Observability for Startups
Luke Demi

@luke_demi_
No vendors were harmed in the making of this presentation
coinbase
SERVICE UNAVAILABLE

Coinbase is currently down for maintenance. Please try again later or see our status page for more information.
Web Service Time Breakdown
Web Service Time Breakdown

![Web Service Time Breakdown Graph]

- MongoDB
- Ruby
Web Service Time Breakdown
Funny thing is Coinbase supposedly uses AWS, so capacity isn’t an issue unless they don’t use autoscaling and other features, are too cheap to pay for more AWS resources, or are just straight up lazy.

They do this every time the price pumps. If you believe they can’t predict and appropriately scale their service to handle the obvious influx of new users, you are delusional. They do this purposefully.
We finally began to ask the hard questions about what was happening in our environment.
Lesson: Good instrumentation will surface your problems. Bad instrumentation will obscure them.
What is good instrumentation?

all we want is to know when and why our services break
Which really means:

- Low latency (under 1 minute)
- Long retention
- Low granularity
- High cardinality
- In-house ("secure"?)
- Fast aggregations
- Intuitive interface
Logging

Rich, open-ended context around discrete events

Pros

- Audit trails
- Debug/error messages
- High cardinality events
- Extreme granularity (down to the microsecond!)
- Very expensive (not optimized for storage)
- Slow aggregations

Cons
Tracing

End-to-end visibility into real requests

Pros

● Entire lifecycle of a request or transaction
● Granular timings from the perspective of the application
● Magical

Cons

● Lacks full context - aggregations are approximations
● Performance penalties
Metrics

- Depth of a queue (gauge)
- Count of incoming requests (counter)
- Fast aggregations
- Storage/retention optimized (burst-proof!)

Pros

Cons

Structured, lightweight and easy to aggregate

- Lacks granularity and cardinality for deeper analysis
Logs provide many, many, many of the features we wanted:

- Low latency (under 1 minute) ✔
- Long retention ❌
- Low granularity ✔
- High cardinality ✔
- In-house ("secure"?) ✔
- Fast aggregations ?
- Intuitive interface ?
We piled so much into logs...
40 nodes * 64GB RAM = 2.5TB
= 12-24 hours of logs

... so aggregating over 7 days is a little bit painful
What was wrong with tripling down on Elasticsearch/Kibana?

1. Bad interface for building alerts
2. Challenges around long term retention and management
3. Iffy aggregation support
Hiring an engineer to do log admin is like hiring a neurosurgeon to water your plants.

Splunk, Elastic, Loggly. Only one won't require an admin.
Lesson: Hammers don't work for everything. Sometimes you need a screwdriver?
● Low latency (under 1 minute) ✔
● Long retention ❌
● Low granularity ✔
● High cardinality ✔
● In-house ("secure"?) ✔
● Fast aggregations ❓
● Intuitive interface ❓
● + Alerts ❌
● + Stability ❌
Maybe Kibana was just a bad metrics provider?
What does Datadog Provide?

- Low latency (under 1 minute)
- Long retention
- Low granularity
- High cardinality
- In-house ("secure"?)
- Fast aggregations
- Intuitive interface
- + Alerts
- + Stability
Looks like we're compensating for something?

- Low latency (under 1 minute)
- Long retention
- Low granularity
- High cardinality
- In-house ("secure"?)
- Fast aggregations
- Intuitive interface
- + Alerts
- + Stability
Cardinality?!
10s flush time for aggregations

Δt

DATADOG
10s flush time for aggregations

$\Delta t$

aggregations stored per tag

DATADOG
definitely
a darker
square
definitely
40 px
x
40 px
definitely STILL
a darker square

definitely 40 px 
  x 
  40 px
● A lot can happen in 10 seconds...
● Pre aggregations feel kind of dumb
  ○ We're limiting the questions we can ask in the future
● No built in context/discoverability
● What about high cardinality?
Lesson: The grass isn't always greener on the other side of the observability fence.
<table>
<thead>
<tr>
<th></th>
<th>Datadog (Metrics)</th>
<th>Kibana (Logs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retention</td>
<td>15 months</td>
<td>7 days</td>
</tr>
<tr>
<td>Granularity</td>
<td>10s by default</td>
<td>Infinite</td>
</tr>
<tr>
<td>Cardinality</td>
<td>~10,000 unique tag combinations per metric name</td>
<td>Infinite</td>
</tr>
<tr>
<td>Aggregations</td>
<td>Fast/Instant</td>
<td>Slow</td>
</tr>
<tr>
<td>Alerting (PagerDuty/Slack)</td>
<td><strong>Supported</strong></td>
<td>Unsupported</td>
</tr>
</tbody>
</table>
When Nirvana?

- Low latency (under 1 minute)
- Long retention
- Low granularity
- High cardinality
- In-house ("secure"?)
- Fast aggregations
- Intuitive interface
- + Alerts
- + Stability
Is a perfect solution too much to ask?
Alternative 2

Double down into events

kibana

honeycomb

facebook

data dog
Questions?

@luke_demi_