# Event Sourcing at Yammer

Michał Rutkowski (mrutkowski@yammer-inc.com) Dmitry Stratiychuk (dstratiychuk@yammer-inc.com) Philipp Fehre (pfehre@yammer-inc.com)



- Challenges Yammer faced
- Why event sourcing?
- How we've rolled it out
- What we've learned
- The future

- Challenges Yammer faced
- Why event sourcing?
- How we've rolled it out
- What we've learned
- The future

- Challenges Yammer faced
- Why event sourcing?
- How we've rolled it out
- What we've learned
- The future

- Challenges Yammer faced
- Why event sourcing?
- How we've rolled it out
- What we've learned
- The future

- Challenges Yammer faced
- Why event sourcing?
- How we've rolled it out
- What we've learned
- The future

### What is Yammer?

# An Enterprise Social Network facilitating better and faster communication within an organization.

III Office 365 Yammer	Microsoft	🔅 ? Michal Rutkowski 🌄	III Office 365 Yammer	Microsoft	🌲 🌣 ? Michal Rutkowski 🎧
★ ● ★ …	Inbox	Create Message		Y Yammer Engineering	√Joined 👔 🌣
Q Search	UNREAD 19 ALL Q		Q, Search		
MICROSOFT GROUPS	New constraints		MICROSOFT GROUPS	Yann Armand - June 3 at 6:10pm	SEARCH
Productionize Vent	from montain a set survey on care for the set from the set of the set of the set the set of the		Productionize Vent	No more need of vagrant for workfeed basic needs	Q. Search this group
😁 Yammer On-Call	<ul> <li>Bian Warten: Workfeed started dropping Veriff-blisher jobs, so I got alerted on that, I think that means that WF tried a</li> </ul>	numb 10m	🍅 Yammer On-Call	Happy Friday SF and Redmond fellows, London is already playing with this since 2 weeks and it is	
Yammer Services Team	New consention		Yammer Services Team	time to share with everybody.	INFO Edit
Y Yammer Engineering Yammer Royal Engineering	👔 🐘 Hann Editivityile is 🌢 Yanner Dir Gal	✓ ∅ Jun 2	Y Yammer Engineering Yammer Royal Engineering 1	No more need to install/setup vagrant on your mac to: - run workfeed unit/functiona/inunit tests. - run bundle install lopdate Gemilia.logda	Big Board
Yammer Service Oriented Archite	That insisten with wentable functing tig guildlab latency and and increase in SDTs. I edeployed wentable and that seems to Genes (Thang I) I believe so. Wavefront said they'd face a fits in the next version, which they said would be pushed to out		Yammer Service Oriented Archite		Yammer Engineering Documents
Yammer Messaging Domain Team 1		0	Yammer Messaging Domain Team 2	- run rails generate	Tech Lead Guide
¥ammer Managed Storage	New constantion	• Online Now 🌣 🔔	Yammer Managed Storage	Jscript/workfeed.rb will prepare and run a docker based playground with WF, postgres, memcache and freede-test. All fit in 2Gb of ram and only need 10 minutes download (max time measured in London) and boot in a	Online Now 🌣 🔔
Yammer Infrastructure Team	tions paying a cought of dama having to do with hig Tis pay for any payor. Hertigent is capable of preventing down to 1	Search for People	Yammer Infrastructure Team		Engineer Search for People
Yammer Security Private	<ul> <li>Michael Rutkowski: I'm 0000 kill Thursday. I'll work on that once I'm back.</li> </ul>		Yammer Security Private		Project E New Hiri Software Development
C hanner Acure Initiative	No consider		Yammer Azure Initiative	expand >	
Tammer Redmand Team	4 March Gaush in D Yannar Yann	Process Engineering M.	S Yammer Redmond Team		Aggrega
Yammer Search Domain	In comp of you already toxin, Lithiadii one binached years ago and data him. We nolliase uses in case in the "ranket" (a. 4 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/nexu/audi-factors take over mark sublicting c toither and pin e. 8 2000 A. Stanish - nizzed 100;//nexu.of/pedia.com/		Yammer Search Domain 1		Yammer Stratlychuk Dmitry
😪 Yammer Ruby Services			Vammer Ruby Services		What Ma
Yammer Reliable Data Storage	Unext may fine the set of the		Yammer Reliable Data Storage		Candida: 💮 Clark Breyman Principal Software Engl., 🛛
Yammer Eng Leadership			Yammer Eng Leadership		Vagrant
+ Create a new group  Discover more groups			+ Create a new group Discover more groups		Git and J Mario Caropreso ( Workfee
Michał Rutkowski	Non-pitche menage	SWE II, Glossarie Cop. L.	Michał Rutkowski		A/B testi
Microsoft	Michael Rothowski	SWE II, DIOSSANE CIIJI, C-	https://www.yammer.com/microsoft.com/notes/589249	+ IVE REPLY SHARE ···	So You'r

# What is important?

- Our customers:
  - -SLA
  - Performance
- Velocity: Ability to build and A/B test features fast

# What is important?

- Our customers:
  - -SLA
  - Performance
- Velocity: Ability to build and A/B test features fast

### The Team

We want to help rest of the engineering team in:

- building quality features fast,
- while meeting our SLA commitments

To that end we:

- drive discussion and adoption of architectural patterns
- adopt and if necessary build tooling to facilitate that

### The Team

We want to help rest of the engineering team in:

- building quality features fast,
- while meeting our SLA commitments

To that end we:

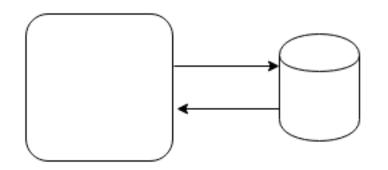
- drive discussion and adoption of architectural patterns
- adopt and if necessary build tooling to facilitate that

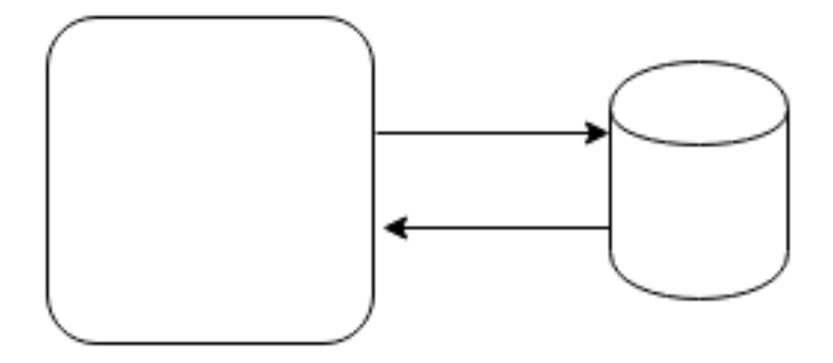
- Better tooling and process for release management:
  - Continuous delivery
  - Load testing
- Best practices for service design and development:
  - Testing for failure
  - Ensuring QoS
- Inter-service integration patterns

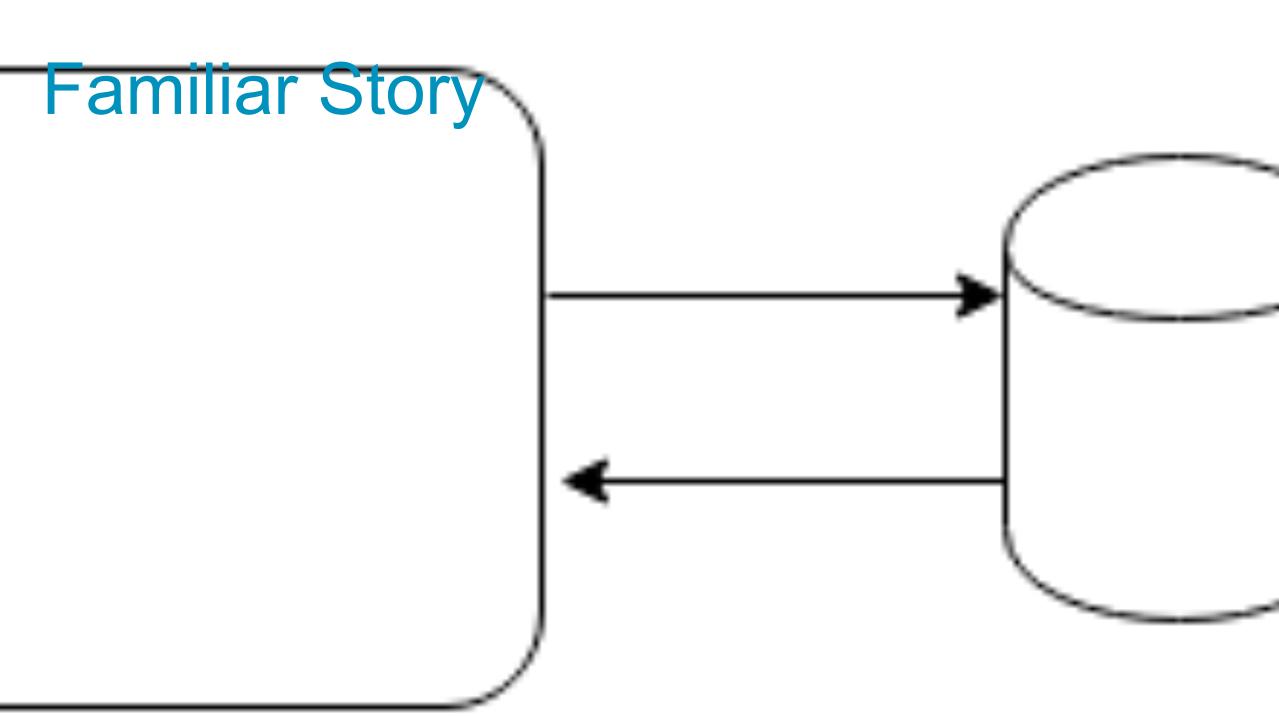
- Better tooling and process for release management:
  - Continuous delivery
  - Load testing
- Best practices for service design and development:
  - Testing for failure
  - Ensuring QoS
- Inter-service integration patterns

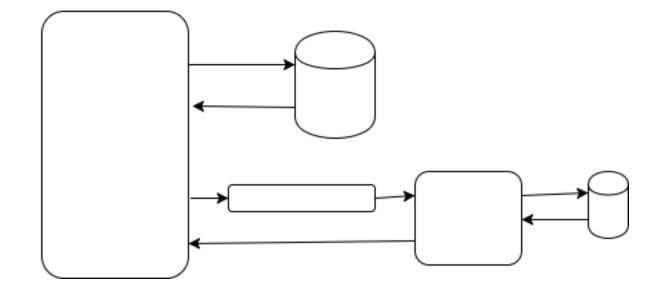
- Better tooling and process for release management:
  - Continuous delivery
  - Load testing
- Best practices for service design and development:
  - Testing for failure
  - Ensuring QoS
- Inter-service integration patterns

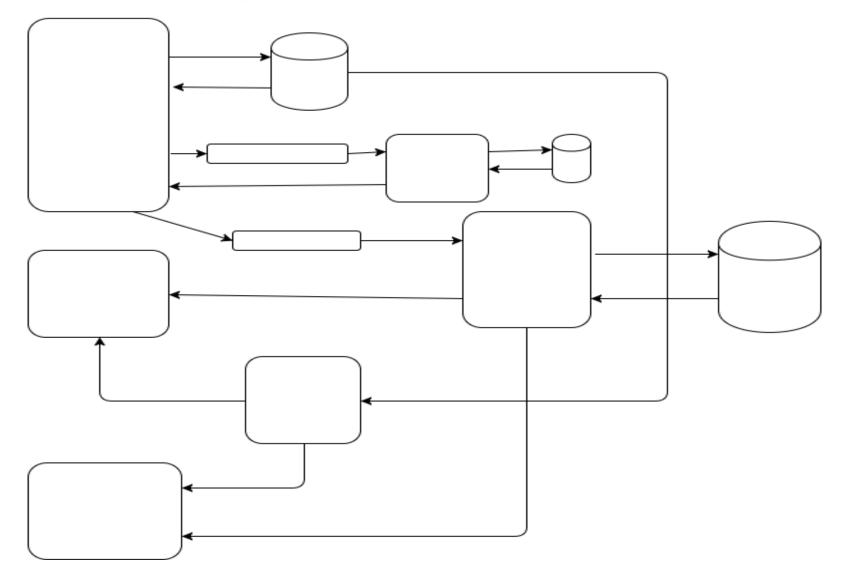
- Better tooling and process for release management:
  - Continuous delivery
  - Load testing
- Best practices for service design and development:
  - Testing for failure
  - Ensuring QoS
- Inter-service integration patterns

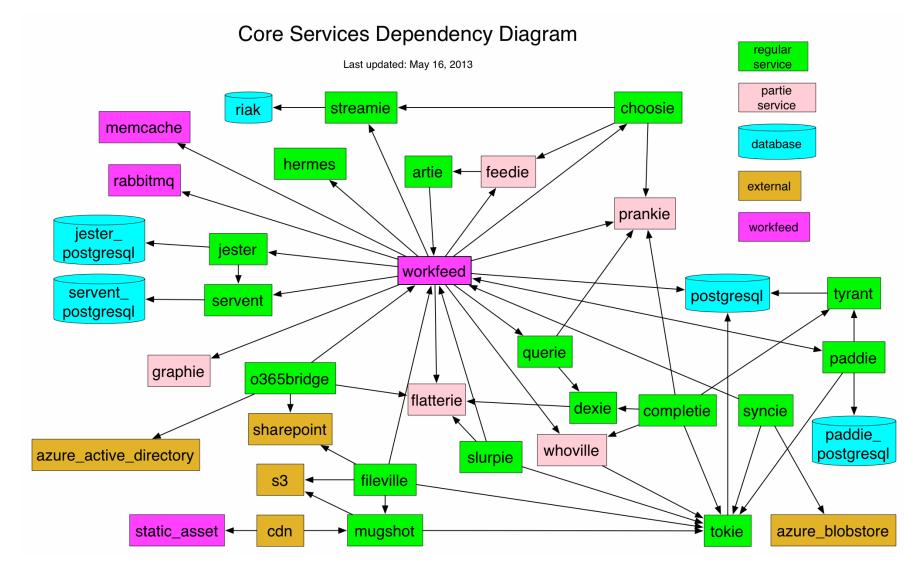












That was mid-2013

No one even tries to draw this diagram **now!** 

Feature development become too slow:

- Too many inter-service dependencies
- Overly chatty services
- Too many inter-team, cross-time-zone dependencies

Feature development become too slow:

- Too many inter-service dependencies
- Overly chatty services
- Too many inter-team, cross-time-zone dependencies

Feature development become too slow:

- Too many inter-service dependencies
- Overly chatty services
- Too many inter-team, cross-time-zone dependencies

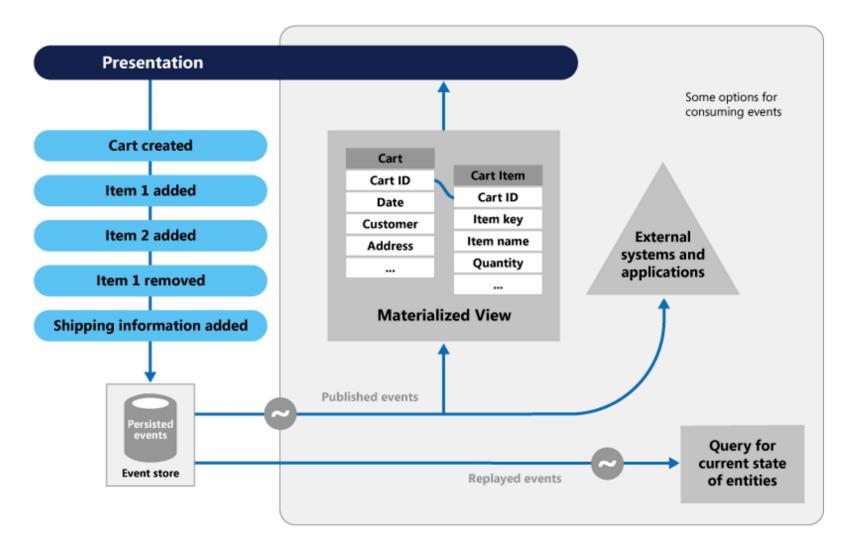
- Too many external dependencies on the read/write path
- Shared DB
- Un-expected, transitive dependencies
- Cascading failures (despite circuit breaking)
- Very easy to make a breaking code change

- Too many external dependencies on the read/write path
- Shared DB
- Un-expected, transitive dependencies
- Cascading failures (despite circuit breaking)
- Very easy to make a breaking code change

- Too many external dependencies on the read/write path
- Shared DB
- Un-expected, transitive dependencies
- Cascading failures (despite circuit breaking)
- Very easy to make a breaking code change

- Too many external dependencies on the read/write path
- Shared DB
- Un-expected, transitive dependencies
- Cascading failures (despite circuit breaking)
- Very easy to make a breaking code change

- Too many external dependencies on the read/write path
- Shared DB
- Un-expected, transitive dependencies
- Cascading failures (despite circuit breaking)
- Very easy to make a breaking code change

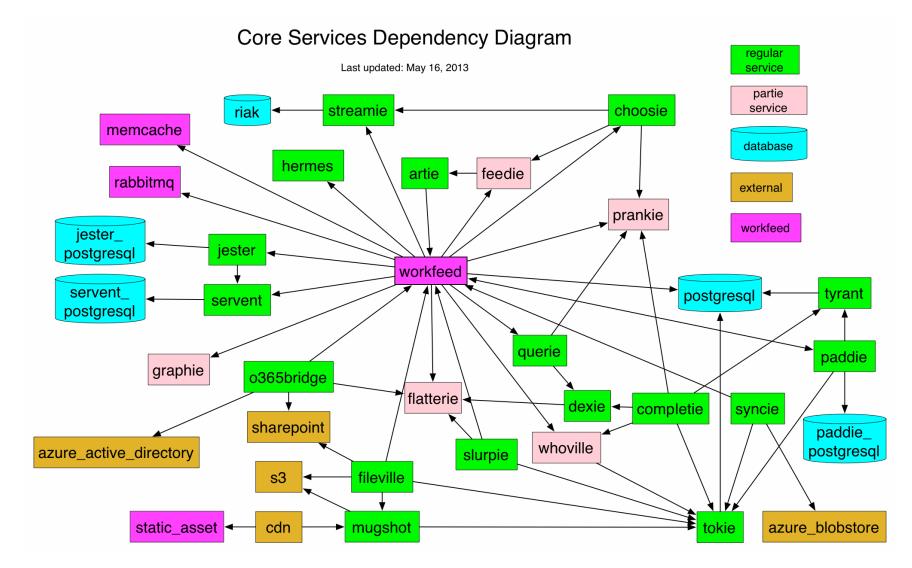


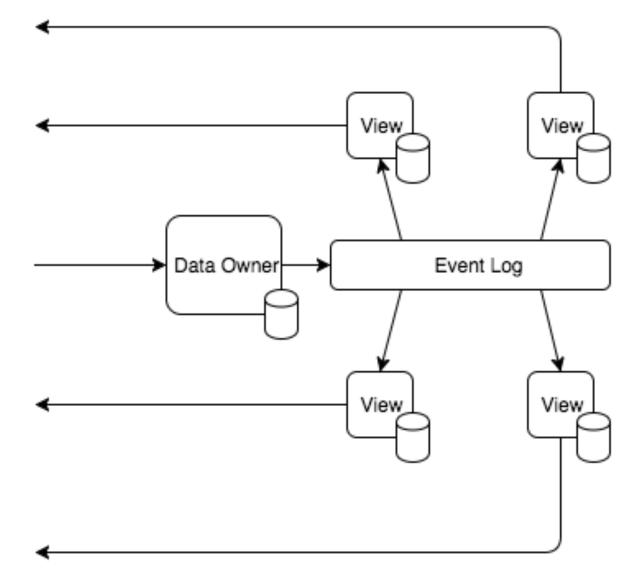
- One Data Owner Service

   Publishes Events
   Persists in DB
- View Services
  - Consume Events
  - Materialized Views (local DB)

- One Data Owner Service
   Publishes Events
   Persists in DB
- View Services
  - Consume Events
  - Materialized Views (local DB)

# **Distributed Monolith**





- Less runtime dependencies (SLA, performance)
- Less chattiness (performance, velocity)
- Loose Coupling (velocity, SLA)
- Events / Not Commands (velocity vs. 1-1 coupling)
- Cheap to setup/backfill new service (velocity)

- Less runtime dependencies (SLA, performance)
- Less chattiness (performance, velocity)
- Loose Coupling (velocity, SLA)
- Events / Not Commands (velocity vs. 1-1 coupling)
- Cheap to setup/backfill new service (velocity)

- Less runtime dependencies (SLA, performance)
- Less chattiness (performance, velocity)
- Loose Coupling (velocity, SLA)
- Events / Not Commands (velocity vs. 1-1 coupling)
- Cheap to setup/backfill new service (velocity)

- Less runtime dependencies (SLA, performance)
- Less chattiness (performance, velocity)
- Loose Coupling (velocity, SLA)
- Events / Not Commands (velocity vs. 1-1 coupling)
- Cheap to setup/backfill new service (velocity)

- Less runtime dependencies (SLA, performance)
- Less chattiness (performance, velocity)
- Loose Coupling (velocity, SLA)
- Events / Not Commands (velocity vs. 1-1 coupling)
- Cheap to setup/backfill new service (velocity)

# Challenges

- We can't make it happen overnight
- There are a lot of risks:
  - Can this pattern deliver?
  - How long will it take to learn?
  - What stack to use?
  - Cost of tech onboarding?

# Challenges

- We can't make it happen overnight
- There are a lot of risks:
  - Can this pattern deliver?
  - How long will it take to learn?
  - What stack to use?
  - Cost of tech onboarding?

# Two challenges

- Validate Event Sourcing
- Choose and on-board appropriate tech stack

Ideally we can decouple the two, to:

- Validate early
- Deliver value early
- Invest in tech once idea validated

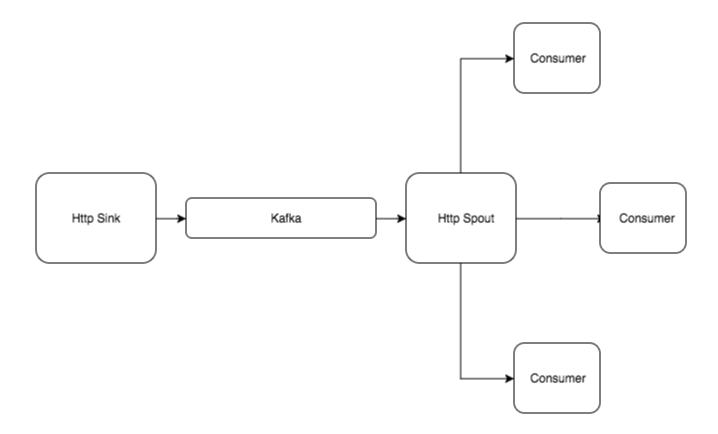
# Two challenges

- Validate Event Sourcing
- Choose and on-board appropriate tech stack

Ideally we can decouple the two, to:

- Validate early
- Deliver value early
- Invest in tech once idea validated

#### Leverage familiar legacy to minimize tech risk.



- Not the greatest design:
  - centralization
  - http proxies, why?
- Built of familiar components we are already operating
- Minimizes tech risk, letting us focus on validation
- Short term, centralization allows for faster iteration

- Not the greatest design:
  - centralization
  - http proxies, why?
- Built of familiar components we are already operating
- Minimizes tech risk, letting us focus on validation
- Short term, centralization allows for faster iteration

- Not the greatest design:
  - centralization
  - http proxies, why?
- Built of familiar components we are already operating
- Minimizes tech risk, letting us focus on validation
- Short term, centralization allows for faster iteration

- Not the greatest design:
  - centralization
  - http proxies, why?
- Built of familiar components we are already operating
- Minimizes tech risk, letting us focus on validation
- Short term, centralization allows for faster iteration

- We've established an API and semantics
- Tooling for consumer/event monitoring/management
- End-to-end test suite (focus on failure handling)
- Automated load and throughput tests

- We've established an API and semantics
- Tooling for consumer/event monitoring/management
- End-to-end test suite (focus on failure handling)
- Automated load and throughput tests

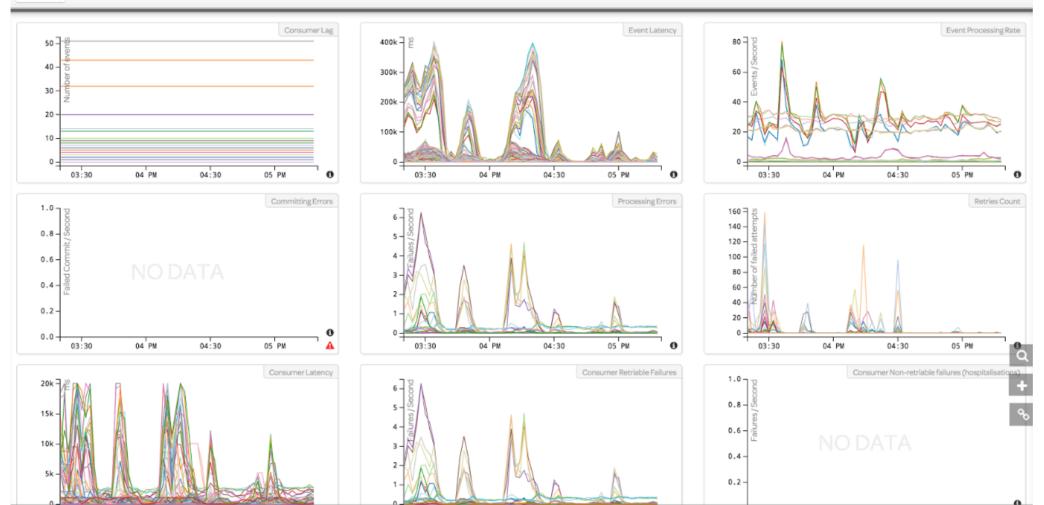
- We've established an API and semantics
- Tooling for consumer/event monitoring/management
- End-to-end test suite (focus on failure handling)
- Automated load and throughput tests

- We've established an API and semantics
- Tooling for consumer/event monitoring/management
- End-to-end test suite (focus on failure handling)
- Automated load and throughput tests

- We've established an API and semantics
- Tooling for consumer/event monitoring/management
- End-to-end test suite (focus on failure handling)
- Automated load and throughput test

### Easy to generate dashboards

Consuming



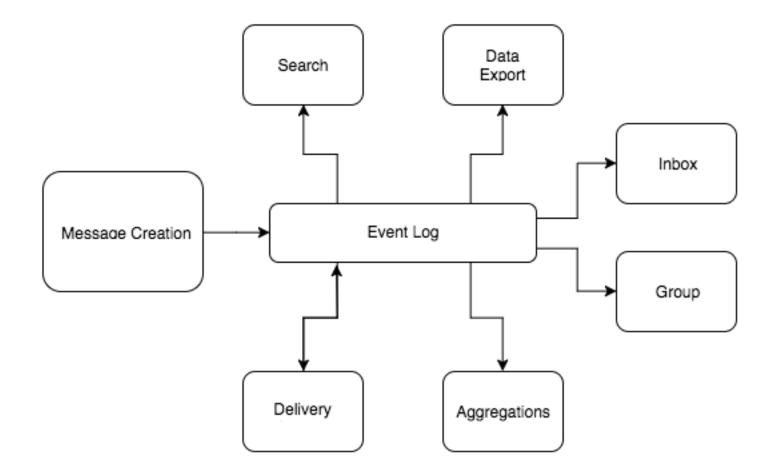
### **Troubleshooting bad events**

All Failing S	Stuck
---------------	-------

Consumer	Торіс	Partition	Host	Read	Lag	Latency	Uncommitted	Retries	Actions
pulse_ogo_create	message	174	ventspout- 1.az2.dm2	43107	0	6 hours	11	21741	0 🗢 🕂 N
stored_feeds_delivery_thread_and_group_delivered	message	60	ventspout- 1.az1.dm2	53668	0	a few seconds	1	11	0 • <b>+</b> X

from the SOA team

# Projects that benefited immediately



### Problems we've faced

- What to publish?
  - Whole pieces of data (potentially unbounded size)
  - IDs, but requires:
    - Immutable versioned data
    - Uniform Resource Identifiers (REST done well)
- There will be multiple publishers
  - Consumers need to deal with it gracefully

### Problems we've faced

- What to publish?
  - Whole pieces of data (potentially unbounded size)
  - IDs, but requires:
    - Immutable versioned data
    - Uniform Resource Identifiers (REST done well)
- There will be multiple publishers
  - Consumers need to deal with it gracefully

# **Adoption Challenges**

- It is a big paradigm shift, it takes time for knowledge to propagate through an organization
- We are not experts either, we are still learning
- But good news, even if imperfect, it already had big impact on how we work

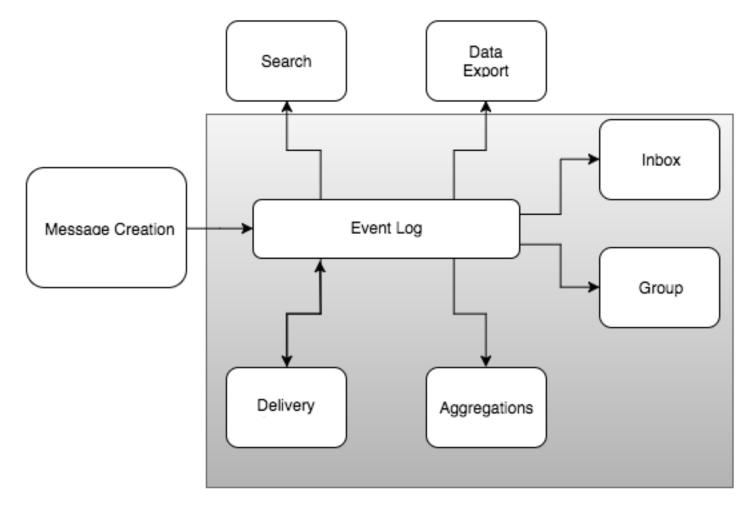
# **Adoption Challenges**

- It is a big paradigm shift, it take time for knowledge to propagate through an organization
- We are not experts either, we are still learning
- But good news, even if imperfect, it already had big impact on how we work

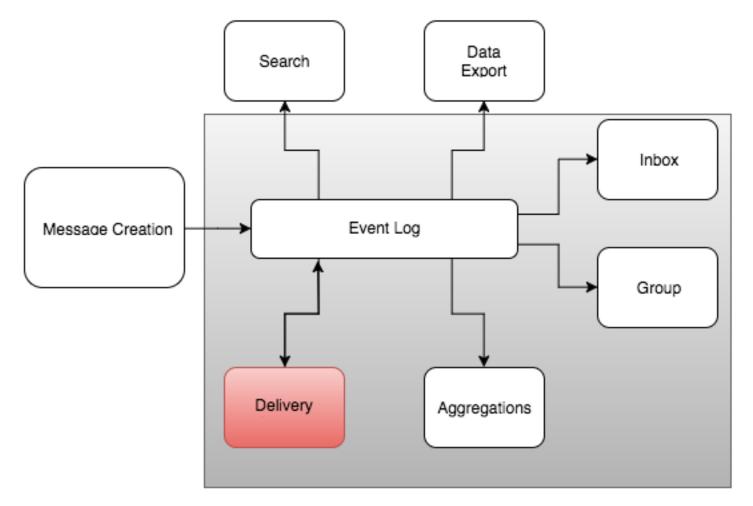
# **Adoption Challenges**

- It is a big paradigm shift, it take time for knowledge to propagate through an organization
- We are not experts either, we are still learning
- But good news, even if imperfect, it already had big impact on how we work

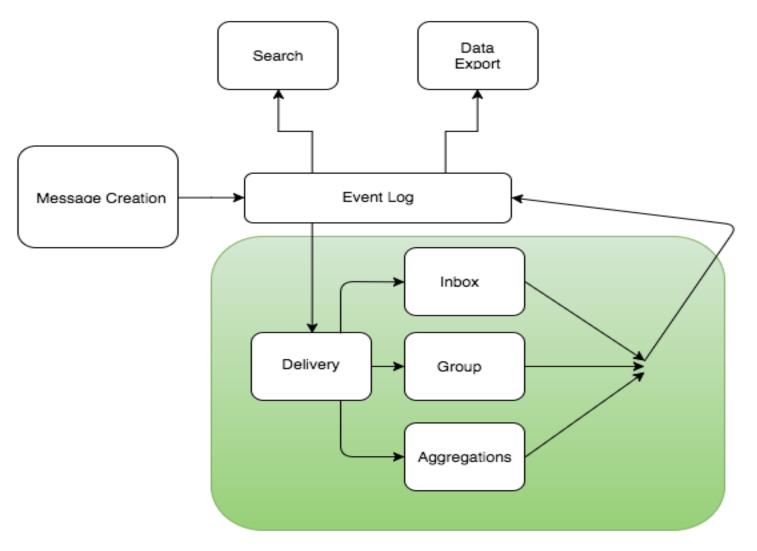
### What we've discovered - Workflows



#### What we've discovered - Workflows



### What we've discovered - Workflows



- Transformation logic outside of view service boundaries
- Stream Processing / CQRS
- We can express them, but too much plumbing
- We need a higher level of abstraction

- Transformation logic outside of view service boundaries
- Stream Processing / CQRS
- We can express them, but too much plumbing
- We need a higher level of abstraction

- Transformation logic outside of view service boundaries
- Stream Processing / CQRS
- We can express them, but too much plumbing
- We need a higher level of abstraction

- Transformation logic outside of view service boundaries
- Stream Processing / CQRS
- We can express them, but too much plumbing
- We need a higher level of abstraction

### What is the Future?

- Move to a fully managed solution
- Provide Rx bindings
- Remove centralization/http proxy components
- Find a solution for Workflows

# **Migration To Event Hubs**

Azure Event Hubs

- Event Log offering from Azure (Kafka, Kinesis)
  - Automated failover within a region
  - No provisioning concerns, simply purchase TUs
- AMQP 1.0 protocol
- Successfully used internally and externally
- Our Metrics pipeline already uses it

## **Migration To Event Hubs**

#### Azure Event Hubs

- Event Log offering from Azure (Kafka, Kinesis)

   Automated failover within a region
   No provisioning concerns, simply purchase TUs
- AMQP 1.0 protocol
- Successfully used internally and externally
- Our Metrics pipeline already uses it

## **Migration To Event Hubs**

**Azure Event Hubs** 

- Event Log offering from Azure (Kafka, Kinesis)
  - Automated failover within a region
  - No provisioning concerns, simply purchase TUs
- AMQP 1.0 protocol
- Successfully used internally and externally
- Our Metrics pipeline already uses it

## **Migration To Event Hubs**

**Azure Event Hubs** 

- Event Log offering from Azure (Kafka, Kinesis)
  - Automated failover within a region
  - No provisioning concerns, simply purchase TUs
- AMQP 1.0 protocol
- Successfully used internally and externally
- Our Metrics pipeline already uses it

#### **RxJava SDK**

- Using Azure SDK
  - backed by ProtonJ
  - Offset tracking
  - Multi-host Consumer with Failover
- Raising the level of abstraction:
  - Data Stream in Event Hubs available as an Observable

### **RxJava SDK**

- Using Azure SDK
  - backed by ProtonJ
  - Offset tracking
  - Multi-host Consumer with Failover
- Raising the level of abstraction:

– Data Stream in Event Hubs available as an Observable

- Azure Service Fabric Reliable Actors
- High Level PAAS offering from Azure
- Based on Project Orleans
- Successfully used by HALO

- Azure Service Fabric Reliable Actors
- High Level PAAS offering from Azure
- Based on Project Orleans
- Successfully used by HALO

- Azure Service Fabric Reliable Actors
- High Level PAAS offering from Azure
- Based on Project Orleans
- Successfully used by HALO

- Azure Service Fabric Reliable Actors
- High Level PAAS offering from Azure
- Based on Project Orleans
- Successfully used by HALO

- Successfully used Event Sourcing to solve our SLA/ Velocity problems caused by a (distributed) monolith
- It addressed both architectural and org aspects
- We did so in an iterative fashion focusing
  - Reducing risk
  - Delivering early
- This is ongoing work

- Successfully used Event Sourcing to solve our SLA/ Velocity problems caused by a (distributed) monolith
- It addressed both architectural and org aspects
- We did so in an iterative fashion focusing
  - Reducing risk
  - Delivering early
- This is ongoing work

- Successfully used Event Sourcing to solve our SLA/ Velocity problems caused by a (distributed) monolith
- It addressed both architectural and org aspects
- We did so in an iterative fashion focusing
  - Reducing risk
  - Delivering early
- This is ongoing work

- Successfully used Event Sourcing to solve our SLA/ Velocity problems caused by a (distributed) monolith
- It addressed both architectural and org aspects
- We did so in an iterative fashion focusing
  - Reducing risk
  - Delivering early
- This is ongoing work

## Thank you!

#### Any Questions?