Observability in DevSecOps

Wider than three pillars

BCS DevSecOps 2025

Andrew Hardie, Devoperative

About me...

- Wrote my first BASIC program in 1969
- On an ASR33:
  - Paper tape!
  - 10 Chars/sec
  - Current loop i/f
  - Line based editor
  - LOUD!
- High status!

About me...

- Then, progressed to a Televideo 920 VDU

- This cost $945, more than the ASR33 ($840)!
- Offered 9600 bps, but modem was 1200 baud.

About me...

- 8” floppy disk,128 Kb
- Hand built first computer
- My first hard disk was 10MB

- My first Open Source software was the Lawrence Livermore BASIC interpreter
- supplied on 9 track mag tape
- I had to convert to 8” floppy
- Using a VAX cluster!

About me...

- Since then…
  - Industrial electronic control systems
    - Razor blades, choc bars, yarn winding, hydraulic bending machine
    - Early microprocessor instruments, e.g. replacing 3-term PID TCs
    - Military systems, artillery, tanks, data recording, crypto, etc
  - Seven parliaments (including the UK House of Commons, to computerise Hansard and Parliamentary Questions),
  - Eight UK Government departments (including BEIS (DTI), MoD, MoJ, Home Office, FCO, CCTA, DHSS,), Metropolitan Police, three Local Authorities, Tony Blair's private office, four palaces (including the UK the Royal Household), the Council of Europe in Strasbourg
  - HSBC, Lloyds Bank, global BI company and two hi-tech startups.

Observability - OLLY to his friends

- Born in 1960
- Father: Rudolf E Kàlmàn (Hungarian-American)
- Birth Cert said: "*to describe mathematical control systems*"
- Graduated as "*Control Theory*"
- Inferring internal states of a system from its external outputs
- Stuff like: $M(t_0, t_1) = \int_{t_0}^{t_1} \varphi(t, t_0)^T C(t)^T C(t) \varphi(t, t_0)\, dt$

OLLY is all grown up now...

- Observability is the hot new thing … but why?
- Containers are black boxes
- Kubernetes is a black hole
- Service Meshes are a black art
- You can't (or *shouldn't be able* to) get inside!
- You know only what they tell you:
  - By themselves
  - By answering your questions

Those three pillars...

- Metrics
- Logs
- Traces
- Various vendors offer products or services based on those three pillars (which they did before) but now claim that it’s Observability
- However...

Metrics

- Metrics are aggregates
  - Over time (e.g. ten second intervals)
  - Of a parameter that is itself an aggregate (CPU, Mem) with multiple overlapping contributory causes
  - Correlation does not imply causation
    - (Error rate may increase along with traffic rate but error percentage can remain the same)
- Metrics/Alerts are inherently *reactive*
- TSDBs do naughty things
  - Compression, e.g. delta encoding, delta of deltas

Metrics #2

- Not only do we have metrics, we have alerts!
- But, metrics & alerts require prior knowledge:
  - Why is this parameter significant?
  - Why is this threshold significant?
- Monitoring is for known/knowns
- Observability is for unknown/unknowns
- The “Fourth Quadrant” – unknown states!
- Not just “What happened?”, but to help us find out “*Why* did it happen?”

Really Important Metrics

- Not fished out of Prometheus (although it is great)
- Ask DORA! (DevOps Research and Assessment)
  - Lead time for changes
  - Deployment frequency
  - Mean time to recovery
  - Change failure rate
  - The 5th metric: from availability to reliability (the other DORA – Digital Operational Resilience Act)
- Thus: Observability is every bit as important in your CI/CD pipeline as it is in platform & running apps!

Logs

- We have logs! Lots of logs!
- We stick them all in ElasticSearch!
- And, our metrics – Noooo! (Cardinality Aggregation!)
- We hook it up to Kibana! Er, ok...
- Result!
- Er, not quite
  - Logs usually unstructured
  - No context – where was this called from?
  - Correlated how?

What are you logging?

- Platform logs
- Deployment logs
- Application logs
  - log event generation in your code!
- Open Telemetry project created for this purpose
  - Tools, APIs and SDKs
- Excellent intro available here:
  - https://opentelemetry.io/docs/concepts/observability-primer/

Traces

- Timestamps added to TCP headers
- Time delays can thus be calculated
- Within and between services
- Examples: Jaeger, Zipkin
- Tracking all the stages of request/response network traffic through your systems
- eBPF is also brilliant for this, and many other things – worthy of a presentation on its own

Spans

- Bounded by the duration of an event in a node
  - Think “unit of work”
- Each step or micro-service creates a span
- As rich content as possible
  - Trace, metrics, logs, arbitrary – anything relevant
  - Metadata
  - Logs - platform and application
  - Think: multi-disciplinary information capture
- Structured records – e.g. JSON, key/values

Distributed Traces

- Think Trace plus Span
- Made up of one or more spans
  - Linked back to previous step by span ID
- Shows what happens with the flow of a request through the distributed (e.g. microservices) system from start to finish
- With identifying metadata - ‘baggage’
- Now, we are getting closer to Observability

Properly Instrumented Apps

- Applications must emit *signals*:
  - Traces, Metrics, Logs, events
- Application is properly instrumented when it is not necessary to add more instrumentation or code to identify and troubleshoot a problem
- Application devs should not just leave it to the platform level logging or orchestrator system
- “You write it, you run it” needs to include “you write it, you instrument it, so *you* can debug it!”

Type of data records

- Cardinality
  - Uniqueness of values
  - High-cardinality – e.g. UUID, designed to be unique
- Dimensionality
  - Number of keys
  - The more dimensions, the more correlation can be done over the data to look for patterns of behaviour

Basic rules

- Catch as much as you can, as you don’t know what you might need to solve those unknown fourth-quadrant problems.
- Storage is cheap. Your time is not.
- The more complex and distributed your systems the harder they are to instrument
- ‘Monitoring’ ≠ Observability
- Deployment observability is every bit as important as running systems observability

Fantasy design

- Ship those structured span records to Kafka
- Process using, e.g.:
  - Vector (https://vector.dev) - bought by Datadog
  - Apache Hop (https://hop.apache.org) - new data integration platform – Hop native, or in Spark/Flink
  - Kafka Streams (library for building streaming apps)
  - Spark, Flink (more and more with Kafka)
  - Etc, etc, etc

Kafka Topic Stream concept

- Take the raw input records
  - Enrich the record in various ways (lookups, relations)
  - Write to a new topic stream
  - Fire off alerts where wanted/useful
  - Rinse and repeat...
- The target goal being topic stream(s) with:
  - ***Actionable Intel*** – problem plus as much context as possible to help solve that problem
    - For humans
    - And auto-remediation systems

Plus...

- Observability systems can do far more than just DevOps, debugging and alerts.
- Can serve other audiences in the organisation:
  - BizInfo – how are products/services being used?
  - Audit – who did what and when, with context?
  - Security – same…
  - And more…
- It’s the way ahead!

Er, Wider?

- Why ‘wider’?
  - Just write out everything you can gather
  - Including source/target end stuff where available
  - You don’t know what you might need later
  - To help you with those unknown unknowns

Excellent references on Wide Events:

- https://jeremymorrell.dev/blog/a-practitioners-guide-to-wide-events/
- https://isburmistrov.substack.com/p/all-you-need-is-wide-events-not-metrics
- https://charity.wtf/2022/08/15/live-your-best-life-with-structured-events/

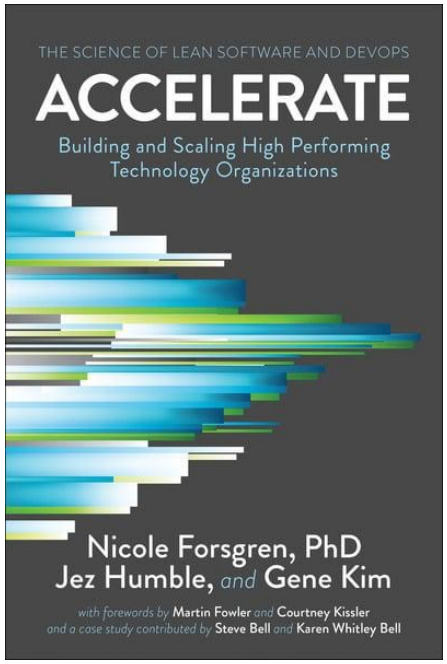
Er, DevSecOps?

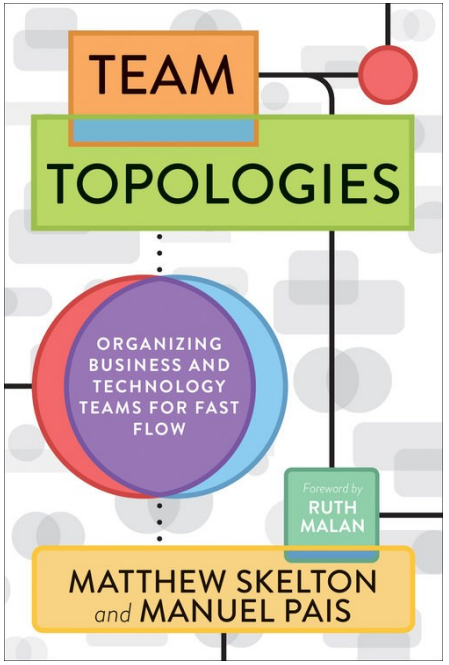
- Surprise: your DevOps CI/CD toolchain again!
- Auditable builds – who built what, when and why?
- Supply chain integrity – using a ‘sheepdip’?
- Dependency import tracking – remember log4j!!!
- If person, component or source in the supply chain is compromised, then I want to know:
- What built artefacts are affected? What references them?
- Where are they stored? Are they eligible for deployment?
- Are they running? Where???

Conclusions

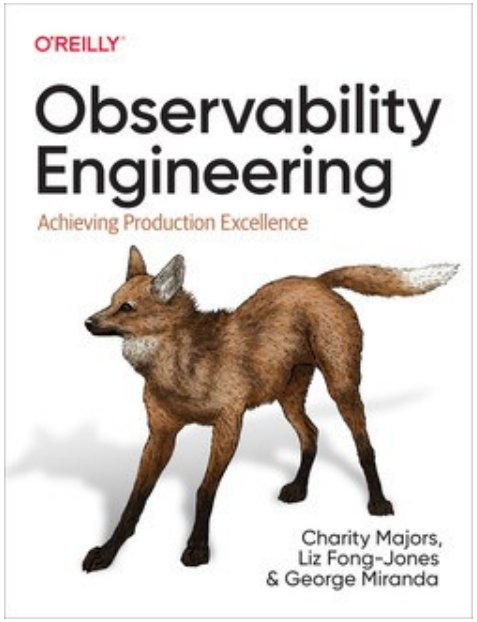
- Observability (true OLLY) must replace ‘monitoring’
- The ‘three pillars’ are not enough, and never were
- Apps cannot be just passive, relying on system level observability – they have to emit signals of their own
- Capture as much as you can, in wide records:
  - High cardinality
  - High dimensionality
- Don’t just rely on text search and GUI queries
- Work with your OLLY data to make it more useful!

README.md


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell


TEAM
TOPOLOGIES
ORGANIZING BUSINESS AND TECHNOLOGY TEAMS FOR FAST FLOW
Foreword by
RUTH MALAN
MATTHEW SKELTON
and MANUEL PAIS


O'REILLY
Site Reliability Engineering
HOW GOOGLE RUNS PRODUCTION SYSTEMS
Edited by Betsy Beyer, Chris Jones,
Jennifer Petoff & Niall Murphy


O'REILLY
Observability Engineering
Achieving Production Excellence
Charity Majors,
Liz Fong-Jones
& George Miranda

Questions?

- Andrew Hardie
- https://www.linkedin.com/in/andrew-hardie-31b982/