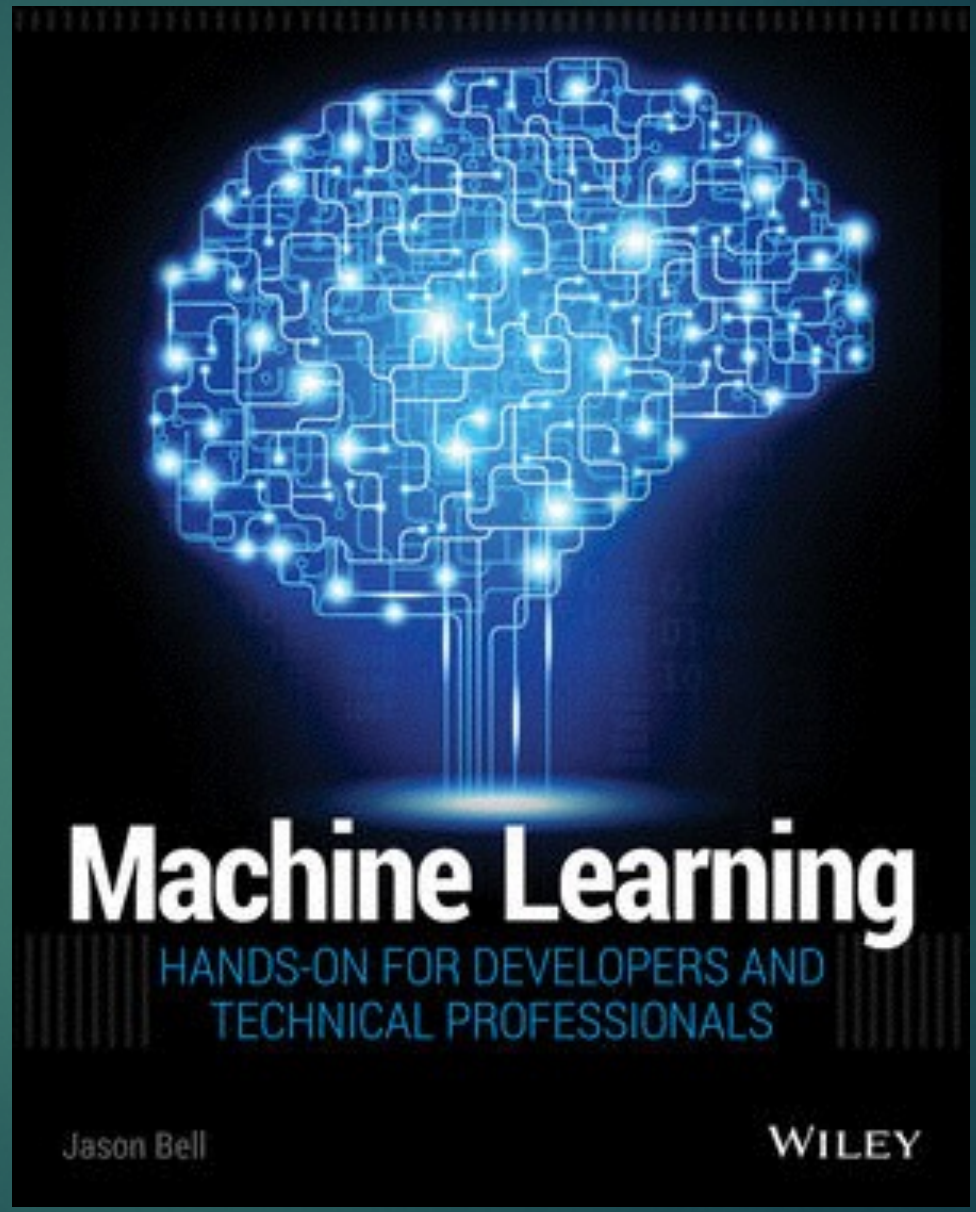
Thoughts, Experiences and Considerations with Throughput and Latency on High Volume Stream Processing Systems Using Cloud Based Infrastructures.

How I Bled All Over Onyx

Work for….

Mastodon C

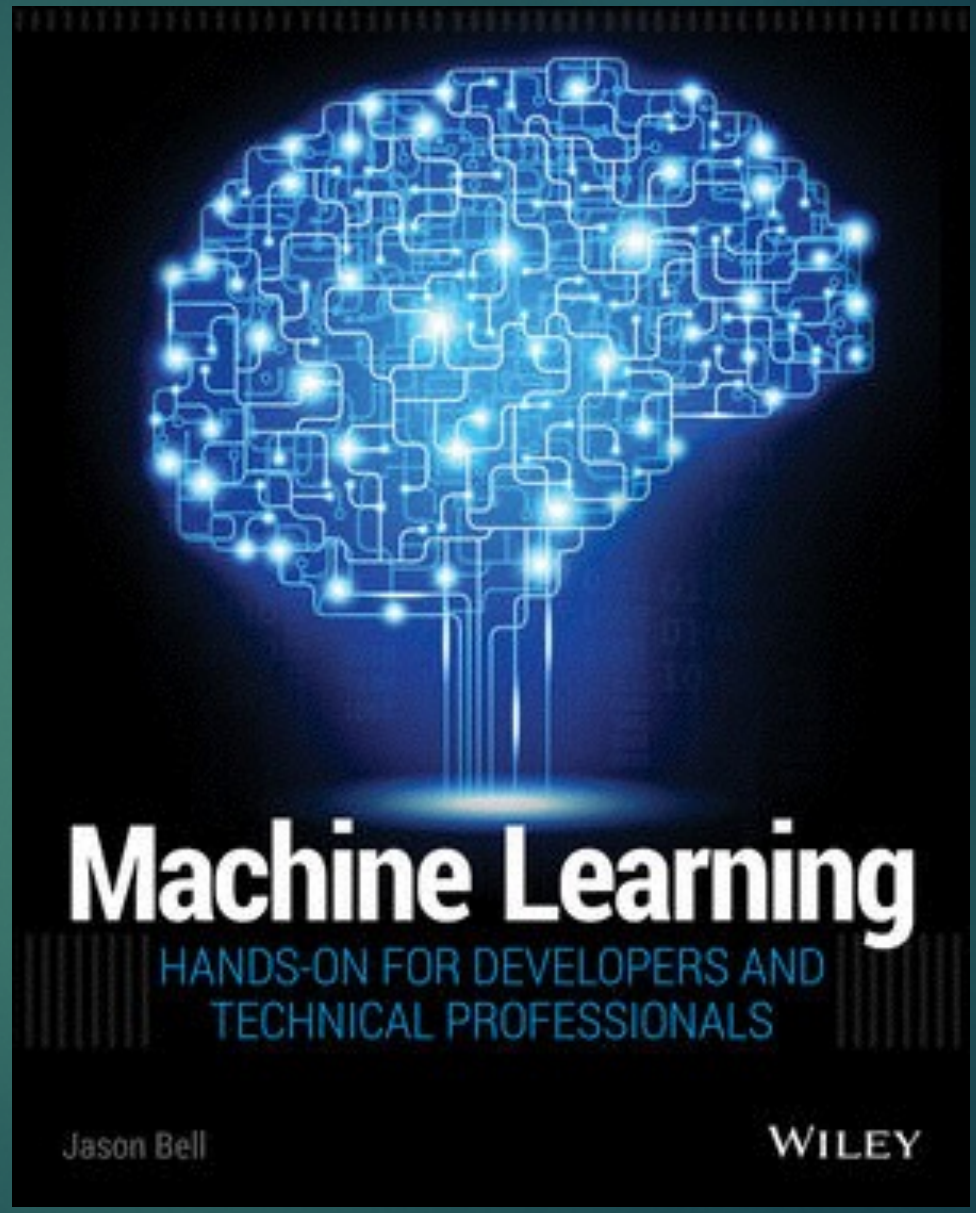
Wrote this….



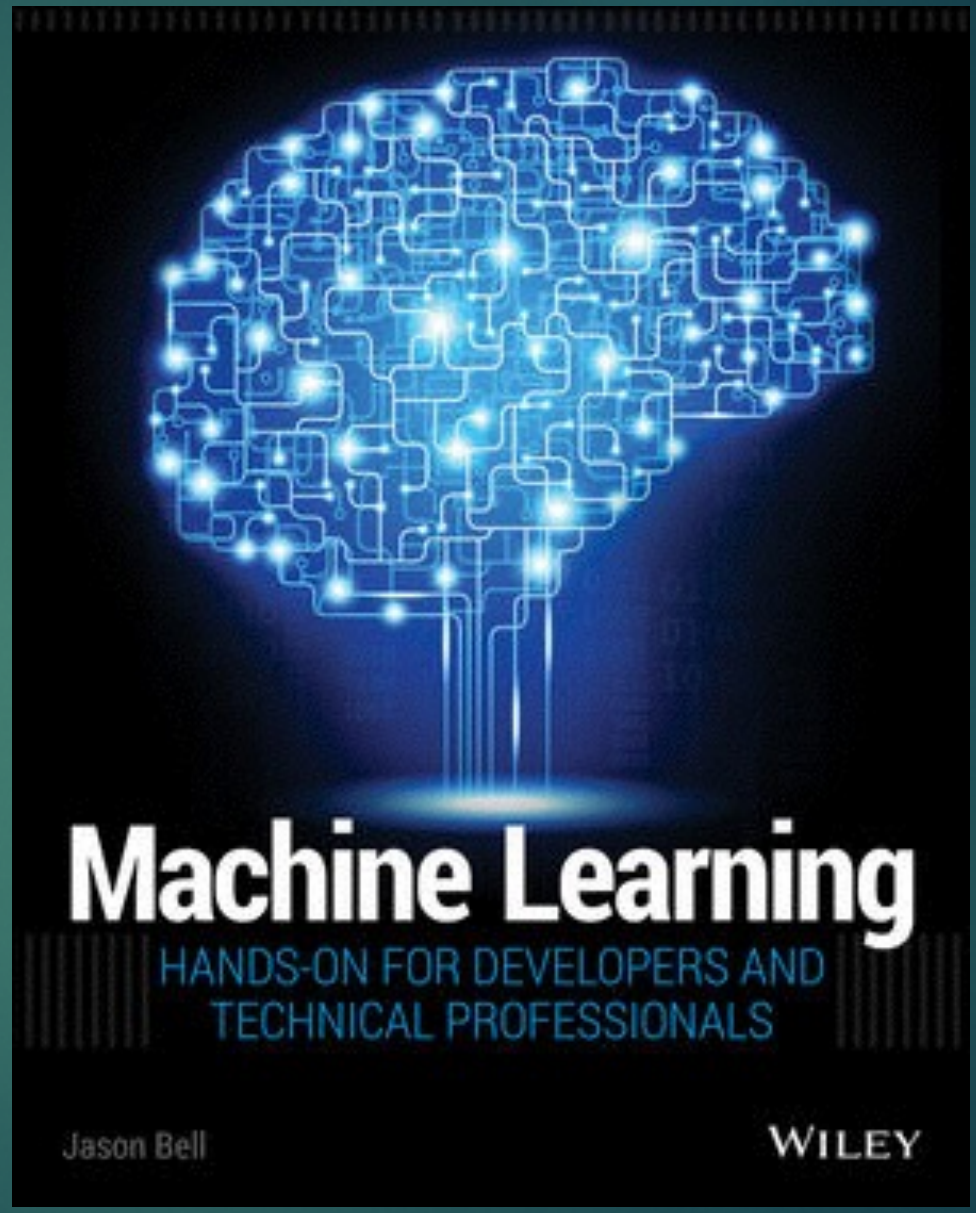
Tweets here:
@jasonbelldata

Tweet heckling is perfectly allowed.

#clojurex



Actually any heckling is perfectly allowed.

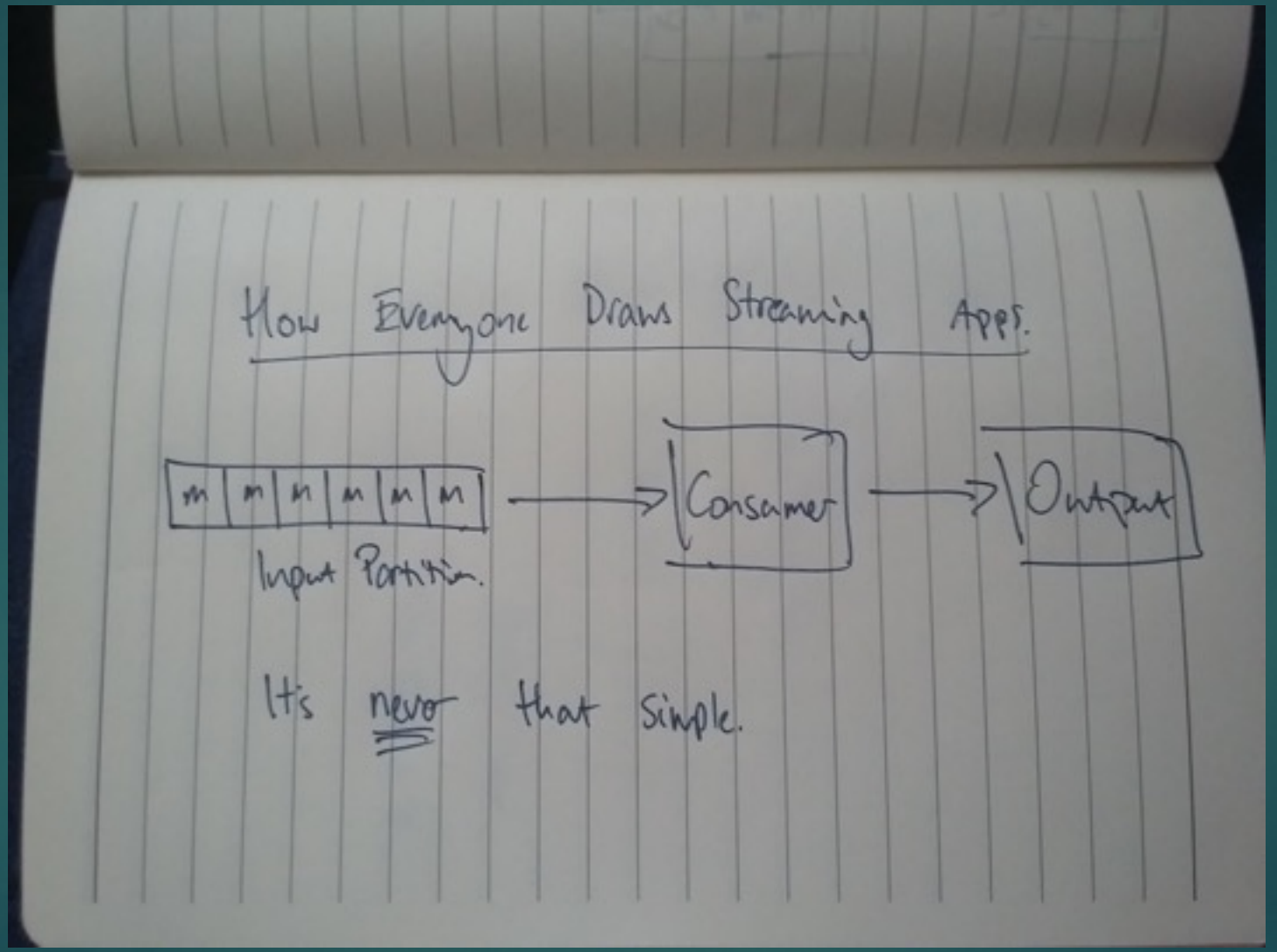


One Day I Got a Question

- "We want to stream in 12TB of data a day….. Can you do that?"

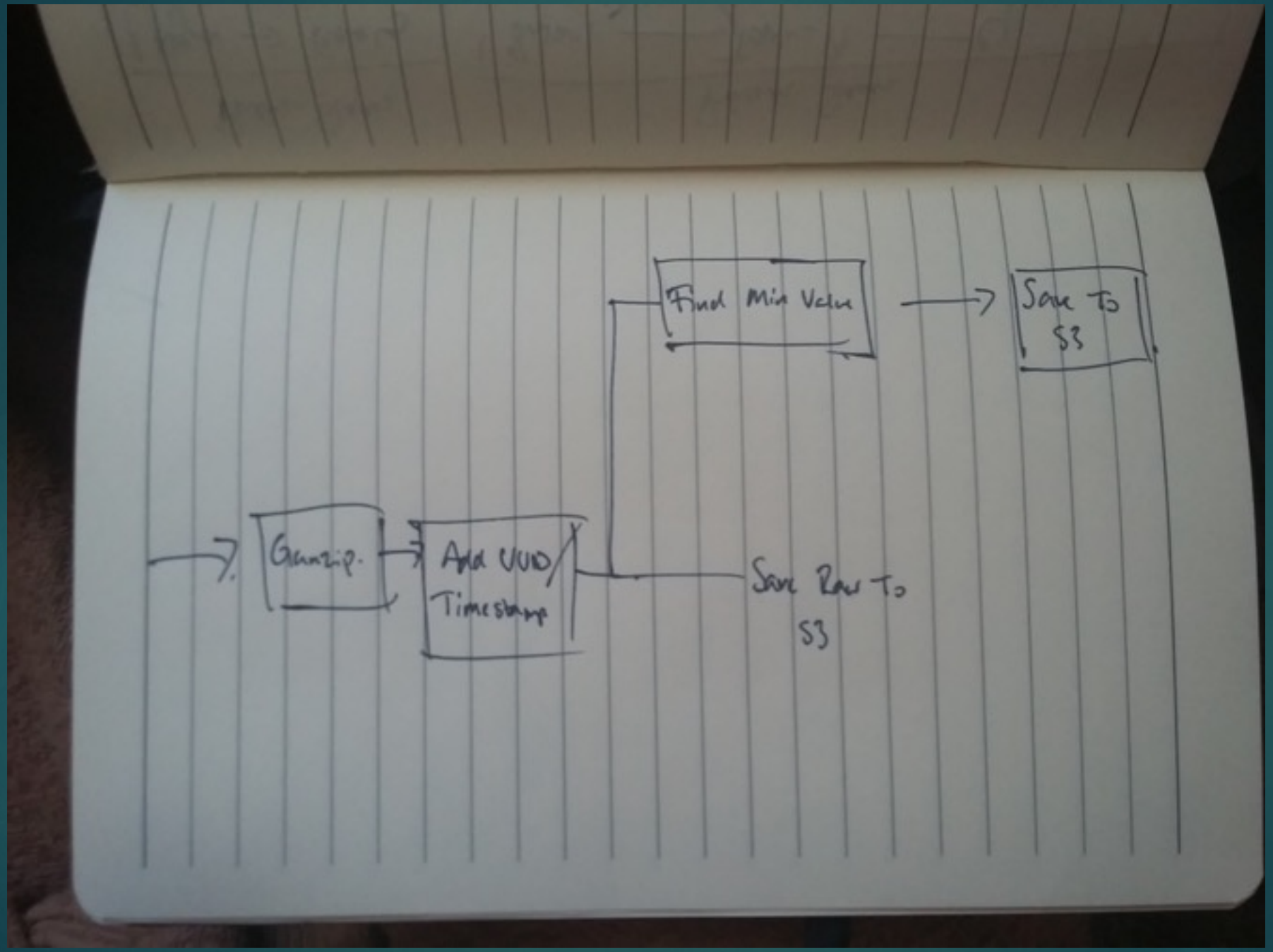
Yeah! We're gonna rock like Cameo '86

A Streaming Application… kind of.



What is Onyx?

- Distributed Computation Platform
- Written 100% in Clojure
- Great plugin architecture (inputs and outputs)
- Uses graph Direct Acyclic Graph workflow
- I spoke about it at ClojureX 2016
- It's very good.....


Find Min Value
Save To S3
Gunzip.
Add UUID Timestamp
Save Raw To S3

Onyx: Workflows as maps.

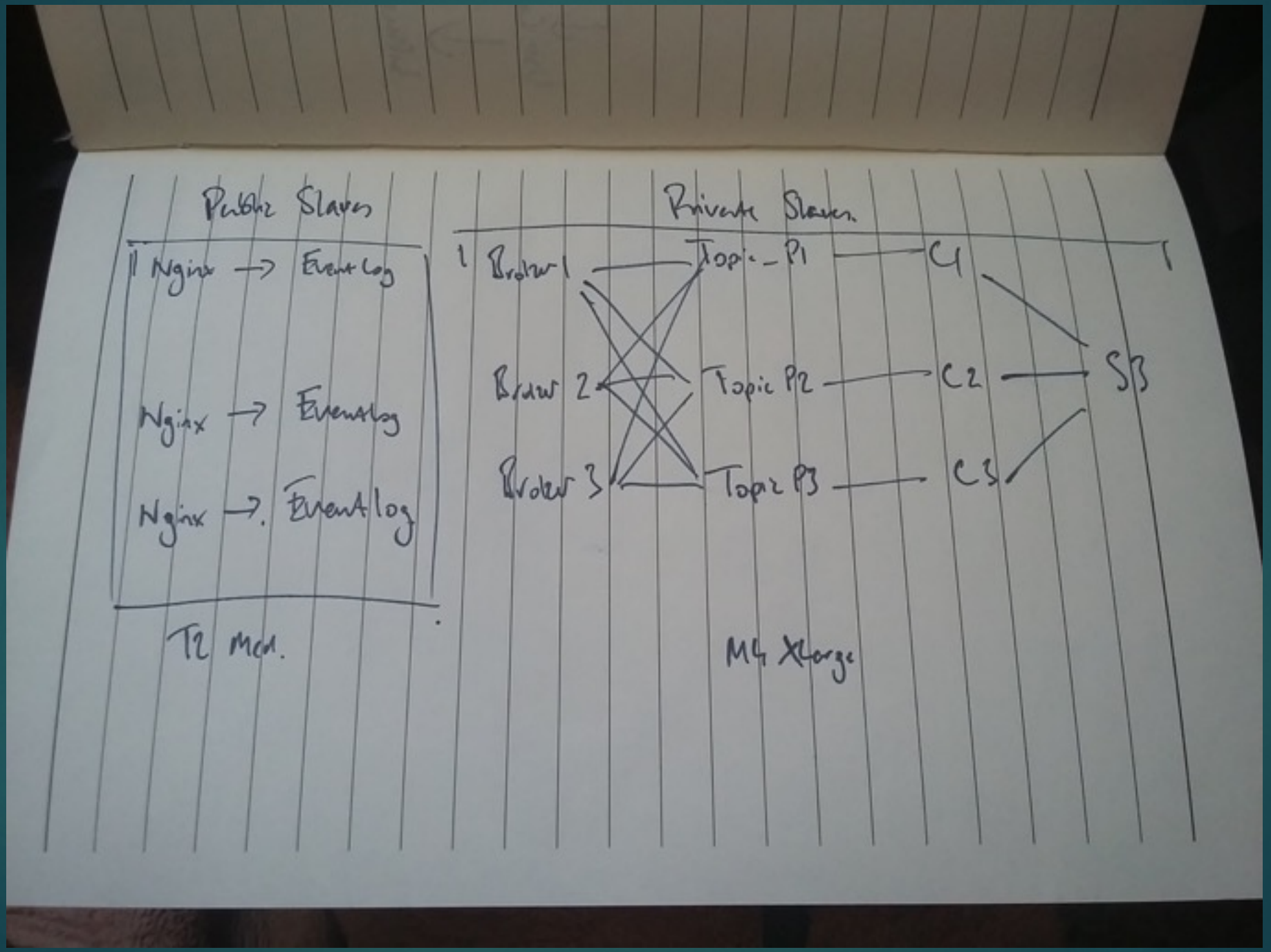
- [[:in :add-fields]
- [:add-fields :dump-raw]
- [:add-fields :find-min]
- [:find-min :dump-min]]

Onyx: Basic Peer Design

- 8 Peers
  - Per Kafka Partition (3)
  - Input task (deserialisation) (1)
  - Assign Date + UUID (1)
  - Write raw to S3 (1)
  - Find min value row (1)
  - Write min value to S3 (1)

Terraform/Marathon/Mesos Deployment

- 3 Masters (m4.large)
- 3 Public Slaves (t2.medium)
- 3 Private Slaves (m4.xlarge)
- 4TB Volumes
- We'd see how it goes from there, at worst d2 instances will do.


Public Slaves
Nginx → Event Log
Nginx → Event log
Nginx → Event log
T2 Med.
Private Slaves
Broker 1
Broker 2
Broker 3
Topic - P1
Topic P2
Topic P3
C1
C2
C3
SB
M4 Xlarge

Great! Let's Build!

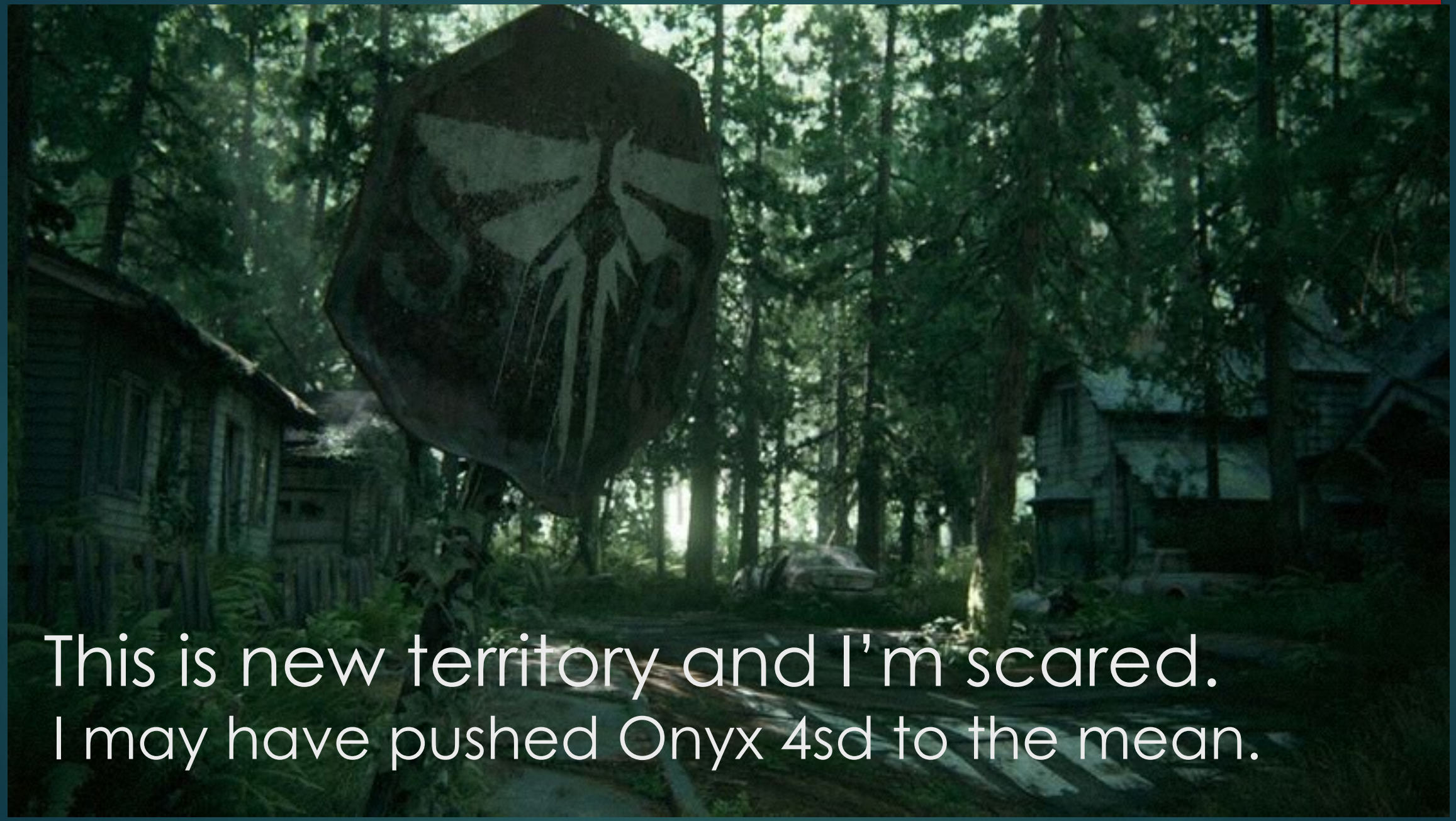
Phase 1 – Testing at 1% Volume

- It's working!
- Stuff's going to S3!
- Tell the client the good news!

Phase 2 – Testing at 2% Volume

- It's working!
- Stuff's going to S3!
- Tell the client the good news!

Phase 3 – Testing at 5% Volume


This is new territory and I'm scared.
I may have pushed Onyx 4sd to the mean.

Know Thy Framework

- Aeron buffer size = 3 x (batch-size x segment size x connections)
  - (1 x 512k x 8) x 3 = 4mb
  - Aeron default buffer is 16mb but then the **twist**
    - Onyx segment size = aeron.term.buffer.size / 8
      - Max message size of 2mb

Know Thy Framework

- Each peer has it's own lifecycle
  - If one task dies the then the whole peer structure can't process data, simple.
  - Make sure you handle lifecycle exceptions.

```
(defn handle-exception [event lifecycle lifecycle-phase e]
  (t/info "Caught exception: " e)
  (t/info "Returning :restart, indicating that this
           task should restart.")
  :restart)
```

Know Thy Framework

- Each peer has it's own lifecycle
  - Heartbeat default is 10 seconds
    - `:onyx.peer/subscriber-liveness-timeout-ms`
    - `:onyx.peer/publisher-liveness-timeout-ms`
    - **If the heartbeat times out the whole job shutdowns.**
    - So I changed it to 30 seconds.

Onyx in Docker containers.

- --shm-size ended up being 10GB
  - OOM killers were frequent
  - Java logging is helpful but still hard to deal with.

```
CGROUPS_MEM=$(cat /sys/fs/cgroup/memory/memory.limit_in_bytes)
MEMINFO_MEM=$(($(awk '/MemTotal/ {print $2}' /proc/
meminfo)*1024)) MEM=$(($MEMINFO_MEM>$CGROUPS_MEM?$CGROUPS_MEM:
$MEMINFO_MEM)) JVM_PEER_HEAP_RATIO=${JVM_PEER_HEAP_RATIO:-0.6}
XMX=$(awk '{printf("%d",$1*$2/1024^2)}' <<< " ${MEM} $
{JVM_PEER_HEAP_RATIO} ") # Use the container memory limit to
set max heap size so that the GC # knows to collect before
it's hard-stopped by the container environment, # causing OOM
exception.
```

One Friday evening…

…two things happened…

1. Watched “12 Days of Christine” and cried.

2. I did a rewrite in Kafka Streams.

- Moved the pure Clojure functions into the streams architecture.
- Used Amazonica to write to AWS S3.

Kafka Streams rewrite…

- Took 2 hours to write and deploy to DCOS.

Kafka Streams rewrite…

- Took 2 hours to write and deploy to DCOS.
- And, by this stage, was slightly tipsy.

Kafka Streams rewrite…

- In deployment Kafka Streams didn't touch more than 2% of the CPU/Memory load on the instance.
- Max memory was 790Mb compared to 8Gb Onyx jobs.

Monday Morning…

- No restarts, still processing data from all partitions!

Monday Morning…

- Now might be a good time to tell the CTO. ;)

So why did Onyx bleed so bad?

- Unzipping messages during deserialisation introduces unknowns across the peers. Very hard to monitor and predict.
- Onyx isn't designed for large messages (kb not mb)

```
(defn deserialize-gzip-message [bytes]
  (try (-> bytes
           ByteArrayInputStream.
           GZIPInputStream.
           io/reader
           slurp)
       (catch Exception e {:error e})))
```

Am I saying Onyx is bad?

- Definitely **not**. It's wonderful. You need to confirm it's wonderful for your use case.

Key Takeaways

- Design on paper first.
- Think about every function the message will visit.
- Think about the fail safes on each function/task (size, timeouts, rogue messages).

Key Takeaways

- Know the key Kafka topic settings.
- LOG.RETENTION.HOURS, LOG.RETENTION.MINUTES, LOG.RETENTION.MS, LOG.RETENTION.BYTES, MESSAGE.MAX.BYTES
- https://dataissexy.wordpress.com/2017/03/10/kafka-diaries-topic-level-settings-you-cant-ignore-part-1-data-streaming/

Key Takeaways

- Every task in the workflow will incur latency and adds up over the duration of the stream.
- Work out your TTL backstop, log retention, on Kafka (default 168 hours).

Key Takeaways

- Using Zookeeper? Monitor it with your life (or someone else's)
- Reduce disk write i/o on Kafka, throw as much RAM at the brokers as you can.
- On Kinesis you don't get the luxury of tuning and is about 70% slower on throughput.

Key Takeaways

- If your customer is not on a 24/7 SLA then work out how much data is going to pass through on the down days, then plan storage accordingly. **Kafka will log while consumers are dead.**

Key Takeaways

- Think the Streaming Rule of 72

The Streaming Rule of 72

- Work out % gain
  - (130 msg/s – 100 msg/s)/100 = 0.3 * 100 = 30%
- 72/30 = 2.4
- You will double the message throughput every 2.4 seconds.

Thank you.

- Ask me questions anytime either here or via Twitter, Linkedin, email and so on.
- If I can help I will help.