observability for developers

How to Get from Here to There

@cyen
@honeycombio
DEV

WRITE → TEST → COMMIT → WRITE → TEST → COMMIT
→ WRITE → TEST → COMMIT → WRITE → TEST → COMMIT
→ WRITE → TEST → COMMIT → WRITE → TEST → COMMIT
→ WRITE → TEST → COMMIT → WRITE → TEST → COMMIT
DEV

OPS

WRITE → TEST → COMMIT → RELEASE

→ DEBUG → FIX
DEV  ★  OPS

"Works on my machine"

"The only good diff is a red diff"
"Observation 1: Change is the most common trigger"

—Subbu Allamaraju, Expedia, Feb 2019
https://m.subbu.org/incidents-trends-from-the-trenches-e2f8497d52ed
DEV

"Works on my machine"

OPS

"The only good diff is a red diff"
THE FIRST WAVE: getting ops folks to code

THE SECOND WAVE: teaching devs to own code in production
The Software Development Process

- Design documents
- Architecture review
- Test-driven development
- Integration tests
- Code review
- Continuous integration
- Continuous deployment
- Observe our code in production
monitoring

The system as black box magic. Thresholds, alerts, system signals like CPU and memory.

Checking and rechecking for known bad behaviors.

observability

The system as a living, adaptable thing. A culture of instrumentation and metadata rather than strictly-defined counters.

Being able to tease out previously-unknown bad behaviors and outliers.
observability

a.k.a. understanding the behavior of a system based on knowledge of its external outputs.

a.k.a. "what is my software doing, and why is it behaving that way?"
"Works on my machine"  "The only good diff is a red diff"

"How is it working for the user?"
What Does Observability-Driven Development look like?
DEBUG PRODUCTION SYSTEMS
Locally: log lines, printfs, debuggers attached to our IDEs

In production: we only have the data we captured when it happened

Make it as easy as possible to add new data as needed
"My data isn’t showing up in Honeycomb!"

+ event_time_delta_sec
IMPROVE IN PROD
"Test in Prod"... doesn’t mean only testing in prod

Testing: for **known knowns**
Monitoring: for **known unknowns**
Observability: for **unknown unknowns**

—Jez Humble
FEATURE FLAGS 💖

IMPROVE
VERIFY
(PROD)

[~] BREAK DOWN
flags.varstring

Jul 5 2017, 2:21 PM – Jul 6 2017, 1:35 AM

Results BubbleUp Raw Data

COUNT

SUM(count)

AVG(cum_write_latency_msec)
IS IT STILL WORKING? LET’S OBSERVE
Watch to make sure reality lines up with expectations

... in the terms that we understand intimately
different build IDs running concurrently

whoa! this build sped things up a lot!
Instrumentation (Getting Data In)
- Best Practices
- Taking the First Few Steps
- Migrating from Unstructured Text Logs
- Stop Searching, Start Analyzing
- Tracing as a New Frontier
BEST PRACTICES FOR INSTRUMENTATION

- Capture contextual, structured data

```json
{
  Timestamp: "2018-03-20T00:47:25.339Z",
  content_length: 172,
  database_dur_ms: 15.79283,
  endpoint: "/posts/15",
  method: "PUT",
  request_dur_ms: 72.446625,
  render_dur_ms: 25.31729,
  service_name: "api",
  user_token: "2e6cfd4"
}
```
BEST PRACTICES FOR INSTRUMENTATION

- Capture contextual, structured data
- Common set of nouns and consistent naming
BEST PRACTICES FOR INSTRUMENTATION

- Capture contextual, structured data
- Common set of nouns and consistent naming
- Instrument from the perspective of what you can control

**USER**
- user_id
- endpoint
- request_dur_ms
- response_status_code

**APP**
- hostname
- active_queue
- query_sql
- caller_fn
- database_dur_ms
- num_rows_returned

**DATABASE**
TAKING THE FIRST FEW STEPS

» Describe your basic "unit of work" and identify where it "enters" the system
TAKING THE FIRST FEW STEPS

▸ Describe your basic "unit of work" and identify where it "enters" the system

▸ Identify metadata to help you isolate unexpected behavior in your business logic

<table>
<thead>
<tr>
<th>Your Infra</th>
<th>Your Deploy</th>
<th>Your Business</th>
<th>Your Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>- hostname</td>
<td>- version / build</td>
<td>- customer</td>
<td>- payload characteristics</td>
</tr>
<tr>
<td>- machine type</td>
<td>- feature flags</td>
<td>- shopping cart</td>
<td>- timers</td>
</tr>
</tbody>
</table>
TAking the First Few Steps

- Describe your basic "unit of work" and identify where it "enters" the system
- Identify metadata to help you isolate unexpected behavior in your business logic
- Experiment! Add temporary fields when needed to validate hypotheses
TAKING THE FIRST FEW STEPS

- Describe your basic "unit of work" and identify where it "enters" the system
- Identify metadata to help you isolate unexpected behavior in your business logic
- Experiment! Add temporary fields when needed to validate hypotheses
- Prune stale fields (if necessary)
MIGRATING FROM UNSTRUCTURED TEXT LOGS

2019-01-25T01:30:23.743Z Enqueued task
2019-01-25T01:30:24.120Z Task processed, returning 42 entries
2019-01-25T01:30:24.212Z Task complete (email sent to foobar@example.com)
2019-01-25T01:30:26.014Z Enqueued task
2019-01-25T01:30:26.214Z Enqueued task
2019-01-25T01:30:24.120Z Task errored: unknown constant ::Fixnum
2019-01-25T01:30:26.214Z Enqueued task
2019-01-25T01:30:24.212Z Task complete (email sent to foobar@example.com)
2019-01-25T01:30:24.212Z Task complete, (email sent to bazqux@example.com)

2019-01-25T01:30:29.953Z Task timed out after 6.01 seconds
2019-01-25T01:30:32.762Z Enqueued task
2019-01-25T01:30:32.993Z Task processed, returning 7 entries
2019-01-25T01:30:33.132Z Task complete (email not found, noop)
2019-01-25T01:30:34.243Z Task processed, returning 0 entries
2019-01-25T01:30:34.243Z Task complete, (email sent to bazqux@example.com)
MIGRATING FROM UNSTRUCTURED TEXT LOGS

- Identify entities that are relevant to your business logic (and include them in your logs!)

2019-01-25T01:30:29.953Z Task timed out after 6.01 seconds task_id=72 type=process
MIGRATING FROM UNSTRUCTURED TEXT LOGS

- Identify entities that are relevant to your business logic (and include them in your logs!)
- Start introducing structure into your logs

```
Timestamp=2019-01-25T01:30:29.953Z
message=Task timed out after 6.01 seconds
task_id=72
type=process
```
MIGRATING FROM UNSTRUCTURED TEXT LOGS

- Identify entities that are relevant to your business logic (and include them in your logs!)
- Start introducing structure into your logs
- Build up **context** instead of outputting disjoint lines

```plaintext
2019-01-25T01:30:29.953Z Enqueued task task_id=72 type=enqueue target=email
2019-01-25T01:30:23.743Z Enqueued task task_id=72 type=enqueue target=email
2019-01-25T01:30:29.953Z Task timed out after 6.01 seconds task_id=72 type=process
```

```plaintext
Timestamp=2019-01-25T01:30:29.953Z  target=email
message=Task timed out after 6.01 seconds  queue_dur_ms=200
task_id=72  timeout_dur_ms=6010
```
STOP SEARCHING, START ANALYZING

- Logs were conceived to store and find history, not for analytics

2019-01-25T01:30:23.743Z Enqueued task
2019-01-25T01:30:24.120Z Task processed, returning 42 entries
2019-01-25T01:30:24.212Z Task complete (email sent to foobar@example.com)
2019-01-25T01:30:26.014Z Enqueued task
2019-01-25T01:30:26.214Z Enqueued task
2019-01-25T01:30:24.120Z Task errored: unknown constant ::Fixnum
2019-01-25T01:30:29.953Z Task timed out after 6.01 seconds
2019-01-25T01:30:32.762Z Enqueued task
2019-01-25T01:30:34.243Z Task processed, returning 0 entries
2019-01-25T01:30:34.243Z Task complete, (email sent to bazqux@example.com)
STOP SEARCHING, START ANALYZING

- Logs were conceived to store and find history, not for analytics
- Logs are no longer human-scale — they are machine-scale
STOP SEARCHING, START ANALYZING

- Logs were conceived to store and find history, not for analytics
- Logs are no longer human-scale — they are machine-scale
- Visualizations are necessary to identify an outlier as a trend or an anomaly
TRACING AS A NEW FRONTIER

- Tracing: not just for concurrent or distributed systems
### TRACING AS A NEW FRONTIER

- Tracing: not just for concurrent or distributed systems

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Event Details</th>
<th>Task ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-01-25T01:30:23.743Z</td>
<td>Enqueued task</td>
<td>72</td>
</tr>
<tr>
<td>2019-01-25T01:30:24.120Z</td>
<td>Enqueued task</td>
<td>74</td>
</tr>
<tr>
<td>2019-01-25T01:30:24.212Z</td>
<td>Task processed, returning 42 entries</td>
<td>74</td>
</tr>
<tr>
<td>2019-01-25T01:30:26.014Z</td>
<td>Task complete (email sent to <a href="mailto:foobar@example.com">foobar@example.com</a>)</td>
<td>74</td>
</tr>
<tr>
<td>2019-01-25T01:30:26.214Z</td>
<td>Enqueued task</td>
<td>77</td>
</tr>
<tr>
<td>2019-01-25T01:30:24.120Z</td>
<td>Task errored: unknown constant ::Fixnum</td>
<td>77</td>
</tr>
<tr>
<td>2019-01-25T01:30:29.953Z</td>
<td>Task timed out after 6.01 seconds</td>
<td>72</td>
</tr>
<tr>
<td>2019-01-25T01:30:32.762Z</td>
<td>Enqueued task</td>
<td>78</td>
</tr>
<tr>
<td>2019-01-25T01:30:34.243Z</td>
<td>Task processed, returning 0 entries</td>
<td>78</td>
</tr>
<tr>
<td>2019-01-25T01:30:34.243Z</td>
<td>Task complete, (email sent to <a href="mailto:bazqux@example.com">bazqux@example.com</a>)</td>
<td>78</td>
</tr>
</tbody>
</table>
### TRACING AS A NEW FRONTIER

- Tracing: not just for concurrent or distributed systems
- A series of related log lines can, in fact, share a lot in common with a trace

<table>
<thead>
<tr>
<th>service_name</th>
<th>trace_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>span_id</td>
</tr>
<tr>
<td>duration_ms</td>
<td>parent_id</td>
</tr>
</tbody>
</table>

- trace_id: 1
  - span_id: A
  - span_id: B, parent_id: A
  - span_id: C, parent_id: B
TRACING AS A NEW FRONTIER

- Tracing: not just for concurrent or distributed systems
- A series of related log lines can, in fact, share a lot in common with a trace
- Tracing will be commonplace in 2019

[0]: https://monitoring.love/articles/2019-predictions/
TRACING AS A NEW FRONTIER

- Tracing: not just for concurrent or distributed systems
- A series of related log lines can, in fact, share a lot in common with a trace
- Tracing will be commonplace in 2019
- Aggregate analysis of traces is still key
DEV

WRITE → TEST → COMMIT → RELEASE → OBSERVE

TEST

OBSERVE
DEVELOPMENT (DEV) → OPERATIONS (OPS)

WRITE → TEST → COMMIT → RELEASE → OBSERVE

TEST

OBSERVE
DEV

OPS
DEV❤️OPS
DEVS, OUR MISSION:

- Stop writing software based on intuition, start backing it up with data
- Teach observability tools to speak more than "Ops"
- ??? (← ask lots of questions and validate hypotheses)
- Profit!
ASK NEW QUESTIONS

SHIP BETTER SOFTWARE

thanks!

@cyen
@honeycombio