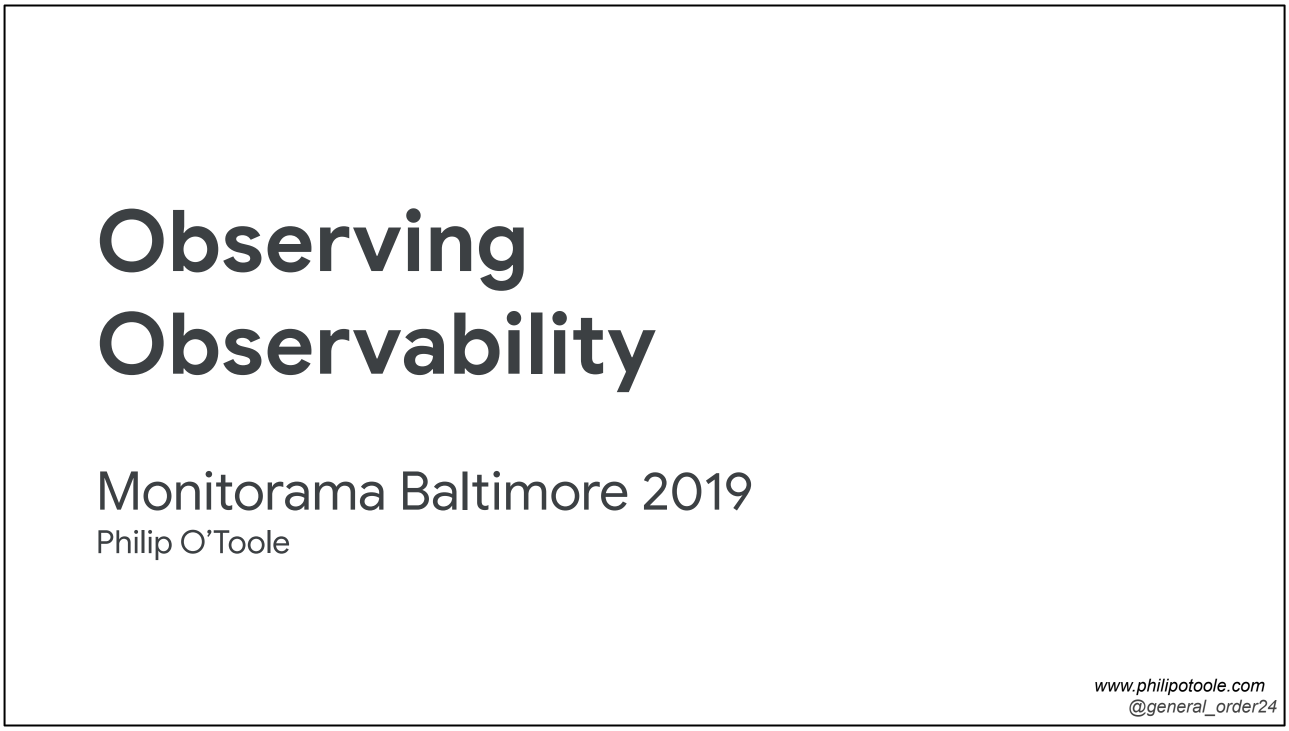


Observing Observability

Monitorama Baltimore 2019

Philip O'Toole

www.philipotoole.com
@general_order24

Hello everyone, and welcome to *Observing Observability*. My name is Philip O'Toole, and I'm going to share my journey, as a developer and manager, through various companies that have built Observability systems. I've watched the tools -- and development teams -- evolve over time, and picked up a few lesson, along the way that I'd like to share. I hope you'll be able to apply these lessons to your own development team, tool choices, and Observability practices.

*

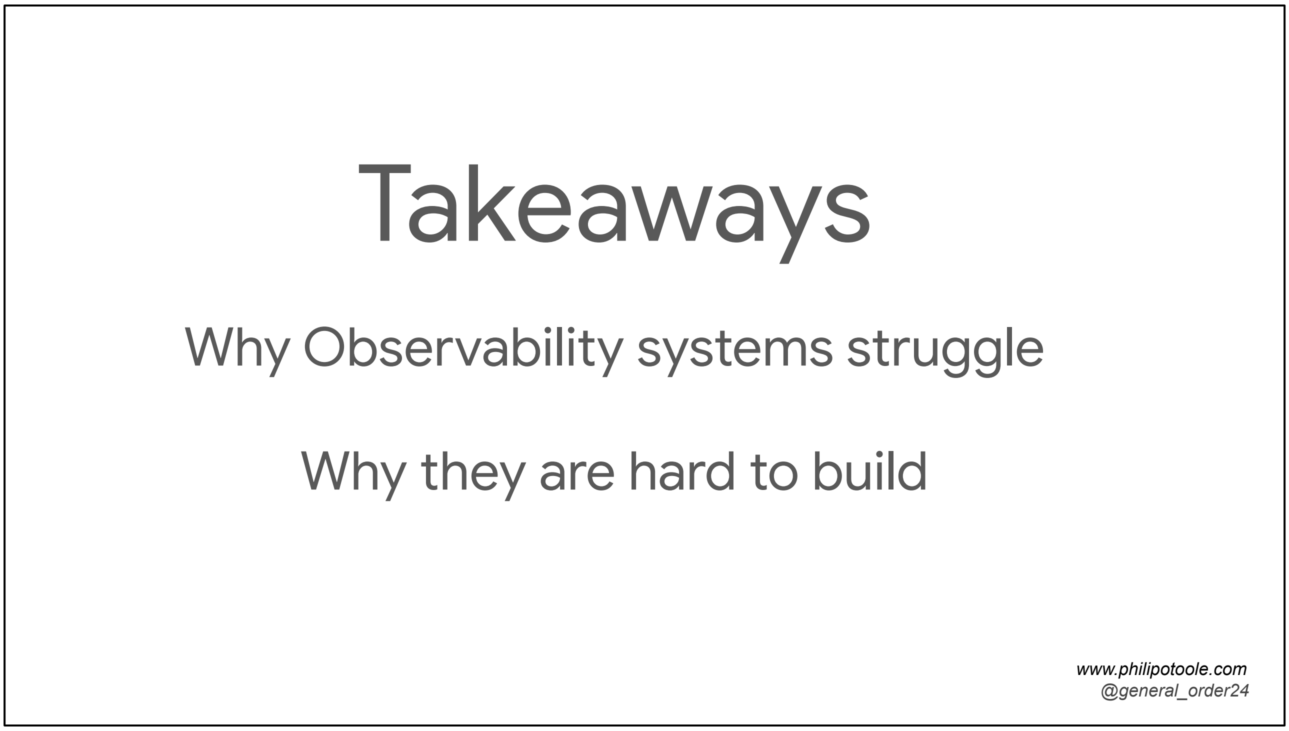
Google Cloud

**The following are my opinions, not my employer's!*

www.philipotoole.com
@general_order24

A little about me. I manage two software development teams within GCP -- specifically Stackdriver. Stackdriver is GCP's Observability platform, which allows GCP users to monitor and observe the systems and software they, in turn, run on GCP.

One other thing before we get going. What you'll hear today are my opinions. My opinions are based on my education, my training, and my experience. Different people have different experiences, so they'll have different opinions. I don't make any claims that what I say comes down from heaven, and these are my views, not my employer's. Now, with that out of the way….



Takeaways

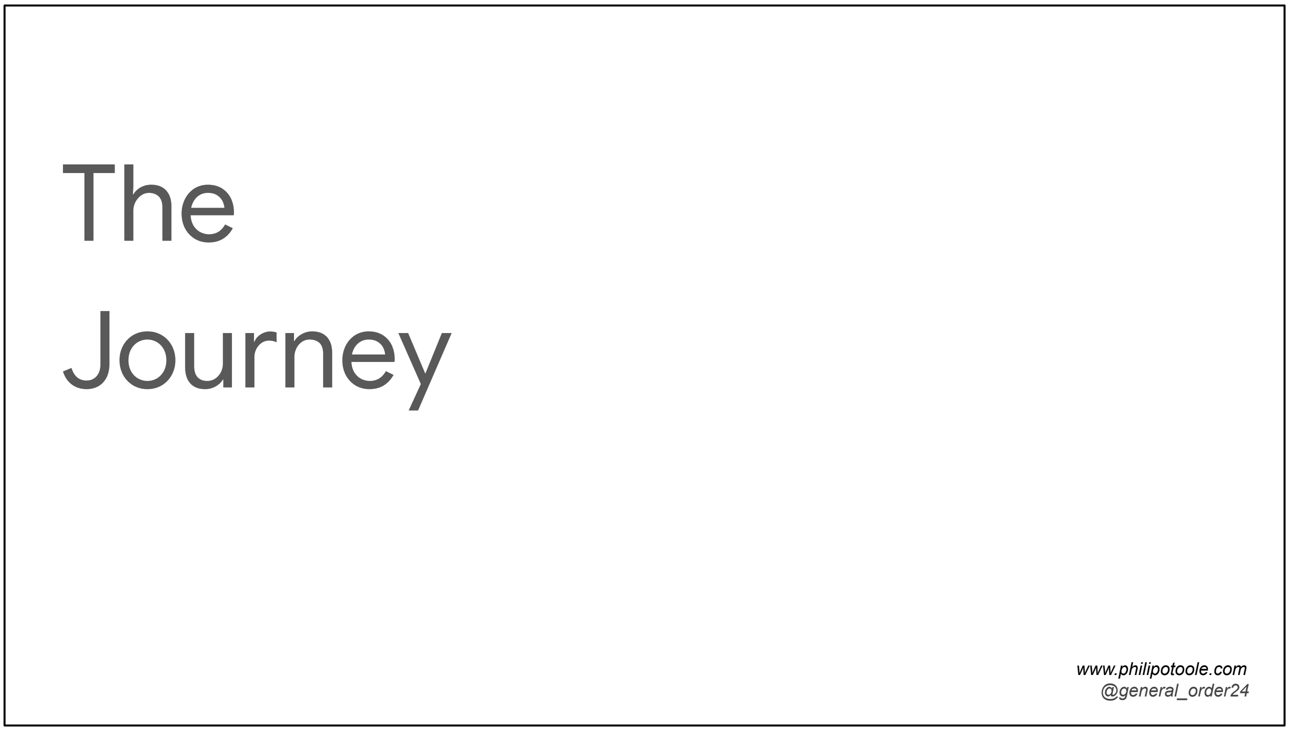
Why Observability systems struggle

Why they are hard to build

www.philipotoole.com
@general_order24

This is a talk of twos.

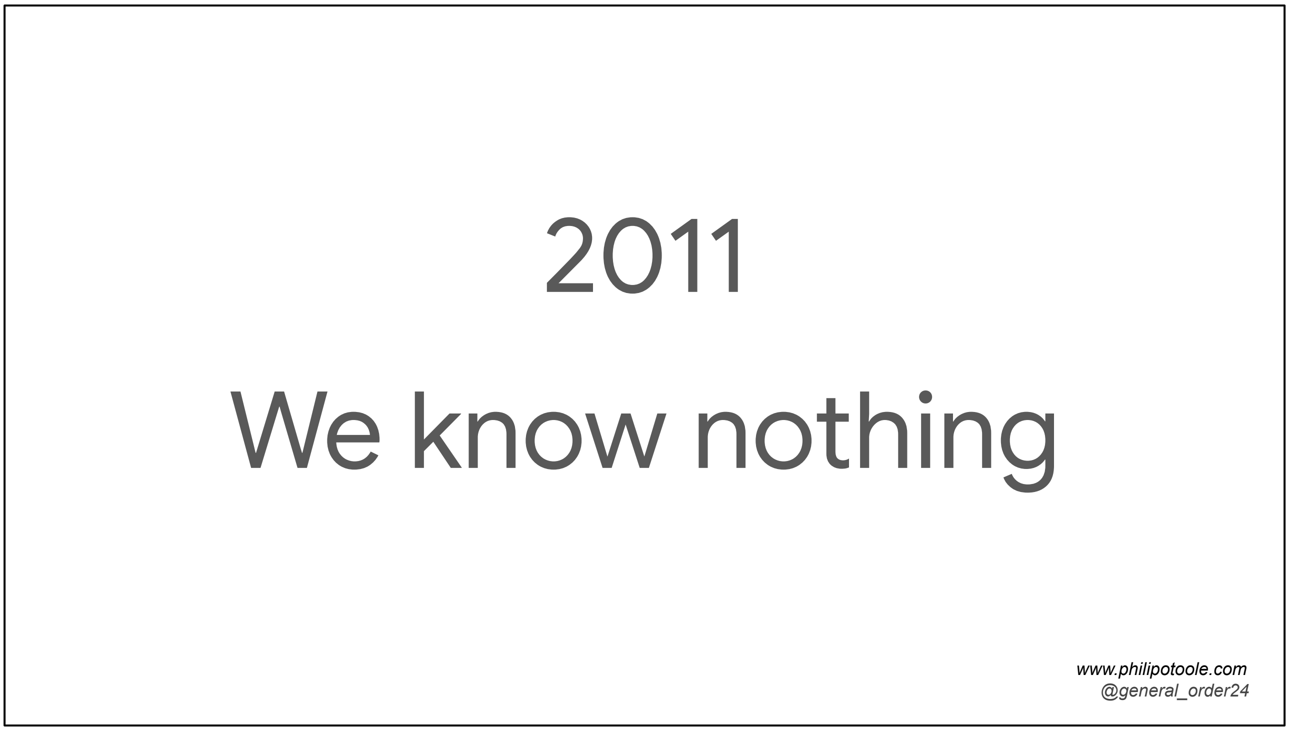
- There are two main takeaways
- And there are two parts to my talk -- the journey and the lessons.



The Journey

www.philipotoole.com
@general_order24

Come with me on a short journey from Zero Observability to Observability at planet-scale. It all starts back in 2011.



2011
We know nothing

www.philipotoole.com
@general_order24

It’s 2011, and I run a **development team for a fast-growing networking firm** in San Francisco, called Riverbed Technology. It’s an advanced company for the time, but it’s **got some characteristics** many of us wouldn’t even recognise today. It’s got:

- An in-house, super-protective IT department. And who can blame them? They work for a publicly-traded company.
- A Microsoft Exchange Server, closely guarded and nursed, sitting in a locked room.
- Development teams, who throw software over the wall, unprepared to build software for anything but networking switches.

But times they are a-changing, and my team needs to build a new licensing server **in the cloud** for our customers. So we build a Django web-based application and deploy it to AWS. It’s a single VM, with a single RDS instance. **The IT department -- not my team -- guards it closely, protects it, treating it like a special snowflake server.**

And the night before Thanksgiving 2011, while I’m having dinner with my in-laws to be, **the licensing server goes dark.**

Licensing Portal goes dark...

...and it's 24 hours until all customer appliances go unlicensed.

https://www.pinterest.com/pin/478859372862314902/

www.philipotoole.com
@general_order24

I get a call from IT. **They don't know what to do, and I don't know how to debug it**. I need the logs, but they are on the VM's disk. I can't get at them, and IT won't give me access. We're a public company, and we don't allow development engineers onto machines that contain customer data (**SOX**). And now I see it. **The logs shouldn't be on a VM, on the other side of a network I couldn't access. What are we doing to do?**

This was **never an issue with networking appliances**. With those you can walk up to them -- or the customer does -- plug in your serial cable, and read what's going on.

So what did we do? We did **voice-based log analysis --** in other words, I had the IT staff read me the logs lines over the phone, while my Thanksgiving dinner went cold. It was Zero Observability. And it sucked.

2012
Something is wrong

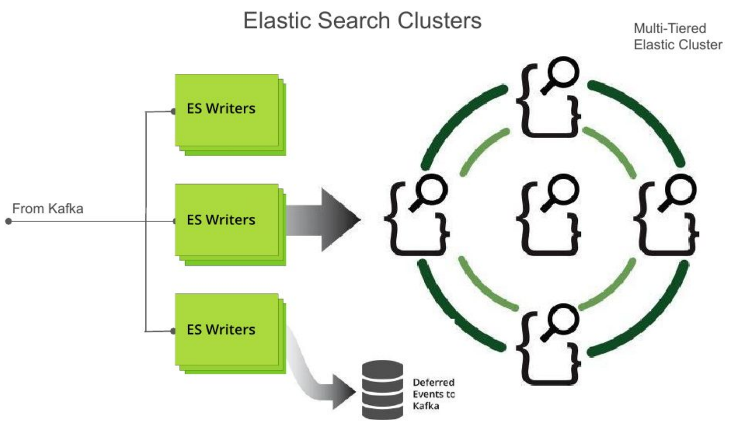
www.philipotoole.com
@general_order24

Now it’s a year later -- 2012 -- and it’s **clear to me that the operational -- and observability tools -- are already falling short when it comes to the next generation of computer systems**.

So it’s inspired to me to join **Loggy**, a cloud-based log analytics system. **The idea was clear! Get the log data off the machines, make it searchable, and those problems I experienced at Riverbed are gone**. **Separate the signals from the systems!** Access the logs from anywhere! It was genius. And there really is nothing like trying to solve **a problem you viscerally appreciate**.

Big pipelines and big indexers

- Lots of logs
- ...even more logs
- Nothing but logs
- ...and cardinality



www.philipotoole.com
@general_order24

At Loggly we built a groundbreaking system for real time log indexing and search. I think it's fair to say that you can see the **influence** in many of today's systems. As far as I know we were the **first team** to put Apache Kafka in front of Elasticsearch. And to build a product out of it.

But we made many mistakes. Because we were a Log Analytics company, **we basically never used time-series to monitor our production systems**. We used logs for everything, **and nothing but logs**. Of course, log data is key -- it's one of the two, maybe three, pillars of Observability systems. But **logs** -- in their basic form -- **don't show you trends**. They can tell you what happened, but don't tell you what the trend is -- **and certainly not if something was close to breaking**. Not without more processing anyway. **Time series does that for you**.

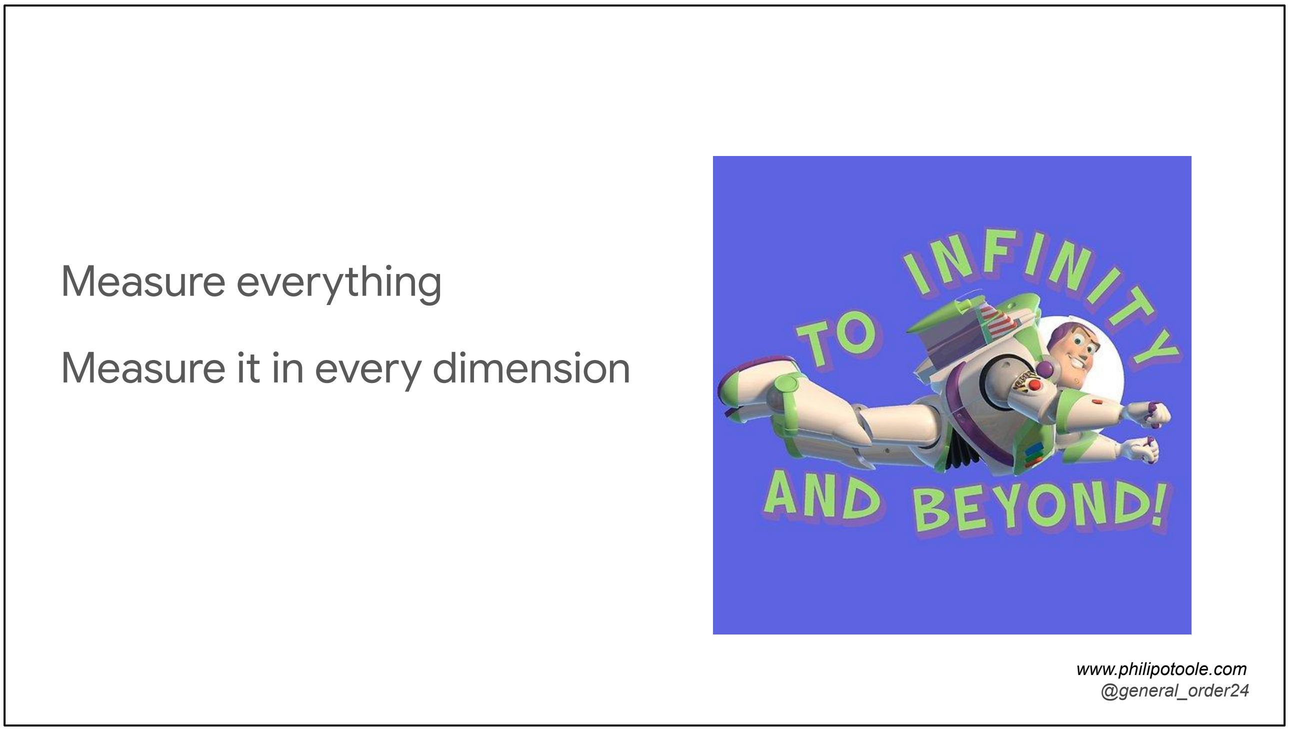
So now I had gone from **Zero Observability to Partial Observability**. It was better, but **outages still sucked**. And I mean really sucked. But another interesting thing was happening. **The tools we did use for monitoring -- Nagios, logs, basic alerting, were struggling to tell us what was happening with our large scale, clustered, distributed system**. I was developing a cutting-edge observability system, but the tools were struggling to monitor **this** generation of computer systems. **Noticeably the *dimensions* on which our data pipelines and clusters needed to be monitored was large and getting much larger** -- the nodes, the shards, the indexes, the customer accounts, the queries. **This was something which was more significant than I realised at the time.**

2014

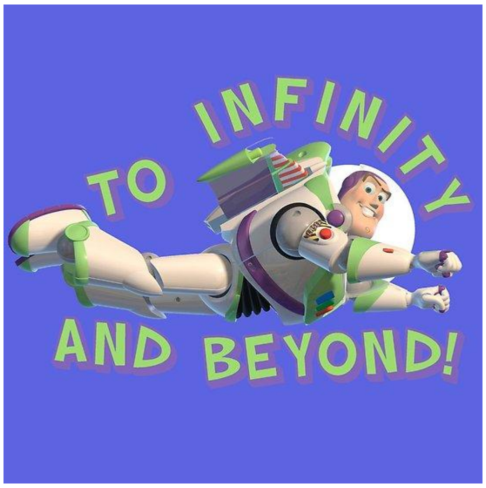
Something is still wrong

www.philipotoole.com
@general_order24

So it’s 2014. I had seen what had happened when you try to monitor large computer systems without the full picture. **While a partial picture was better than none, it was still pretty bad**.


Measure everything

Measure it in every dimension



www.philipotoole.com
@general_order24

I knew where I needed to go next

From Loggly I went to a short-lived company, named Jut. **Jut had a vision, and wanted to build an Observability system that supported both logs and metrics, via a single tech stack and unified query language (more on this later)**. After the logs-only experience at Loggly it was appealing, but the company didn't survive. However, what Jut really did was introduce me to **the power of time-series data**.

And since I was now a time-series believer, after Jut I joined the core development team at InfluxDB.

At InfluxDB I helped build an great time series database, and we built it in Go. **Suddenly one could measure everything!** And I noticed something interesting. InfluxDB was **designed specifically to measure systems with very large numbers of dimensions** -- what I now knew to call ***cardinality***.

Observability Challenges

- It’s hard for the tools to keep pace
- Observability systems are often **driving** the pace

www.philipotoole.com
@general_order24

And then I realised it.

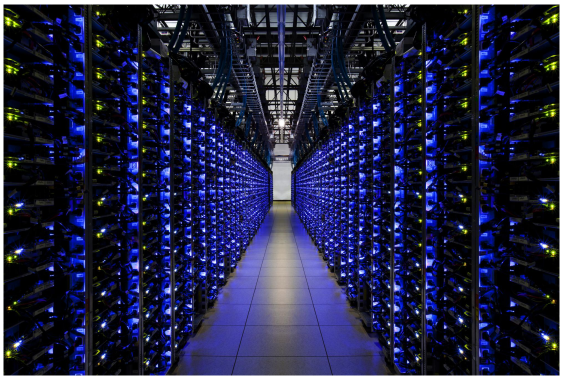
Every generation of monitoring system was being built such that it could monitor **the previous generation of computer system, but often not the current generation**. Logs solved Riverbed’s IT problems, but only Loggly solved Riverbed’s Cloud observability problems. Next InfluxDB would have solved Loggly’s monitoring problems, with its superb support for high cardinality time-series measurement.

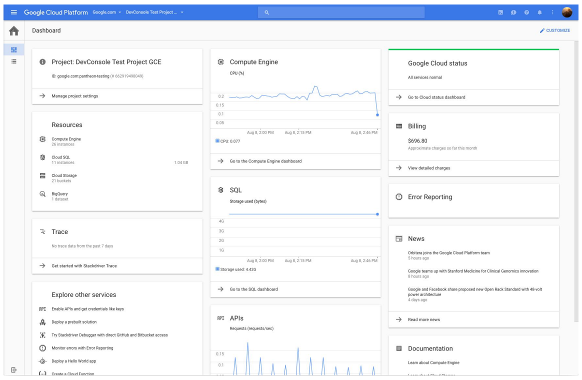
And as Observability systems themselves get more complex -- they become highly available, replicated, clustered, distributed -- they are one of the key forces driving creation of even more sophisticated computer infrastructure such as containers and Kubernetes.

And systems such as Kubernetes are, in turn, driving creation of even more sophisticated observability and monitoring systems!

Observability systems are often first solving the previous generation’s Observability systems monitoring challenges!

Google Cloud Platform





Google Cloud Platform Google.com DevConsole Test Project ...

Dashboard

CUSTOMIZE

Project: DevConsole Test Project GCE
ID: google.com:pantheon-testing (# 662919498049)
Manage project settings

Resources
Compute Engine 26 instances
Cloud SQL 11 instances 1.04 GB
Cloud Storage 21 buckets
BigQuery 1 dataset

Trace
No trace data from the past 7 days
Get started with Stackdriver Trace

Explore other services
Enable APIs and get credentials like keys
Deploy a prebuilt solution
Try Stackdriver Debugger with direct GitHub and Bitbucket access
Monitor errors with Error Reporting
Deploy a Hello World app
Create a Cloud Function

Compute Engine
CPU (%)
0.2
0.15
0.1
0.05
Aug 8, 2:00 PM Aug 8, 2:15 PM Aug 8, 2:46 PM
CPU: 0.077
Go to the Compute Engine dashboard

SQL
Storage used (bytes)
4G
3G
2G
1G
Aug 8, 2:00 PM Aug 8, 2:15 PM Aug 8, 2:46 PM
Storage used: 4.42G
Go to the SQL dashboard

APIs
Requests (requests/sec)
0.15
0.1

Google Cloud status
All services normal
Go to Cloud status dashboard

Billing
$696.80
Approximate charges so far this month
View detailed charges

Error Reporting

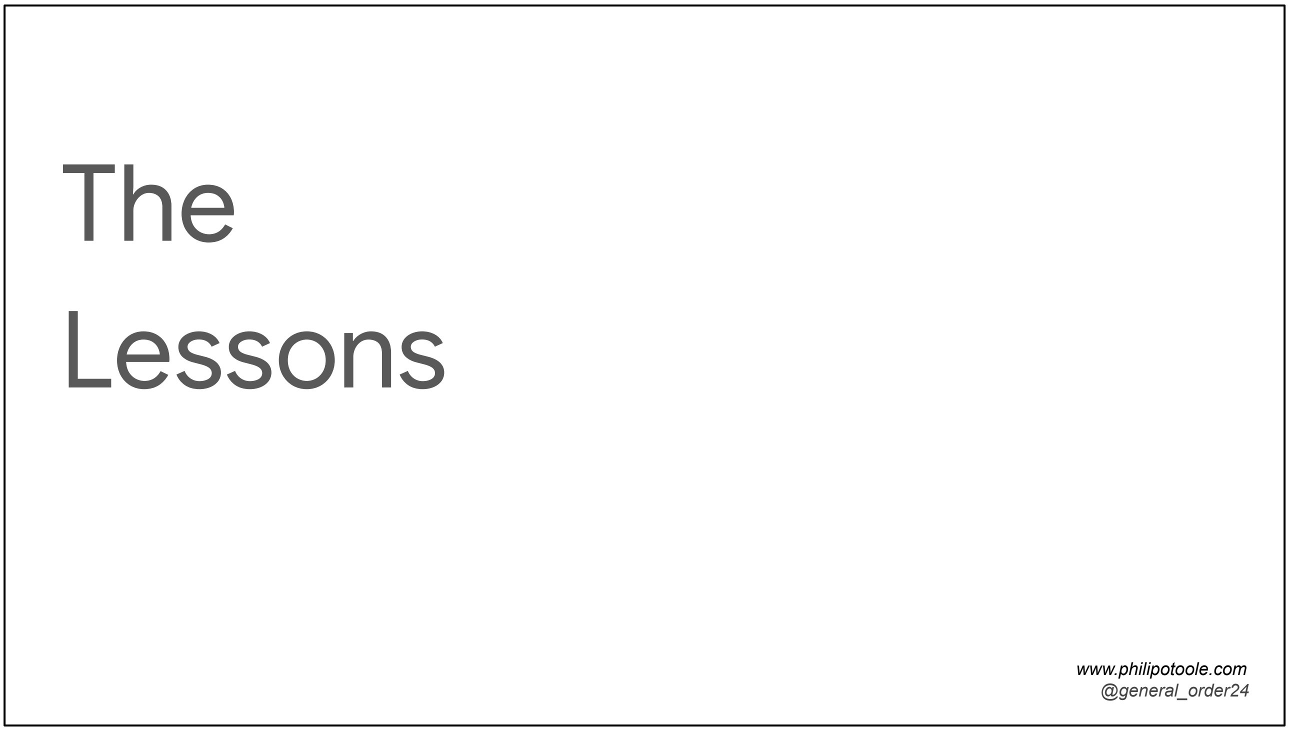
News
Orbitera joins the Google Cloud Platform team
5 hours ago
Google teams up with Stanford Medicine for Clinical Genomics innovation
8 hours ago
Google and Facebook share proposed new Open Rack Standard with 48-volt power architecture
4 days ago
Read more news

Documentation
Learn about Compute Engine

Stackdriver

www.philipotoole.com
@general_order24

And now I'm **at GCP**. At Google we make great use of both logs **and** time series, **SLO-targets are an integral part of design**, and alerting is core.
We're not perfect, but I feel like I've gone from **zero observability at Riverbed to planet-scale observability at Google**.



The Lessons

www.philipotoole.com
@general_order24

Anyway, **that was the journey.**

Next let's talk about the lessons I've learned along the way, and how they might apply to your **teams, tools, and technical choices**.

1

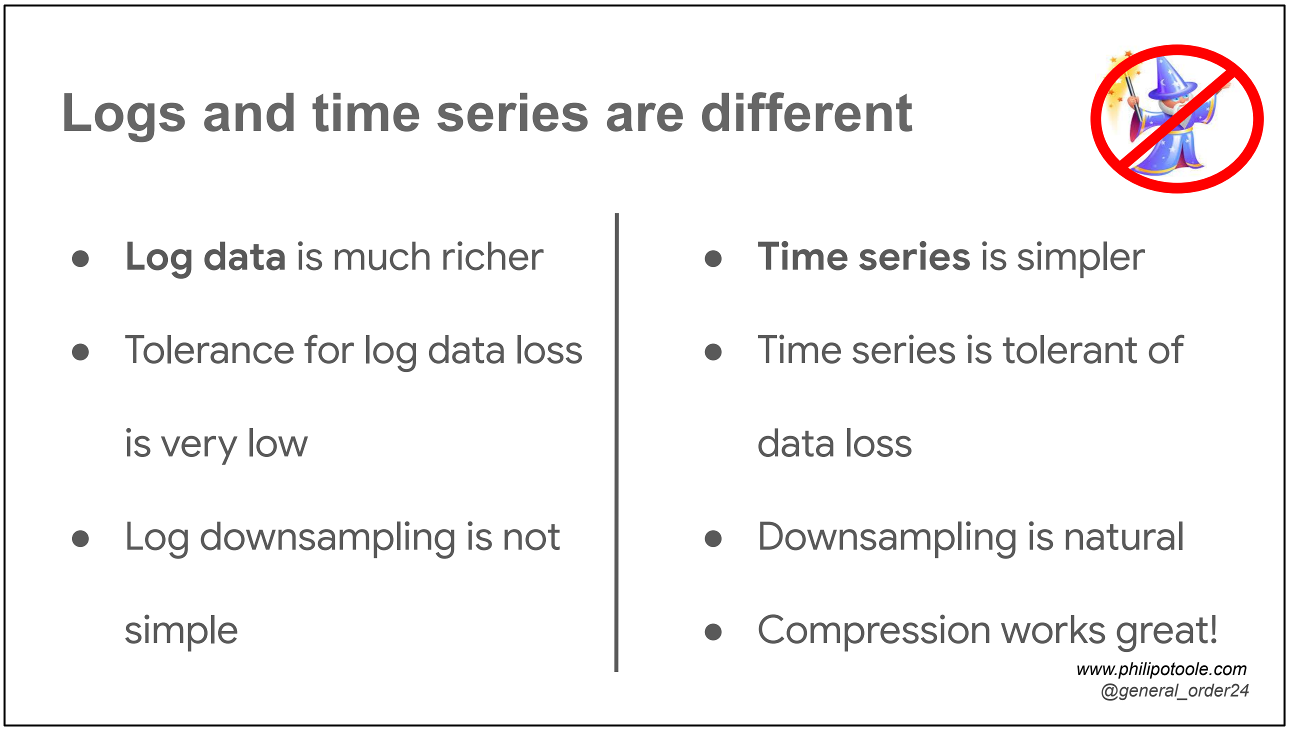
Lesson #1

Logs and time series are different

www.philipotoole.com
@general_order24

This one may be seem obvious, but I've been involved with a number of attempts to build a single system that handles both well. Sometimes even I've wondered if logs and time series are just different sides of the same event. **But in every single Observability company I've worked, which has tried to unify these data types in an efficient and high-performance manner, underneath the covers there has always been two distinct storage technologies.**

That is why there is a wizard up there, with a line through him. **There is no magic spell.**



Logs and time series are different

- **Log data** is much richer
- Tolerance for log data loss is very low
- Log downsampling is not simple

- **Time series** is simpler
- Time series is tolerant of data loss
- Downsampling is natural
- Compression works great!

www.philipotoole.com
@general_order24

What is it about log data, and time series, that is so different? **Log data** is usually **much richer**, often requiring **full-text search support**. This gets you into **inverted indexes**, rich **query languages**, and large data storage **overheads**. And the **tolerance for data loss is very low** - a single log message could be critical to understanding what has happened to a system. Compare this with time series data -- you can lose some data, and probably still do fine. **Time series** data **can be downsampled** too, but I've never encountered a compelling -- or at least simple -- case for samping logs. And with techniques such as delta encoding, **time series can also be compressed** very effectively, without any loss of information.

Time series are pointers, but logs tell you what happened.

Now this doesn't mean a SaaS can't be built that does a great job working with both types of data. But if someone tells you their **database** supports both cases really well, be very sceptical.

2

www.philipotoole.com
@general_order24

Lesson #2

Everyone thinks a new query language is the solution.

www.philipotoole.com
@general_order24

Out of the 5 companies mentioned in my story, 4 of them have seriously considered creating their own query language. 3 of them actually have, but only 1 has gained any traction -- and that's *flux* from InfluxData.

So you should know that the set of query languages is much, **much larger than the set of query languages that have had any success**. **SQL** -- and SQL-like -- still **dominate the market** place. And one of the key reasons for **InfluxDB's early success** was its SQL-like query language.

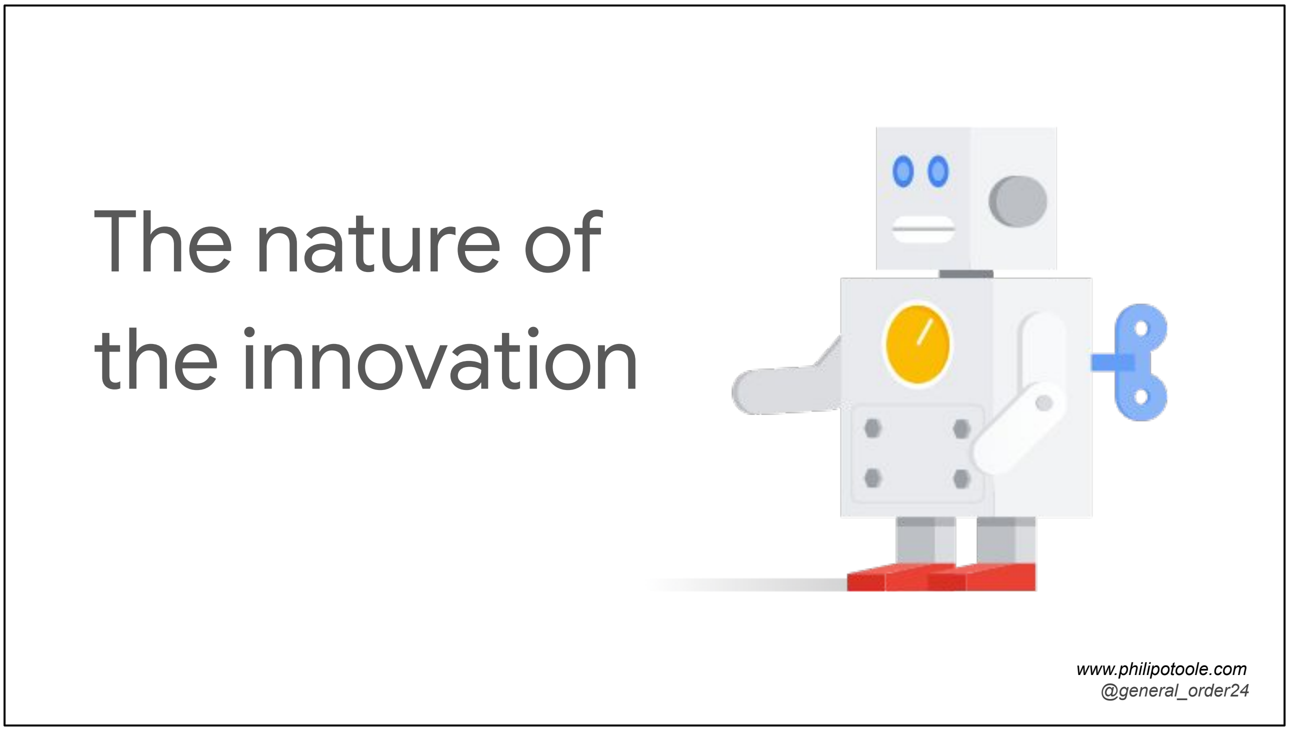
Now this is not to say that your query language isn't the solution to your monitoring challenges -- there are definitely custom query languages inside Google, with a particular one for time series data. But SQL-like languages dominate. And **developing your own query language is huge investment** -- I saw this at Jut -- and it can distract from **solving other, more pressing issues**. And oiften these query languages promise a unified interface **for interrogating both logs and time series**, but because those two data sources as so different, the **language design has to make sacrifices that limit its coherence or simplicity**.

So, like systems that promise superior performance for both logs and data, be wary of systems that advocate that their query language is their special sauce.

3

www.philipotoole.com
@general_order24

Lesson #3



The nature of the innovation

www.philipotoole.com
@general_order24

I think there is a **popular image out there, that small software startups are a hotbed of technical innovation** -- that they are producing amazing technical wonders everyday. This is definitely the perception of the general public. And obviously to certain extent it's true -- there are many innovative companies represented here today.

But startups are disproportionately innovating when it comes to business models and productizing existing technology. Academic-level technical innovation is **primarily happening at large firms**.

Throughout most of my small company experience, we **took large chunks of technology developed at larger firms and repurposed them for our uses**. At **Loggly** we used **Kafka**, which came out of LinkedIn. At **InfluxData**, core elements of **TSM storage engine is based upon the Gorilla storage engine** which came out of the Facebook. One **notable exception has been Elasticsearch**, which originated with a startup. But Elasticsearch has Apache Lucene at its core, which was developed elsewhere by Doug Cutting. But at all these **small companies** were were **laser-focused on gaining customers, and how we could tweak our business models and onboarding practises to win those customers** -- and keep them.

So I don't think the **image** of a startup triggering groundbreaking technical breakthroughs is **quite as common as one might think**.

What's the lesson? I think it's **when evaluating a new, small player's Observability**

offering, understand **what it’s built on**. If it’s brand new software from end-to-end, I would say it’s less likely to be successful -- and actually do what it says -- than a company that is **combining pre-existing technology in innovative new ways**, and combining it with a **new type of business model**.

4

www.philipotoole.com
@general_order24

Lesson #4

It’s a crowded market

www.philipotoole.com
@general_order24

Let’s look at the next observation.

Engineers want to solve the problems they understand best. The story I shared earlier shows this in action. And the tools and systems built in Observability are often the best in the business, especially from a computer science point-of-view.

But, boy, is the market crowded. Look at the state of Marketing software by contrast. Some of **biggest software companies in world are in the advertising and marketing space**, and yet much **marketing software is not in as strong a place technically**. Believe I know, **I once spent two years building it**. Yet there are many, Observability vendors, making technically amazing software, **all of which is doing the same thing**. Why?

Too many software developers build what they know, but maybe they shouldn’t.

The crowded market has **other effects too** -- some obvious, some more subtle. The obvious one is captured by one of the most famous xkcd cartoons ever -- **there are so many standards**.

The more subtle one is the **effect a crowded market has on the Observability development teams themselves**. The teams within such organisations are **fragmented, sharded, multiplied** -- as they have to deal with the ever-growing challenges of **OS-support**, **client library support, language support, data collection protocols, data models, query dialects, and so on**. The crowded,

fractured market seeps in, as though by osmosis, into the development teams. The requirement to work for everyone, everyone, is daunting.

But there is a lesson here too. If you’re going to build a successful Observability product, **differentiation is key**.

5

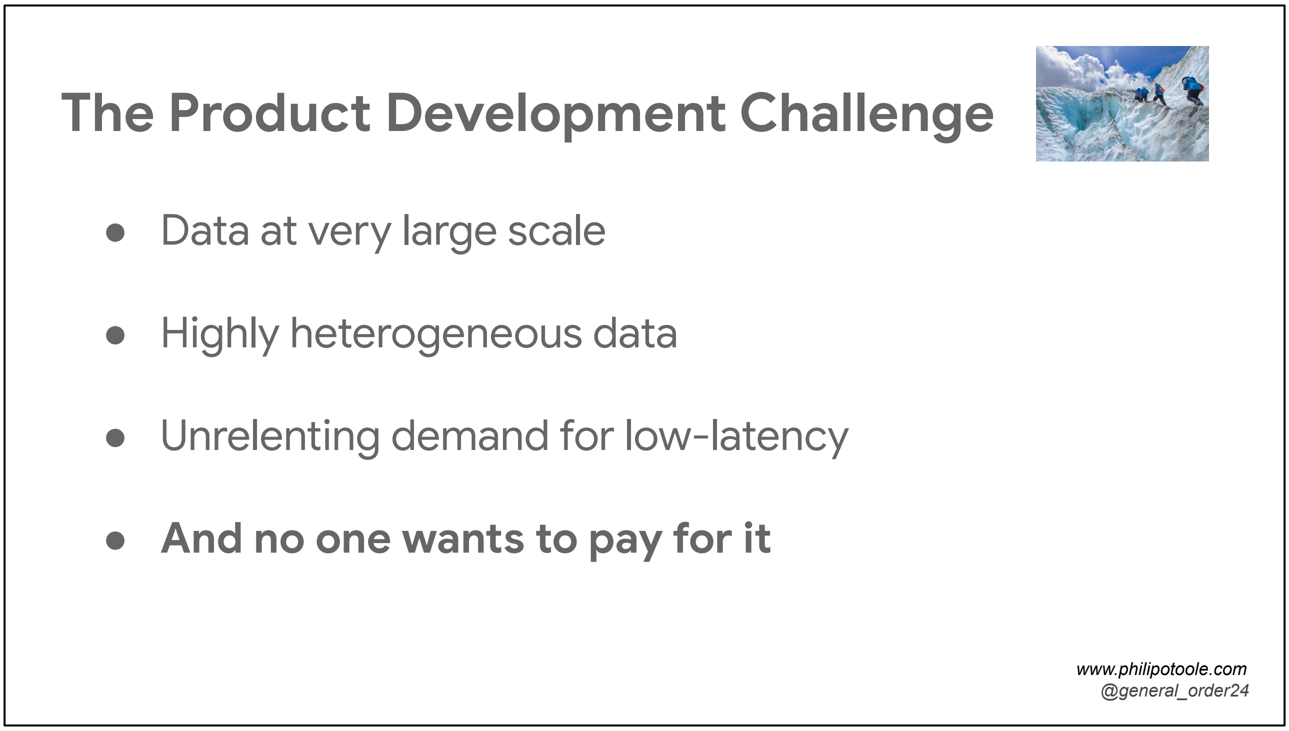
www.philipotoole.com
@general_order24

Lesson #5

The product development challenge

www.philipotoole.com
@general_order24

Building effective Observability systems presents a **daunting computer science challenge**. Let’s think about why.



The Product Development Challenge

- Data at very large scale
- Highly heterogeneous data
- Unrelenting demand for low-latency
- **And no one wants to pay for it**

www.philipotoole.com
@general_order24

- The data is **usually at very large scale**, since it's almost always machines that are emitting it. That's **challenge #1**.
- In many cases the **data is highly heterogeneous** -- both structured and unstructured data must be supported well. Sometimes binary data too. **Challenge #2**
- There is an **unrelenting demand for low-latency systems** -- **both at ingest time and at query time**. We know how to build systems today that will offer either high-performance ingest, followed by querying some longer time later. Or the systems will offer fast queries, but will require preprocessing the received data first (or perhaps handle relatively little data) -- introducing ingest latency. But Observability systems **must try to address both requirements simultaneously**. This is **challenge #3 -- performance is demanded at both ends of the system**.
- **And finally, challenge #4: no one wants to pay for it.** Doesn't everyone complain about Splunk in this dimension?

There are other non-technical challenges too. As engineers in this industry we're building **tools that are often -- if not mostly -- used by people in a bad emotional state at the time**. I often feel this **leaks into our development and Product Management teams**, much like the pernicious effects of the crowded market.

What's the lesson?

I think we, as an industry, still **need to do a better job communicating the value of**

our work, and the systems we build -- that's my challenge to all of you.

The contrast with Computer Security has become clear Sometime during the last 5 years, **Security made the leap from being perceived as a cost center to now a highly valuable partner**. It's probably the high-profile data breaches that have occurred in that time, and geopolitical implications of such failures.

The question is this: does that mean we're going to need a high-profile systems failure, with geopolitical implications, before people are happy to pay for Observability systems?

Takeaways

www.philipotoole.com
@general_order24

And that’s it! Let me remind you of the takeaways.

Why Observability systems struggle

- They can't keep up with the generation infrastructure changes

Why they are hard to build

- The main data pillars are so different
- Query languages only get you so far
- Crowded market
- It's CS-hard and no one wants to pay

www.philipotoole.com
@general_order24

Thank you!

Links

http://www.philipotoole.com/2019-monitorama

Acknowledgements

Nathen Harvey & Marie Cosgrove-Davies

www.philipotoole.com
@general_order24

But let me finish with this. Building these systems is **amazing work, technically challenge, and central to what has become the industry of the last 50 years**. So let's keep doing it, and doing it well.