Submit your questions

Ask and upvote questions via the PagerDuty mobile app! We will do Q&A at the end of the session.

Find Session > Session Engagement
getting comfortable in prod
to improve your life in dev

@cyen
@honeycombio
first, some background…
DEV

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

"Works on my machine"

OPS

"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
THEN

NOW
DEVOPS

"Works on my machine"

"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
it’s all about sharing
SOFTWARE OWNERSHIP
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?"
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.
DEV  →  OPS

WRITE  →  TEST  →  COMMIT  →  RELEASE

💥  →  DEBUG  →  FIX
Now we have the data coming in. Now

“it no longer feels like a scary fucking conundrum” ← direct quote from the maker of the spreadsheet

5:44 PM - 4 Sep 2019
… why devs, again?
The Software Development Process

- Design documents
- Architecture review
- Test-driven development
- Integration tests
- Code review
- Continuous integration
- Continuous deployment
- Observe our code in production
--- FAIL: TestUnitTest (0.00s)

talk_test.go:10: —

expected: 4 (type int)

actual: 5 (type int)
"Works on my machine"

"The only good diff is a red diff"
DEV$\rightarrow$ PROD still observability
prod, part of the dev process?
The Software Process

- Design documents
- Architecture review
- Test-driven development
- Integration tests
- Code review
- Continuous integration
- Continuous deployment

(Wait for exception tracker to complain)

when deciding…

WHAT to build

HOW TO build it

WHETHER it works
("test in prod")
Locally: log lines, printfs, debuggers attached to our IDEs

What’s causing our code to deviate from expectations?

Stop "pulling straws"—quantify pain, and start prioritizing.
HOW TO

▸ Know what "normal" really is
▸ Events (instrumentation) can be like DEBUG statements in prod
▸ What and how we build should be informed by reality
Complex systems have an infinitely long list of black swan failure scenarios

"Test in Production" to experiment and check hypotheses

Feature flags + observability = 💛
but this is hard.
make prod feel more like dev
TOOLS SHOULD SPEAK MY LANGUAGE

- As a dev, traditional monitoring tools don't tie back to the concepts I deal with in my code

  - AWS availability zone
  - CPU utilization
  - kafka partition
  - Cassandra hostname
  - $YOUR_BIZ-relevant ID
  - time to render
  - API endpoint
  - payload size
  - build ID
  - client OS
TOOLS SHOULD SPEAK MY LANGUAGE

- As a dev, traditional monitoring tools don't tie back to the concepts I deal with in my code

AWS availability zone

- us-east-1
- us-west-1
- eu-west-1
- us-west-2
- eu-central-1

customer ID
Jan 29 2019, 10:18 PM ~ Jan 30 2019, 12:18 AM

Results

<table>
<thead>
<tr>
<th>build_id</th>
<th>COUNT</th>
<th>COUNT_DISTINCT(server_hostname)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18817</td>
<td>144,388</td>
<td>60</td>
</tr>
<tr>
<td>18812</td>
<td>83,639</td>
<td>60</td>
</tr>
</tbody>
</table>
TOOLS SHOULD SPEAK MY LANGUAGE

- As a dev, traditional monitoring tools don't tie back to the concepts I deal with in my code

AND LET ME ITERATE
SHARE PATTERNS WHERE POSSIBLE

- Tracing helps production feel even more familiar: can map a trace directly to my code structure

```python
while len_p < nb_primes:
    for i in p[:len_p]:
        if n % i == 0:
            break
    else:
        p[len_p] = n
        len_p += 1
        n += 1
```
PROD SHOULD FEEL LIKE DEVELOPMENT?
<table>
<thead>
<tr>
<th>name</th>
<th>service_name</th>
<th>0s</th>
<th>10s</th>
<th>20s</th>
<th>30s</th>
<th>40s</th>
<th>50s</th>
<th>55.16s</th>
</tr>
</thead>
<tbody>
<tr>
<td>/api.RetrieverService/Fetch</td>
<td>poodle</td>
<td>55.148s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/api.RetrieverService/Fetch</td>
<td>retriever</td>
<td>55.147s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>head.startQuery</td>
<td>retriever</td>
<td>0.3451ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>head.fetchMerged</td>
<td>retriever</td>
<td>53.230s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FetchPartialMerge</td>
<td>retriever</td>
<td>27.086s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FetchPartialMerge</td>
<td>retriever</td>
<td>29.264s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>merging</td>
<td>retriever</td>
<td>28.889s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>1.311s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>401.0ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>388.3ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>398.7ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>385.0ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>962.2ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>785.1ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>809.1ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>793.4ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>828.4ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td>841.4ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mergeBucket</td>
<td>retriever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fields**

- duration.ms: 27086.312459
- global.availability_zone: us-east-1d
- global.build_id: 110870
- global.env: production
- global.infra_type: aws_instance
- global.instance_type: 
- global.instance_type: 
CHANGE CAN BE INCREMENTAL

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)

2019-01-25T01:30:29.953Z Task timed out after 6.01 seconds
task_id=72

Timestamp=2019-01-25T01:30:29.953Z
message=Task timed out after 6.01 seconds
task_id=72
CHANGE CAN BE INCREMENTAL

2019-01-25T01:30:23.743Z Enqueued task task=72
2019-01-25T01:30:24.120Z Enqueued task task=74
2019-01-25T01:30:24.212Z Task processed, returning 42 entries task=74
2019-01-25T01:30:26.014Z Task complete (email sent to foobar@example.com) task=74
2019-01-25T01:30:26.214Z Enqueued task task=77
2019-01-25T01:30:24.120Z Task errored: unknown constant ::Fixnum task=77
2019-01-25T01:30:29.953Z Task timed out after 6.01 seconds task=72
2019-01-25T01:30:32.762Z Enqueued task task=78
2019-01-25T01:30:34.243Z Task processed, returning 0 entries task=78
2019-01-25T01:30:34.243Z Task complete, (email sent to bazqux@example.com) task=78
at the end of all of this...
DEV OPS
WRITE → TEST → COMMIT → RELEASE → OBSERVE

TEST

OBSERVE
OPS: share the great responsibility (and great power!)

DEVS: embrace observability, bring production closer to development.
thanks!

@cyen
@honeycombio

ASK NEW QUESTIONS
SHIP BETTER SOFTWARE

CURIOUS? TRY play.honeycomb.io
We value your feedback!

Take the session survey through the PagerDuty Summit App!

Find Session > Session Feedback > Track Session Survey