



Building a financial data pipeline with Kafka

Opinions expressed are those of the author and may not be shared by all personnel of Man Group plc ('Man'). These opinions are subject to change without notice, and are for information purposes only and do not constitute an offer or invitation to make an investment in any financial instrument or in any product to which any member of Man's group of companies provides investment advisory or any other services. Any forward-looking statements speak only as of the date on which they are made and are subject to risks and uncertainties that may cause actual results to differ materially from those contained in the statements. Unless stated otherwise the source of all information is Man Group plc and its affiliates as of the date on the first page of this material. Unless stated otherwise this information is communicated by AHL Partners LLP which is registered in England and Wales at Riverbank House, 2 Swan Lane, London, EC4R 3AD and is authorised and regulated in the UK by the Financial Conduct Authority.

100% systematic quant

\$19bn+ AUM

Everyone writes code

65+ Tech

30 years experience

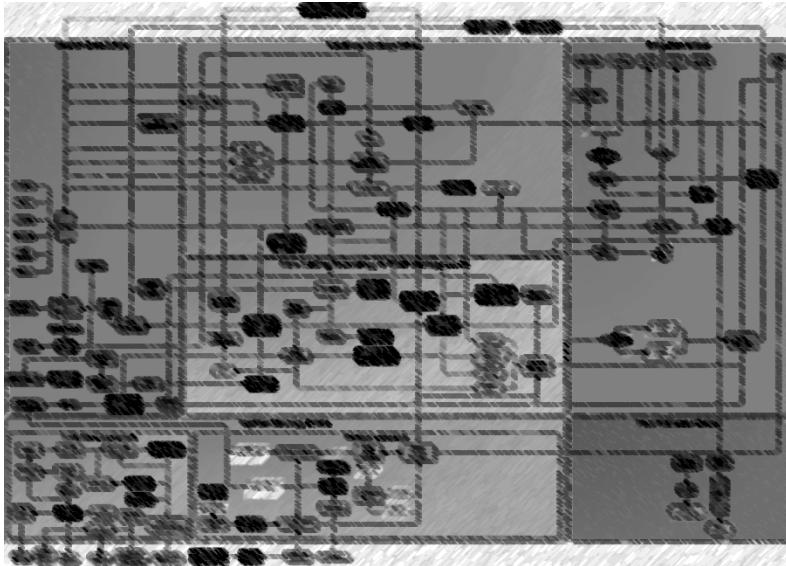
Open culture

Open source software



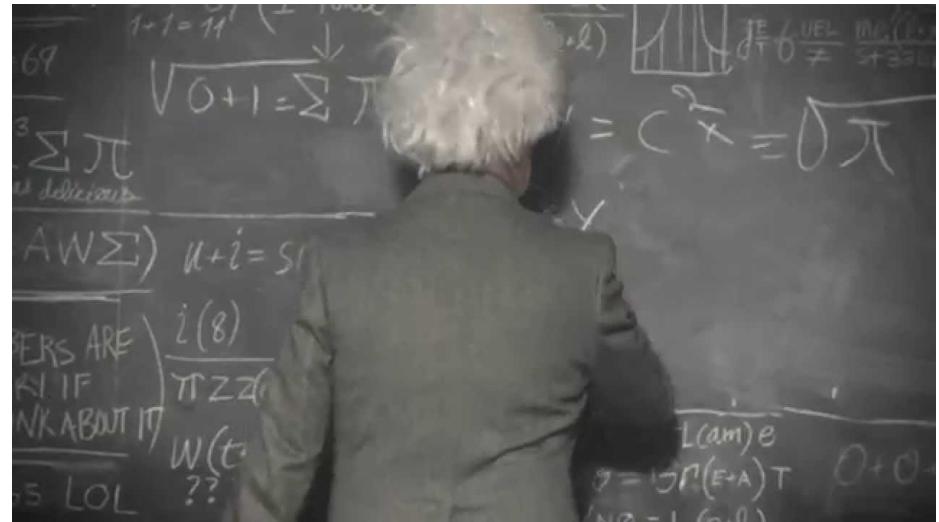
theano





25 Core technologists

- Linux infrastructure
- Data engineering
- Core platform
- Core trading
- Electronic execution
- Risk systems



35 Research technologists

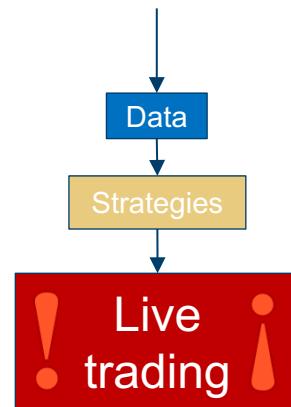
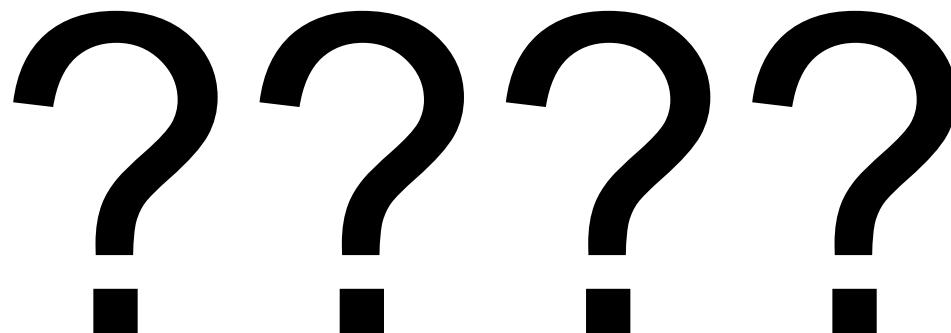
- Across 10 research teams
- Support research
- Implement strategies
- Maintain strategies

The problem

Something happens somewhere on



Something happens somewhere on 



The AHL streaming data pipeline

From the sky



Something happens somewhere on

Tick feeds

Thompson Reuters Enterprise Platform

Tick data

Tick
collectors

Tick
collectors

Tick
collectors

Tick
processor

AHL market
data ticks

! Live
trading !

Kafka

Zookeeper

AHL market
data ticks

AHL market
data ticks

Tick
processor

Processed
data

Processed
ticks

Tick
processor

MongoDB
Arctic tickstore

MongoDB
Arctic

! Live
trading !

Something happens somewhere on 

↓
Tick feeds

Thompson Reuters Enterprise Platform

↓
Tick data

```
2018-02-07 15:59:18,712 INFO [sub_worker_6] LoggingSubscriber - Type: UPDATE Subject: BPOD.TRI.US
    PROD_PERM:      Long:          14003
    DelayedEID:     Long:          0
    UpdType:        String:       ASK
    ASK_MMID1:      String:       UN
    ASKXID:         String:       NYS
    ASK_TIME:        LocalTime:    15:59
    ASK_TIME1:       LocalTime:    15:59:18
    ASK_TIM_MS:      Long:        57558635
    AskPrice:        Double:      40.87
    AskLSc:          String:       UN
    LstAIsInd:       Long:        0
    AskTime:         LocalTime:    15:59:18
    QUOTIM:          LocalTime:    15:59:18
    QUOTIM_MS:       Long:        57558635
    ASK:             Double:      40.87
    ASKSIZE:         Double:      1.0
```

Something happens somewhere on 

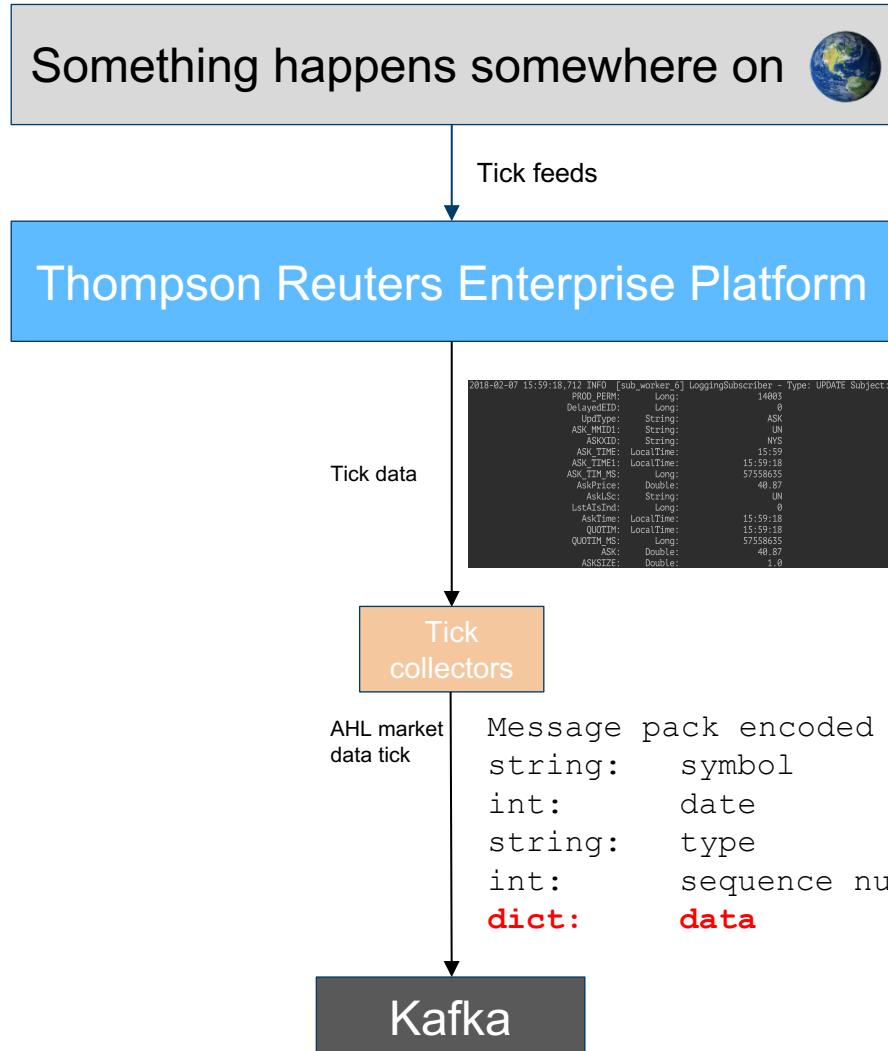
↓
Tick feeds

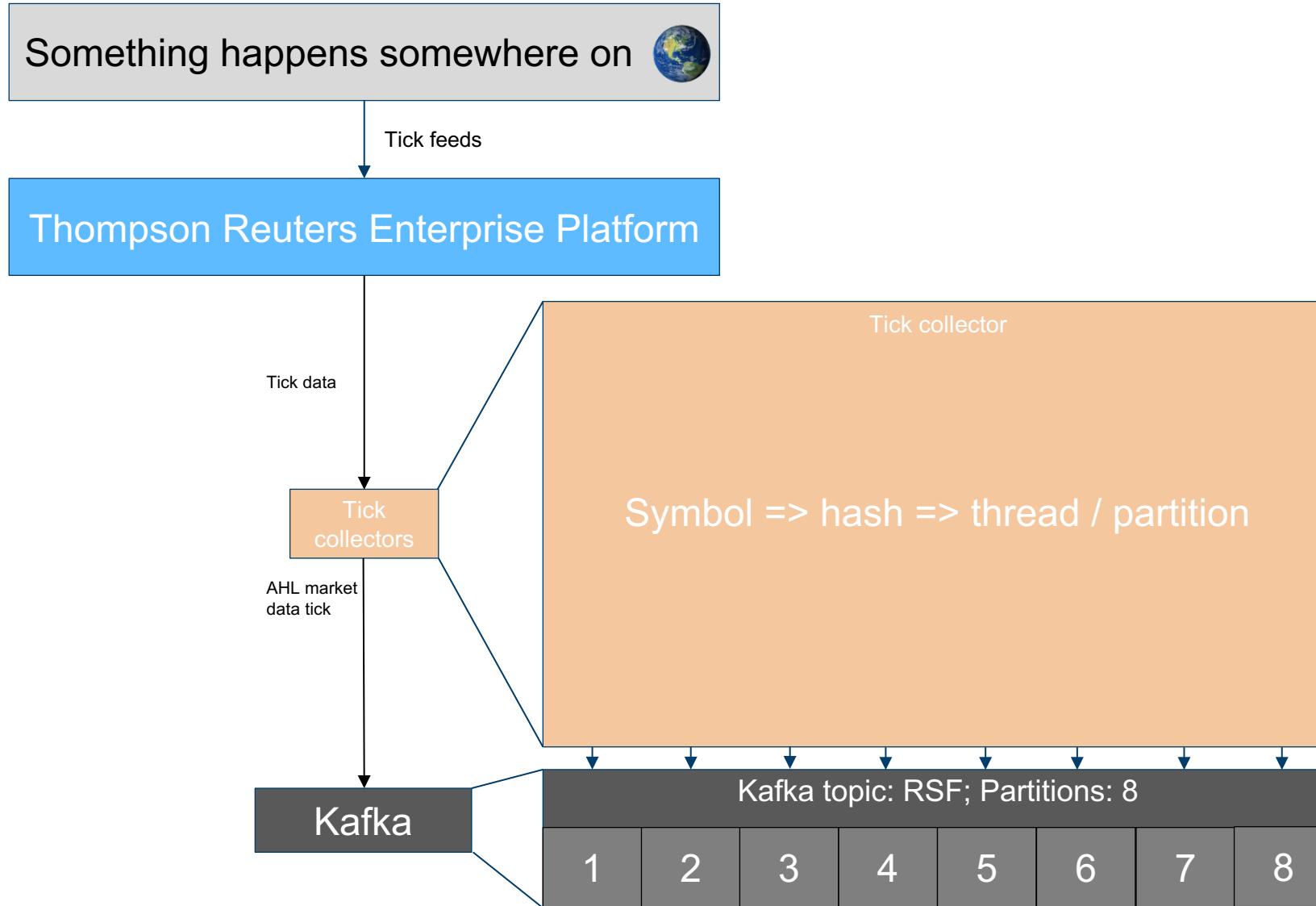
Thompson Reuters Enterprise Platform

↓
Tick data

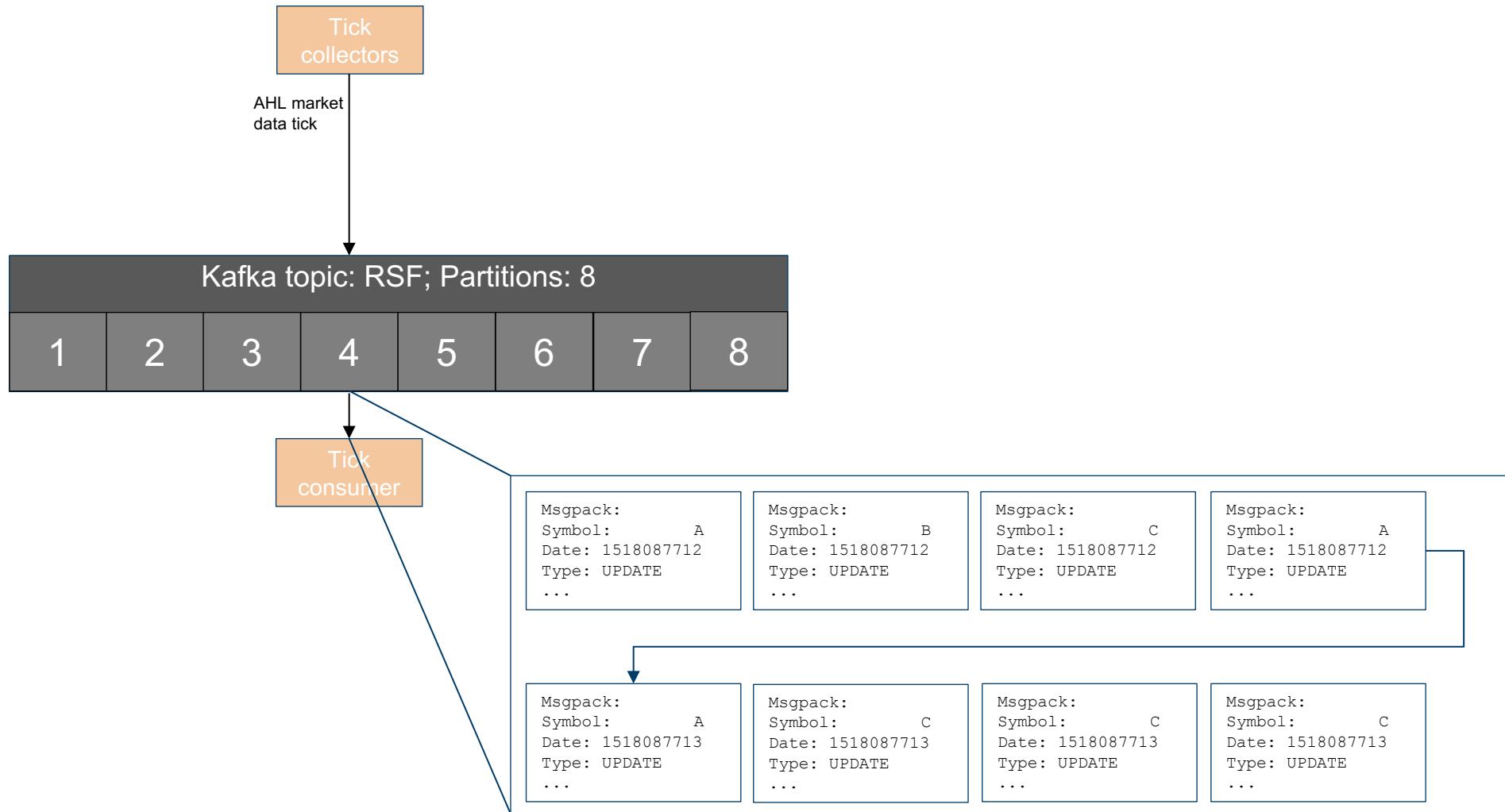
```
2018-02-07 15:59:18,712 INFO [sub_worker_6] LoggingSubscriber - Type: UPDATE Subject: BPOD.TRI.US
    PROD_PERM:      Long:          14003
    DelayedEID:     Long:          0
    UpdType:        String:       ASK
    ASK_MMID1:      String:       UN
    ASKXID:         String:       NYS
    ASK_TIME:        LocalTime:   15:59
    ASK_TIME1:       LocalTime:   15:59:18
    ASK_TIM_MS:     Long:        57558635
    AskPrice:       Double:      40.87
    AskLSc:         String:       UN
    LstAIsInd:      Long:          0
    AskTime:        LocalTime:   15:59:18
    QUOTIM:         LocalTime:   15:59:18
    QUOTIM_MS:      Long:        57558635
    ASK:            Double:      40.87
    ASKSIZE:        Double:      1.0
```

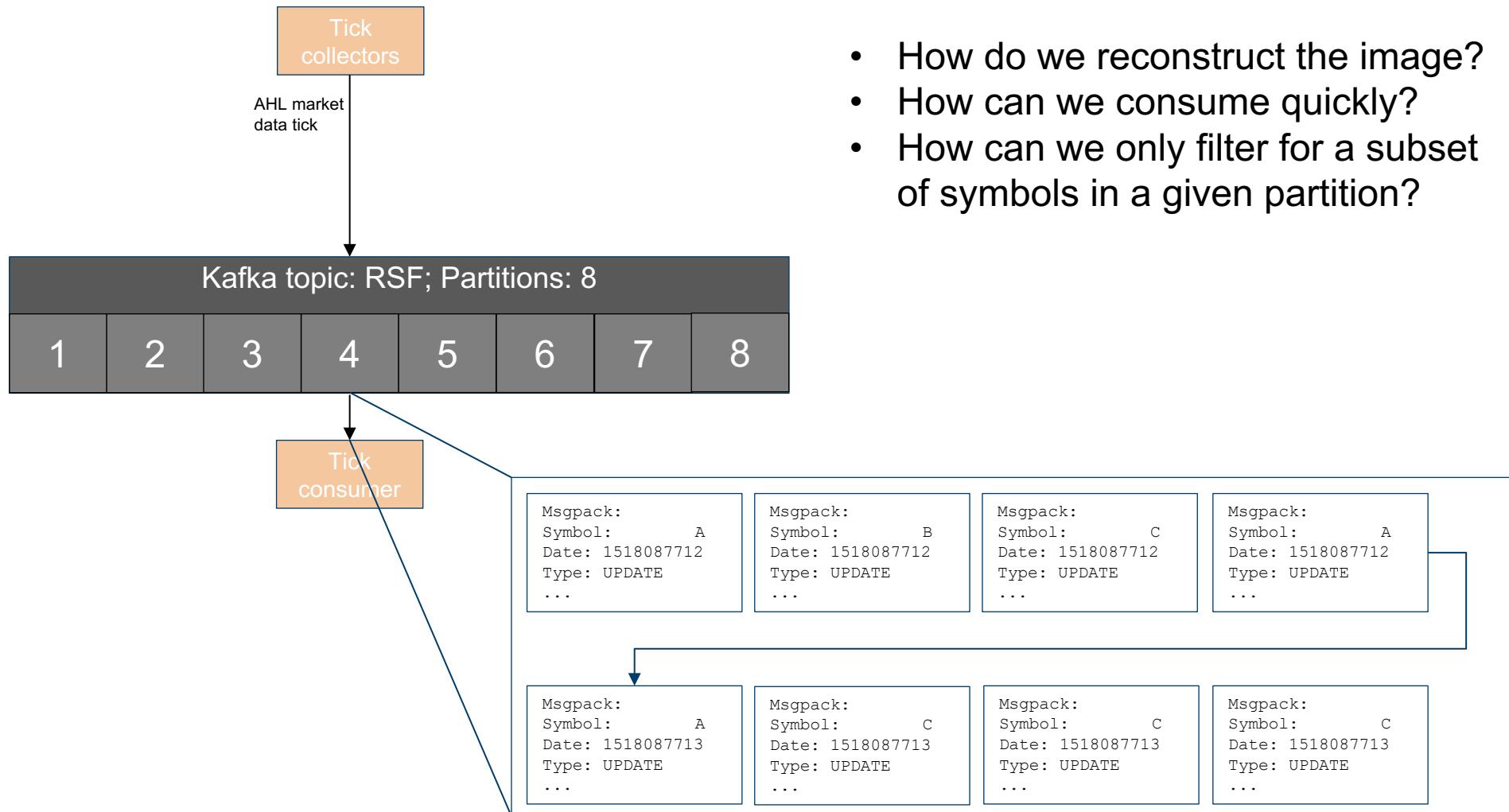
x2 billion a day



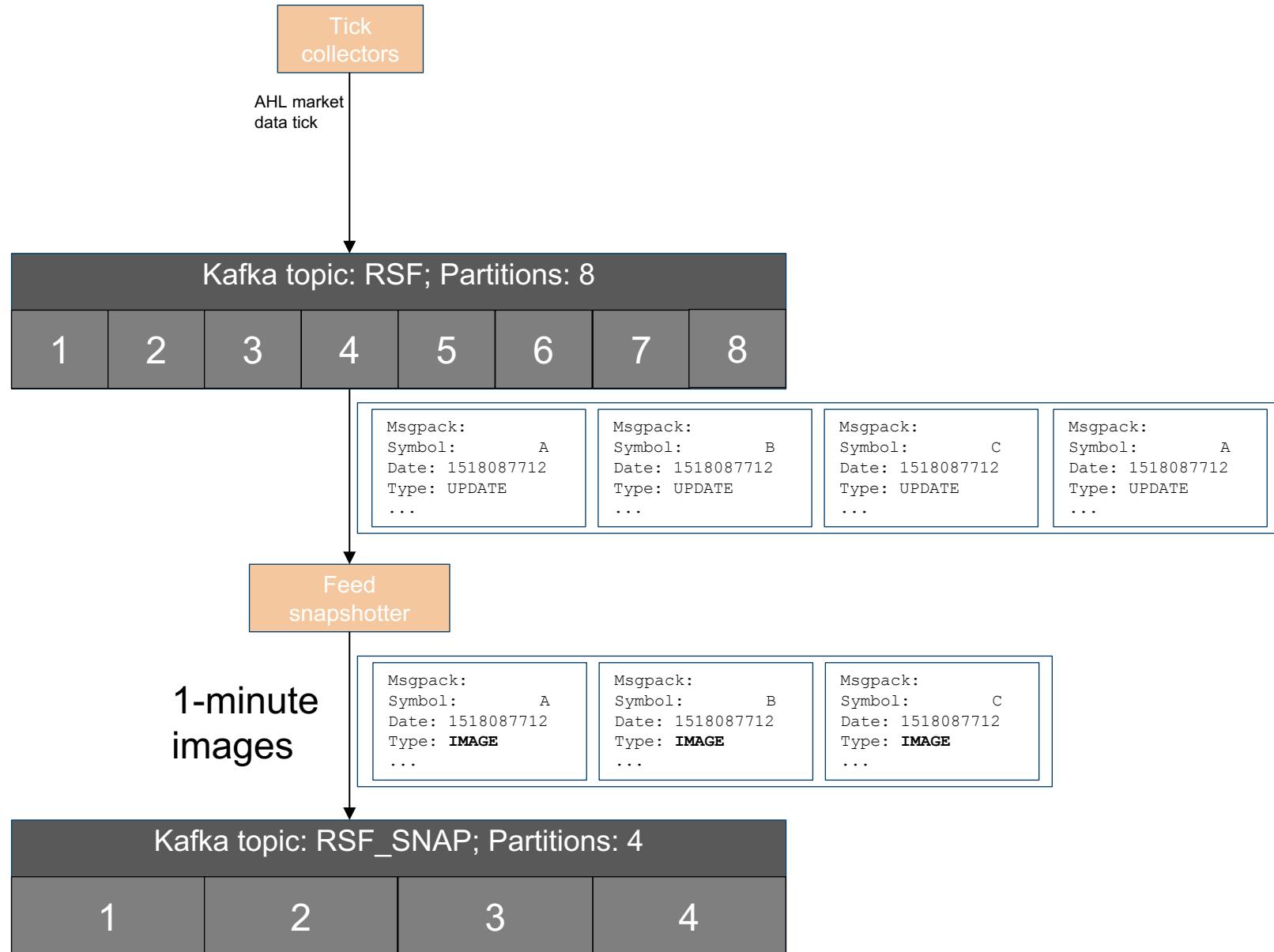


The AHL streaming data pipeline





The AHL streaming data pipeline



The AHL streaming data pipeline

AHL Kafka client



- AHL is a Python shop.
- Only one mature Python Kafka client in 2013 – kafka-python. Kafka-python is **pure Python** implementation. It's also (relatively) slow to consume ☹
- We use a custom Kafka message set decoder written in Cython. ~10x speed up. Filtering based on Symbol done in Cython.
- Our Kafka clients (Java and Python) talk in terms of timestamps rather than Kafka offsets – does binary search into the partition behind the scenes.
- Time-based search now implemented natively in Kafka (since 0.10.1.0) on the broker side.
- Kafka-python is also slow to produce ☹ Python client creates a pool of producers. Maps from symbol -> producer -> partition.

The AHL streaming data pipeline

How is this deployed? How do we know it works?

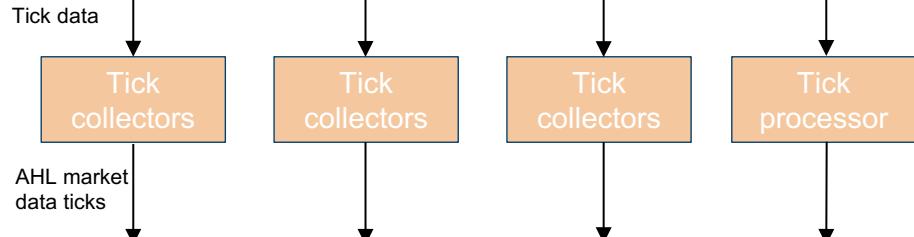


Something happens somewhere on

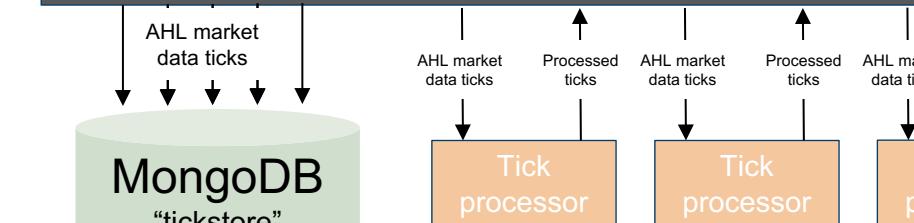
Tick feeds

- **Lots** of moving parts
- How is each component deployed?
- How is each component monitored?

Thompson Reuters Enterprise Platform



Kafka

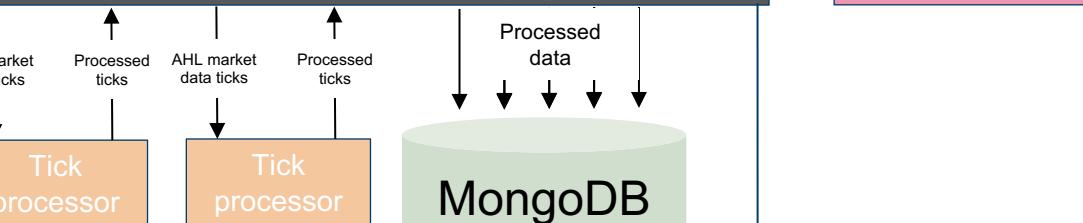


MongoDB "tickstore"



Live trading

Zookeeper



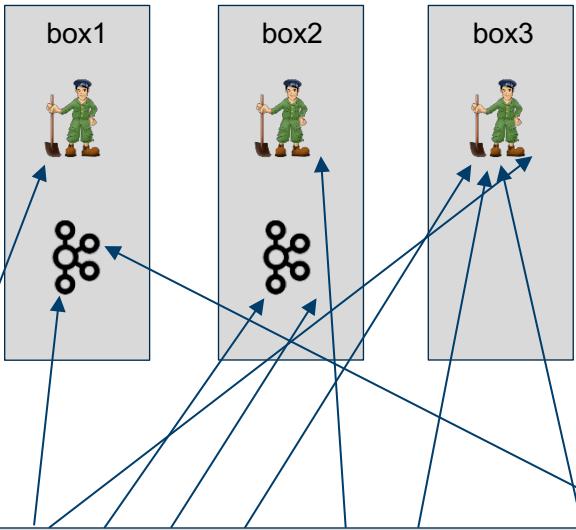
Live trading

The AHL streaming data pipeline

How is each component deployed?



Old



Lots and lots of AHL services

Key

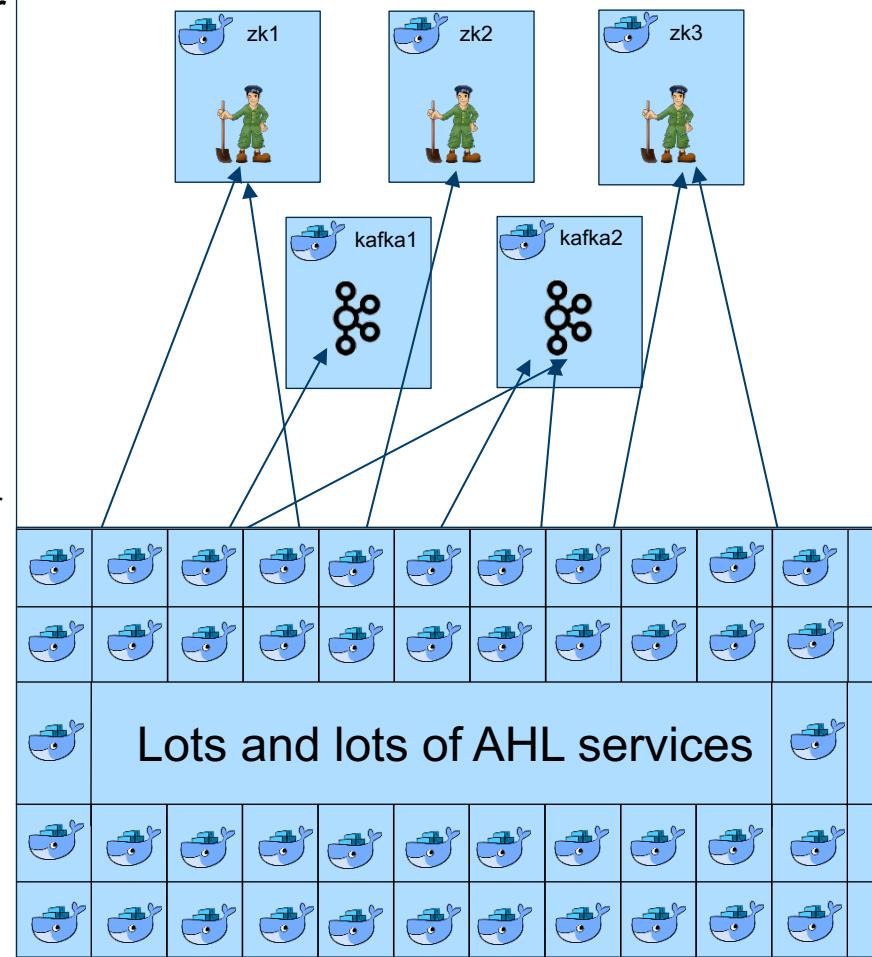
= Zookeeper

= Kafka

= Bare metal

= Docker container

New



Lots and lots of AHL services

The AHL streaming data pipeline

How is each component deployed?



- At AHL we've drunk *some* of the Docker kool-aid.
- Dockerize everything!
- Docker-compose deploy everything!
- **No** Kubernetes, docker swarm, linkerd, etcd, etc.
 - We use manually managed CNAMEs heavily for aliasing/indirection
- We use our own internal solution for deploying docker-compose services
 - Heavily integrated with Bitbucket. Engineers and Researchers are **not allowed to touch the live trading system.**
 - Services are created and destroyed on merge. Merges are only permitted by support staff.
- For basic container administration, every box runs an instance of an agent called **Dockit**.



A screenshot of a Bitbucket merge request interface. At the top, there are status indicators for 6 reviewers, with 5 marked as 'Approved' (green checkmark) and 1 marked as 'Pending' (yellow circle). To the right are buttons for 'Merge' and 'Revert'. Below this is a section titled 'MERGE ISSUES' with the subtext 'There is 1 issue preventing you from merging this pull request.' A detailed error message follows: '⚠ You have insufficient permissions to update 'live'. Check your branch permissions with the project administrator.'.

The AHL streaming data pipeline

How is each component deployed?



■ Dockit

Dockit 1.26.0

No Dockfile loaded, showing all containers

Config

Name	Loaded
.dockit.auth	2018-01-20 17:08

Containers

Name	Uptime	Actions
dockit mongo_inst	19 days	 
ALL		
dockit_dockit_1	19 days	 
mongo_instances_config	Exited (0) 2 weeks ago	 

Log: 

```
{"time": "2018-02-08T17:13:32Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/containers", "status": 200, "latency": 119864, "latency_human": "119.864ms"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/configs", "status": 200, "latency": 63, "latency_human": "63µs"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/nameofprefixes", "status": 200, "latency": 10, "latency_human": "10µs"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/configstatus", "status": 200, "latency": 41, "latency_human": "41µs"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/images", "status": 200, "latency": 27230, "latency_human": "27.230ms"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/containers", "status": 200, "latency": 137986, "latency_human": "137.986ms"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/configstatus", "status": 200, "latency": 26, "latency_human": "26µs"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/nameofprefixes", "status": 200, "latency": 10, "latency_human": "10µs"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/configs", "status": 200, "latency": 304, "latency_human": "304µs"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/log/dockit_dockit_1?tail=20", "status": 200, "latency": 32798, "latency_human": "32.798ms"}, {"time": "2018-02-08T17:13:35Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/images", "status": 200, "latency": 137458, "latency_human": "137.458ms"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/containers", "status": 200, "latency": 74, "latency_human": "74µs"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/configs", "status": 200, "latency": 137458, "latency_human": "137.458ms"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/nameofprefixes", "status": 200, "latency": 23, "latency_human": "23µs"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/configstatus", "status": 200, "latency": 48, "latency_human": "48µs"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/images", "status": 200, "latency": 28593, "latency_human": "28.593ms"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.145", "method": "GET", "uri": "/v1/containers", "status": 200, "latency": 130594, "latency_human": "130.594ms"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/configstatus", "status": 200, "latency": 41, "latency_human": "41µs"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/configs", "status": 200, "latency": 78, "latency_human": "78µs"}, {"time": "2018-02-08T17:13:38Z", "remote_ip": "10.200.31.70", "method": "GET", "uri": "/v1/nameofprefixes", "status": 200, "latency": 13, "latency_human": "13µs"}]
```

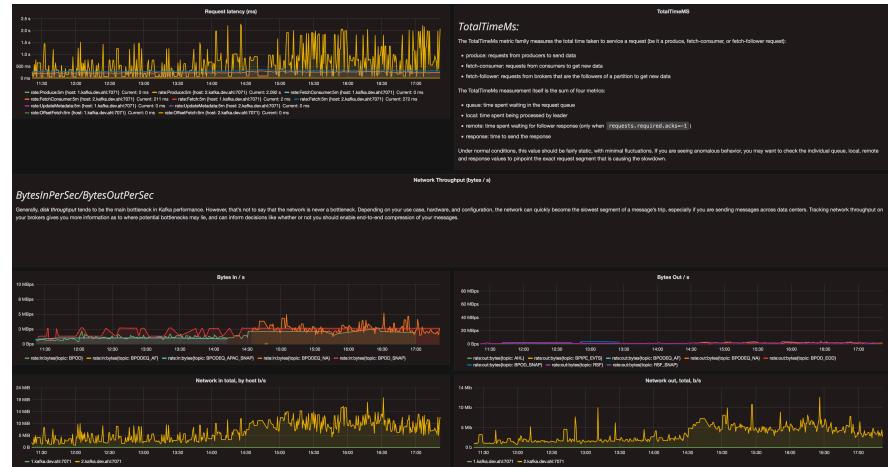
The AHL streaming data pipeline

How is each component monitored?



- We **need** to know when something goes wrong
- Prometheus + nagios + grafana + alerta

Kafka



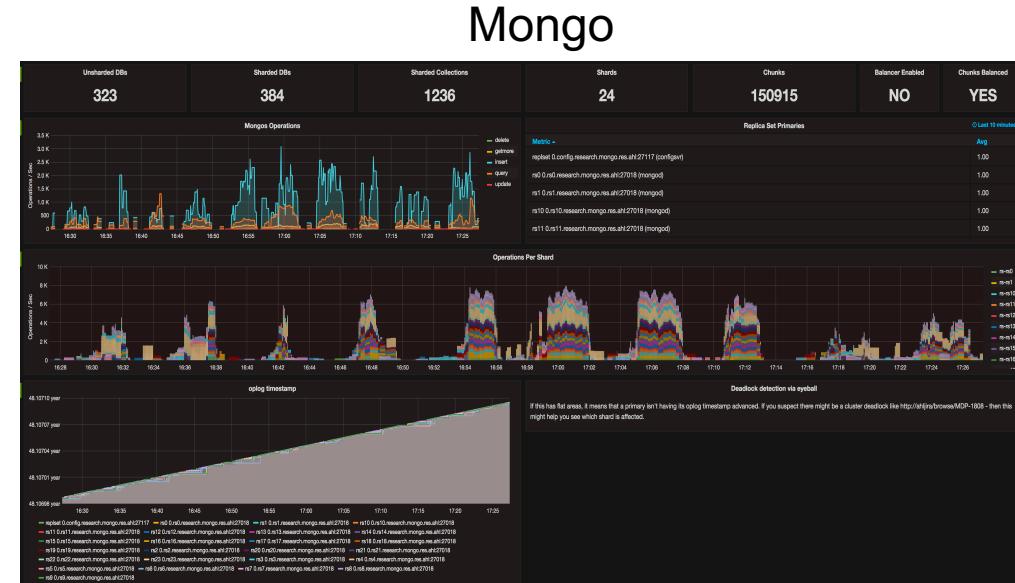
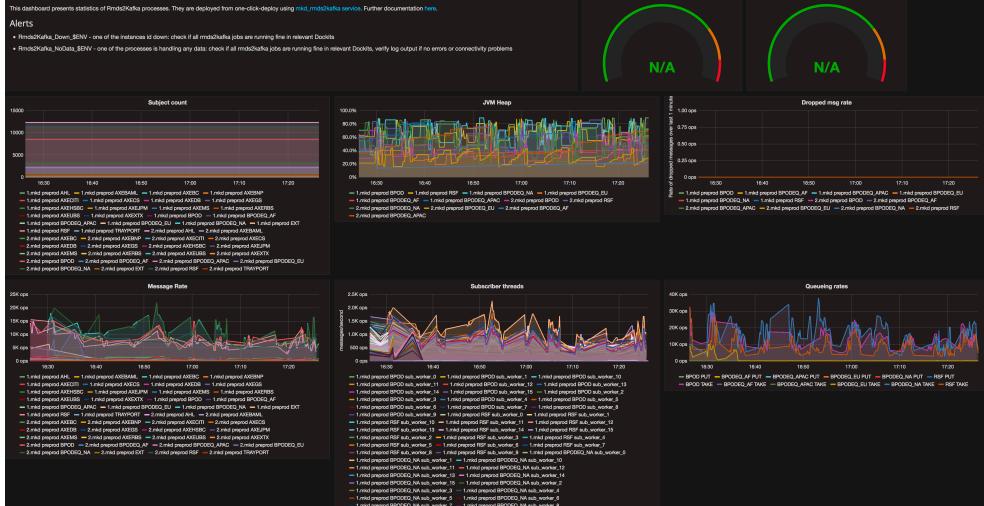
Tickprocessor

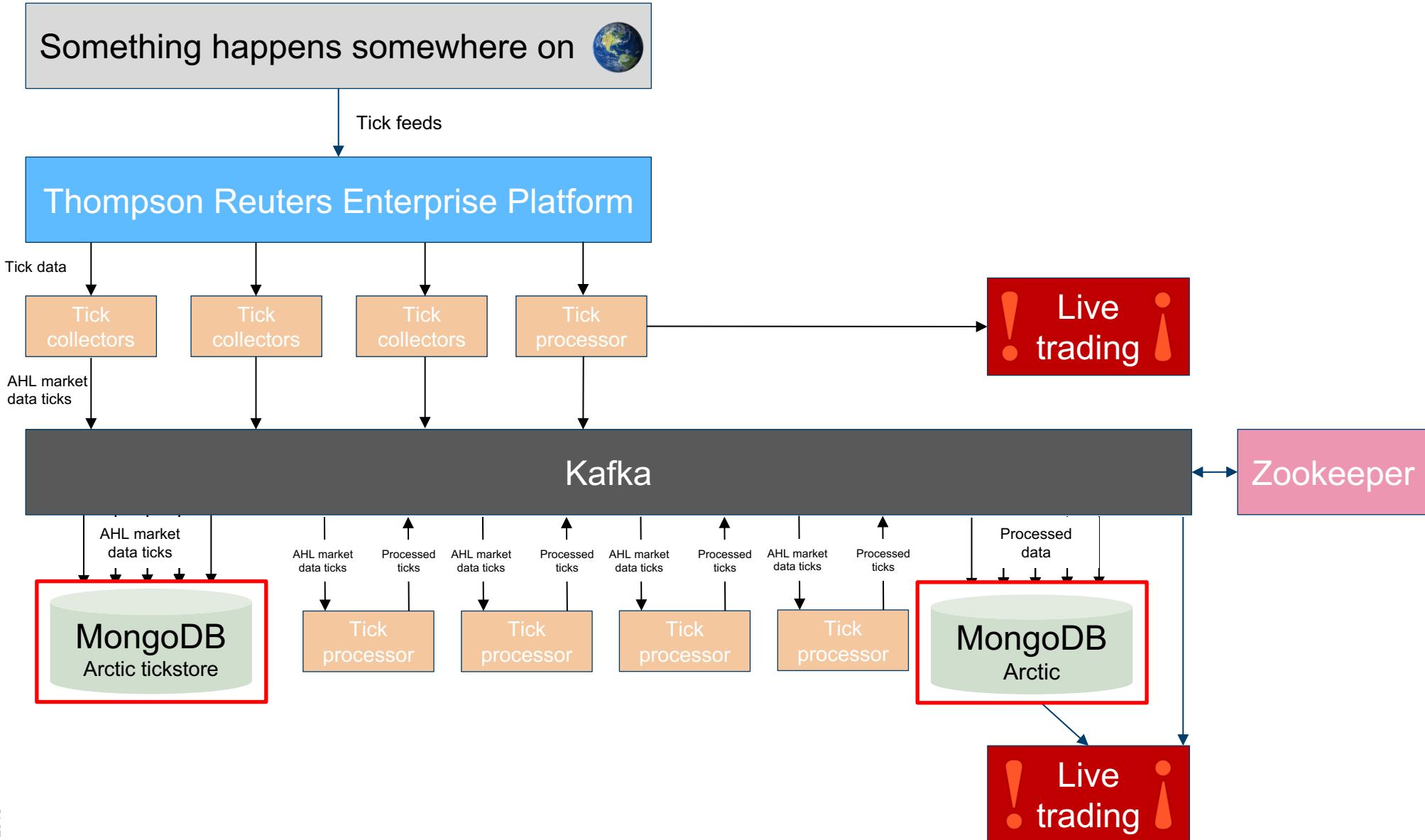


The AHL streaming data pipeline

How is each component monitored?

- We **need** to know when something goes wrong
 - Prometheus + nagios + grafana + alerta





- Data sizes we deal with...

- | | | |
|------------------|-------------------------|-----------------------------|
| • ~1MB x 1000s | 1x a day price data | 10k rows (30 years) |
| • ~0.5GB x 1000s | 1-minute data | 4M rows (20 years) |
| • ~1GB x 1000s | 10k x 10k data matrices | 100M cells (30 years) |
| • ~30TB | Tick data | 200k msgs/s
>1B msgs/day |

- ... and different shapes

- Time series of prices
- Event data
- News data
- Metadata
- What's next?

- Requirements

- Easy to use – *and we mean easy*
- Fast – *as fast as local files*
- Scalable – *unbounded in data-size and number of clients*
- Agile – *any data shape; new shapes; iterative development*
- Complete – *all data behind the simple API*

The AHL streaming data pipeline

Arctic – our time series database



- We store all our data, of every type and shape, in Arctic. Arctic is an optionally versioned, columnar, time-series database backed by Mongo, which is entirely open source...

[manahl / arctic](https://github.com/manahl/arctic)

 Watch ▾ 108  Star 897  Fork 232

[README.md](#)

 Arctic TimeSeries and Tick store

[circleci](#)  [build](#)  [coverage](#) 96% [health](#) 91% [gitter](#) [join chat](#)

Arctic is a high performance datastore for numeric data. It supports [Pandas](#), [numpy](#) arrays and pickled objects out-of-the-box, with pluggable support for other data types and optional versioning.

Arctic can query millions of rows per second per client, achieves ~10x compression on network bandwidth, ~10x compression on disk, and scales to hundreds of millions of rows per second per [MongoDB](#) instance.

Arctic has been under active development at [Man AHL](#) since 2012.

- Check it out at <https://github.com/manahl/arctic>
- Every tick, every EOD sample, every downsampled 1-minute-bar is stored permanently in Arctic.
- **Everything is retrievable - nothing is thrown away!**



Man AHL Technology

A screenshot of a GitHub page for the 'arctic' repository. The page title is 'arctic Arctic TimeSeries and Tick store'. It features a 'arctic' logo with a blue mountain-like graphic. Below the title are buttons for 'circleci passing', 'coverage 93%', 'GITTER', and 'JOIN CHAT'. A text block describes Arctic as a high-performance datastore for numeric data, supporting Pandas, numpy arrays, and pickle objects. It also mentions MongoDB support and active development since 2012.

@ManAHLTech



■ We're hiring!

mparrott@ahl.com