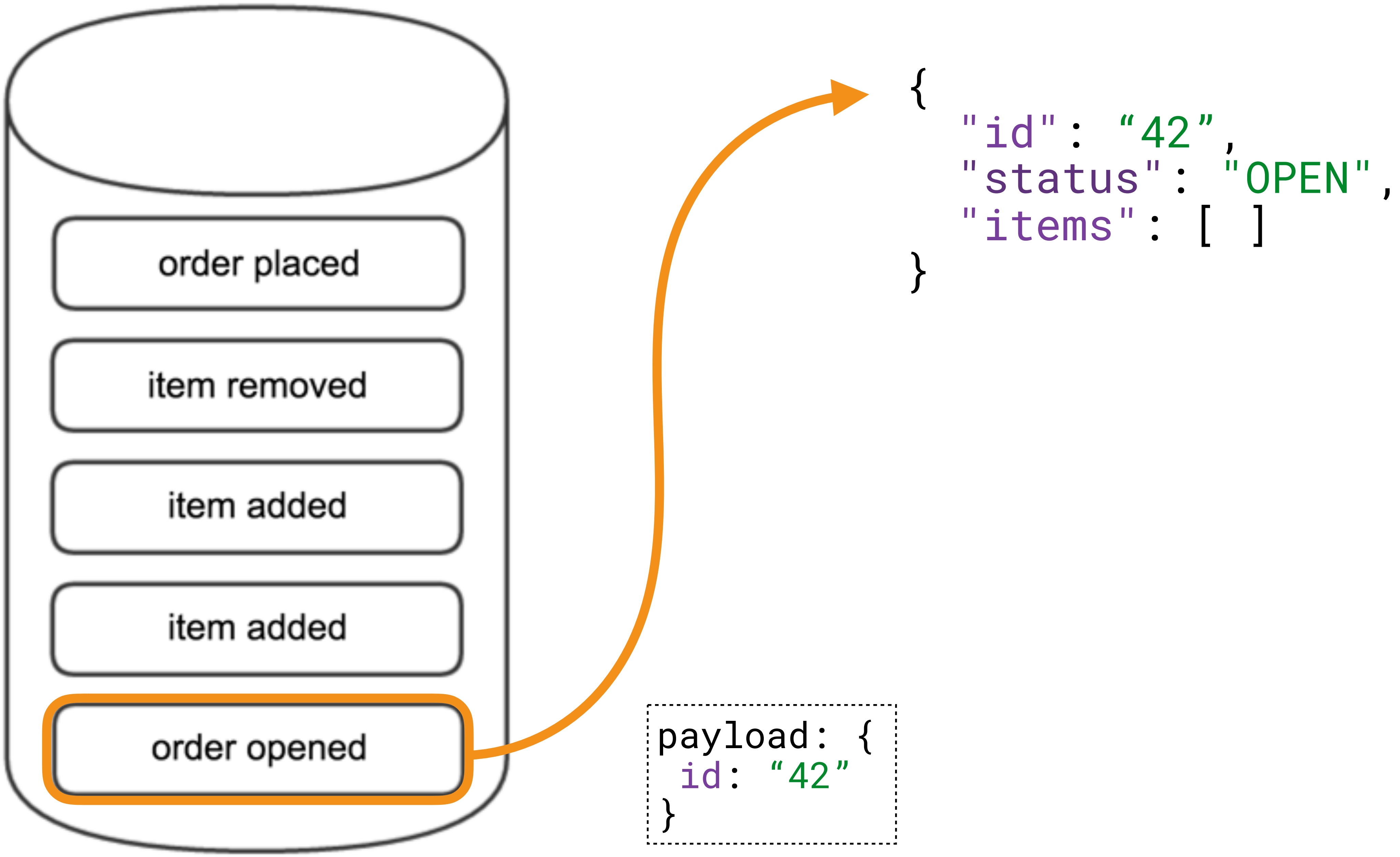
Example



{ }

AGGREGATES

Example



order placed

item removed

item added

item added

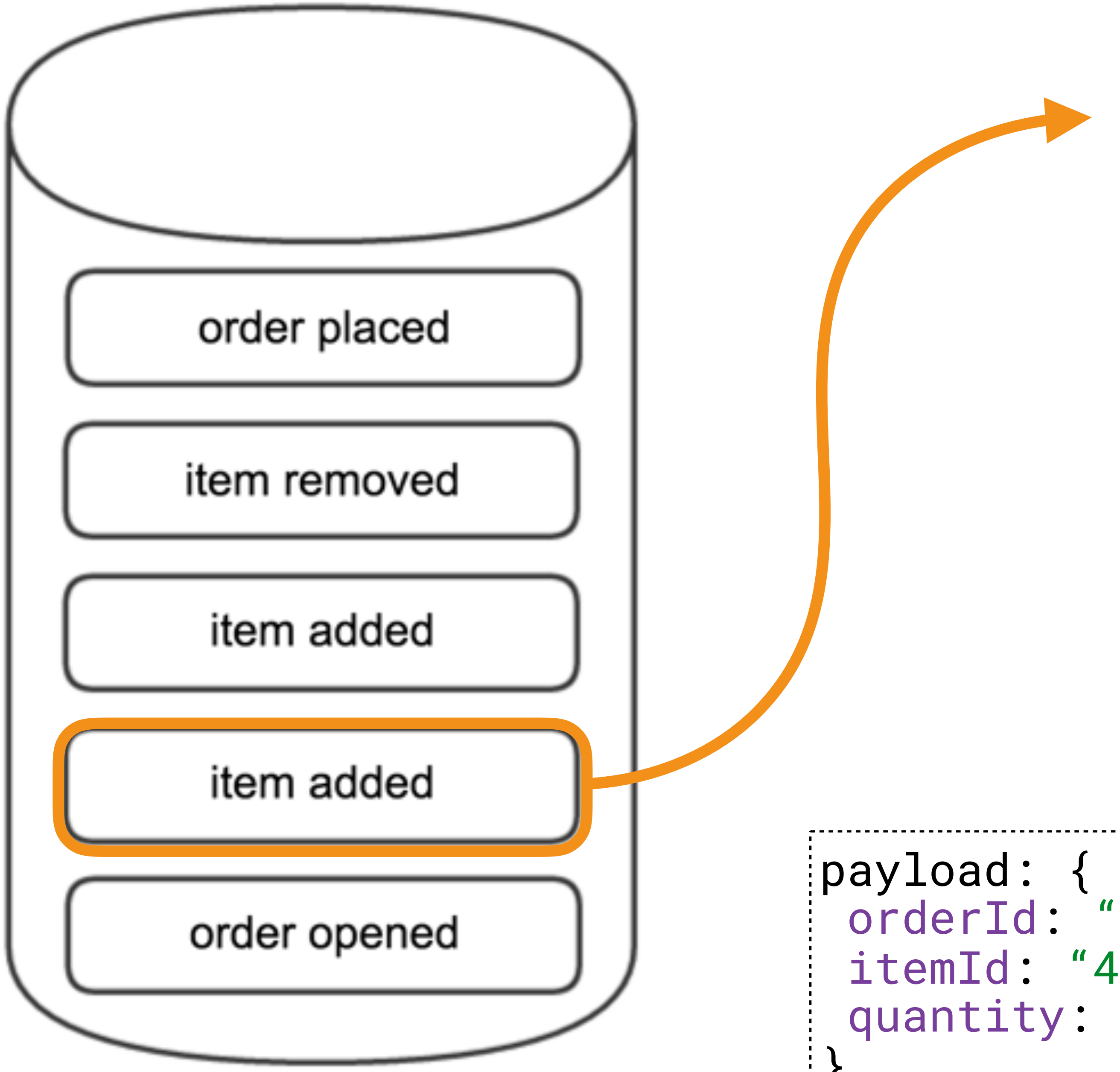
order opened

```
payload: {
 id: “42”
}
```

```
{
   "id": “42”,
   "status": "OPEN",
   "items": [ ]
}
```

AGGREGATES

Example

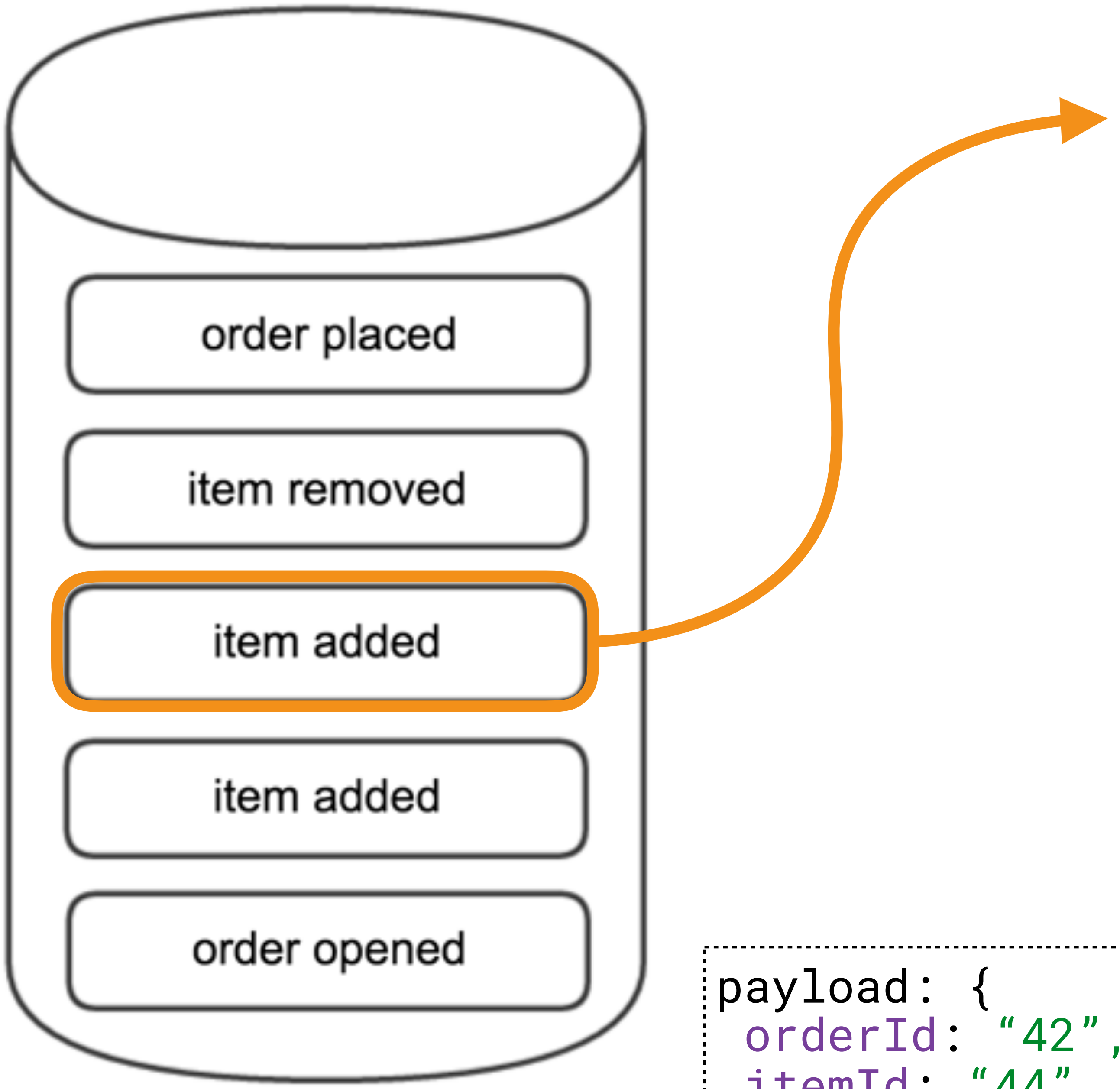


```
{
  "id": “42”,
  "status": "OPEN",
  "items": [
    {
      "id": “43”,
      “quantity": 1
    }
  ]
}
```

```
payload: {
 orderId: “42”,
 itemId: “43”,
 quantity: 1
}
```

AGGREGATES

Example

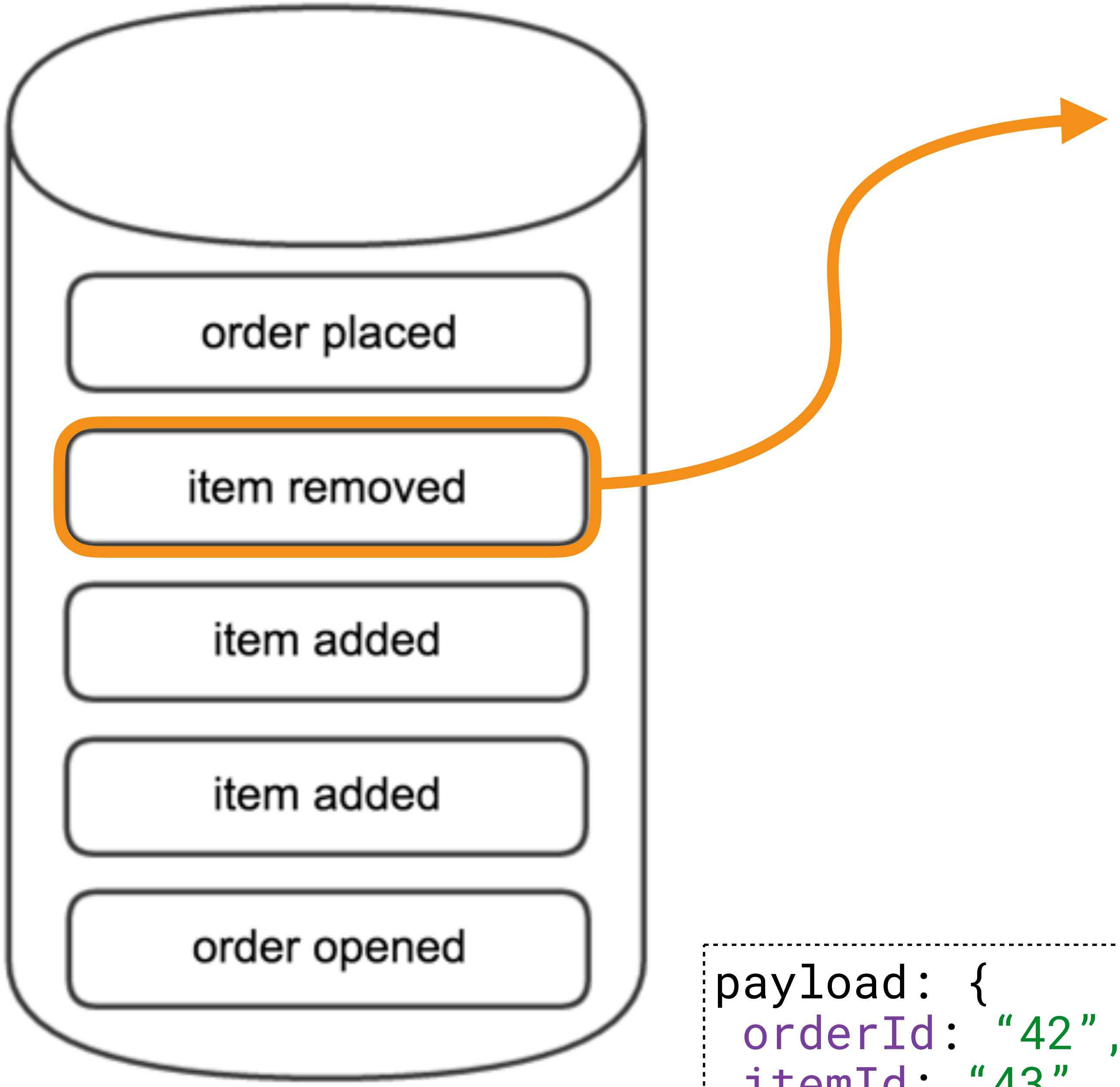


```
{
  "id": “42”,
  "status": "OPEN",
  "items": [
    {
      "id": “43”,
      “quantity": 1
    },
    {
      "id": “44”,
      “quantity": 2
    }
  ]
}
```

```
payload: {
 orderId: “42”,
 itemId: “44”,
 quantity: 2
}
```

AGGREGATES

Example

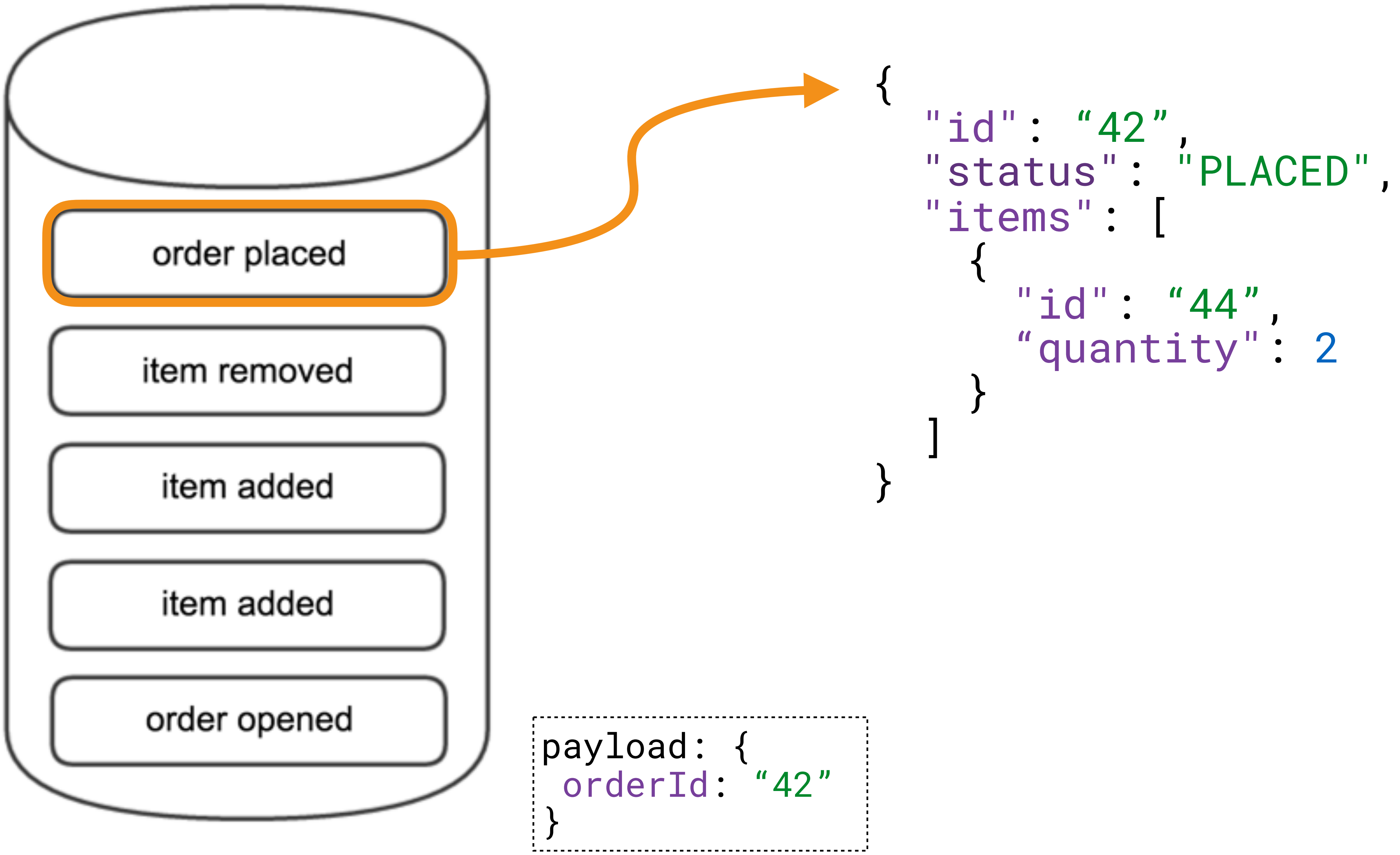


```
{
  "id": “42”,
  "status": "OPEN",
  "items": [
    {
      "id": “44”,
      “quantity": 2
    }
  ]
}
```

```
payload: {
 orderId: “42”,
 itemId: “43”
}
```

AGGREGATES

Example

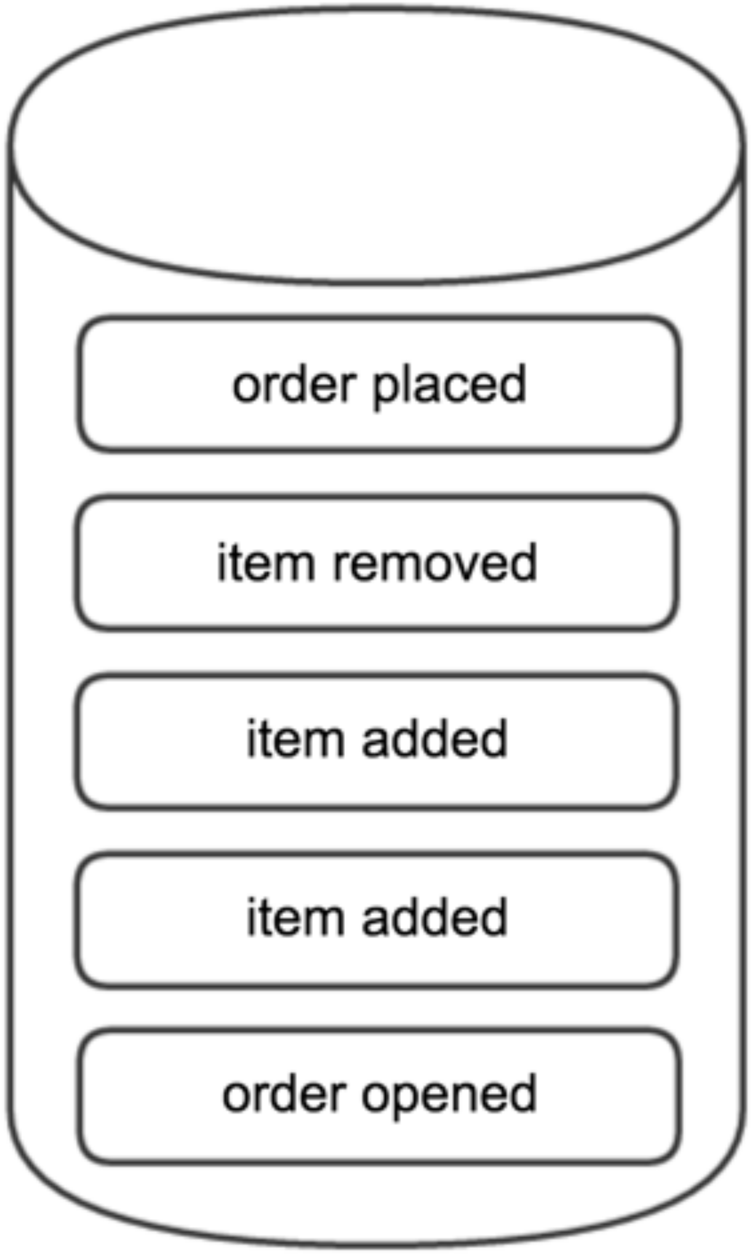


```
{
  "id": “42”,
  "status": "PLACED",
  "items": [
    {
      "id": “44”,
      “quantity": 2
    }
  ]
}
```

```
payload: {
 orderId: “42”
}
```

AGGREGATES

Example



order placed

item removed

item added

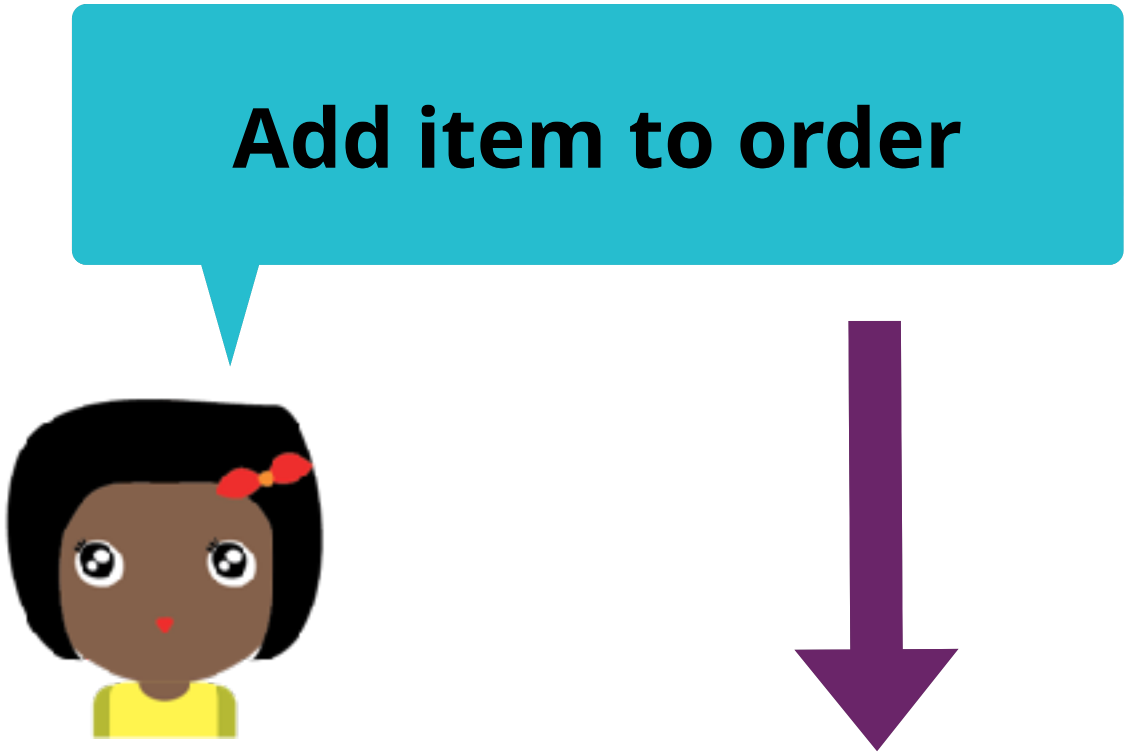
item added

order opened

Order Aggregate

```
{
  "id": “42”,
  "status": "PLACED",
  "items": [
    {
      "id": “44”,
      “quantity": 2
    }
  ]
}
```

COMMANDS



Add item to order

```
{
  "type": “com.restaurant.AddItemToOrderCommand”,
  "payload": {
    "orderId": “be596c8e…”,
    "itemId": “5d09509c…”,
    "quantity": 1
  },
  "timestamp": “2017-03-25 08:48:51 -03:00”
}
```

COMMANDS

Handling a command

```
(state, command) -> [event]
```

COMMANDS

How do clients send commands?

I want to **add an item** to my order!

?

COMMANDS

Reviewing REST

Resource Collection

```
host/api/orders
```

COMMANDS

Reviewing REST

Resource

```
host/api/orders/8659a0d6
```

COMMANDS

Reviewing REST

Nested Resource

```
host/api/orders/8659a0d6/items/5d1a8457
```

COMMANDS

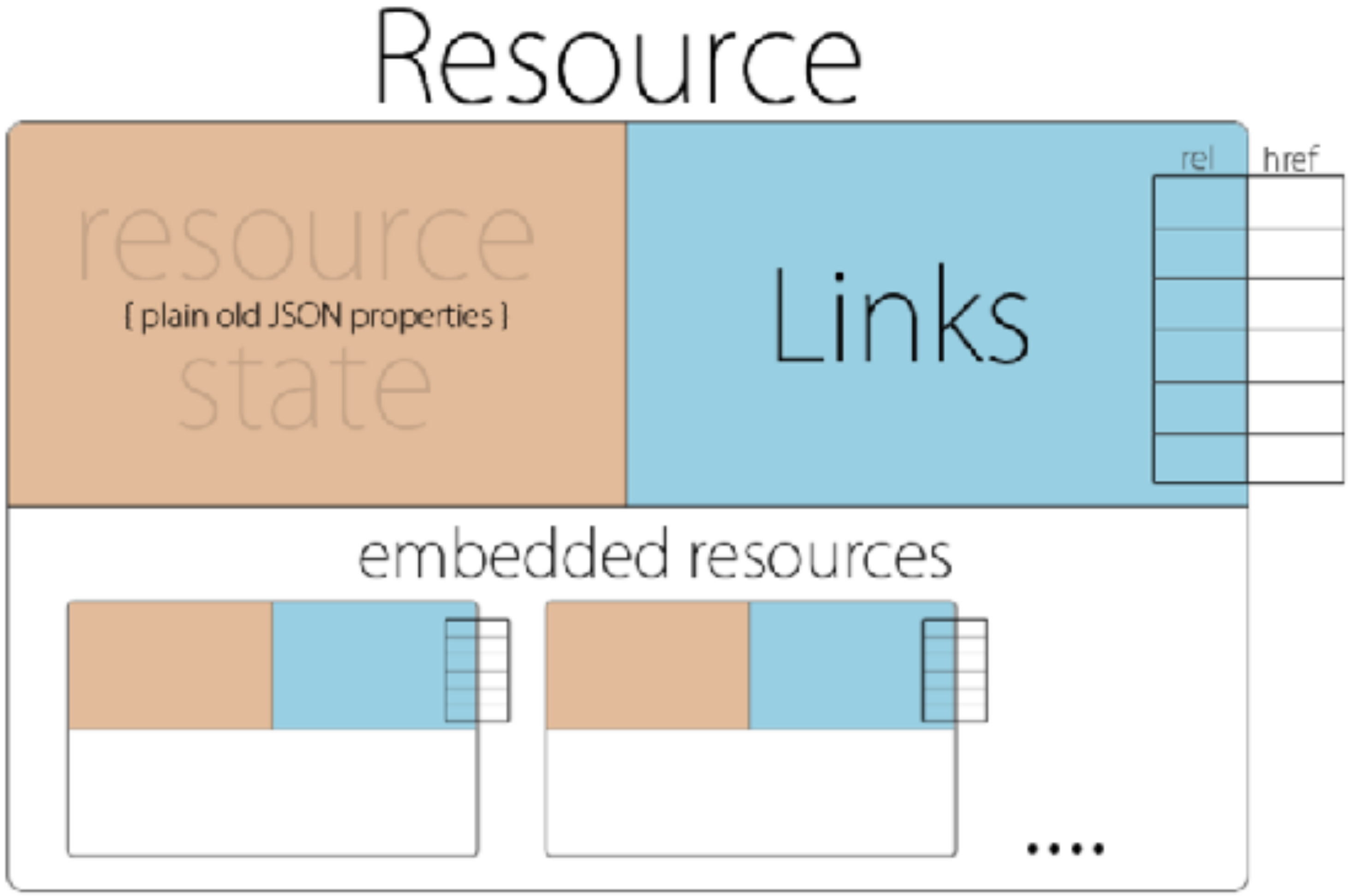
Reviewing REST

HTTP Methods

GET
POST
PUT/PATCH
DELETE

Reviewing REST

Hypermedia Formats



COMMANDS

Commands can be resources

COMMANDS

Option #1

Generic “Commands” nested-resource

```
POST /orders/8659a0d6/commands

  {
    “type": “AddItemToOrderCommand”,
    "menuItem": “/menu/items/8d30d99“,
    “quantity": 2
  }
```

COMMANDS

Option #2

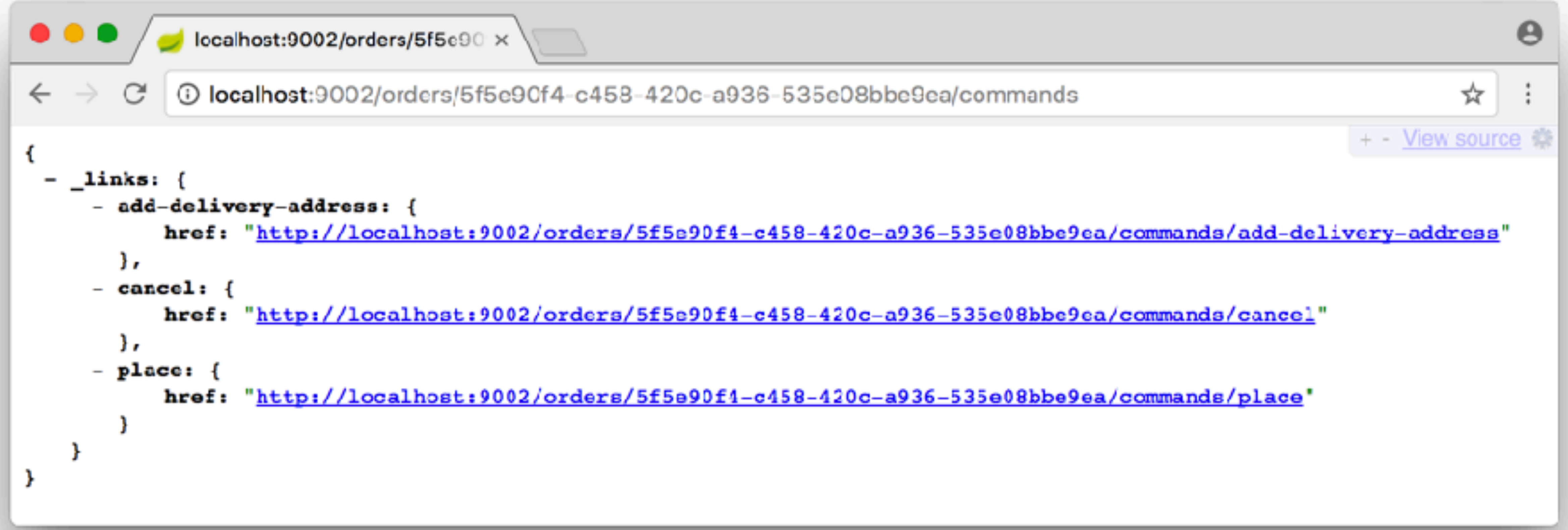
Command URI

```
POST /orders/8659a0d6/items/commands/add

          {
            "menuItem": “/menu/items/8d30d99“,
            “quantity": 2
          }
```

COMMANDS

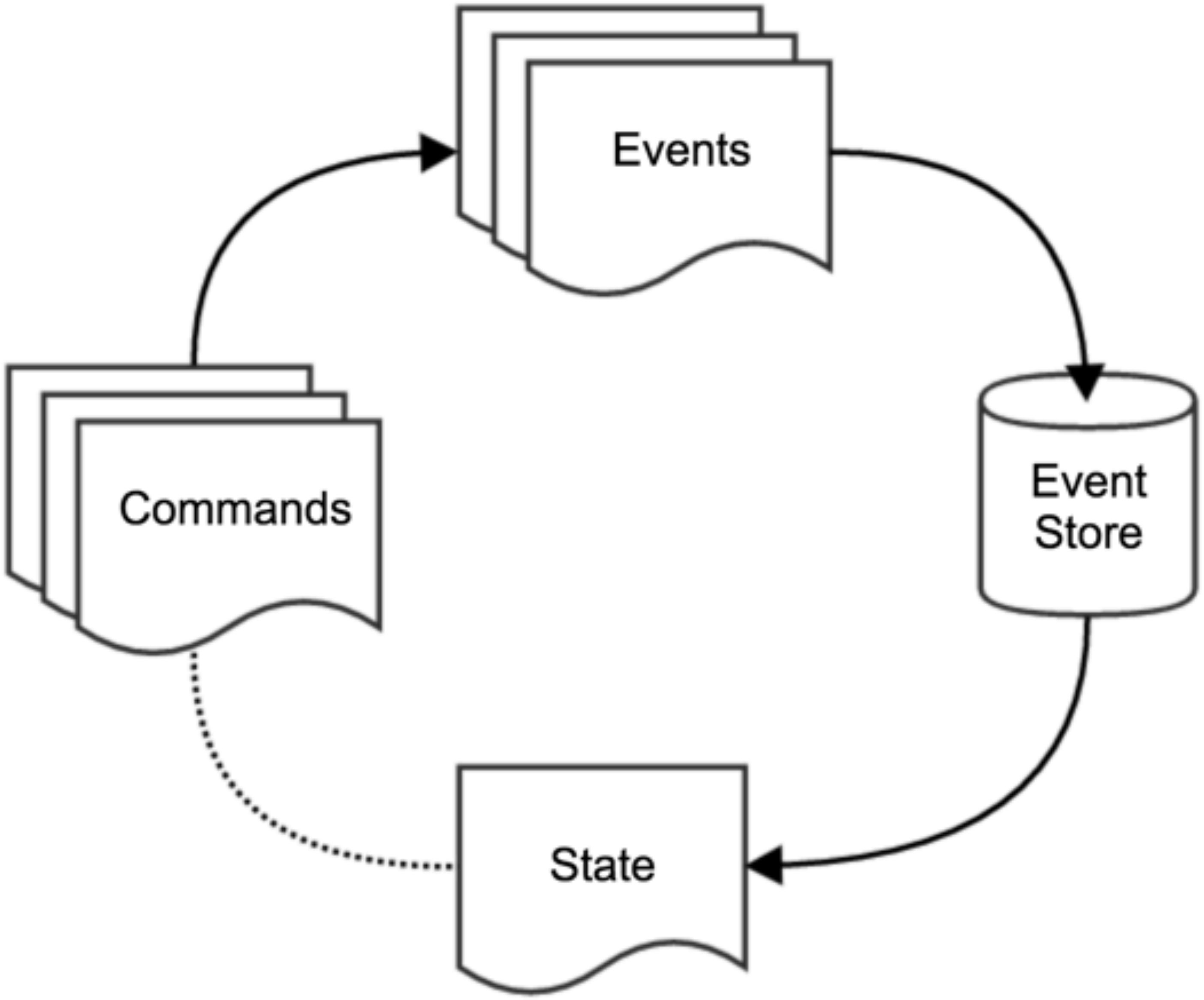
Command discoverability



localhost:9002/orders/5f5e90f4-c458-420c-a936-535e08bbe9ea/commands

```
{
  - _links: {
      - add-delivery-address: {
            href: "http://localhost:9002/orders/5f5e90f4-c458-420c-a936-535e08bbe9ea/commands/add-delivery-address"
        },
      - cancel: {
            href: "http://localhost:9002/orders/5f5e90f4-c458-420c-a936-535e08bbe9ea/commands/cancel"
        },
      - place: {
            href: "http://localhost:9002/orders/5f5e90f4-c458-420c-a936-535e08bbe9ea/commands/place"
        }
    }
}
```

EVENT SOURCING



Queries?

QUERIES

Bad news 😔

The **Event Store** is not good for queries

Most applications

- A few **writes**
- A lot of **reads**

ThoughtWorks®

CQRS

CQRS

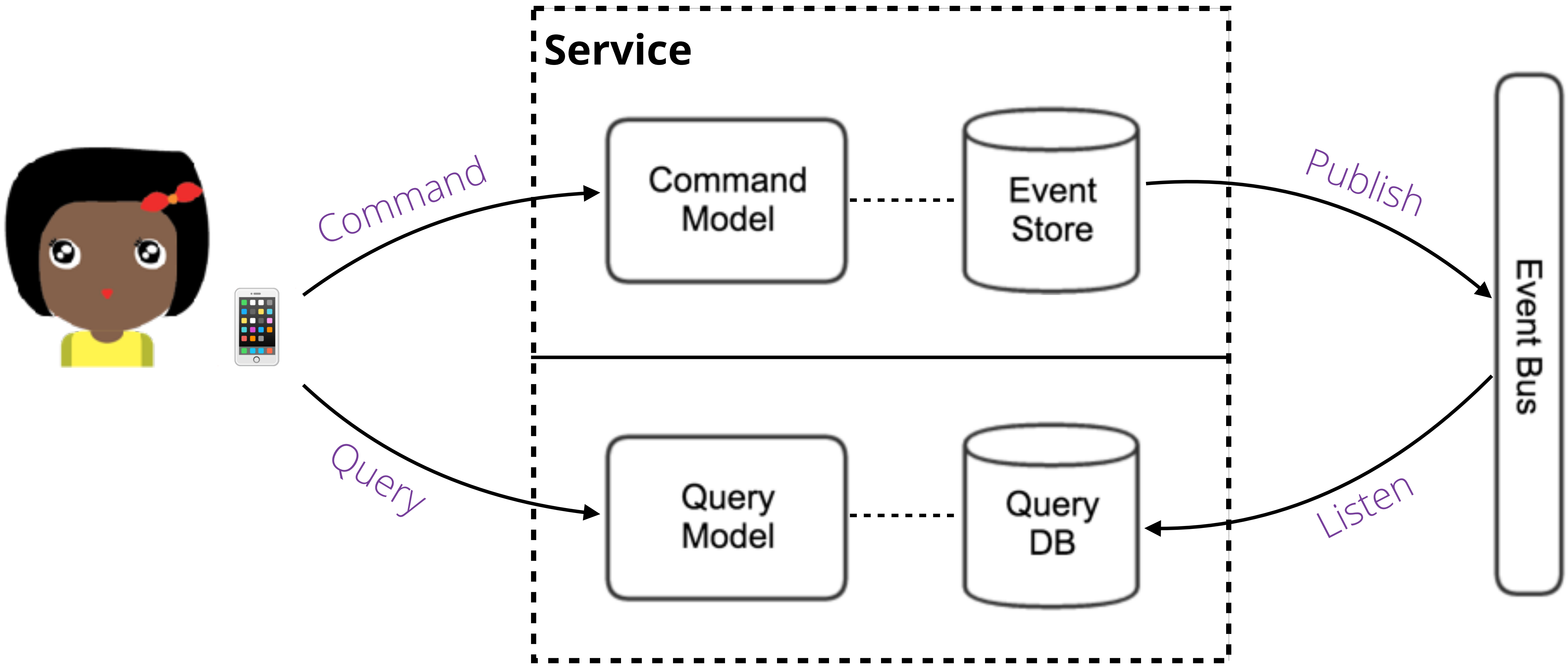
Command ------------- Write ✍️

Query ------------------------- Read 📖

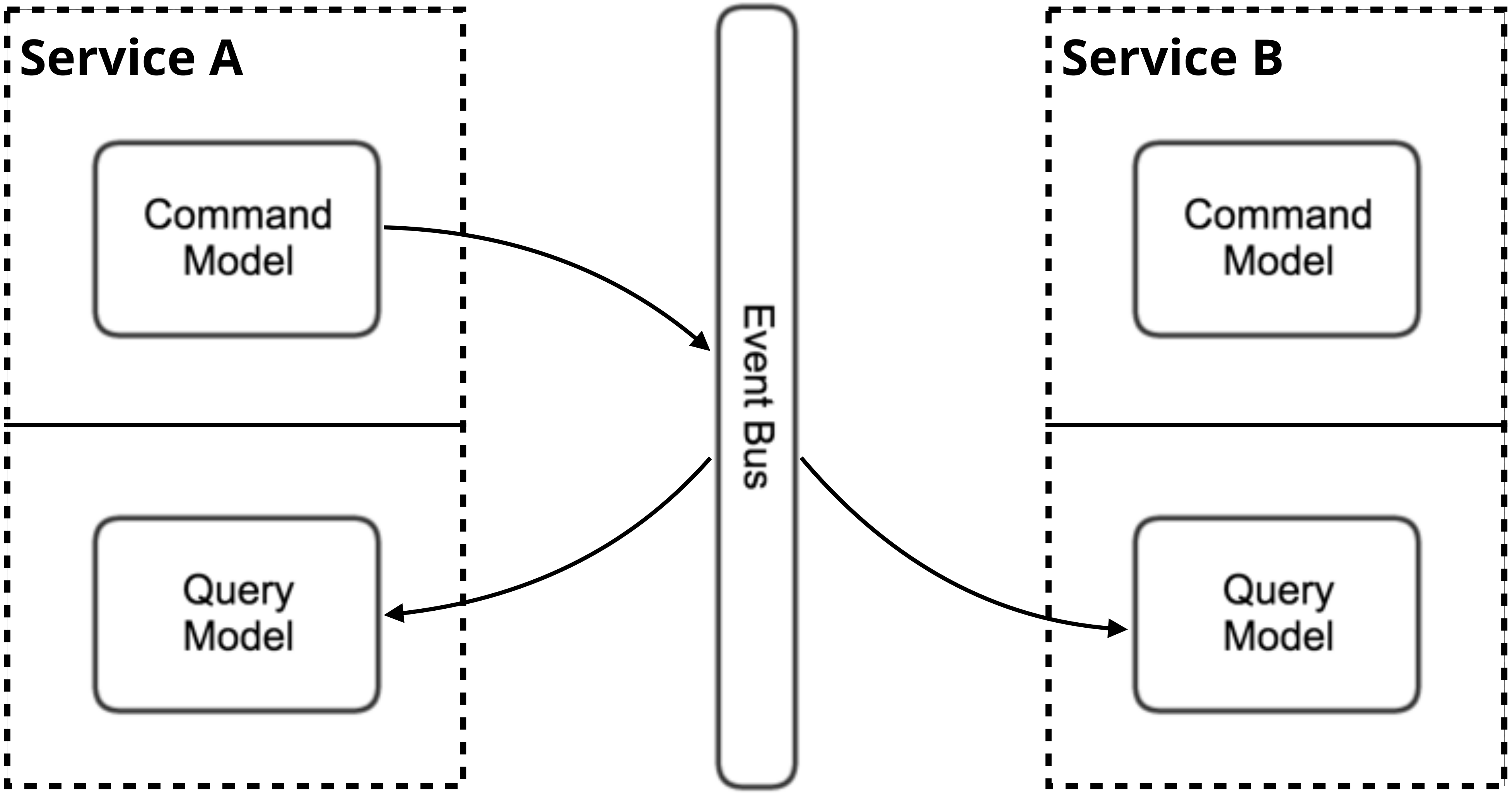
Responsibility

Segregation

CQRS


Service
Command
Command Model
Event Store
Publish
Event Bus
Query
Query Model
Query DB
Listen

CQRS



Service A
Command Model
Query Model
Event Bus
Service B
Command Model
Query Model

CQRS

Command model

- Command definitions
- Event definitions
- The aggregate
- Aggregate repository
- Write only API

Query model

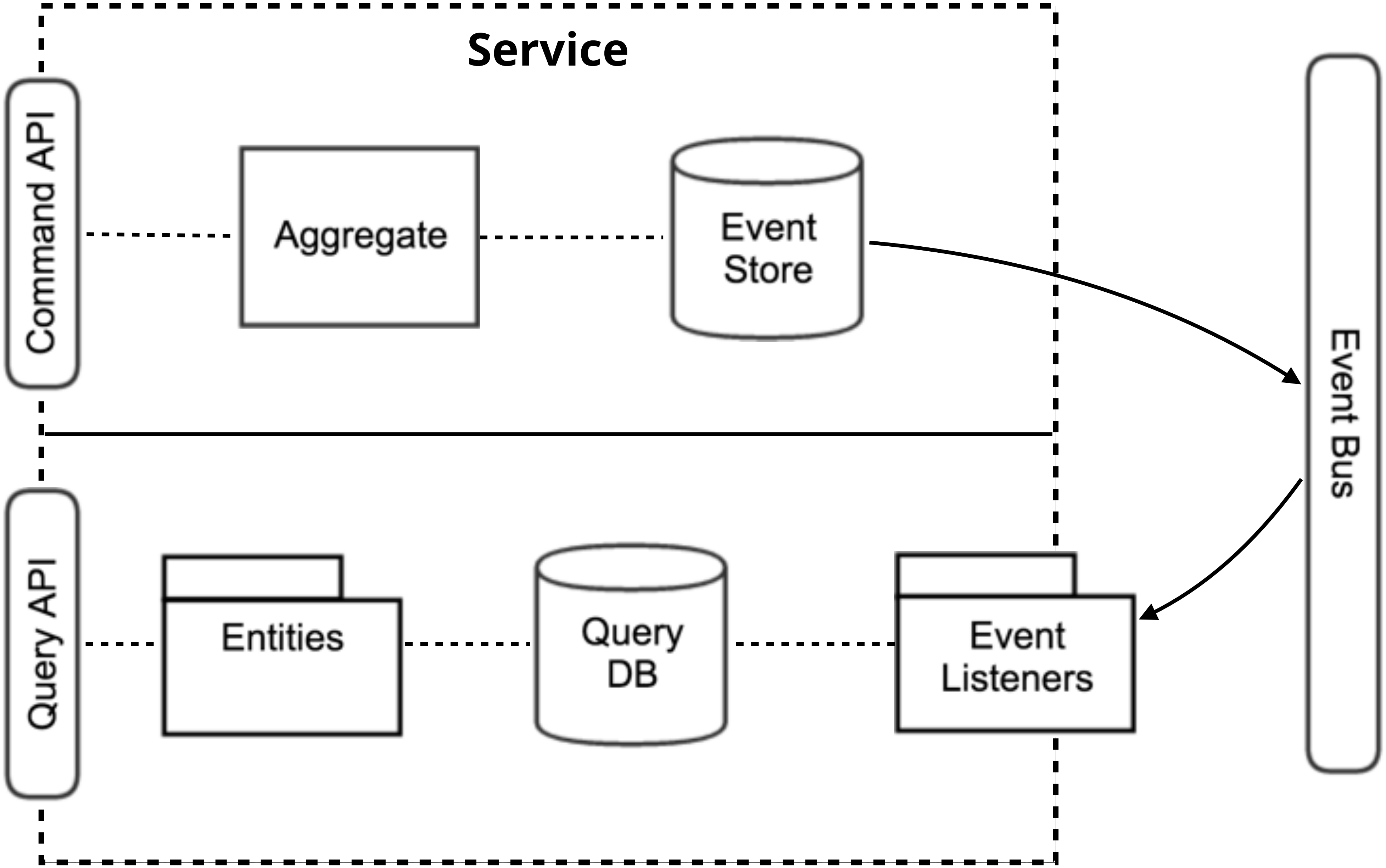
- Event listeners
- Query entities
- Repositories
- Read only API

ThoughtWorks®

DEMO

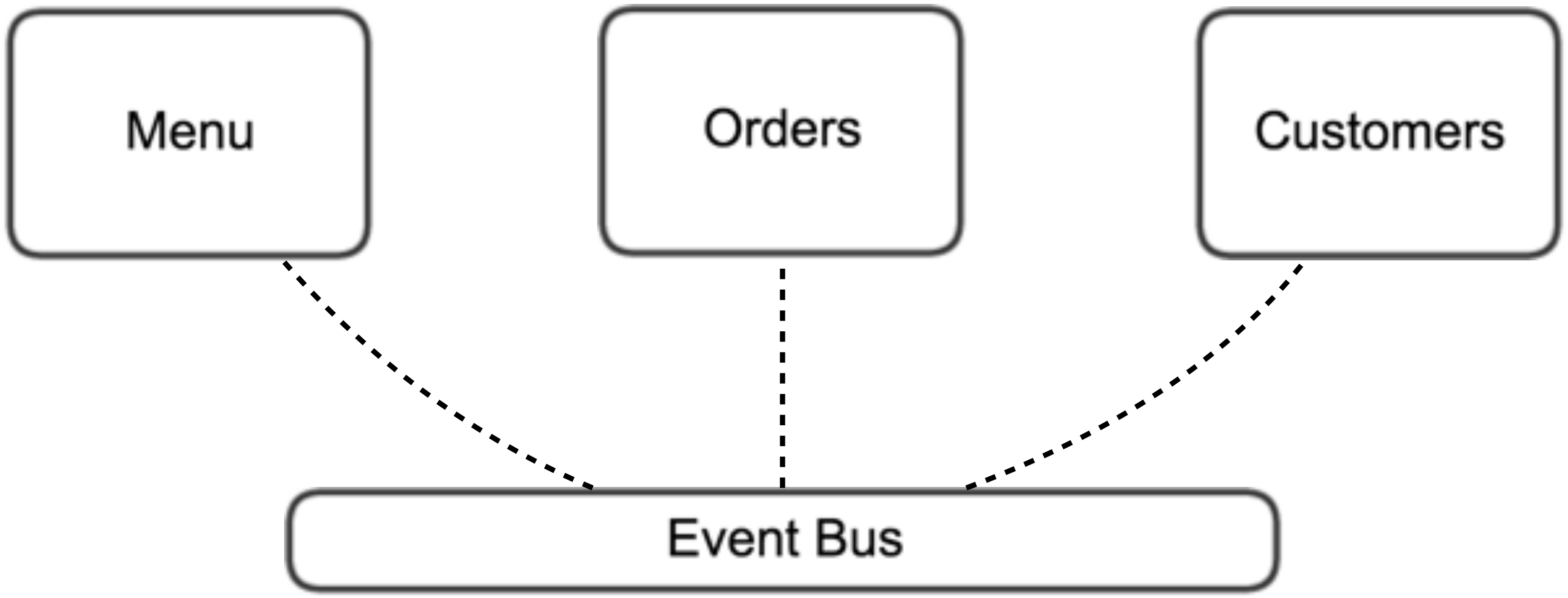
DEMO

Online Restaurant



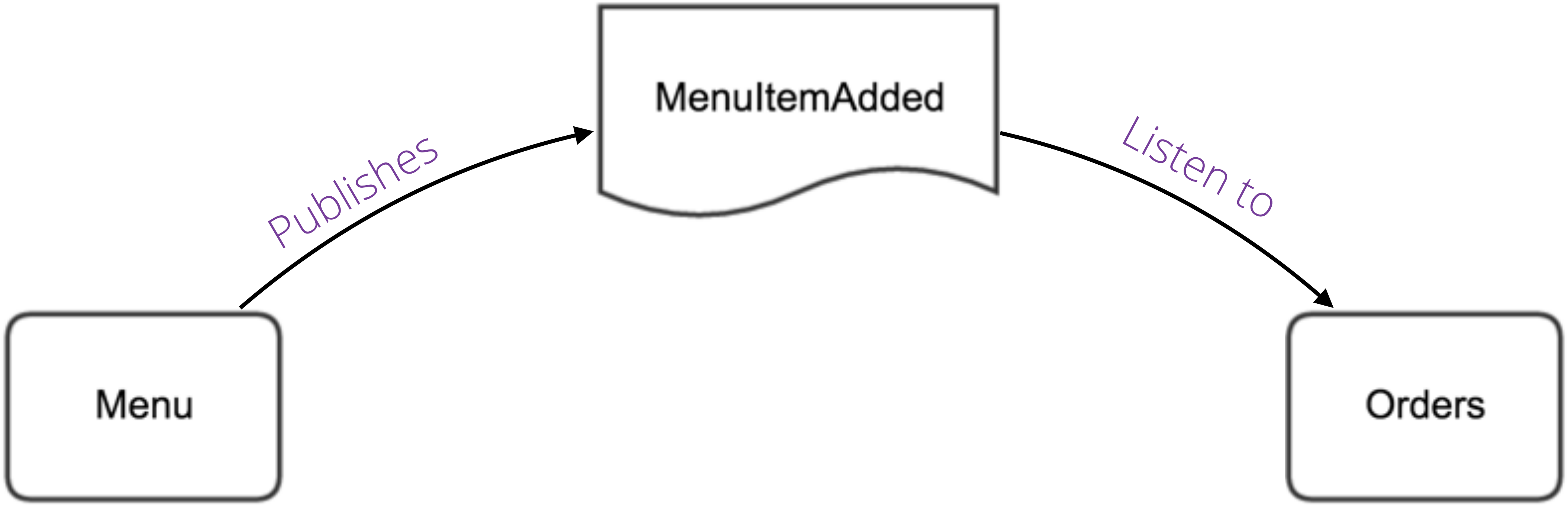
DEMO

Online Restaurant



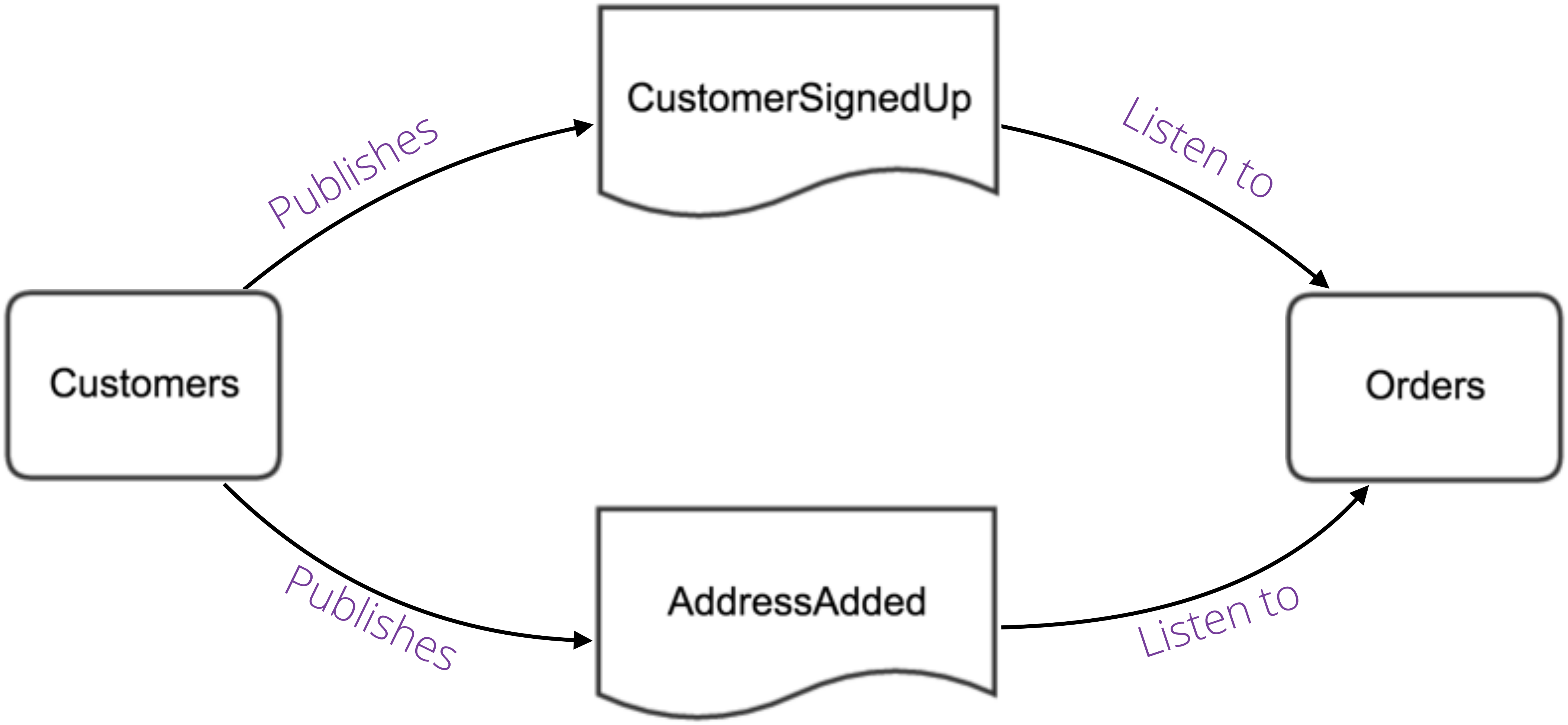
EXAMPLE: ONLINE RESTAURANT

Online Restaurant



EXAMPLE: ONLINE RESTAURANT

Online Restaurant



TECHNOLOGIES

Java

Spring Boot

Axon Framework

Spring Data REST

RabbitMQ

https://github.com/vvgomes/event-driven-restaurant

ThoughtWorks®

SUMMARY

MICROSERVICES + EVENT SOURCING

Benefits

- History based queries
- Audit log by design
- Immutability
- User intent
- Decoupling
- Resilience

Challenges

- Complexity
- Snapshots
- Upcasting
- Race conditions
- Event contracts
- Eventual consistency

ThoughtWorks®

WHEN MICROSERVICES MEET EVENT SOURCING

Vinicius Gomes