

# Loosely Coupled Microservices with Grails

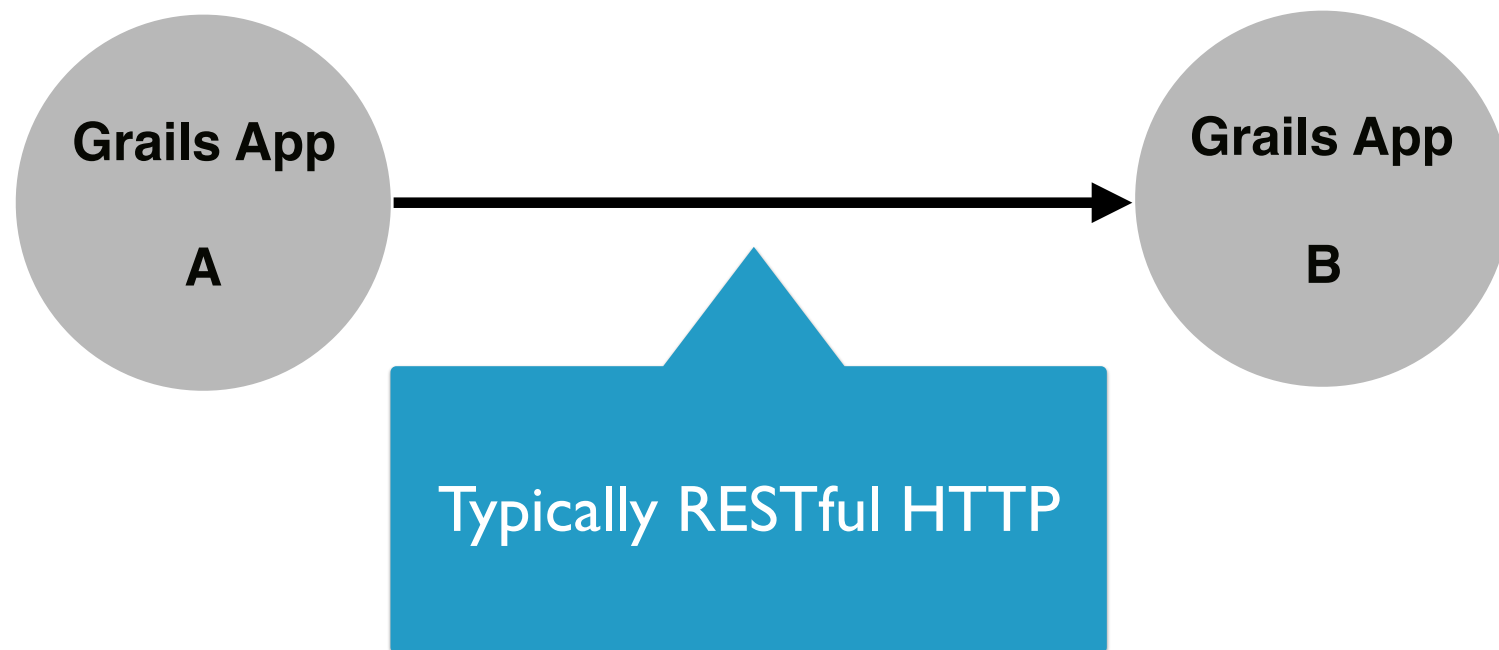
---

Michael Plöd  
@bitboss

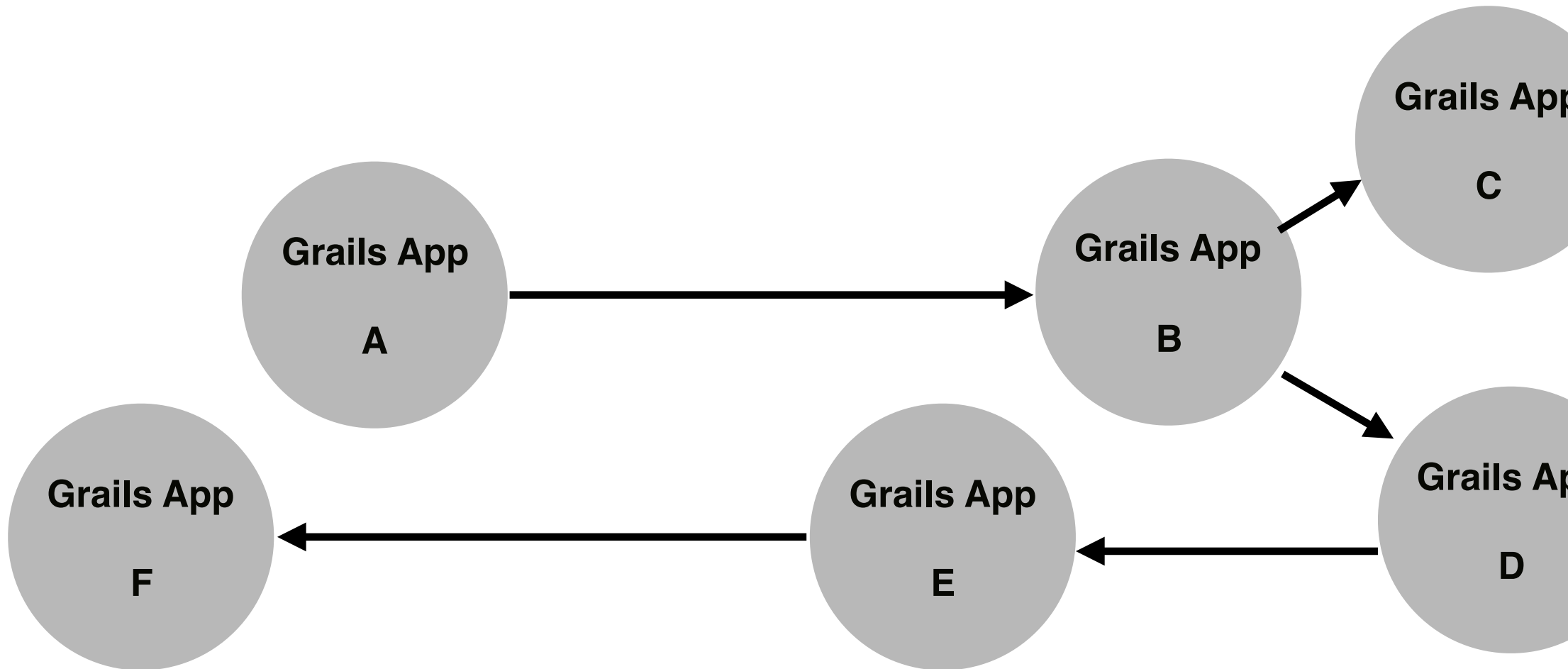


innoQ

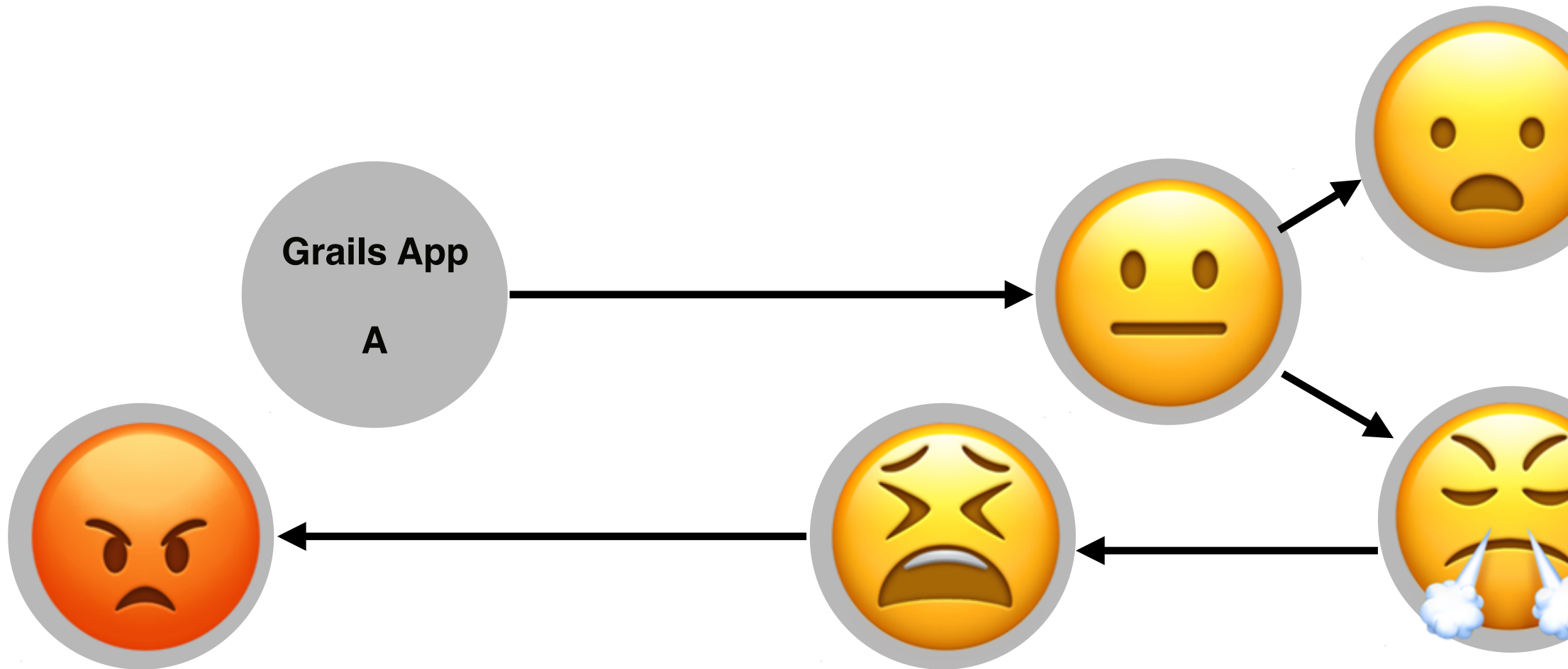
# Interactions between Microservices



# Interactions between Microservices



# Interactions between Microservices





Welcome to the  
**Dark Side**

# Challenges

---



Service Discovery



Latency



Timeouts



Load Balancing




Errors



Stability

# Usual Architecture Patterns

---

- › Event-Driven Architecture
  - › REST
  - › SOA
  - › Microservices / Self-Contained Systems
  - › CQRS
  - › Reactive / Actor Model
- 

# Domain Events

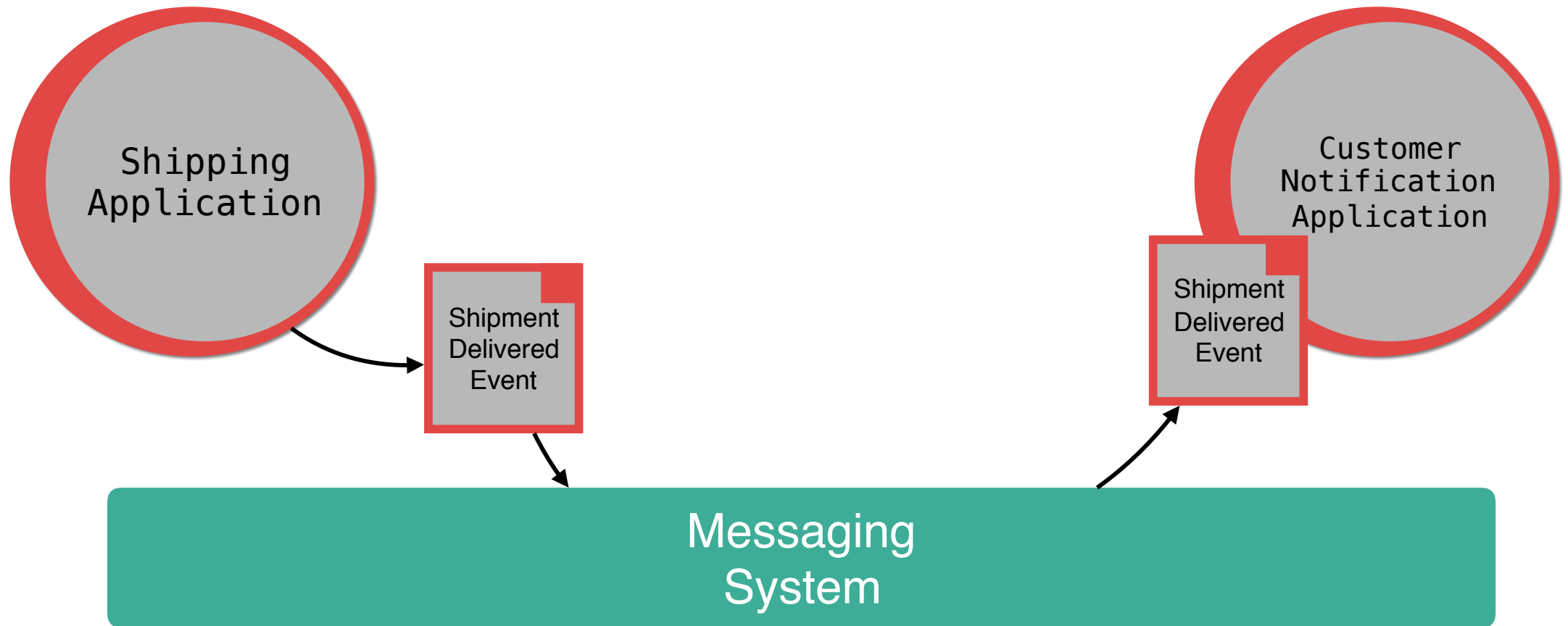
---

- › Applications can issue Domain Events
- › Domain Events are important occurrences in an Domain
  - › UserVerified
  - › ShoppingCartCheckedOut
  - › ShipmentDelivered
- › Domain Events are usually based on eventual consistency
- › Major driver for high degree of decoupling between Microservices

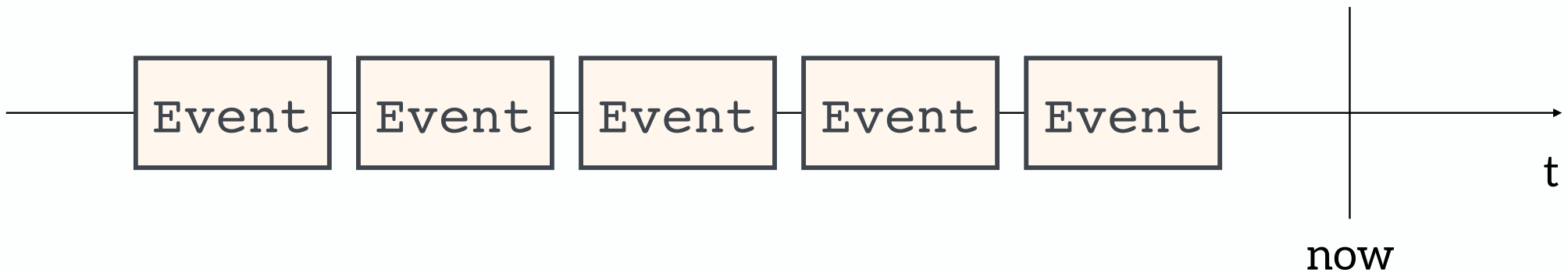


# Domain Events

---



*An event is something  
that happened in the past*



ShipmentDeliveredEvent

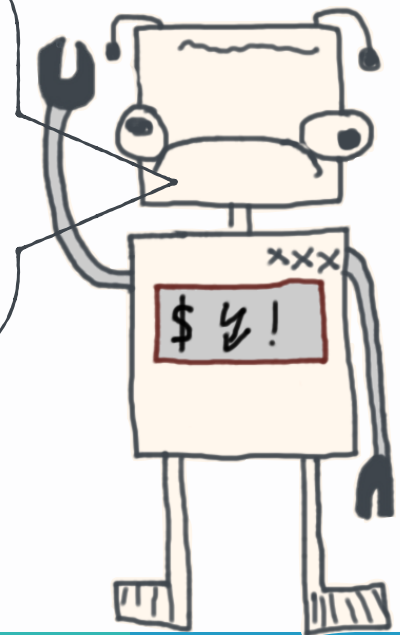
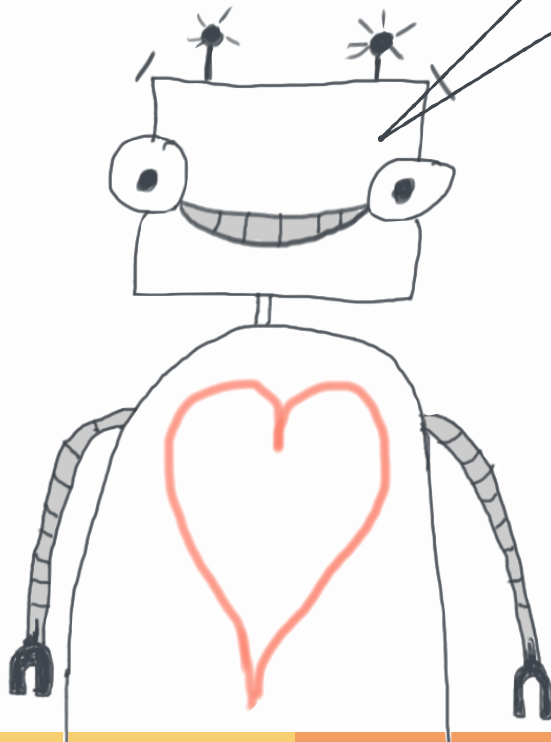
CustomerVerifiedEvent

CartCheckedOutEvent

CreateCustomerEvent

WillSaveItemEvent

DoStuffEvent





An Event is always immutable



**Loan Details  
Entered**

**Financial  
Situation  
Entered**

**Personal  
Information  
Entered**

**Credit  
Application**

**Application  
Submitted**

**Scoring**

...

...

**Let's reuse the ESB  
from the failed SOA  
project**





**No**



**No**

# Talking about Grails



**Loan Details  
Entered**

**Financial  
Situation  
Entered**

**Personal  
Information  
Entered**

**Application  
Submitted**

Use the  
**Grails 3  
Event System**  
for handling of  
internal Events

Use  
**RabbitMQ  
or  
Kafka**  
for handling of  
external Events

# Options for Events

## Event with complete payload

- › The Event contains all relevant data
- › Enables complete async processing
- › Payload is big and can fill up your message broker

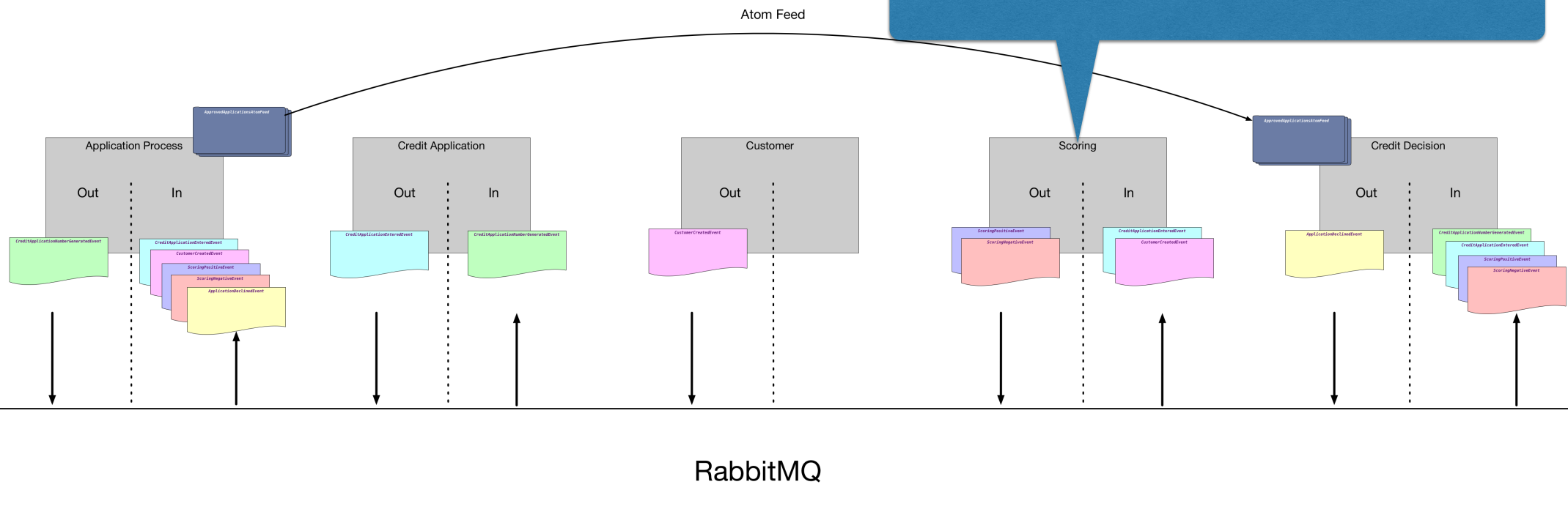
## Event with a REST URL

- › The Event only contains a URL to a REST-Resource and some minimal data, that is usually always needed
- › Small payload that is fast to transport
- › Standard cases work completely asynchronous, more complex cases might require a synchronous roundtrip (without service lookup though)

## Empty Event as a Feed update notification

- › The Event has no payload at all, it is just a notification
- › The payload, be it data or a REST URL, is usually contained in an Atom Feed
- › Event is a polling trigger for the feed

Let's replace the Scoring one with a Grails 3 Microservice!



<https://github.com/mploed/event-driven-spring-boot>  
<https://github.com/mploed/event-driven-grails>



# THANK YOU

**I'll post links to slides and code on Twitter**

*Michael Plöd - innoQ*  
*@bitboss*