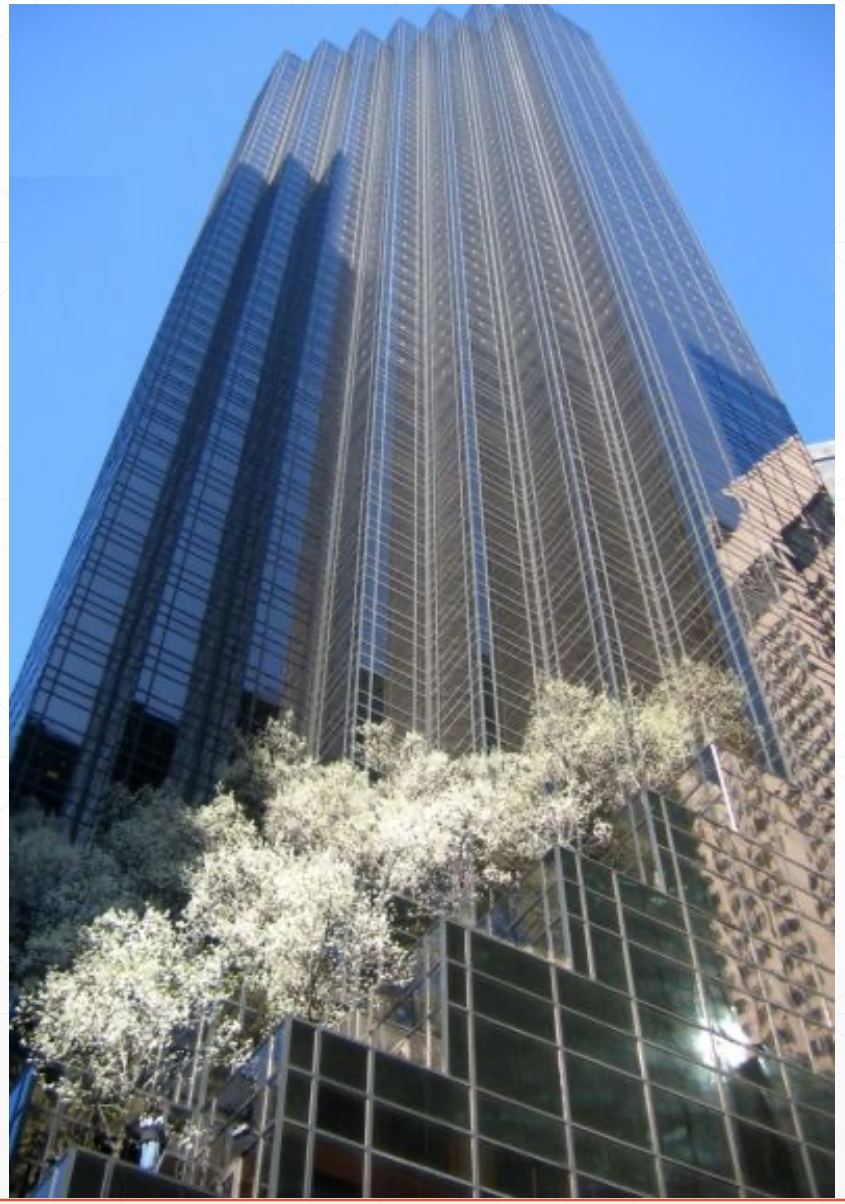
# Getting out of the monolith hell

Domenico Musto



# Monolithic architecture is great

# Monolithic architecture

- Is easy to understand

# Monolithic architecture

- Is easy to understand
- A single code base

# Monolithic architecture

- Is easy to understand
- A single code base
- No integration issues

# Monolithic architecture

- Is easy to understand
- A single code base
- No integration issues
- A single deployment pipeline


Get started
Try it free >
Get in touch!
Join now »
FIND OUT MORE
VIEW DEAL
>> Check it out <<
FREE 30 DAY TRIAL
SIGN UP FOR FREE
Take a tour
CLICK HERE
Click here
Signup now!
GET IT NOW!
Register now
ADD TO CART
Start now
Create your account
Activate my FREE trial
LEARN MORE
Download now
SHOP NOW
SIGN UP

- The code base gets too large and difficult to work with

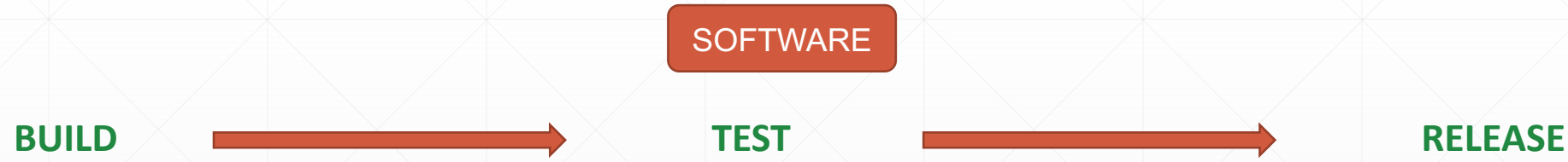
- The code base gets too large and difficult to work with
- The IDE is overloaded

- The code base gets too large and difficult to work with
- The IDE is overloaded
- Continuous delivery is difficult

# SOFTWARE DEVELOPMENT CYCLE

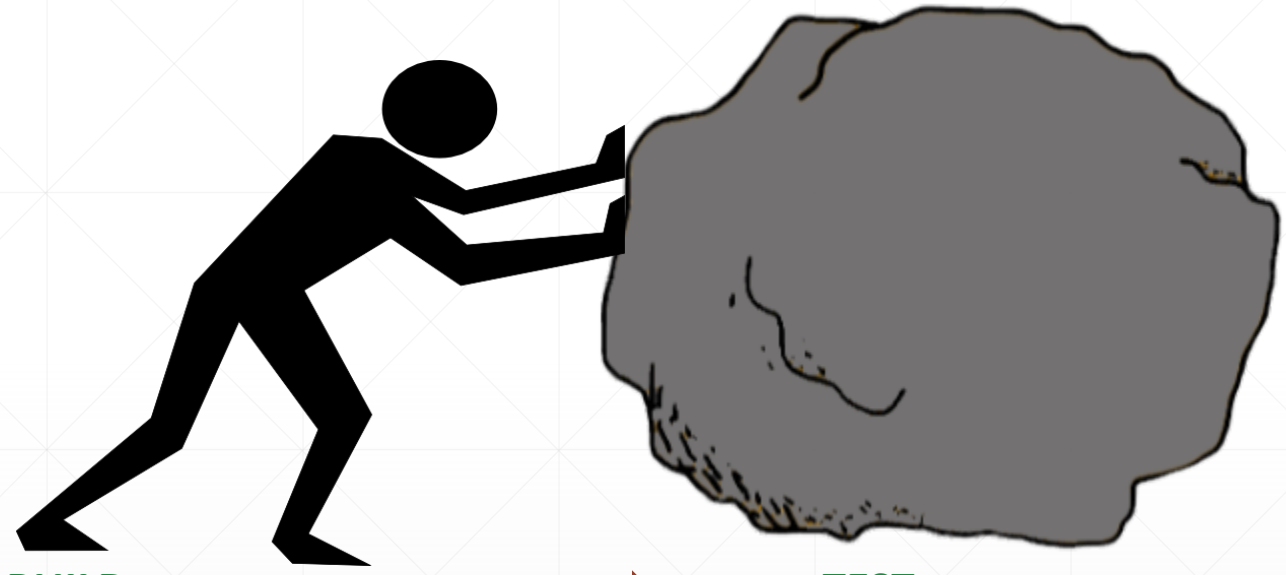
# SOFTWARE DEVELOPMENT CYCLE

# SOFTWARE DEVELOPMENT CYCLE



# SOFTWARE DEVELOPMENT CYCLE

# SOFTWARE DEVELOPMENT CYCLE



BUILD → TEST → RELEASE

- The code base gets too large and difficult to work with
- The IDE is overloaded
- Continuous delivery is difficult
- The system is difficult to scale

- The code base gets too large and difficult to work with
- The IDE is overloaded
- Continuous delivery is difficult
- The system is difficult to scale
- Scaling the dev team is difficult

- The code base gets too large and difficult to work with
- The IDE is overloaded
- Continuous delivery is difficult
- The system is difficult to scale
- Scaling the dev team is difficult
- Hard to adopt new technologies

- The code base gets too large and difficult to work with
- The IDE is overloaded
- Continuous delivery is difficult
- The system is difficult to scale
- Scaling the dev team is difficult
- Hard to adopt new technologies

# The monolith hell

# Distributed system

- Speed up delivery by reducing development friction
- Scale the system
- Scale the development team
- Increase the uptime of your application

# Agenda

- Business capabilities
- Components
- Integration patterns
- From monolith to distributed
- Pitfalls

# Business capabilities

# Identify business capabilities

- Things that the business / product can do

# Identify business capabilities

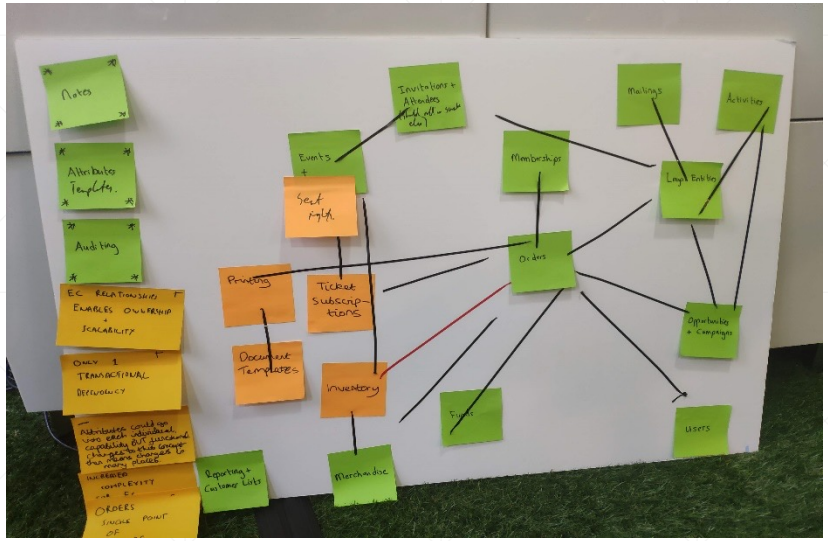
- Things that the business / product can do
- Shared domain space and language

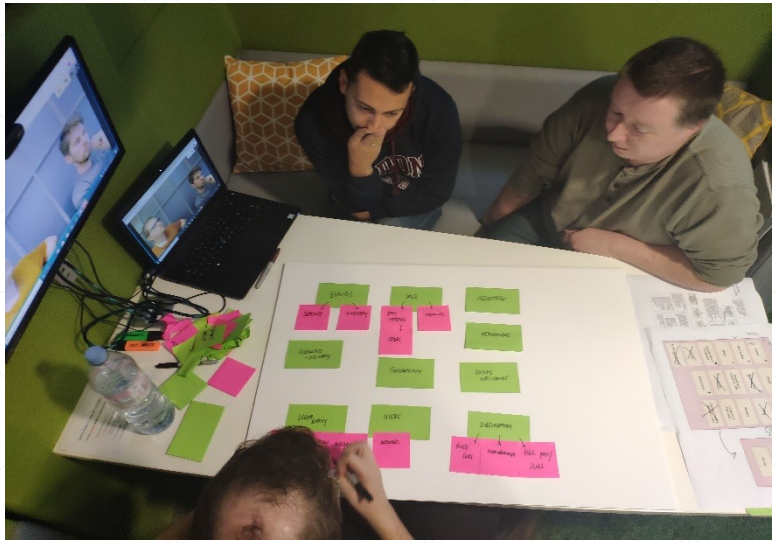
# Identify business capabilities

- Things that the business / product can do
- Shared domain space and language
- A team could be built around

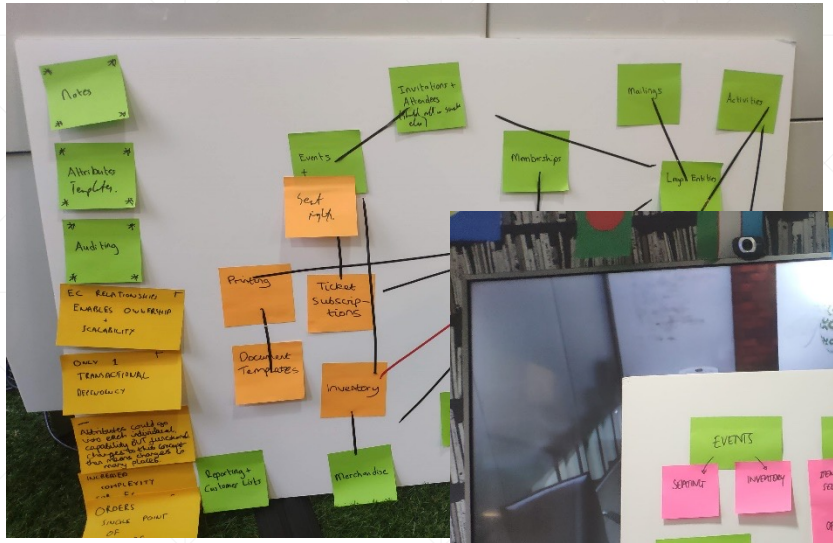
# Identify business capabilities

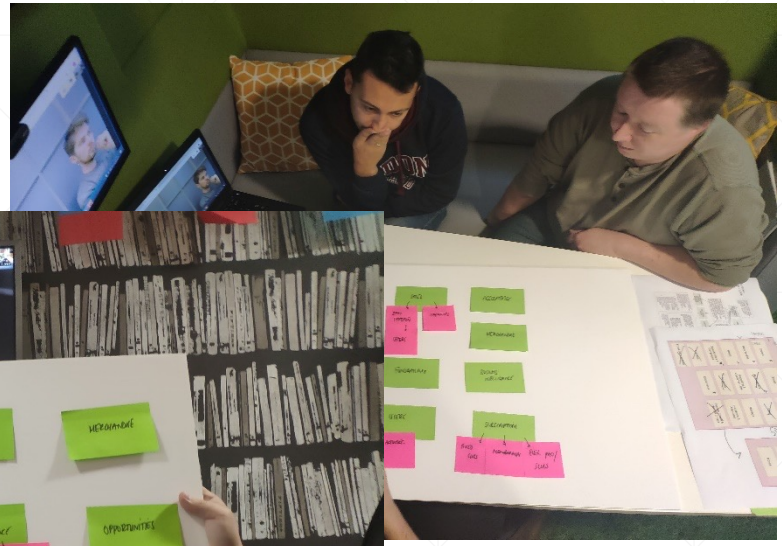
- Things that the business / product can do
- Shared domain space and language
- A team could be built around
- Micro products

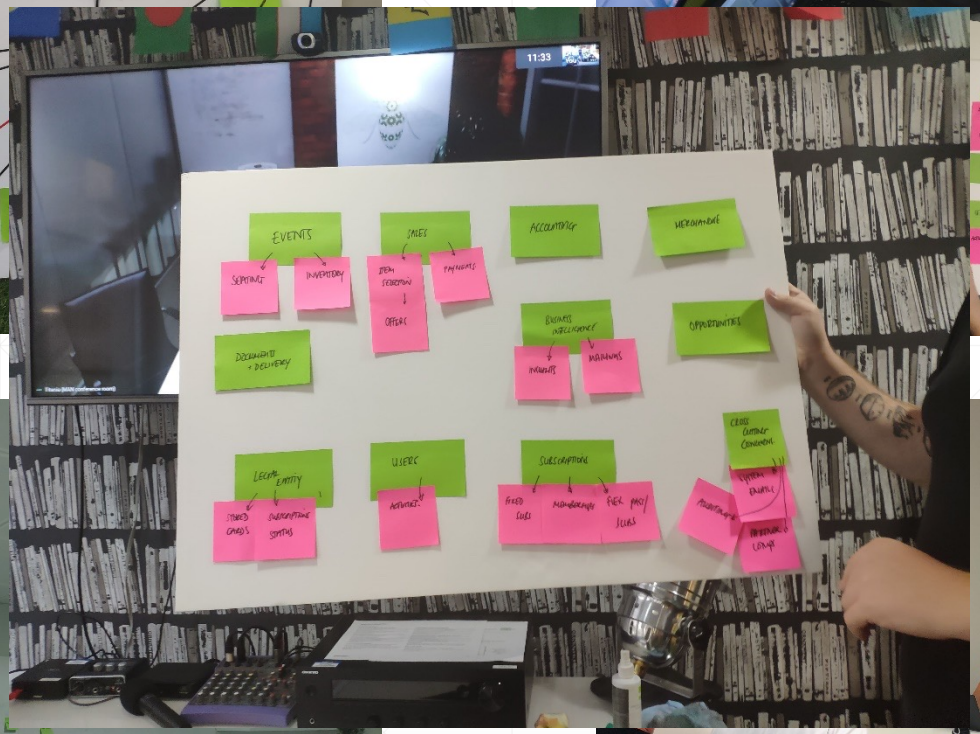

Notes
Attributes Templates
Auditing
Events
Invitations + Attendees
Memberships
Mailings
Activities
Loyalty Entities
Seat Mgmt.
Orders
Ticket subscriptions
Printing
Document Templates
Inventory
Opportunities + Campaigns
Users
Reporting + Customer Lists
Merchandise

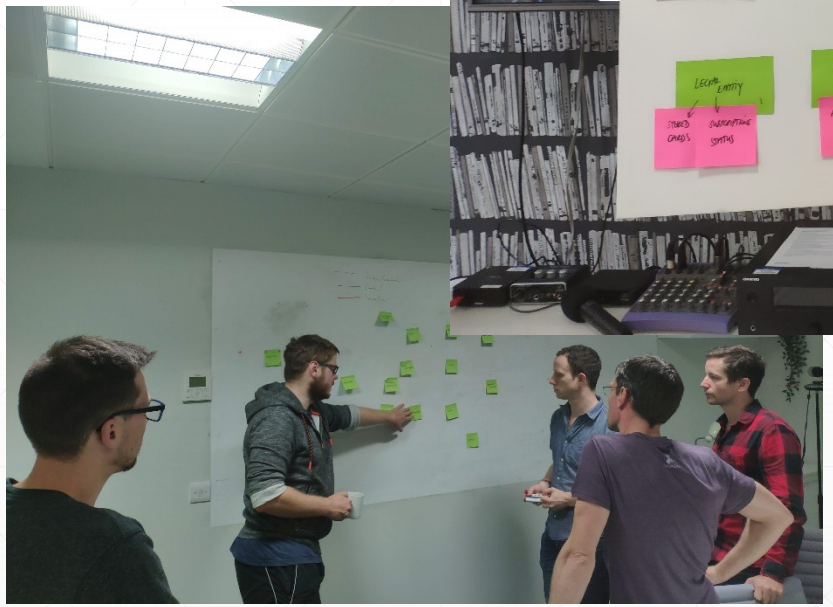



JEAN-MICHEL ASQUIAT

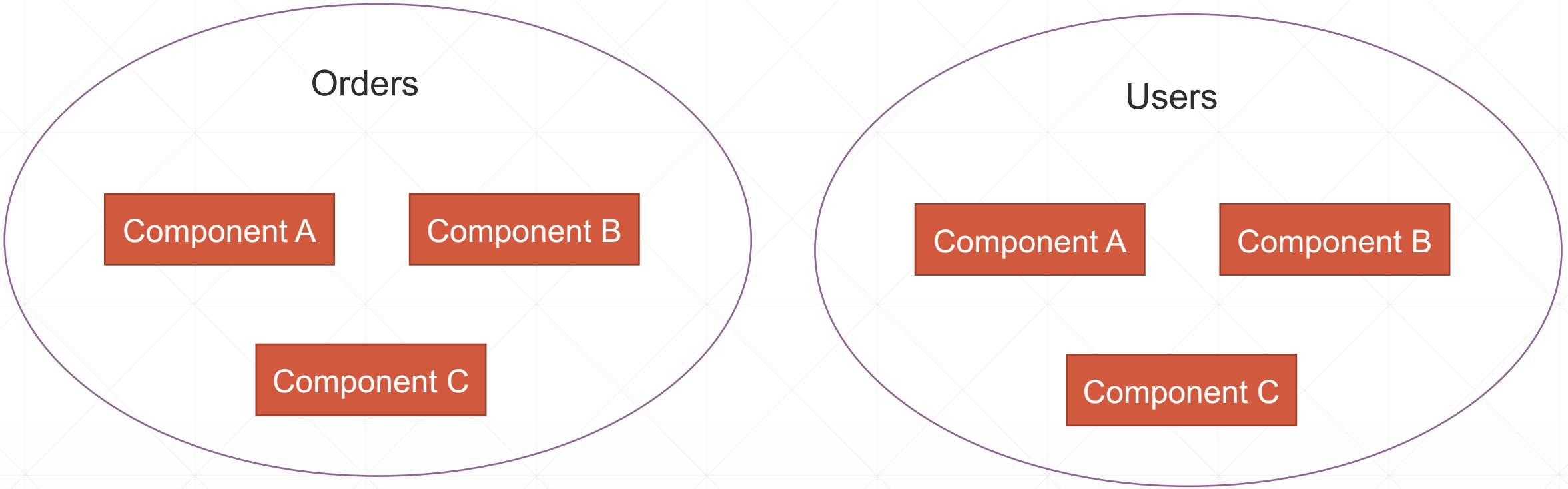

Notes
Invitations + Attendees
Mailings
Activities
Events
Memberships
Attributes Templates
Auditing
Sent Mail
Printing
Ticket subscriptions
Document Templates
Inventory
Reporting + Customer Lists
Merchandise




EVENTS
SALES
ACCOUNTING
INVENTORY
PAYMENTS
OFFERS
OPPORTUNITIES
USERS
SUBSCRIPTIONS



# Components


Orders
Component A
Component B
Component C
Users
Component A
Component B
Component C

# Components

- Autonomous

*Ian Cooper*

# Components

- Autonomous
- Explicit boundaries

*Ian Cooper*

# Components

- Autonomous
- Explicit boundaries
- Bounded context

*Ian Cooper*

# Components

- Autonomous
- Explicit boundaries
- Bounded context
- Decentralized governance

*Ian Cooper*

# Components

- Autonomous
- Explicit boundaries
- Bounded context
- Decentralized governance
- Distributable

*Ian Cooper*

# Integration patterns

## Orders

- Understands and owns orders data
- Drives orders workflows

## Users

- Understands and owns users data
- Drives users workflows

# HTTP VS Messaging

# HTTP

# HTTP


Orders
OrderComponent
https://api.awesome.com/users/32
Users
UsersAPI

# HTTP goodness

- Easy

# HTTP goodness

- Easy
- Synchronous

# HTTP goodness

- Easy
- Synchronous
- Caching

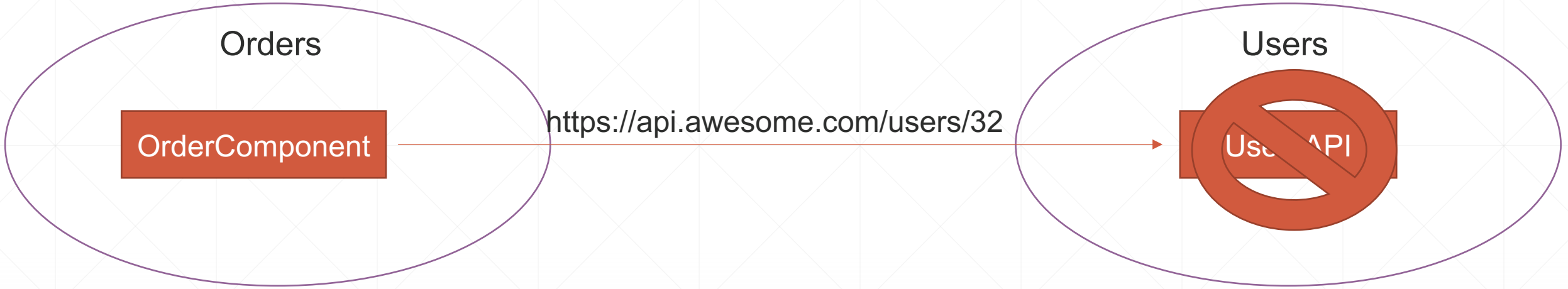
# HTTP weakness

- Reduces service autonomy

# HTTP


Orders
OrderComponent
https://api.awesome.com/users/32
Users
UsersAPI

# HTTP


Orders
OrderComponent
https://api.awesome.com/users/32
Users

# HTTP weakness

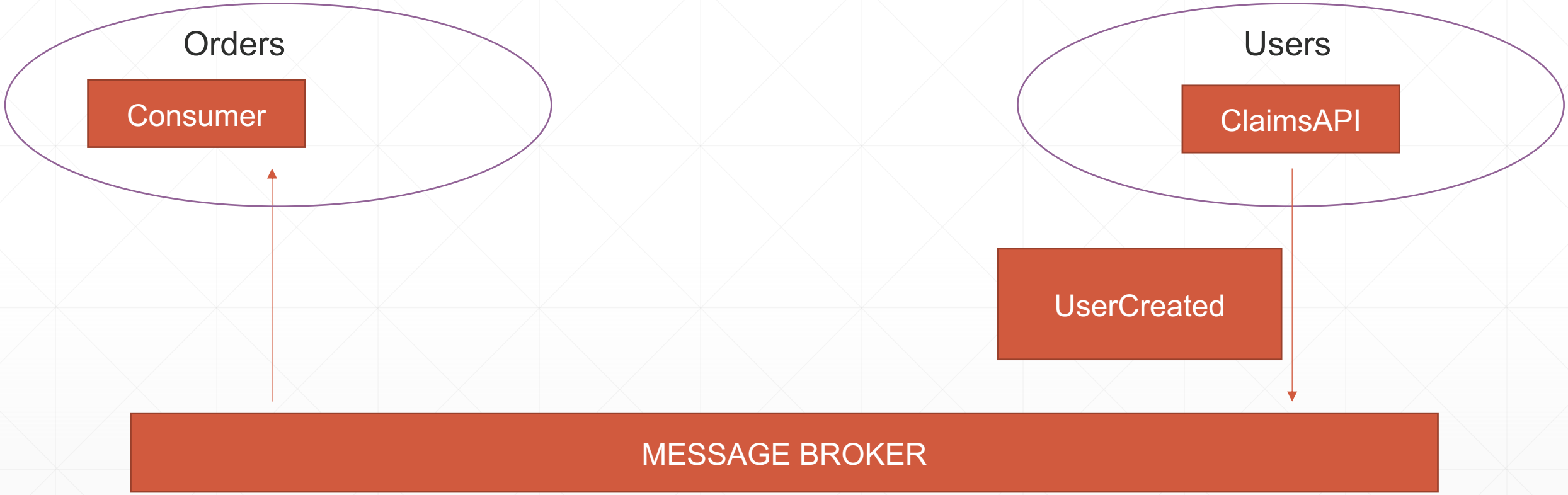
- Reduces service autonomy
- RPC temptation

# HTTP weakness

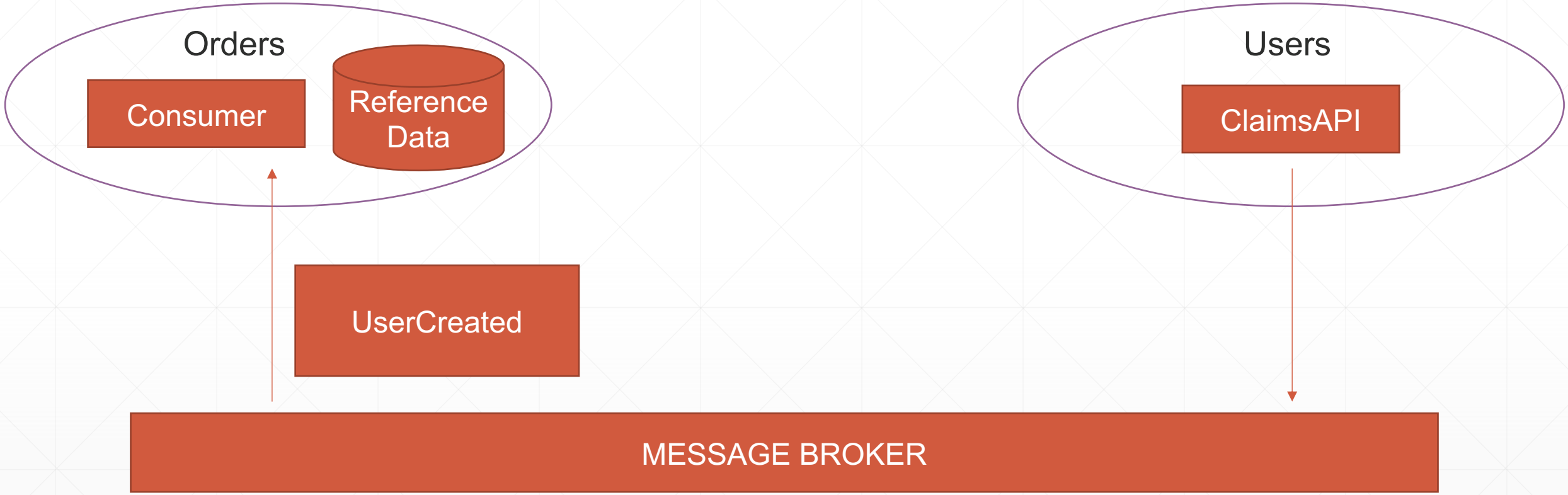
- Reduces service autonomy
- RPC temptation
- Versioning

# Messaging

# Messaging


Orders
Consumer
Users
ClaimsAPI
UserCreated
MESSAGE BROKER

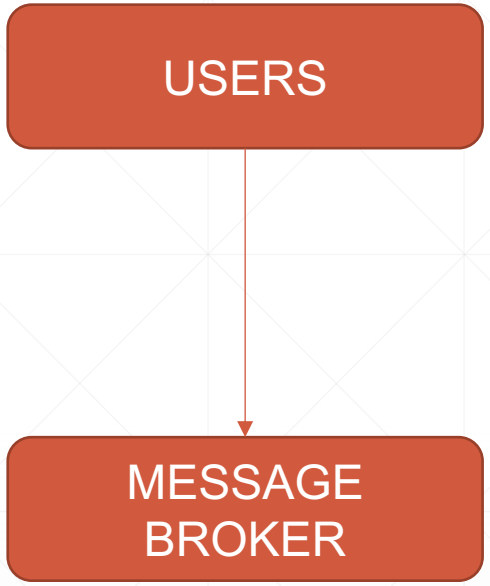
# Messaging


Orders
Consumer
Reference Data
Users
ClaimsAPI
UserCreated
MESSAGE BROKER

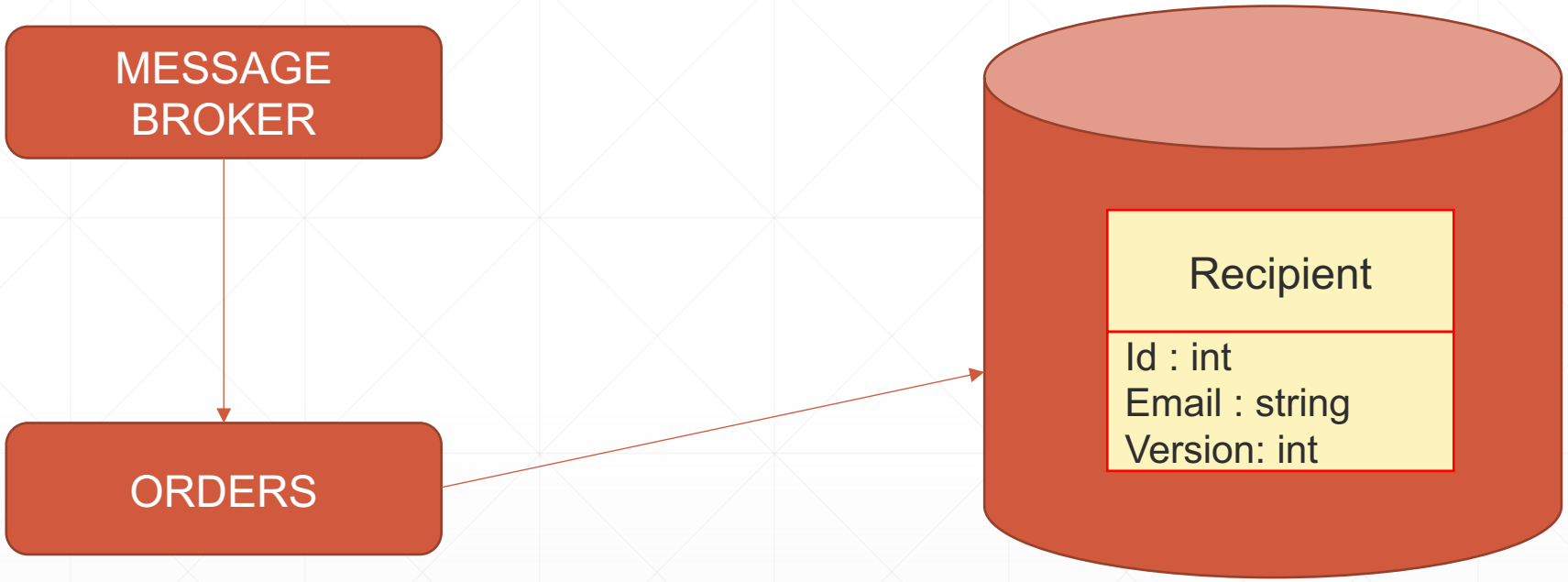
# Reference data

UserCreated

```
{
  Id: 1,
  Username: “domenico”,
  Email: “domenico.musto@gmail.com”,
  Twitter: “@mimmozzo”,
  Version: 1
}
```



# Reference data



# Type of messages

- Events
- Document messages
- Commands

# Event

- Informs the outside world that something has happened

# Event

- Informs the outside world that something has happened
- Past sequence

# Event

- Informs the outside world that something has happened
- Past sequence
- Unique ID

# Event

```
UserDeleted {
        Id: 100,
        UserId: 200
}
```

# Document Message

```
UserCreated {
   Id: 100,
   Name: ‘Domenico’,
   LastName: ‘Musto’,
   Email: ‘domenico.musto@gmail.com’,
   Twitter: ‘@mimmozzo’
}
```

# Command

- A demand to do something

# Command

- A demand to do something
- Imperative sentence

# Command

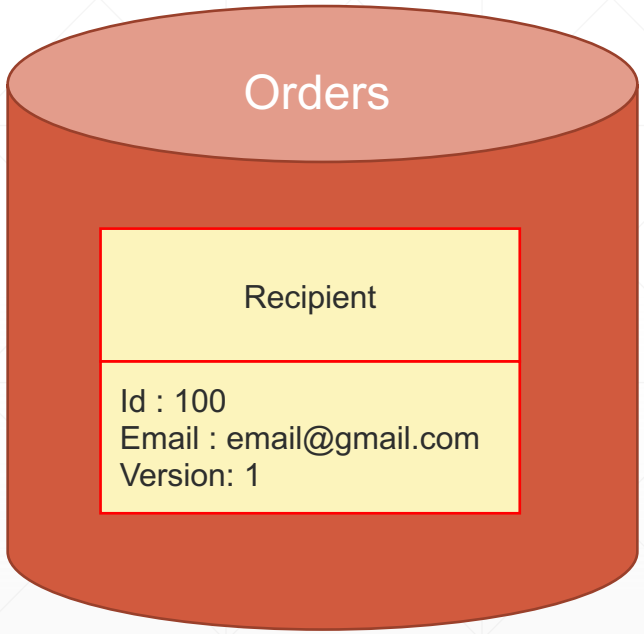
```
SendEmail{
   Id: 100,
   To: ‘domenico.musto@gmail.com’,
   From: ‘info@myecommerce.com’,
   Subject: ‘Welcome’,
   Content: ‘Enjoy’
}
```

# Operating events

# Operating events

- Eventual consistency
- Compensation
- Idempotence

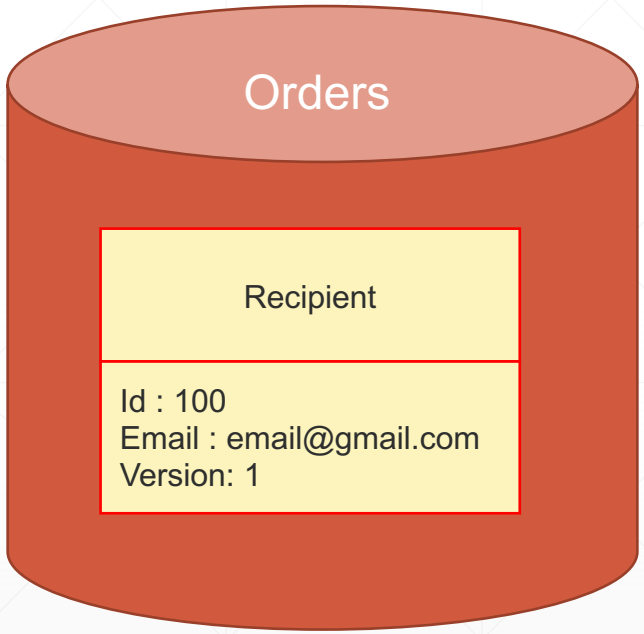
# Operating events – eventual consistency



# Operating events – eventual consistency

UserEmailUpdated

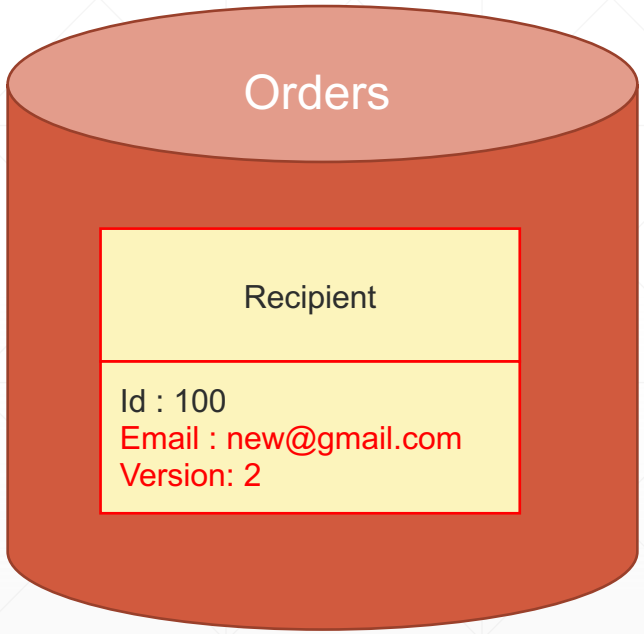
```
{
  Id: 100,
  Email: “new@gmail.com”,
  Version: 1
}
```



# Operating events – eventual consistency

UserEmailUpdated

```
{
  Id: 100,
  Email: “new@gmail.com”,
  Version: 1
}
```



# Operating events – compensation

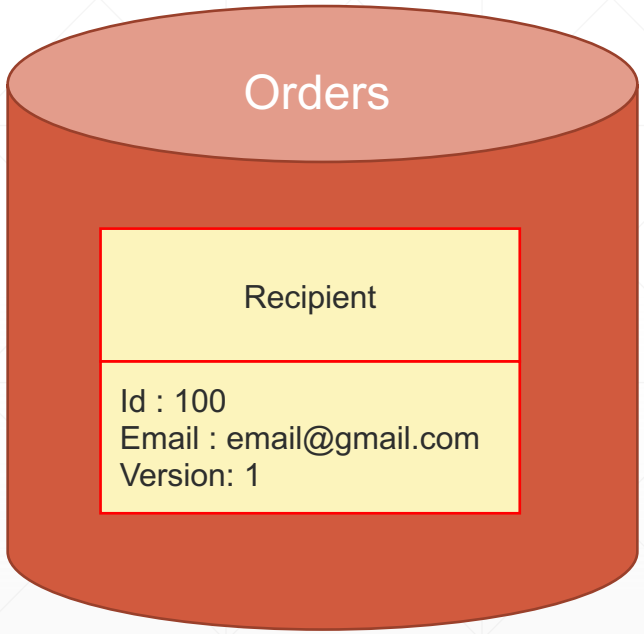
# Operating events – compensation

# Operating events – idempotence

# Operating events – idempotence

UserEmailUpdated

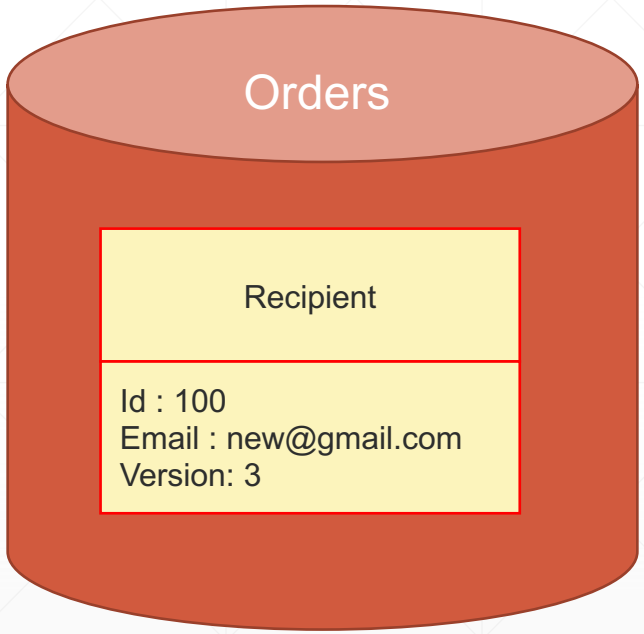
```
{
  Id: 100,
  Email: “new@gmail.com”,
  Version: 3
}
```



# Operating events – idempotence

UserEmailUpdated

```
{
  Id: 100,
  Email: “old@gmail.com”,
  Version: 2
}
```



# Messaging goodness

- Decoupling

# Messaging goodness

- Decoupling
- Service autonomy

# Messaging goodness

- Decoupling
- Service autonomy
- Versioning

# Messaging weakness

- Async

# Messaging weakness

- Async
- Eventual consistency

# Messaging weakness

- Async
- Eventual consistency
- Enforce idempotency

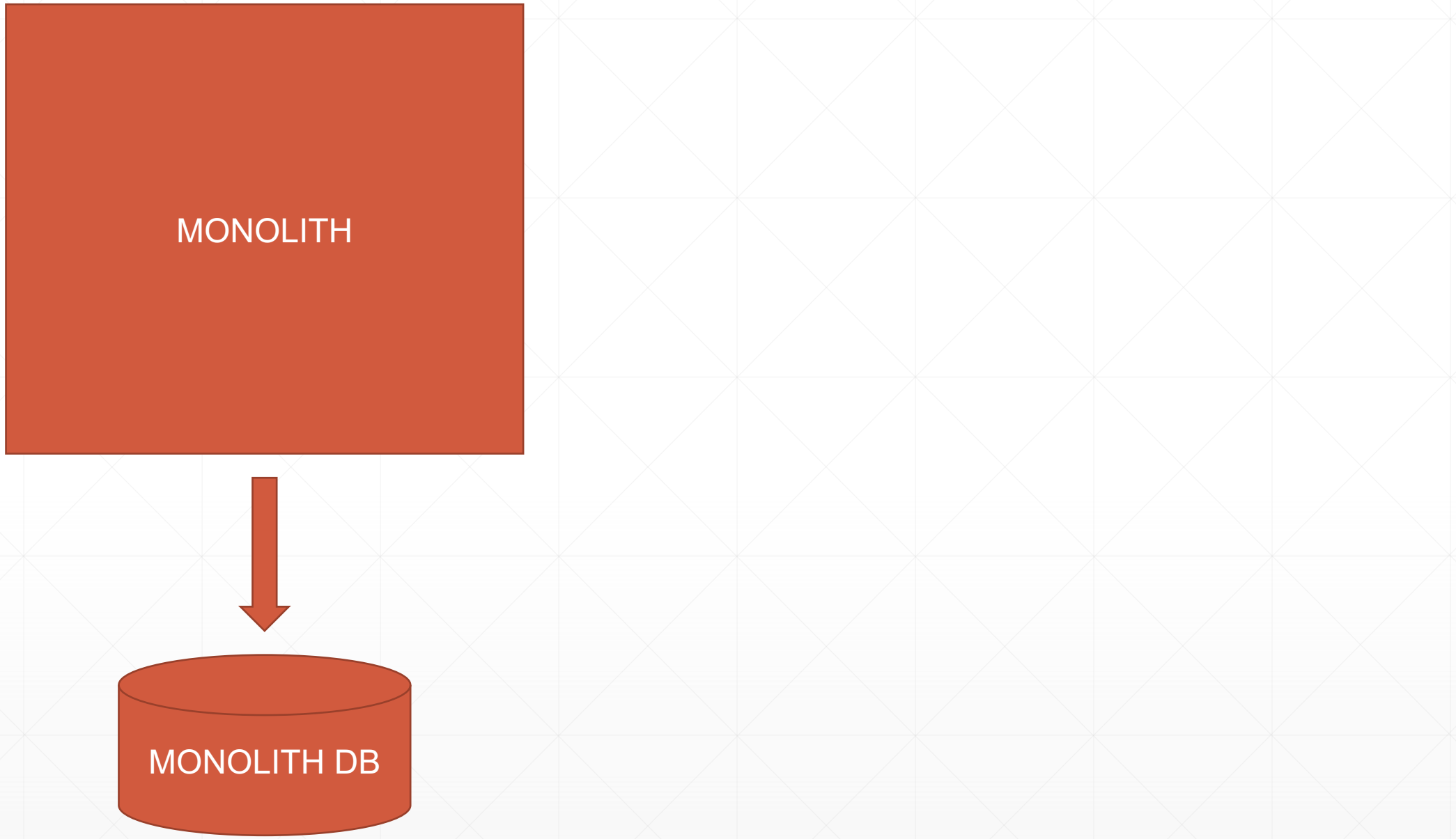
# HTTP or Messaging ?

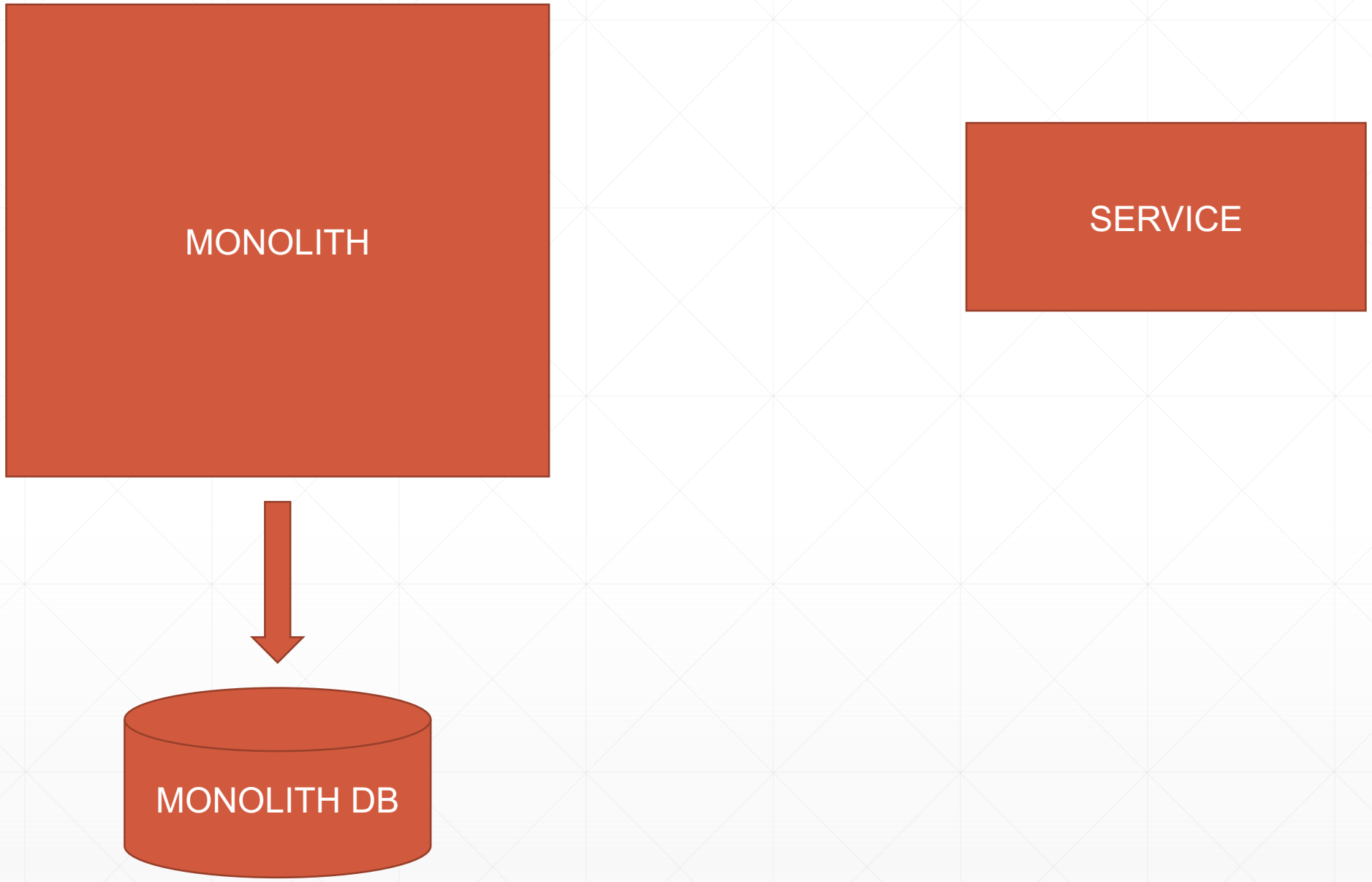
- Async VS Sync

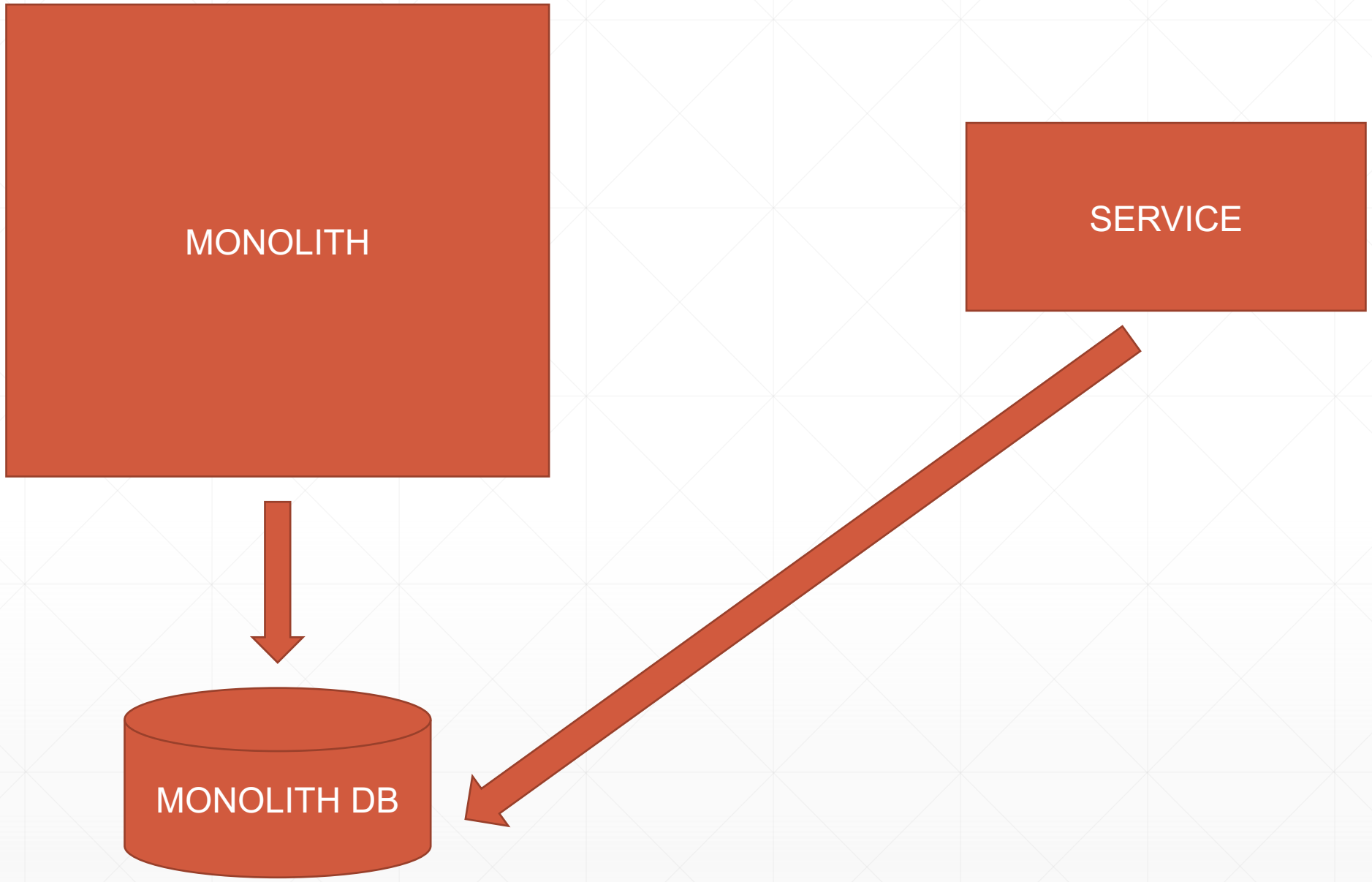
# HTTP or Messaging ?

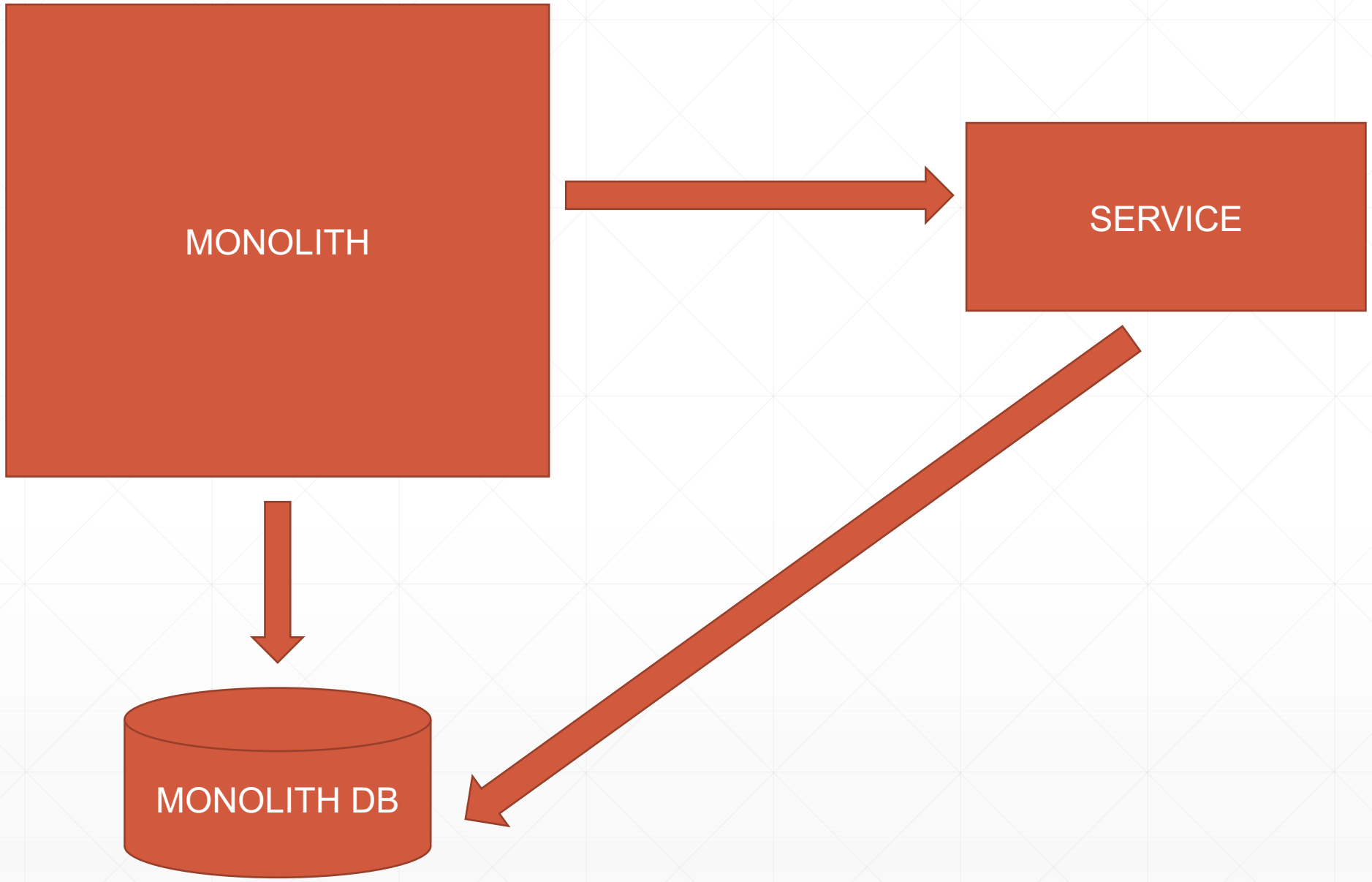
- Async VS Sync
- Cross BCs integration ?

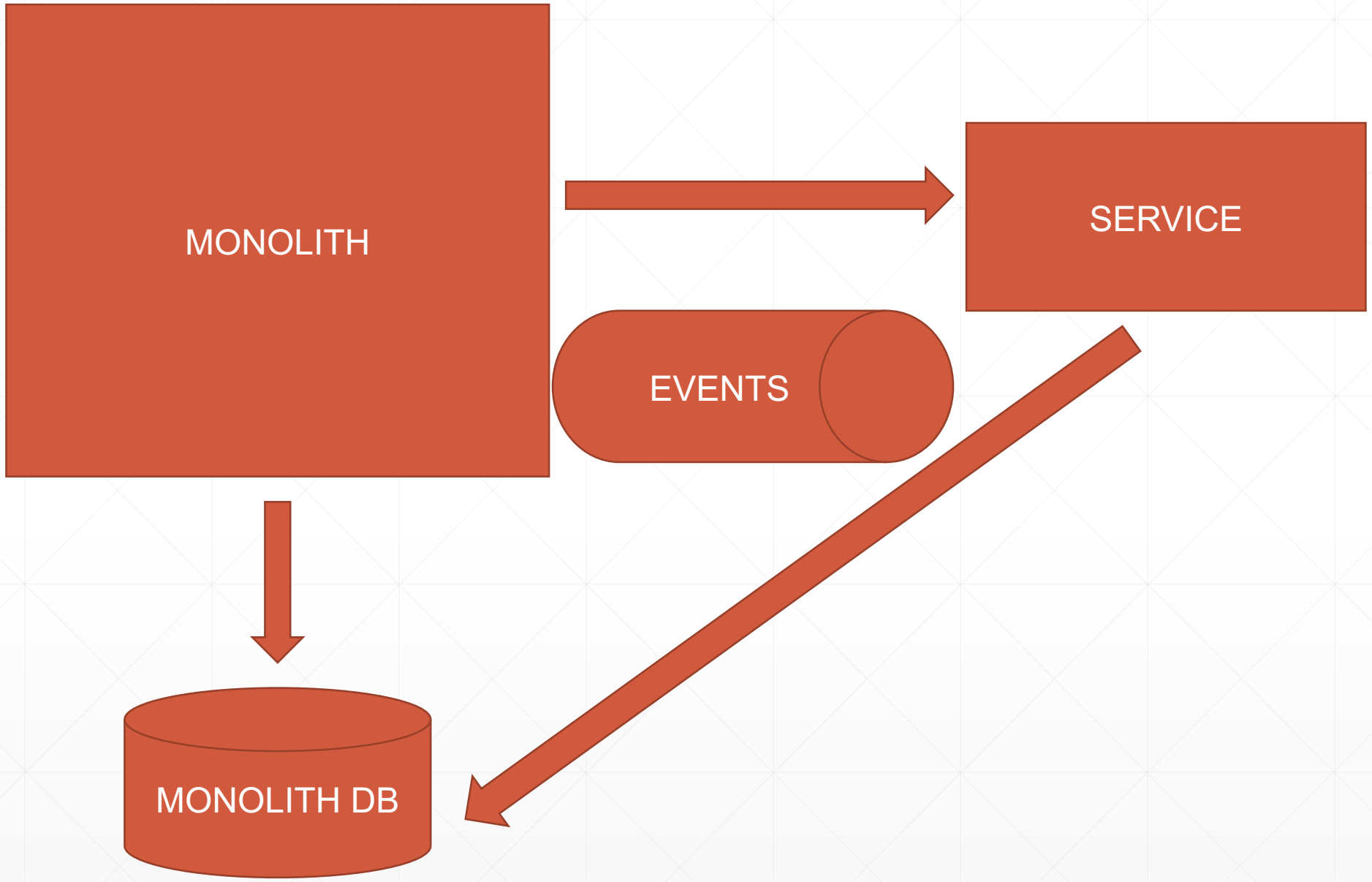
# From Monolith to Distributed

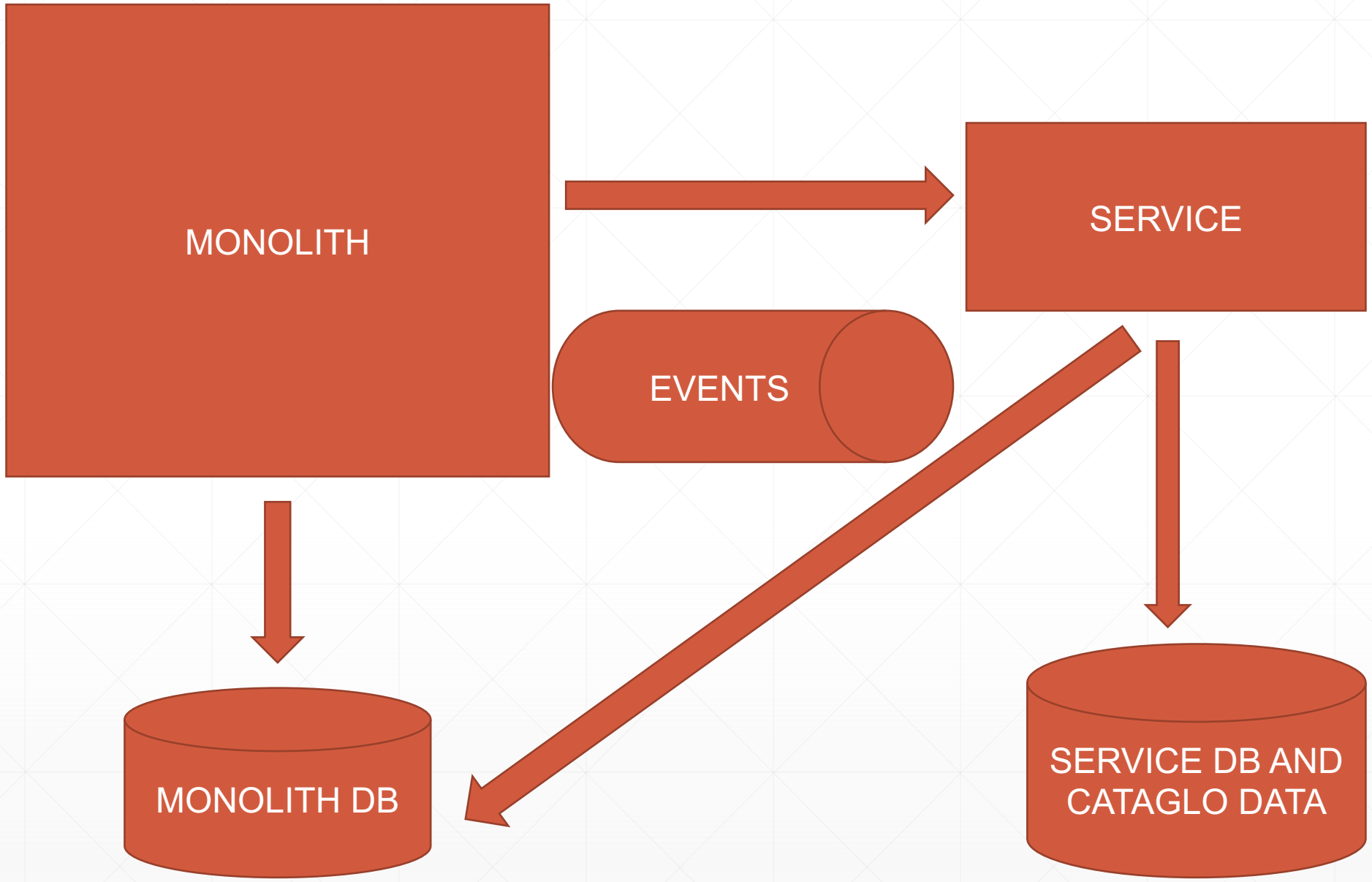

MONOLITH
MONOLITH DB

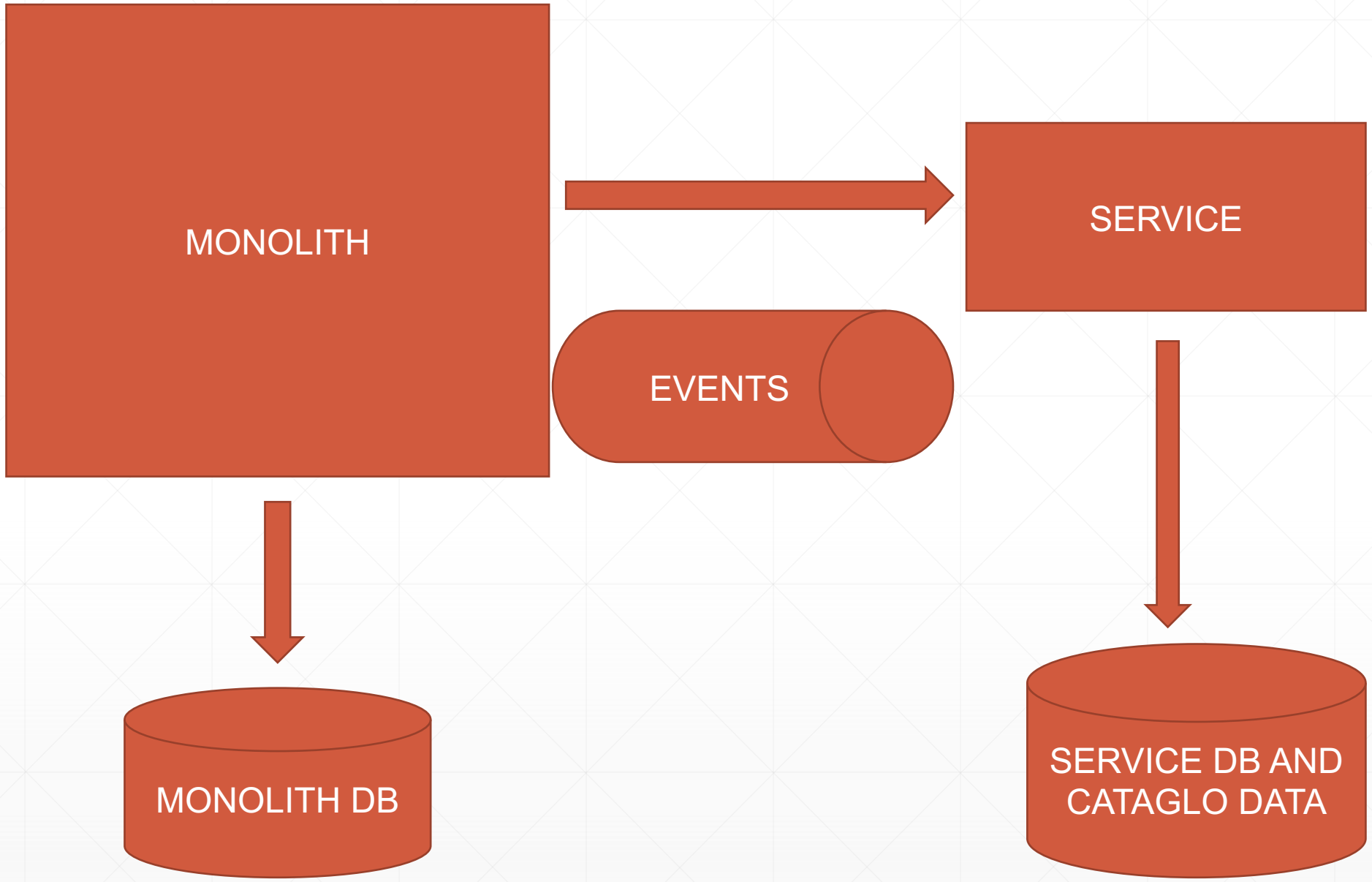

MONOLITH
SERVICE
MONOLITH DB


MONOLITH
SERVICE
MONOLITH DB


MONOLITH
SERVICE
MONOLITH DB


MONOLITH
SERVICE
EVENTS
MONOLITH DB


MONOLITH
SERVICE
EVENTS
MONOLITH DB
SERVICE DB AND
CATAGLO DATA


MONOLITH
SERVICE
EVENTS
MONOLITH DB
SERVICE DB AND
CATAGLO DATA

# Pitfalls

# Pitfalls

- Nano services

# Pitfalls

- Nano services
- Too many services to perform a business workflow

# Pitfalls

- Nano services
- Too many services to perform a business workflow
- Fragmented logic

# Pitfalls

- Nano services
- Too many services to perform a business workflow
- Fragmented logic
- Management and cost overhead

# Distributed system

- Speed up delivery by reducing development friction
- Scale the system
- Scale the development team
- Increase the uptime of your application

# Thanks.

domenico.musto@gmail.com
@mimmozzo