ASYNCHRONOUS

MESSAGING

WITH REBUS

Mogens Heller Grabe

mogens@rebus.fm

https://rebus.fm

@mookid8000

2019-02-06T18:00:00.1337+01:00

"MICROSERVICES"

"SERVICE-ORIENTED ARCHITECTURE"

"DISTRIBUTED SYSTEMS"

BOUNDED CONTEXT

BOUNDED CONTEXT

"DDD deals with large models by dividing them into different Bounded Contexts and being explicit about their interrelationships."

https://martinfowler.com/bliki/BoundedContext.html

BOUNDED CONTEXT OPPORTUNITIES:

MODELING

Greater flexibility with focused models

DISTRIBUTION

Things can be made independent in space/time

INTER-PROCESS COMMUNICATION

SYNCHRONOUS

Pros/Cons?

e.g.
- fetch information for UI
- update entity in domain model
- check status of ongoing process
- fetch details from another bounded context
- (...)

INTER-PROCESS COMMUNICATION

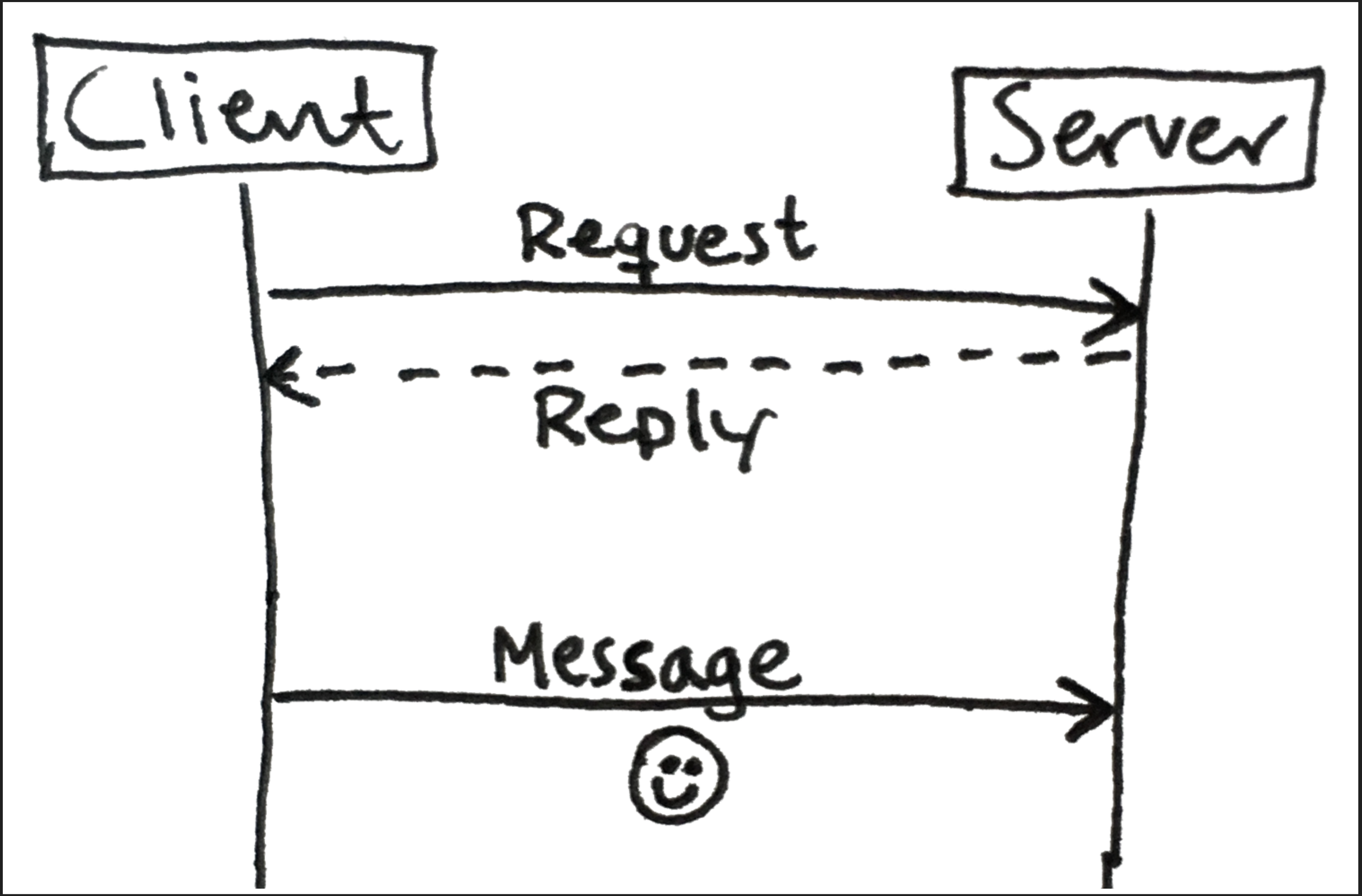
ASYNCHRONOUS

Pros/Cons?

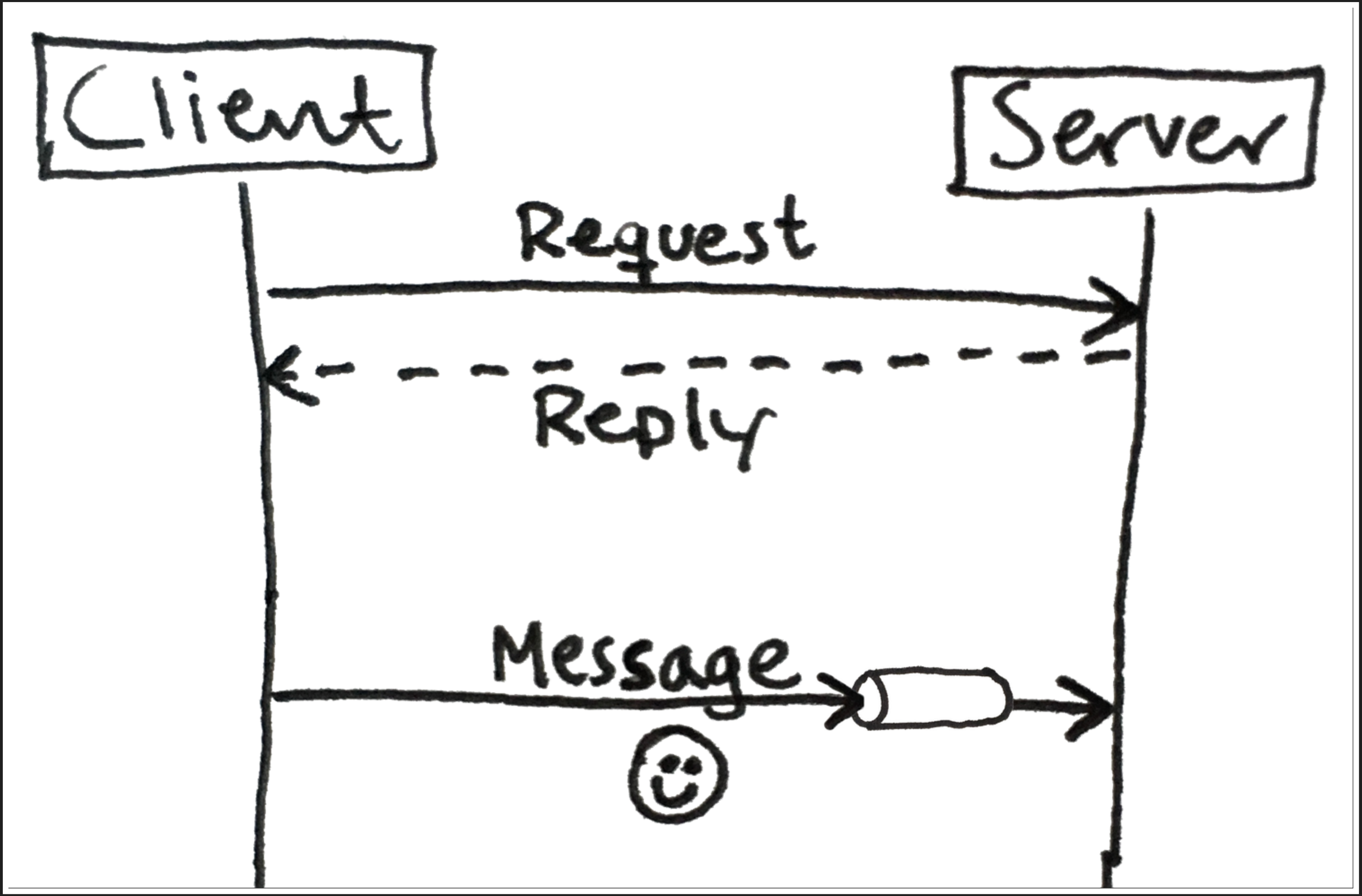
e.g.
- everything else! ;) like:
- process payment when credit card details have been received
- record change in inventory when an order has been placed
- quickly accept loads and loads of work to do

+ more (we'll look some more into this 🤓)

SYNC/ASYNC COMMUNICATION



SYNC/ASYNC COMMUNICATION



ASYNCHRONOUS COMMUNICATION

This is not async:

```
const string url = "http://www.erdetlillelillefredag.dk/api";

var response = await http.GetStringAsync(url);

Console.WriteLine($"Ja? {response}");
```

ASYNCHRONOUS COMMUNICATION

THIS is async:

```
await bus.Send(request);

// if only there was some way I could get a reply.... 🙃
```

ASYNCHRONOUS COMMUNICATION

MESSAGE BROKER

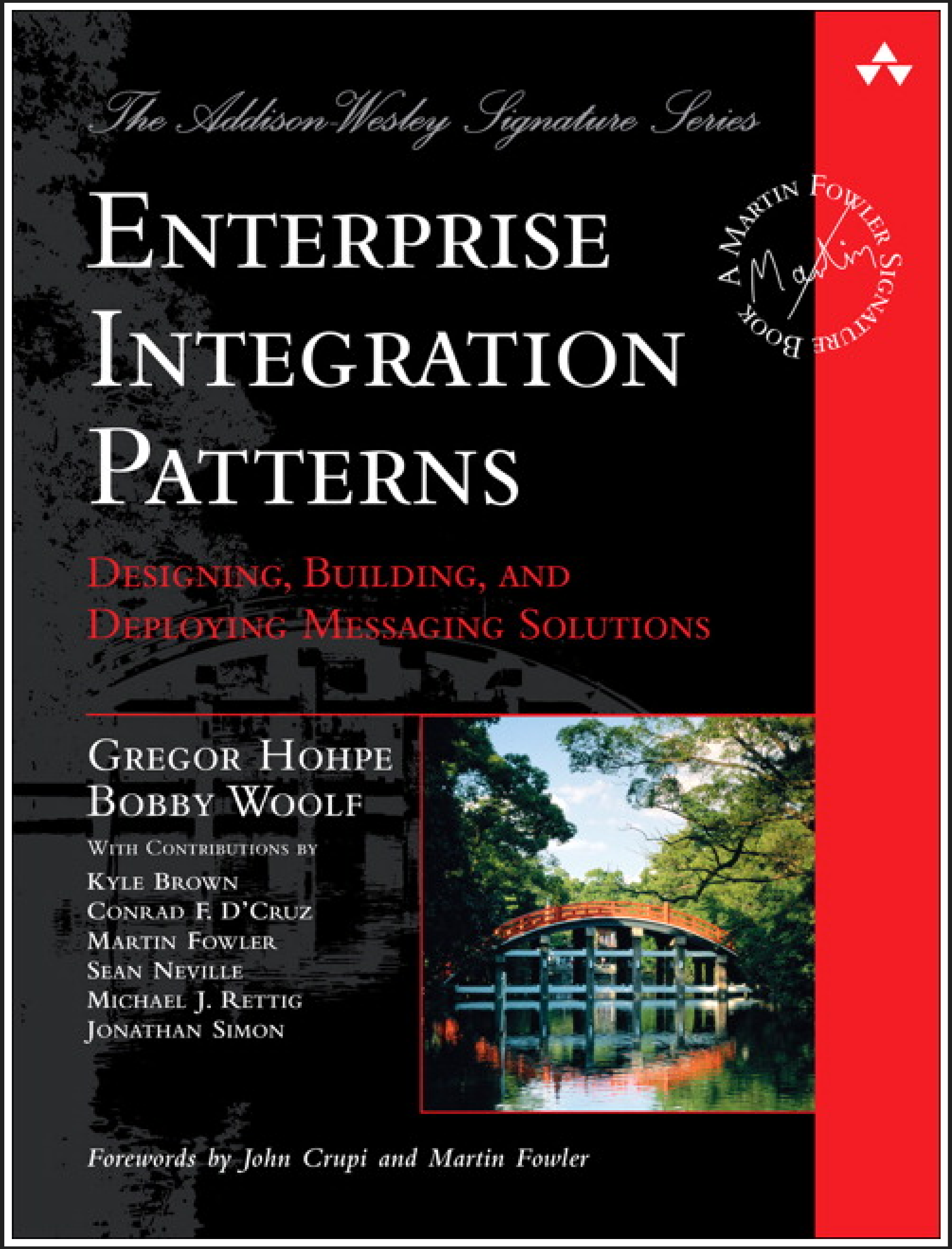
PEER TO PEER

LOG-BASED BROKER

RECOMMENDATION:

WRAP IT UP SOMEHOW

Don't make your systems dependent on their platform.
(more than necessary)



The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS

GREGOR HOHPE
BOBBY WOOLF

WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON

Forewords by John Crupi and Martin Fowler

http://www.enterpriseintegrationpatterns.com/

WHAT?

- Meet Rebus 🚐
- Four "enterprise problems" 🏢
  - Big system
  - Unreliable stuff
  - Spikes in load
  - Stuff that takes time
- More Rebus stuff
- You can leave now 🙂

HOW?

- Slides 📑
- Talk 🤭
- Code 🤓

MOGENS HELLER GRABE

Rebus FM

https://rebus.fm

mogens@rebus.fm

C# / .NET / distributed systems / modern databases / open source software

EXPERIENCE FROM

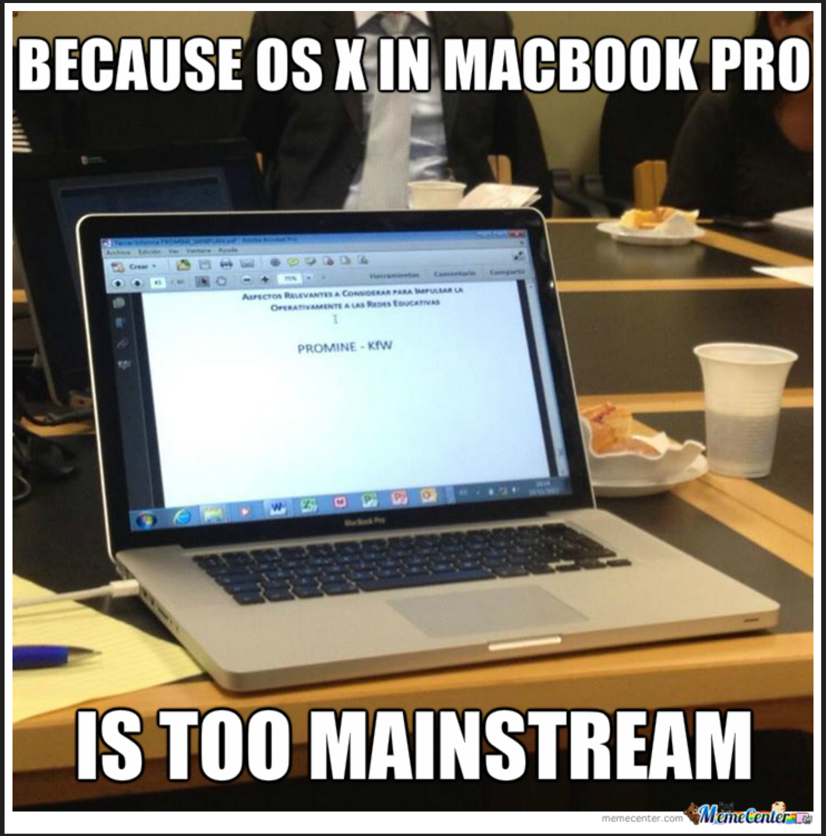
- Finance

  Mortgage deed trading
  Debt collection
- Energy

  Smart grid/Virtual power plant
- Trading

  Oil and power trading

ME



ENTERPRISEY.

YOU

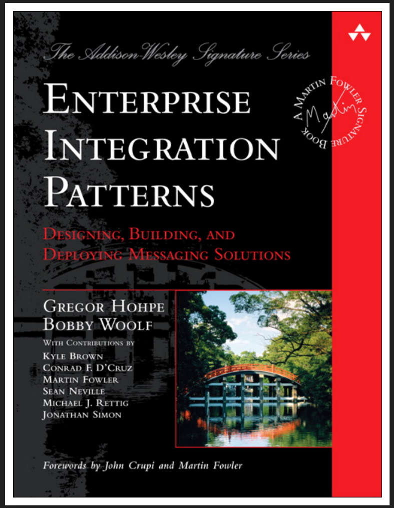
"I'VE GOT POSTGRES ON VINYL."

MEET REBUS

"SERVICE BUS"?

WHAT IS IT?

- Messaging library
- Implementation of some useful messaging patterns



The Addison Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS

A MARTIN FOWLER SIGNATURE BOOK

GREGOR HOHPE
BOBBY WOOLF

WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON

Forewords by John Crupi and Martin Fowler

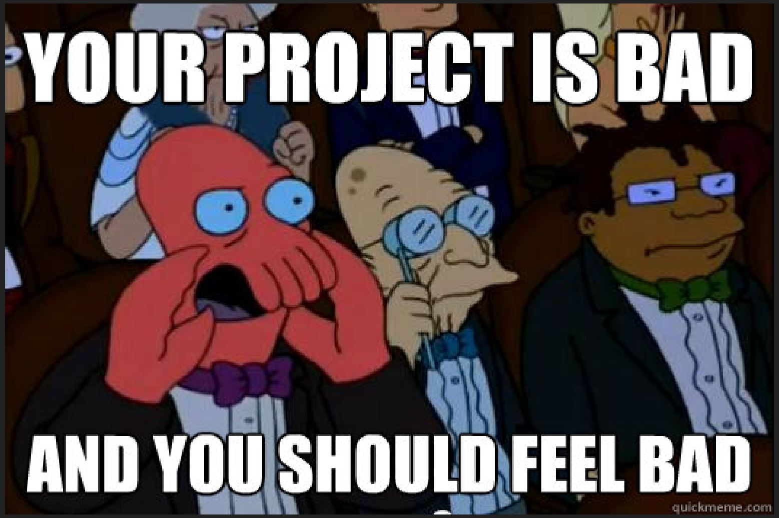
- Layer on top of queues

WHAT IS IT?

- NuGet package(s) (1 + 30+)
- Targets .NET 4.5 + 4.6, and .NET Standard 2.0
- Additional DLLs if you want
    - Amazon
    - Azure
    - MSMQ
    - SQL Server
    - PostgreSQL
    - RabbitMQ
    - MongoDB
    - Castle Windsor
    - Autofac
    - StructureMap
    - Unity
    - Ninject
    - Microsoft Extensions DI
    - Log4net
    - NLog
    - RavenDB
    - Oracle
    - MySQL
    - …

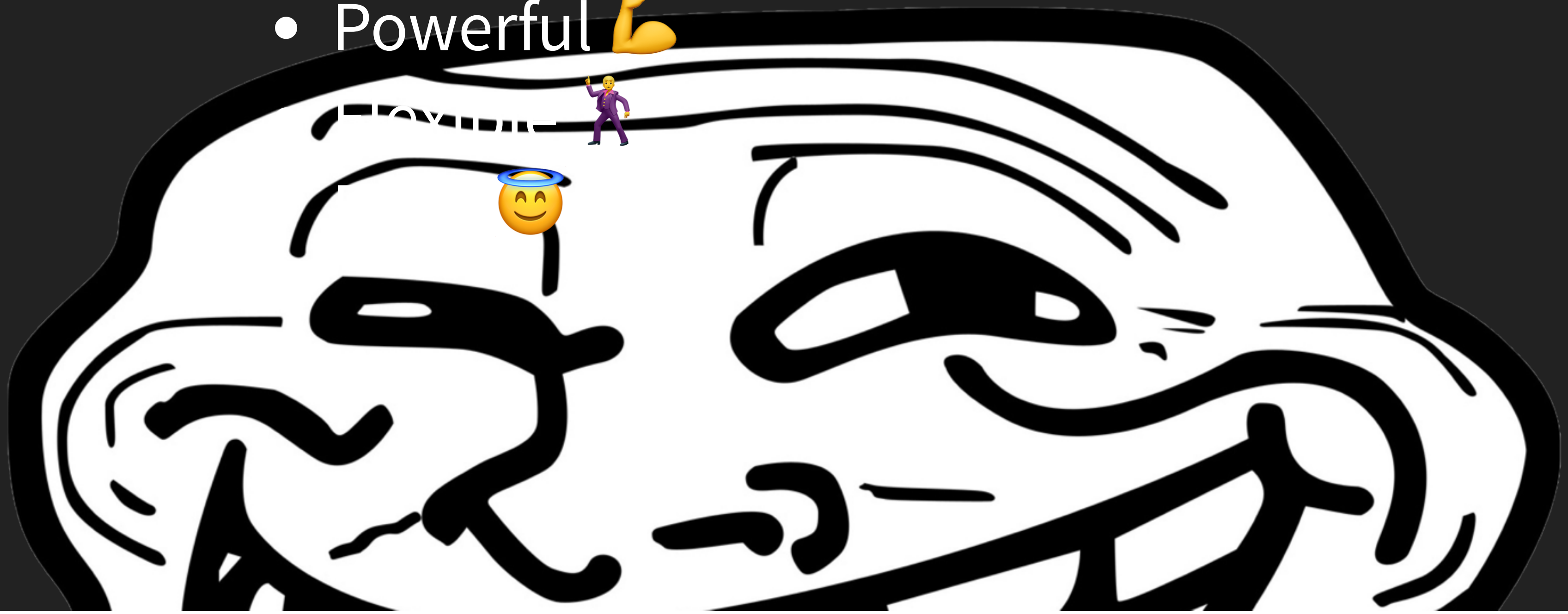
MOTIVATION

- Used to use NServiceBus
- Could not use NServiceBus where I wanted to though
- NServiceBus had really bad error messages
- Wanted to use MassTransit but I couldn't make it work
- Wanted to fork NServiceBus when it was still Apache V2



GOALS

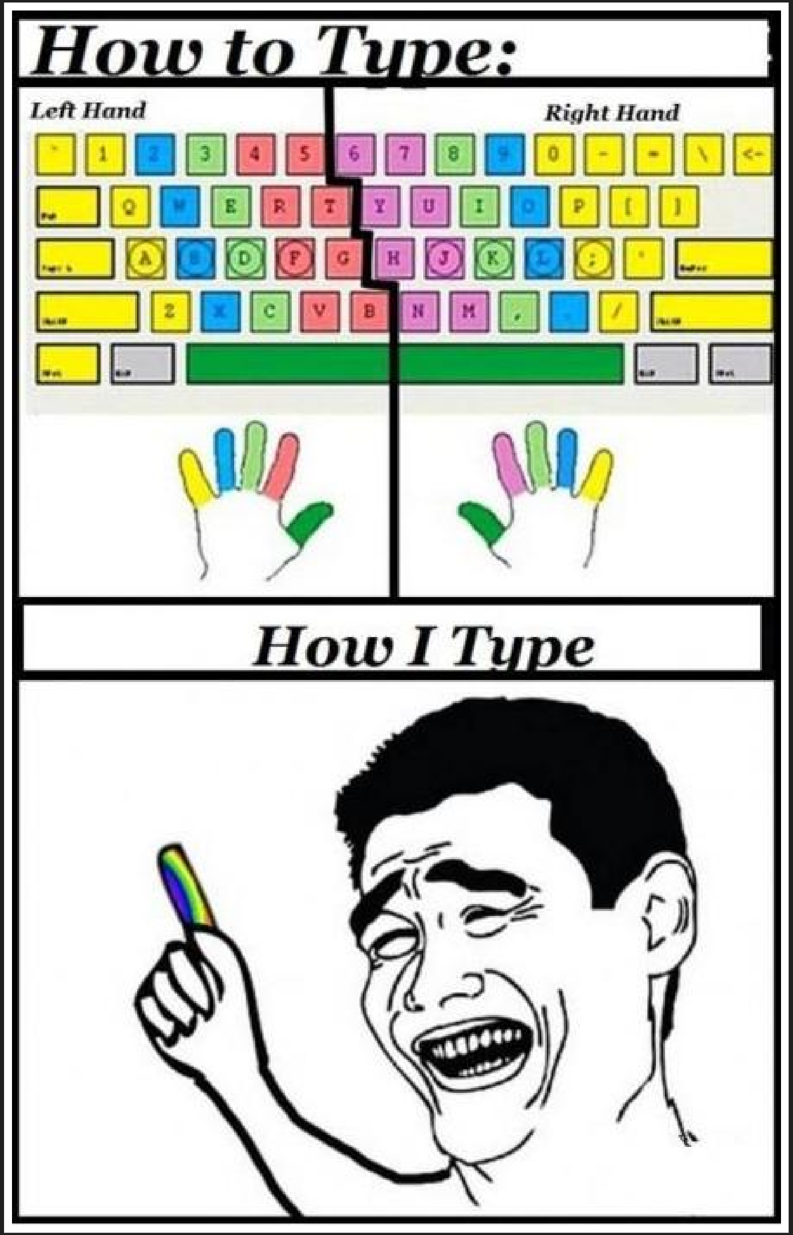
- Welcoming 👐
- A pleasure to use 👌
- Mostly stay out of the way 👀
- Powerful 💪
- Flexible 🕺
- [illegible] 😇



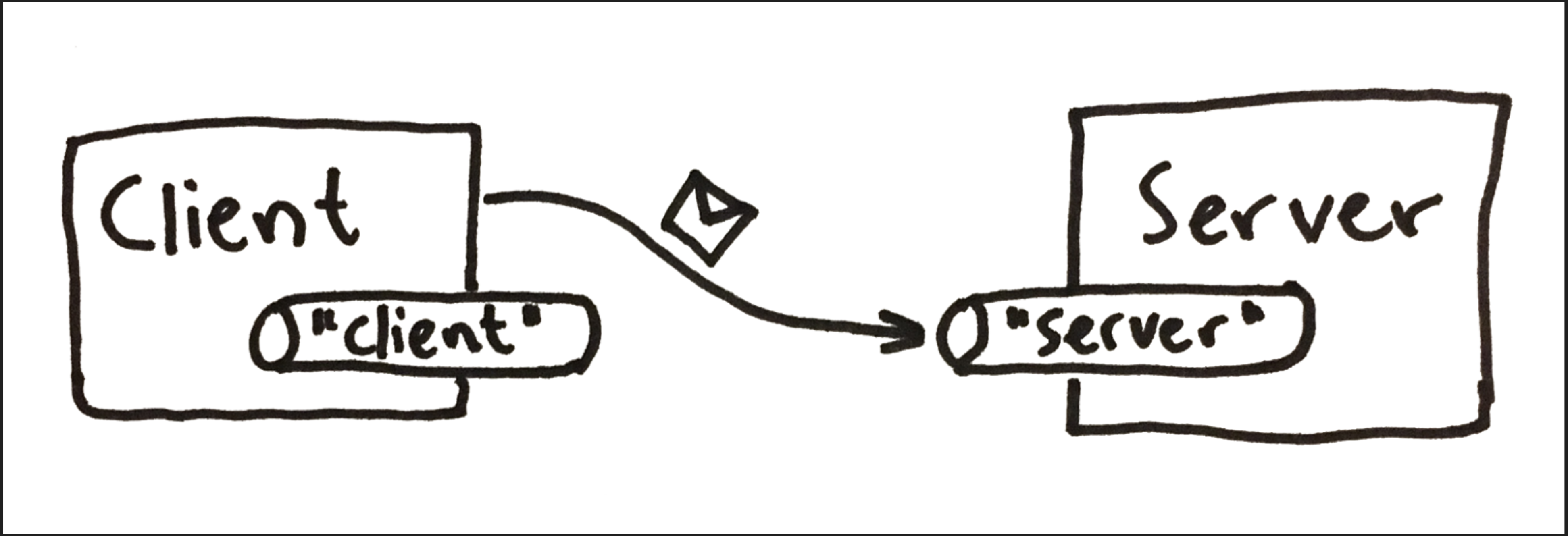
META

- First commit: 20th Sep. 2011
- ~4000 lines of C# 7
- Code on GitHub: https://github.com/rebus-org/Rebus
- Extensions on GitHub too: ?q=rebus.
- Has contributions from > 50 developers besides me
- Binaries available via NuGet http://nuget.org/packages?q=rebus
- Current version: 5.2.1
- Has been making money transactions and controlling power plants since 0.17-alpha

🚀 DEMO 0



DEMO 0 SUMMARY



DEMO 0 SUMMARY

- `await bus.Send(...)` sends to 1 recipient
- The recipient is determined with an "endpoint mapping"
- We mapped `System.String` to the `server` queue

FOUR "ENTERPRISE PROBLEMS"

1: BIG SYSTEM
SIZE

2: UNRELIABLE STUFF

Call other people's fucked up APIs... and get dragged down with them.

3: SPIKES IN LOAD

Once in a while, something extraordinary happens, and then we have too much stuff going on.

4: STUFF THAT TAKES TIME

Stuff that makes stuff happen

...that makes other stuff happen

...that makes other stuff happen

...then waits for a while

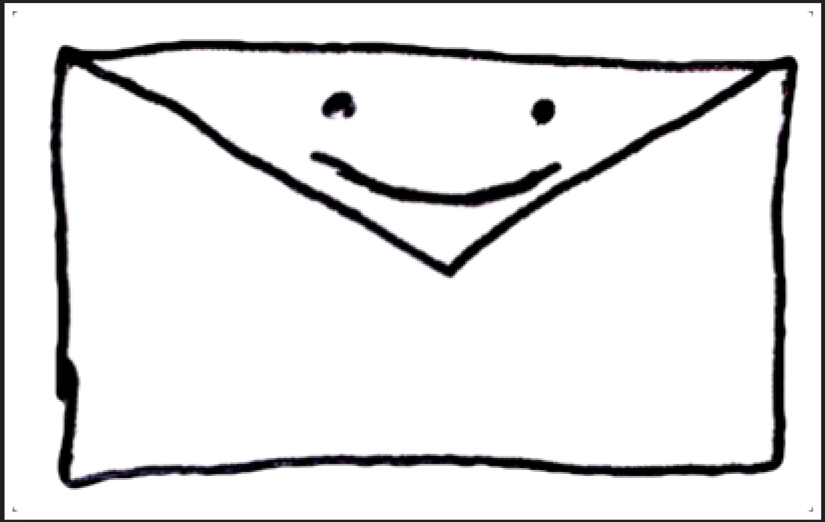
...then other stuff happens

(you probably get it)

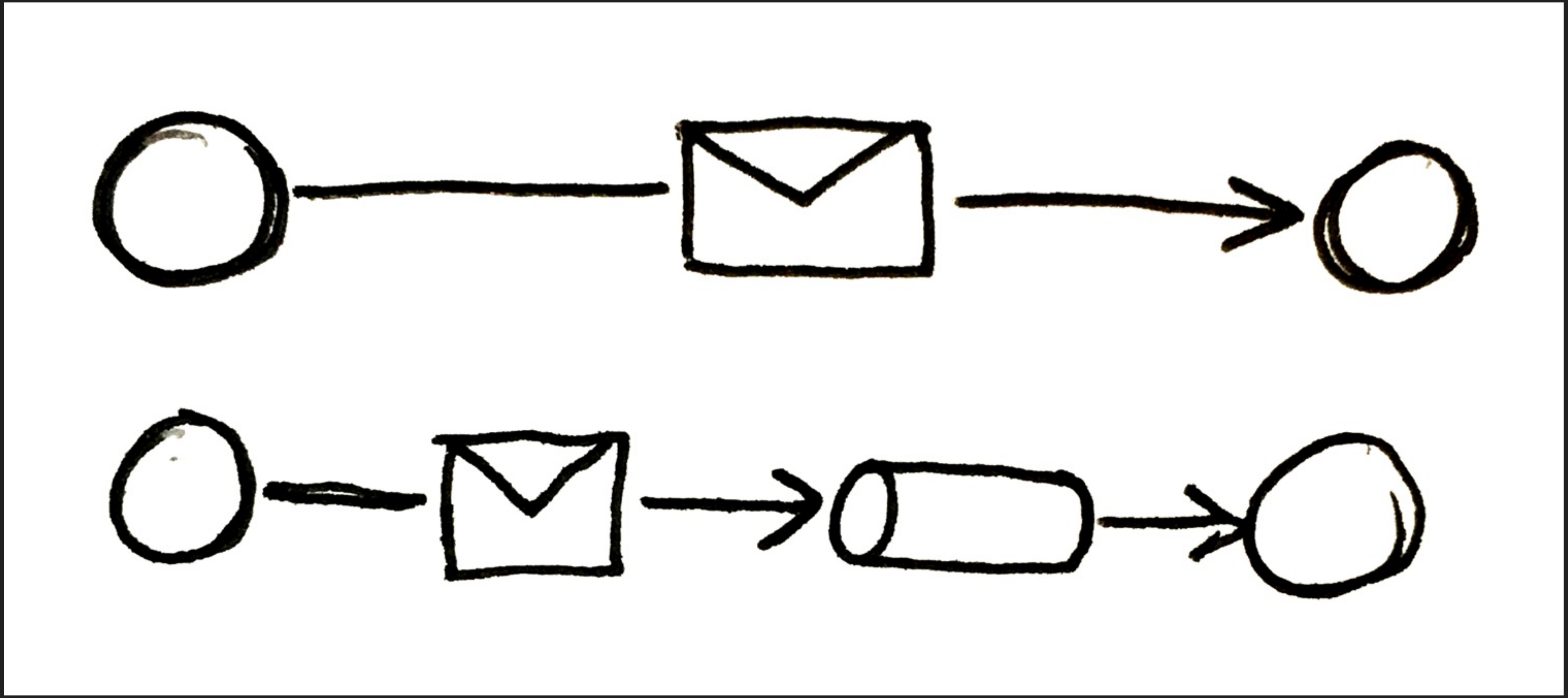
PROBLEM SUMMARY

- Monolith
- Integration
- High load
- Coordination

A SOLUTION: ASYNCHRONOUS MESSAGING



MESSAGING VIA DURABLE, ASYNCHRONOUS QUEUES



WINDOWS FTW!!1

AZURE FTW!!1

AWS FTW!!1

EXAMPLES

The four problems I talked about

1. System that's becoming too big
2. Integration with external parties
3. Unpredictable load spikes
4. Complex logic with coordination and timing

1ST PROBLEM

System that's becoming too big

Scenario: Trading – traders in front office ("Trading") make deals with counterparties, and the billing staff in back office ("Invoicing") ensure that trades get invoiced.

CUES

- bounded context
- pub/sub messaging

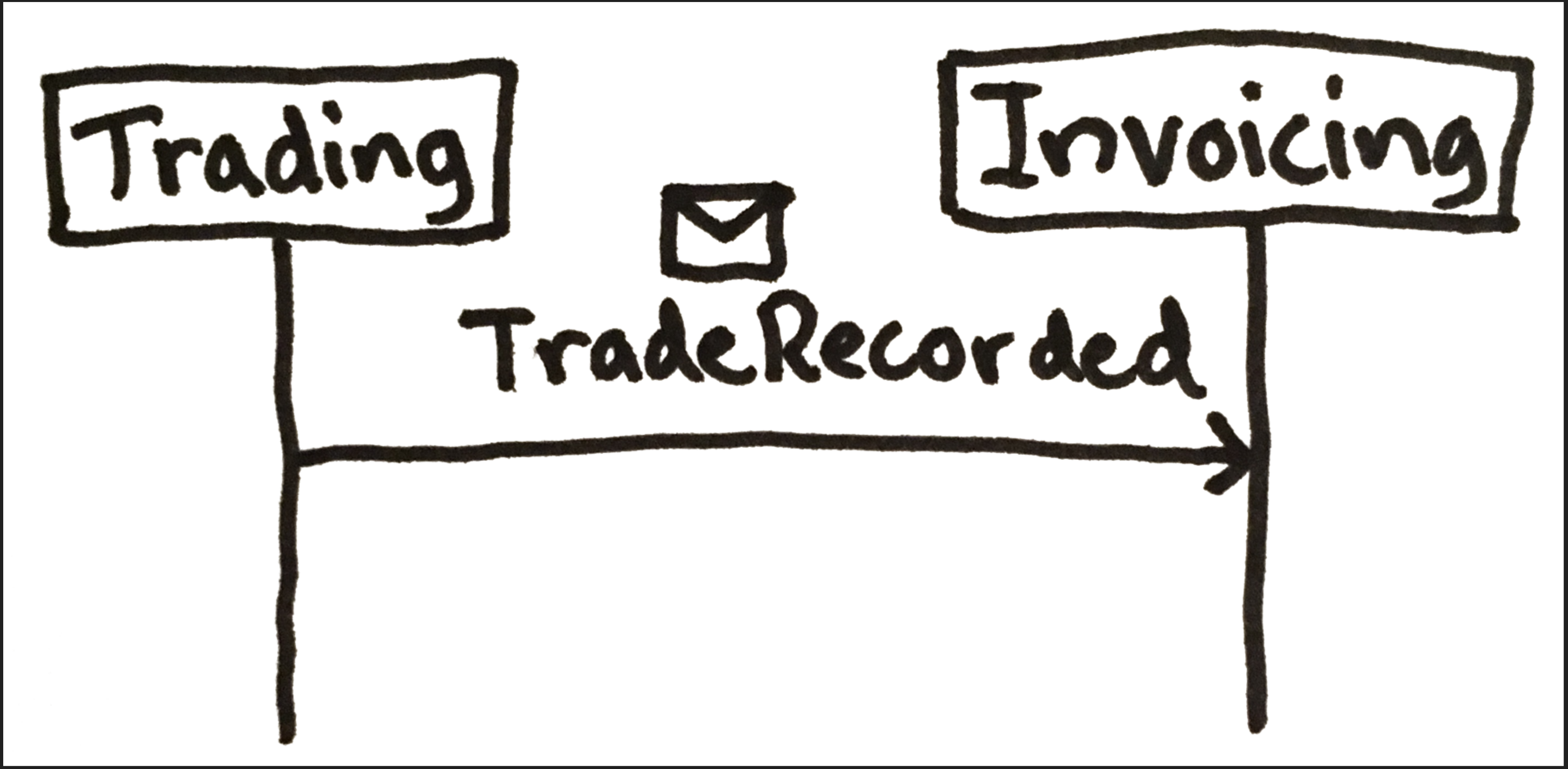
🚀 DEMO 1

Split into separate Trading and Invoicing systems

DEMO 1 SUMMARY

- Two separate bounded contexts...
- ...as two separate processes
- Publish/subscribe messaging
- Connected only by shared message DLL

DEMO 1 SUMMARY



2ND PROBLEM

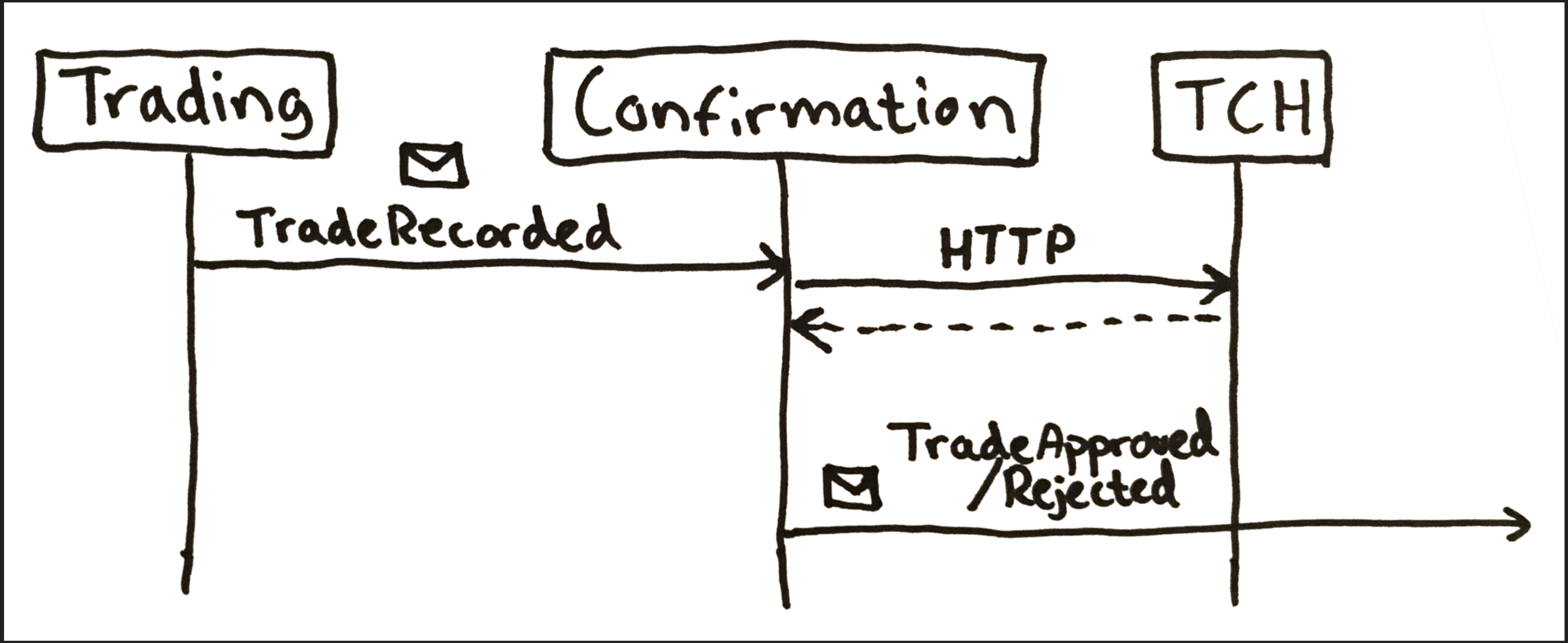
Integration with external party

Summary: When a new trade is made, middle office ("Confirmation") needs to confirm the trade with the counterparty. This can be done automatically by contacting a "clearing house", who is nice enough to provide a HTTP API for us to use.

The **TradeClearingHouseApi** HTTP service is pretty shaky, though..... 😩

CUES

- reliability
- automatic retries
- dead-letter queue


Trading
Confirmation
TCH
TradeRecorded
HTTP
TradeApproved
/Rejected

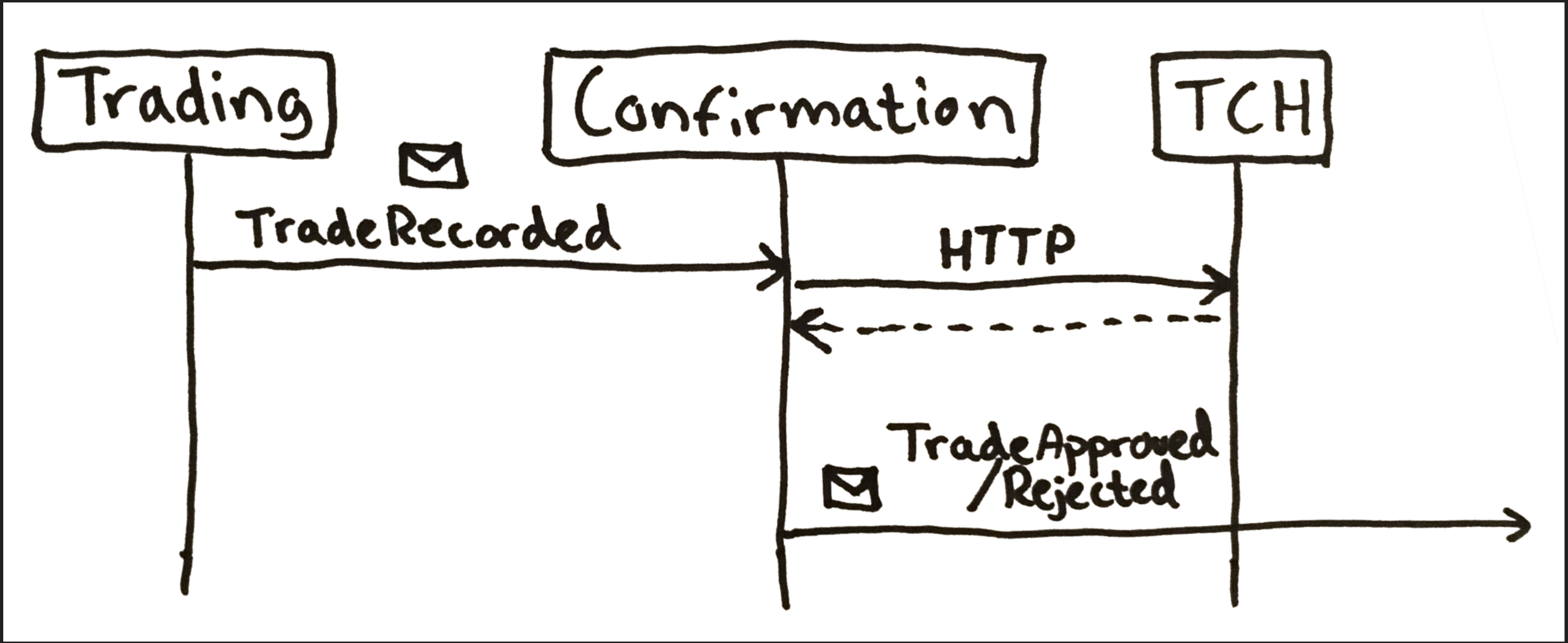
🚀 DEMO 2

External HTTP service calls with automatic retries

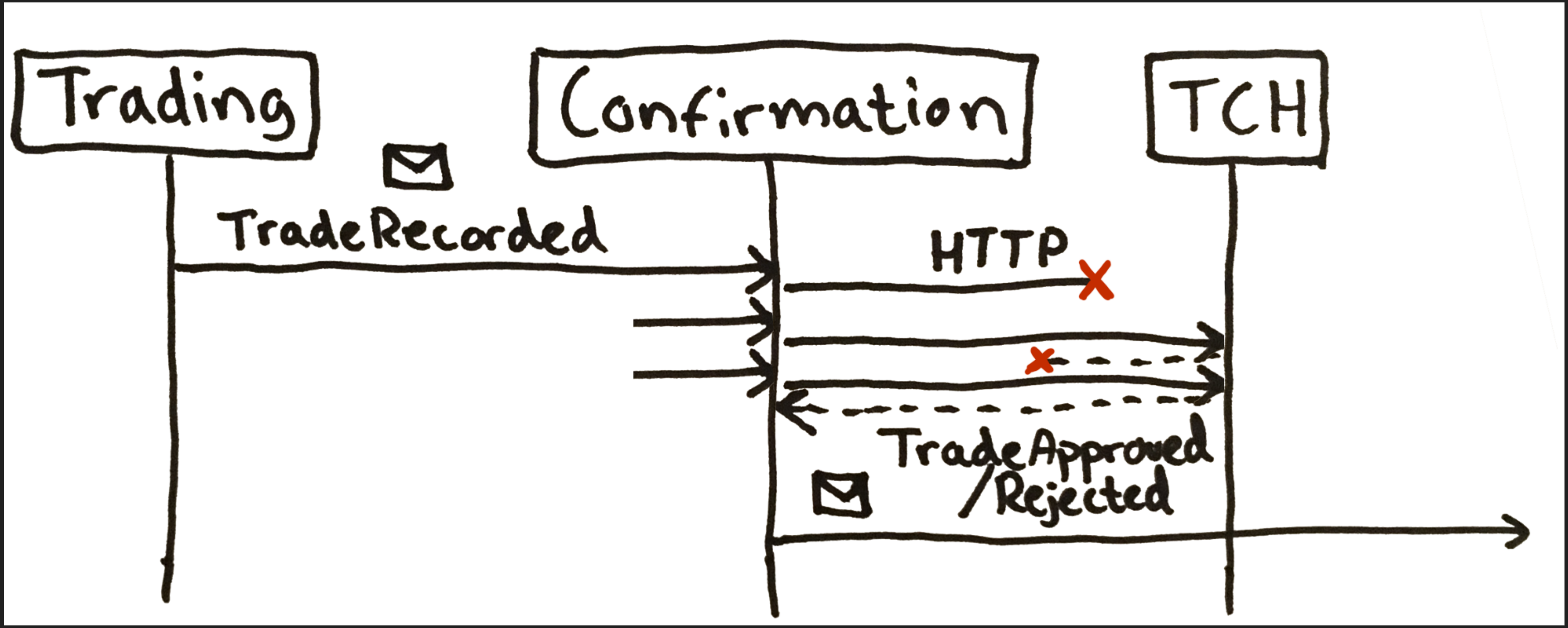
DEMO 2 SUMMARY

- Unreliable operation made more likely to succeed
- Can never end up not succeeding

DEMO 2 SUMMARY



DEMO 2 SUMMARY



3RD PROBLEM

Sudden spikes in load

When the world is trading a lot, the clearing house can have a hard time keeping up.

We're in luck though, because we do all of our work asynchronously and therefore temporally decoupled.

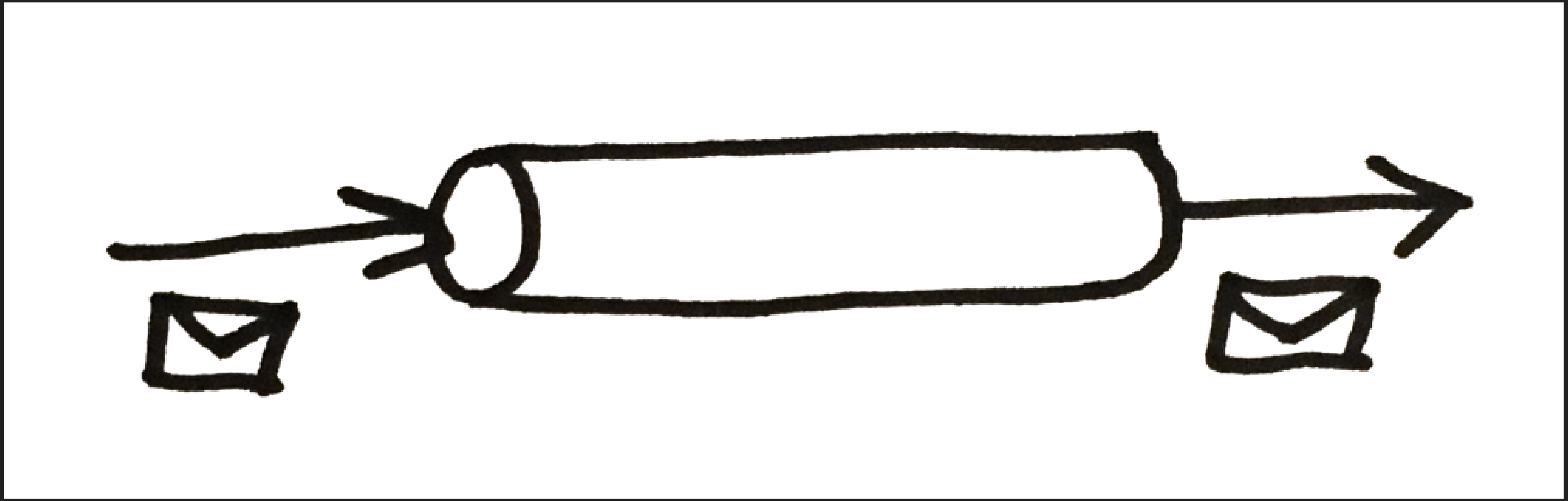
CUES

- durable message queue
- store-and-forward

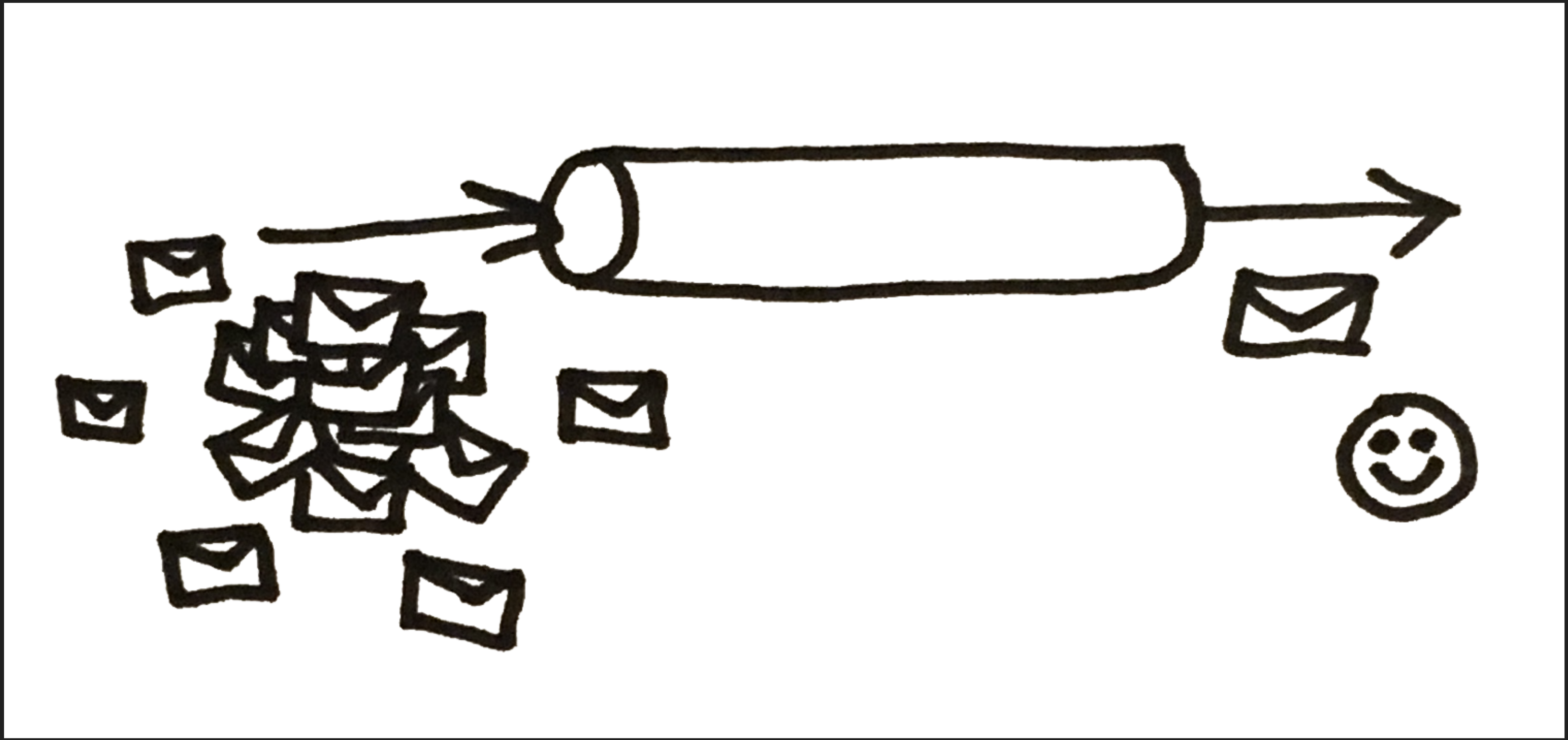
🚀 DEMO 3

Reduce impact of slow parts of the system by using asynchronous messaging.

DEMO 3 SUMMARY



DEMO 3 SUMMARY



4TH PROBLEM

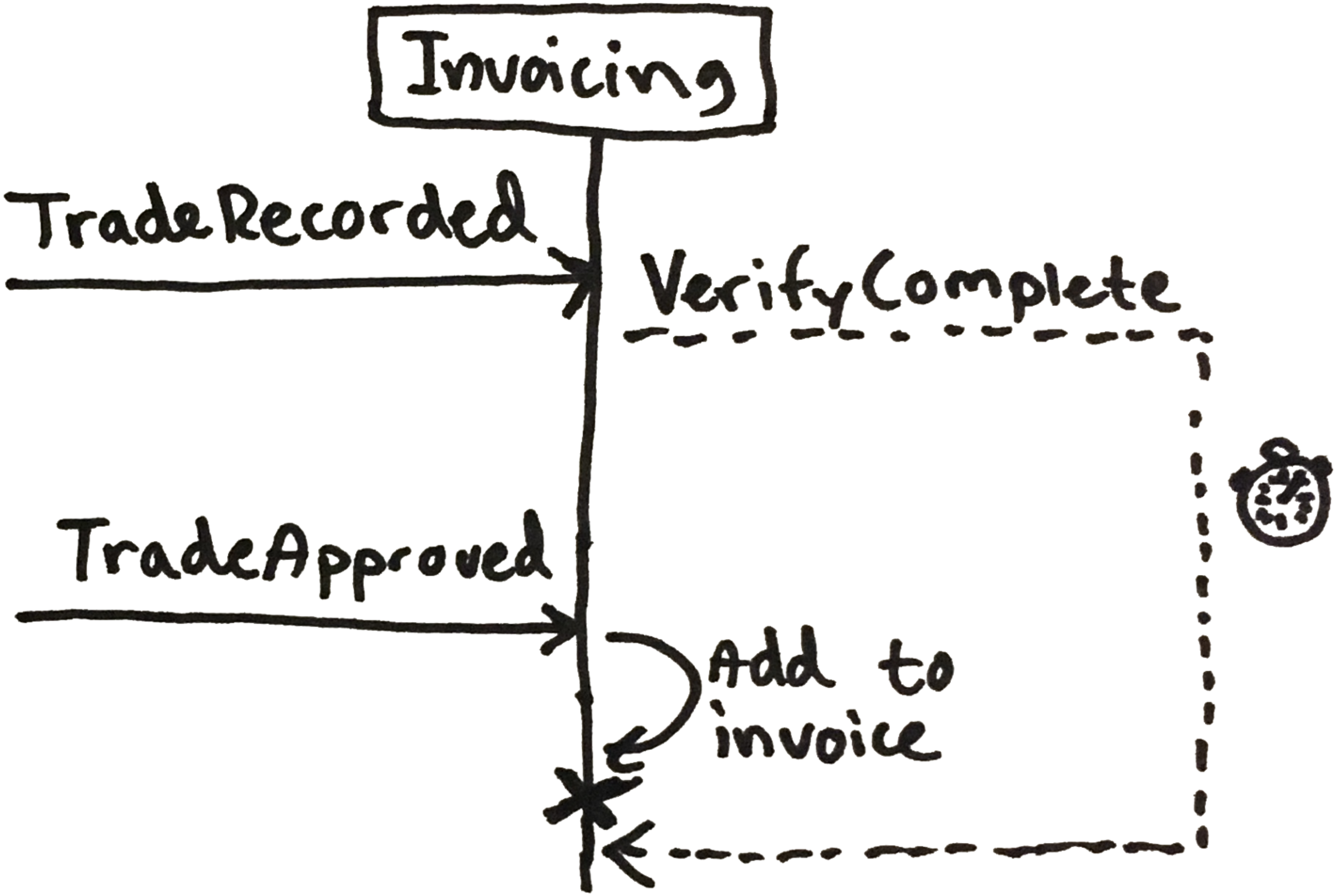
Complex coordination and timing

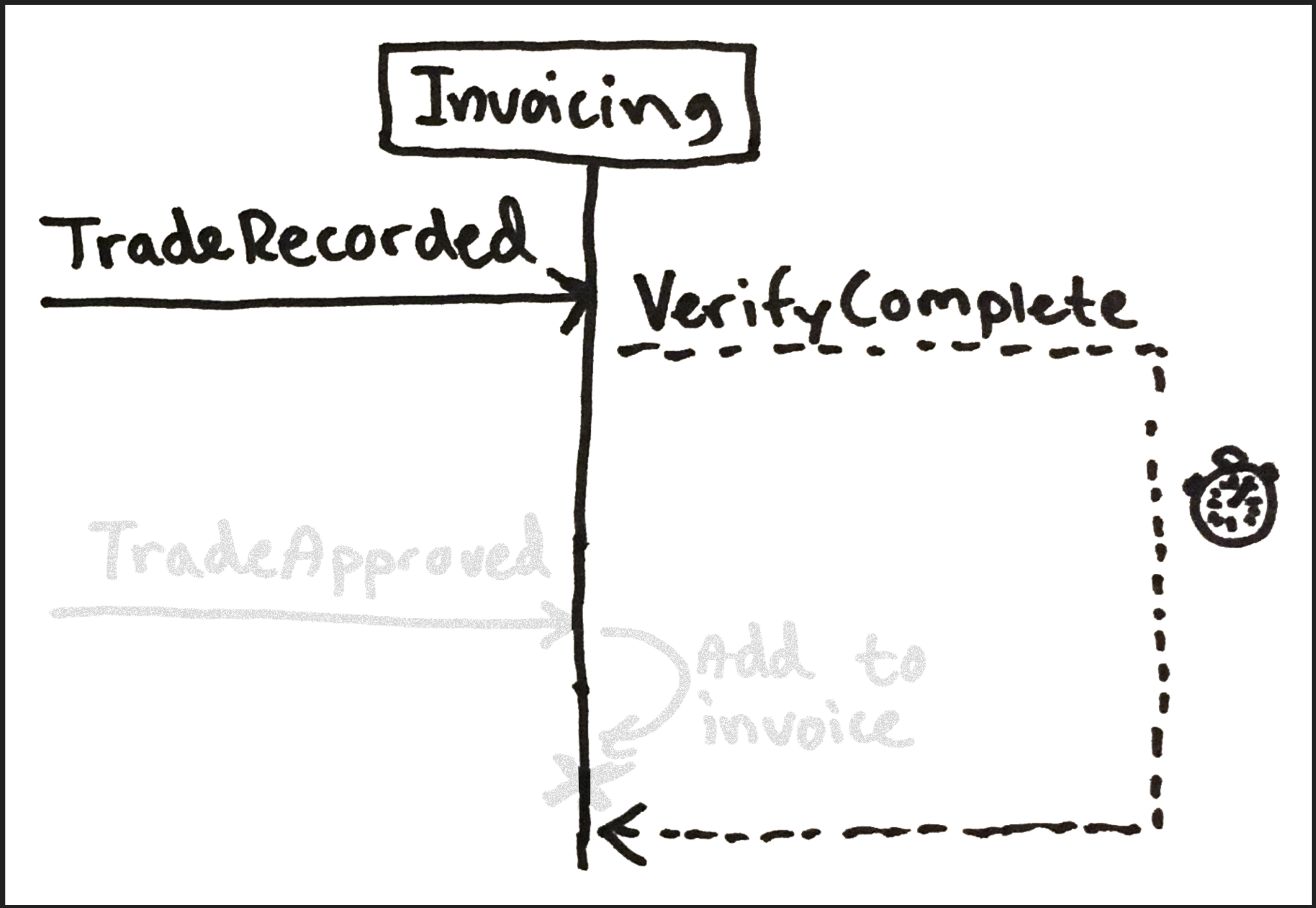
Invoicing must add confirmed trades to invoices (let's just assume rejected trades are handled elsewhere).

If the clearing house fails completely, we must take some kind of alternate action, ensuring that we never "forget" a trade.

CUES

- process manager ("saga")
- timeouts
- compensating actions


Invoicing
TradeRecorded
VerifyComplete
TradeApproved
Add to invoice


Invoicing
TradeRecorded
VerifyComplete
TradeApproved
Add to invoice

🚀 DEMO 4

Complex logic, coordination, timing

DEMO 4 SUMMARY

- Asynchronous operations participating in process
- Process has a timeout with an alternate path
- Messages are correlated with process instance
- Business workflow made explicit

DEMO 4 SUMMARY

- "Saga" = state machine running on messages
- Saga handler `Data` property = current state
- Correlation configuration defines how to find current state

SUMMARY

Messaging provides a model that can...

- help you glue the pieces together when you break down a system into bounded contexts
- help you overcome glitches when you depend on stuff that you do not control
- be used to make processes explicit by correlating events

STUFF I DIDN'T SHOW

Rebus can

- also use RabbitMQ, Azure Service Bus, Amazon SQS, SQL Server, + more as transports
- store subscriptions, sagas, and timeouts in SQL Server, RavenDB, MongoDB, + more
- activate handlers with Castle Windsor, StructureMap, Autofac, Ninject, Unity, Microsoft Extensions DI, SimpleInjector + more
- log with NLog, Log4Net, and Serilog
- do handler pipeline re-ordering
- do polymorphic dispatch
- compress/encrypt message bodies

WHAT NOW?

- Fleet Manager

  Insights into + management of Rebus endpoints.

  Available now via "Rebus Pro"

- Service Discovery

  Integrating modern configuration mgt. tools like Consul, etcd, etc.

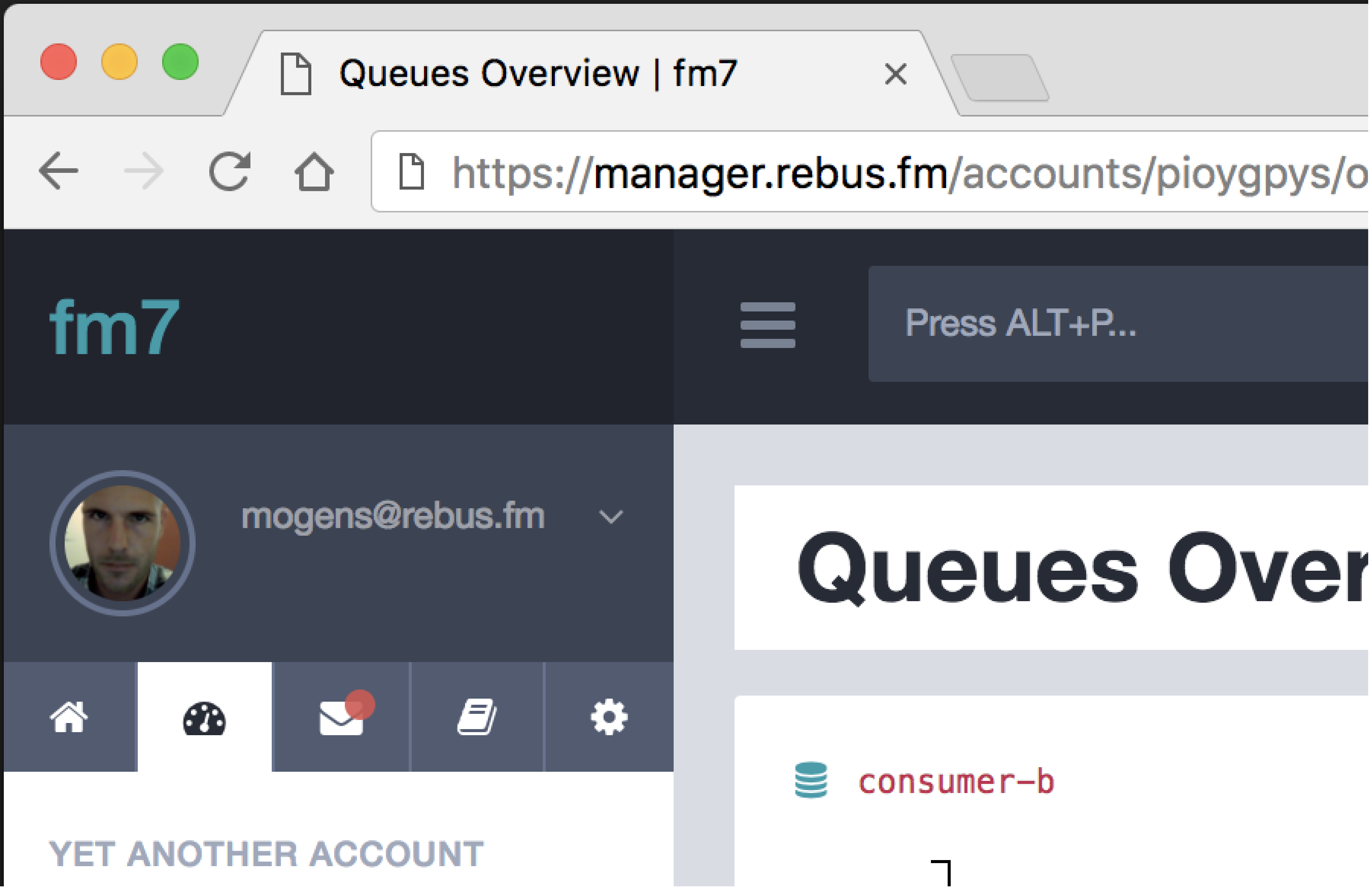
- Additional transports and persistence libraries

  It's never enough.

- Maybe an "outbox"

  Makes it slightly easier to implement idempotency

FLEET MANAGER



Queues Overview | fm7

https://manager.rebus.fm/accounts/pioygpys/o

fm7

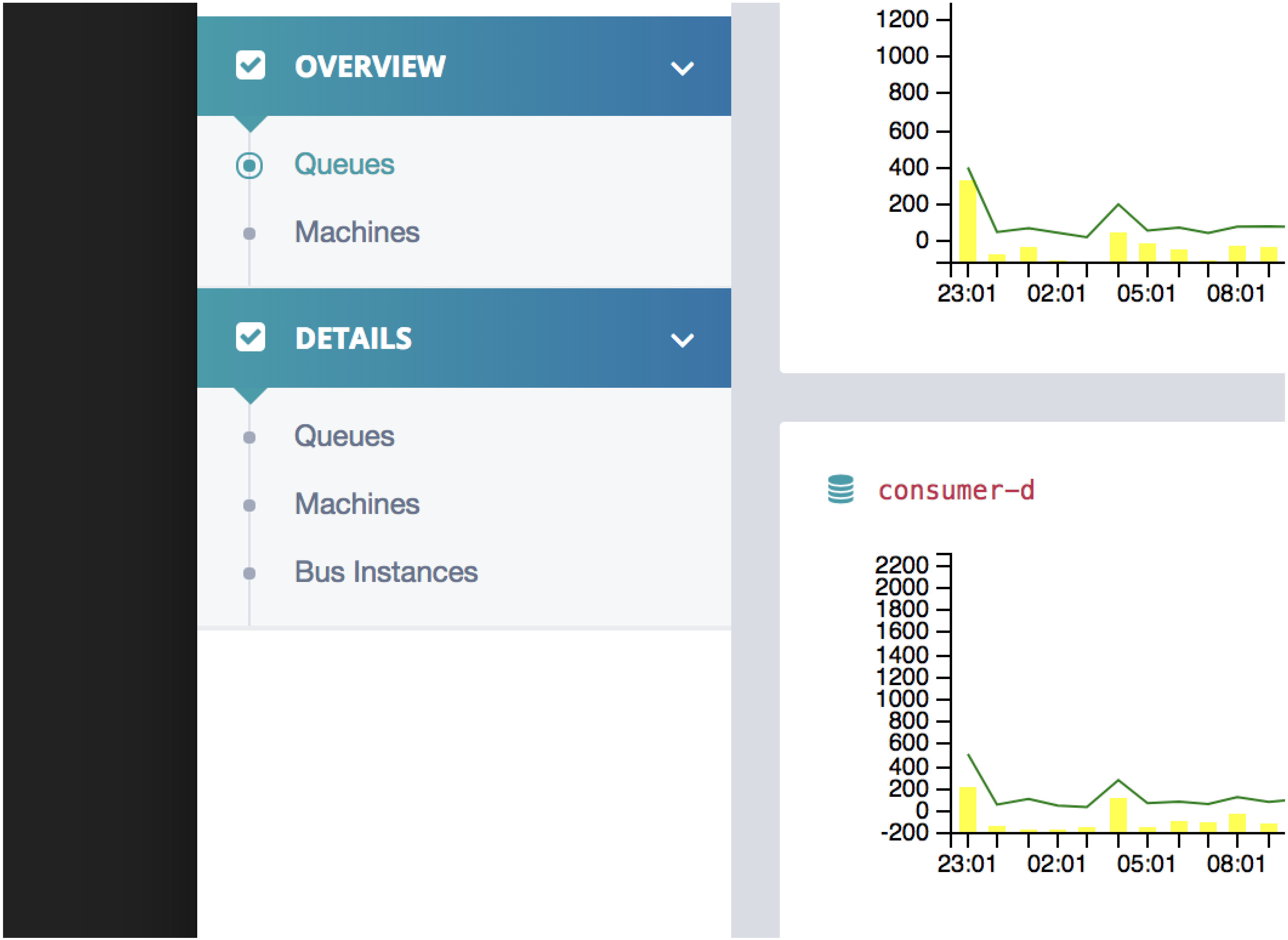
Press ALT+P...

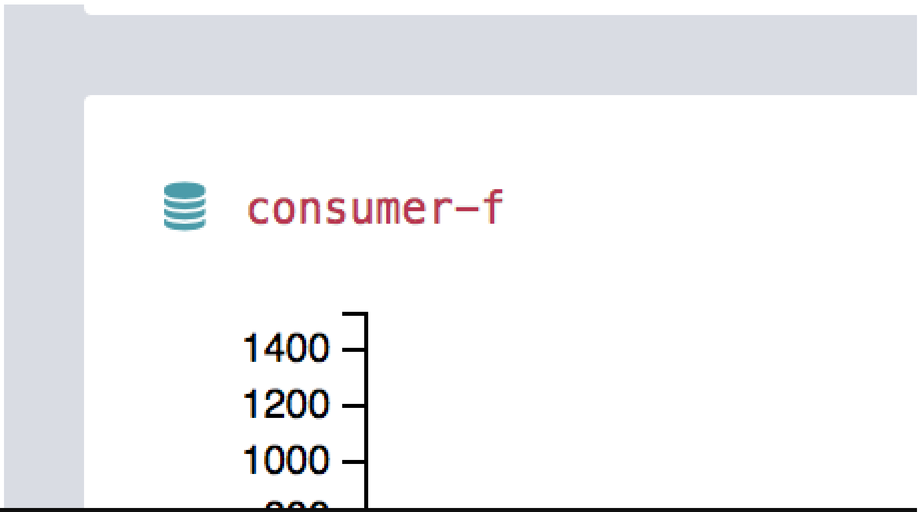
mogens@rebus.fm

Queues Over

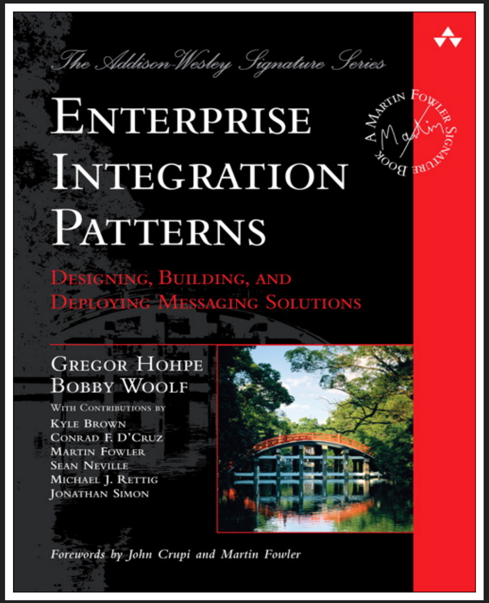
consumer-b

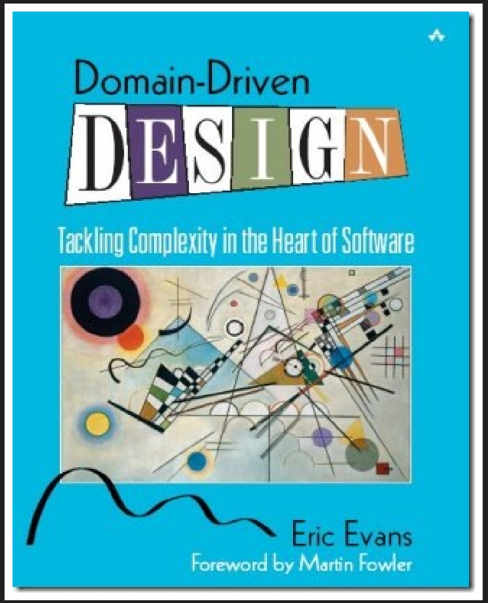
YET ANOTHER ACCOUNT


OVERVIEW
Queues
Machines
DETAILS
Queues
Machines
Bus Instances
consumer-d
1200
1000
800
600
400
200
0
23:01
02:01
05:01
08:01
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
-200
23:01
02:01
05:01
08:01


consumer-f
1400
1200
1000

LITERATURE


The Addison-Wesley Signature Series
Enterprise Integration Patterns
Designing, Building, and Deploying Messaging Solutions
Gregor Hohpe
Bobby Woolf
With Contributions by
Kyle Brown
Conrad F. D'Cruz
Martin Fowler
Sean Neville
Michael J. Rettig
Jonathan Simon
Forewords by John Crupi and Martin Fowler
A Martin Fowler Signature Book


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Free catalog:
Gregor's homepage

Free short version:
DDD quickly (InfoQ)

WANT TO READ MORE?

Check out books & blog posts by Gregor Hohpe, Michael T. Nygard, Udi Dahan, Sam Newman, and Greg Young.

THANK YOU FOR LISTENING!

...and a big thank you to Hakim for creating the immensely awesome reveal.js

Sample code: https://github.com/rebus-org/RebusDemos

rebus.fm

Mogens Heller Grabe

mogens@rebus.fm

https://rebus.fm

@mookid8000