SOUNDCLOUD

Alerting for Distributed Systems
A Tale of Symptoms and Causes, Signals and Noise

SRECon Europe
Dublin, 2016-07-12

Björn “Beorn” Rabenstein, Production Engineer, SoundCloud Ltd.

O(100) engineers
~5% is ProdEng

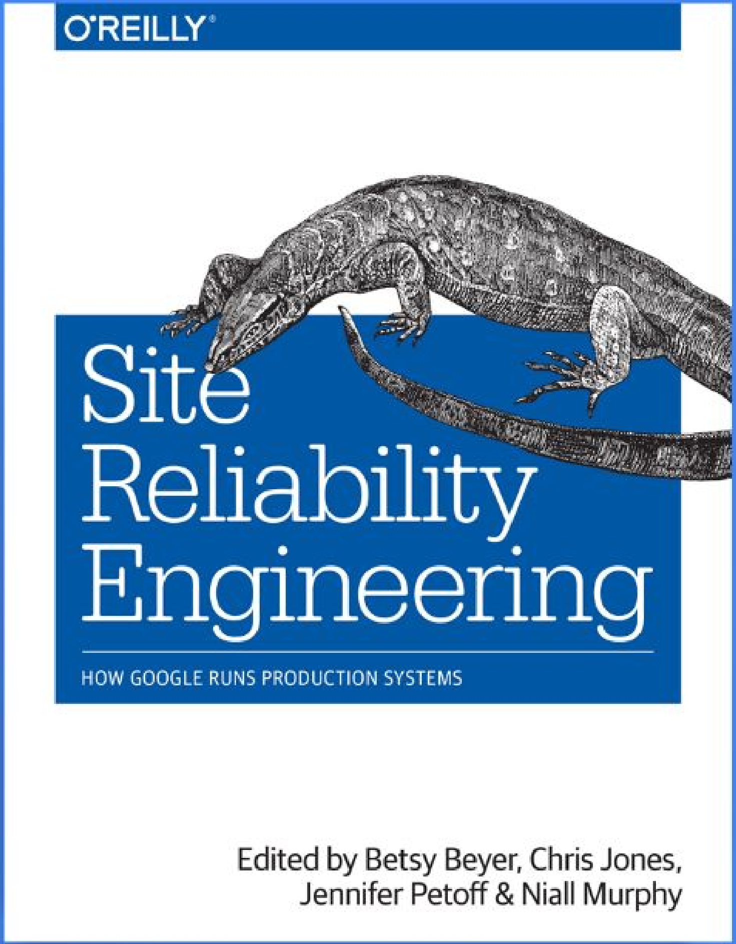
SOUNDCLOUD

“You build it, you run it.”
“True DevOps”
“NoOps”

O(10k) engineers
~5% is SRE

Google

SRE “by the book”

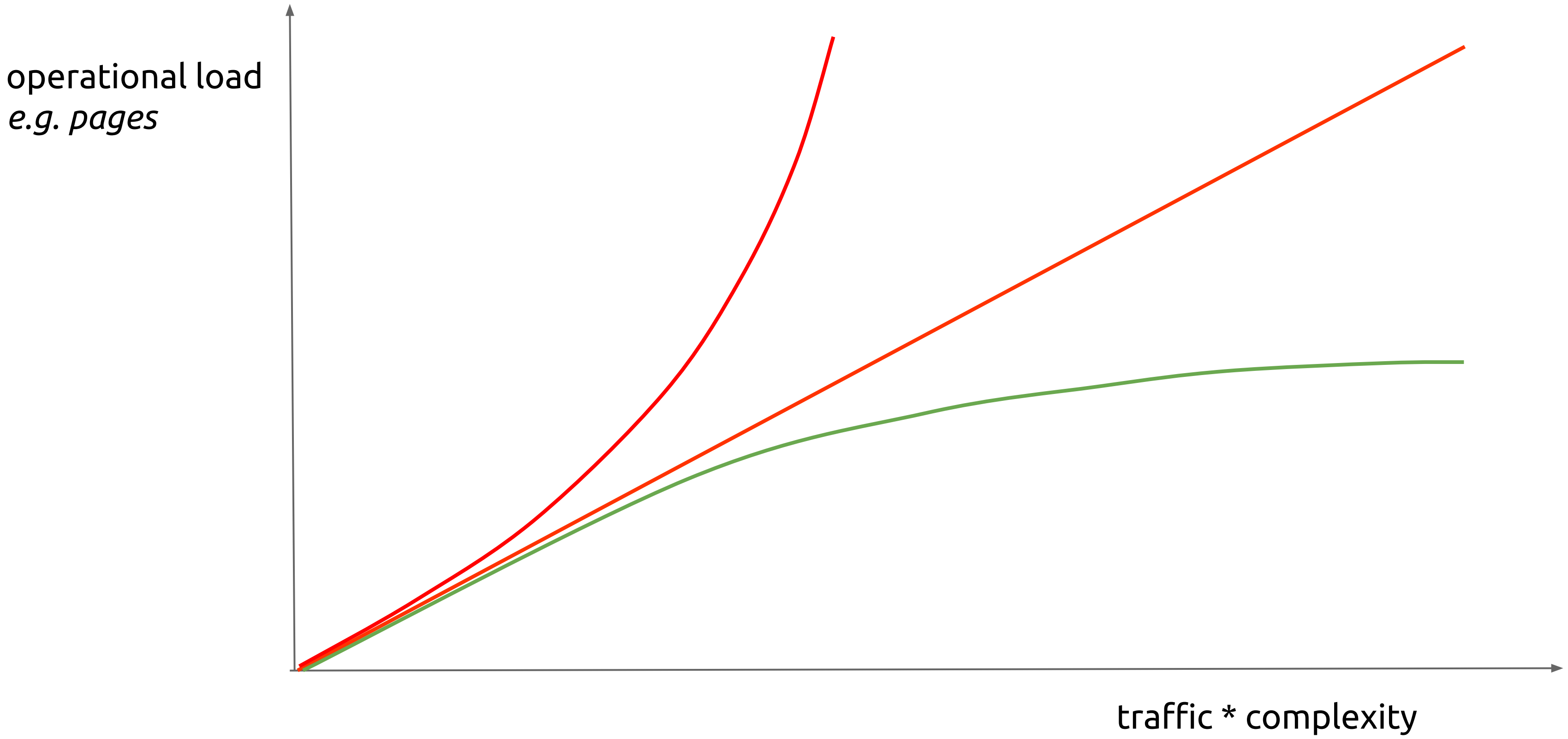


O'REILLY®

Site Reliability Engineering

HOW GOOGLE RUNS PRODUCTION SYSTEMS

Edited by Betsy Beyer, Chris Jones,
Jennifer Petoff & Niall Murphy

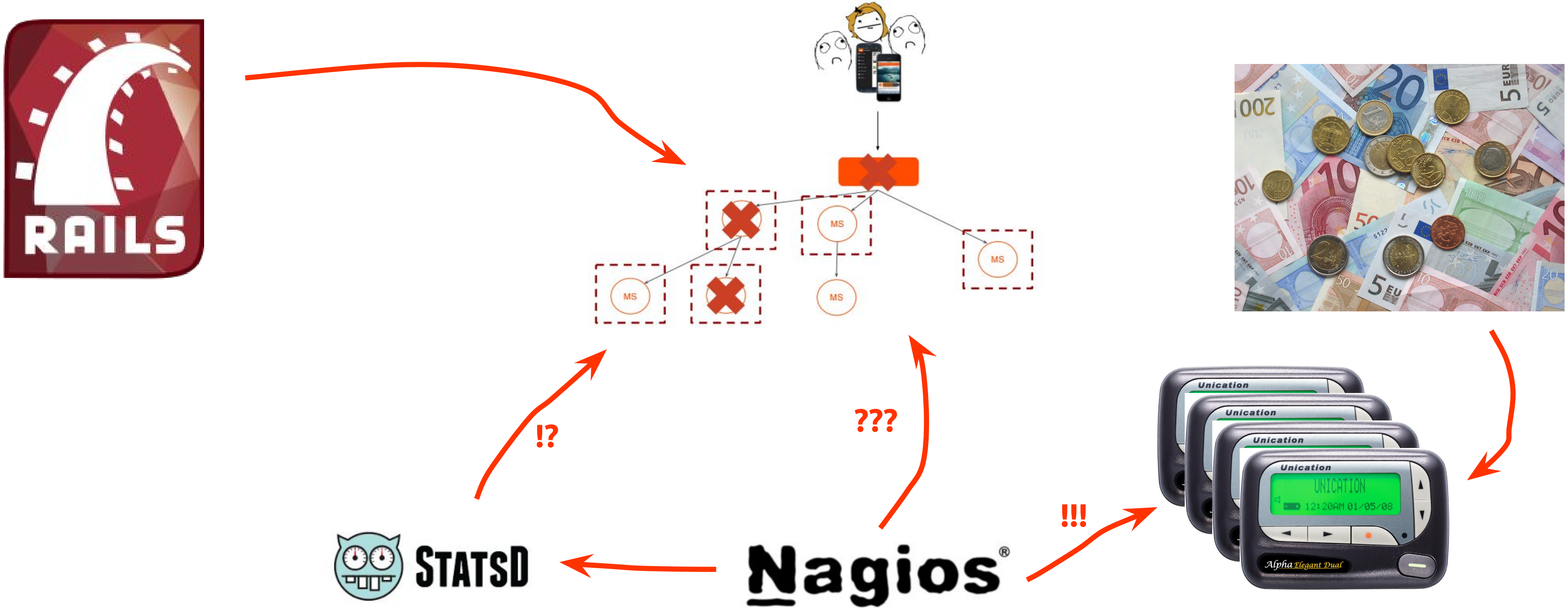

operational load
e.g. pages
traffic * complexity

Our SRE organization has an advertised goal of keeping *operational work* (i.e., toil) *below 50%* of each SRE's time. At least 50% of each SRE's time should be spent on engineering project work that will either reduce future toil or add service features. [...] We share this 50% goal because *toil tends to expand* if left unchecked and can quickly fill *100% of everyone's time.*

Chapter 5: Eliminating Toil

SoundCloud's trajectory 3 years ago

All started with a healthy growth in traffic and features…



Radio pager by Vitachao (Template:Unication) CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)

The three kinds of “alerts”

SRE book calls them *monitoring output*.
Alerts is Prometheus terminology.

Expected response	**SRE book**	**SoundCloud lingo**	**Delivered to**
Act immediately	Alerts	Pages	Pager
Act eventually	Tickets	Tickets / “email alerts”	Issue tracker / email :-(
None (for diagnostics only)	Logs	Informational alerts	Nowhere / dashboards

Sense of urgency and action

Every time the pager goes off, I should be able to react with a s*ense of urgency*. I can only react with a sense of urgency *a few times a day* before I become *fatigued*.

Every page should be *actionable*.

Chapter 6: Monitoring Distributed Systems

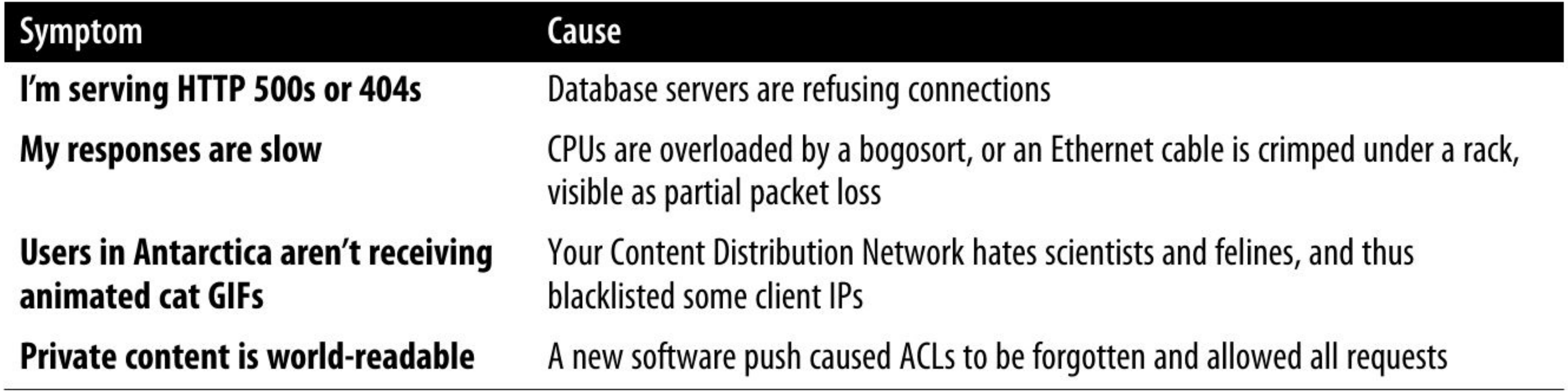
True story: One day, SoundCloud was down, and a single page fired…

“The outage was so bad, more pages should have fired.” *(From a SC Postmortem)*

Symptoms vs. causes

How to make pages more meaningful?

> "What" versus "why" is one of the most important distinctions in writing good monitoring with *maximum signal and minimum noise.*
>
> *Chapter 6: Monitoring Distributed Systems*



Symptom	Cause
I'm serving HTTP 500s or 404s	Database servers are refusing connections
My responses are slow	CPUs are overloaded by a bogosort, or an Ethernet cable is crimped under a rack, visible as partial packet loss
Users in Antarctica aren't receiving animated cat GIFs	Your Content Distribution Network hates scientists and felines, and thus blacklisted some client IPs
Private content is world-readable	A new software push caused ACLs to be forgotten and allowed all requests

Source: Betsy Beyer et al. "Site Reliability Engineering – How Google Runs Production Systems"

Causes and symptoms are loosely bound in distributed systems.

At the scale our systems operate, being alerted for *single-machine failures* is unacceptable because such data is *too noisy to be actionable*.

Chapter 10: Practical Alerting from Time-Series Data

What was thought to be good signals for problems might just be noise today (or worse, you can't say if it is noise or not):

- A machine is down. *Happens all the time.*
- Load average is high. *Really?*
- My network uplink / CPUs / disk / RAM ... are fully utilized. *Good or bad?*

Black-box vs. white-box

Black-box is just perfect for symptom-based alerting, isn't it?

We combine *heavy use of white-box* monitoring with *modest but critical uses of black-box* monitoring. The simplest way to think about black-box monitoring versus white-box monitoring is that black-box monitoring is *symptom-oriented and represents active—not predicted—problems.* […]

For paging, black-box monitoring has the key benefit of forcing discipline to only nag a human when a problem is both *already ongoing and contributing to real symptoms*. On the other hand, for *not-yet-occurring but imminent* problems, black-box monitoring is *fairly useless*.

Chapter 6: Monitoring Distributed Systems

Pros & cons

Black-box:

- End-to-end test “as the user sees it”.
- Probes may be different from current user traffic.
- Tail latency and rare failures only visible over a long time.

White-box:

- Reported latency serving the frontend might be a lie, but reported latency of requests to the backend is “live-traffic probing”.
- Must resist temptation to alert on countless internal details.
- Indispensable to detect imminent problems and to investigate causes.

Imminent problems

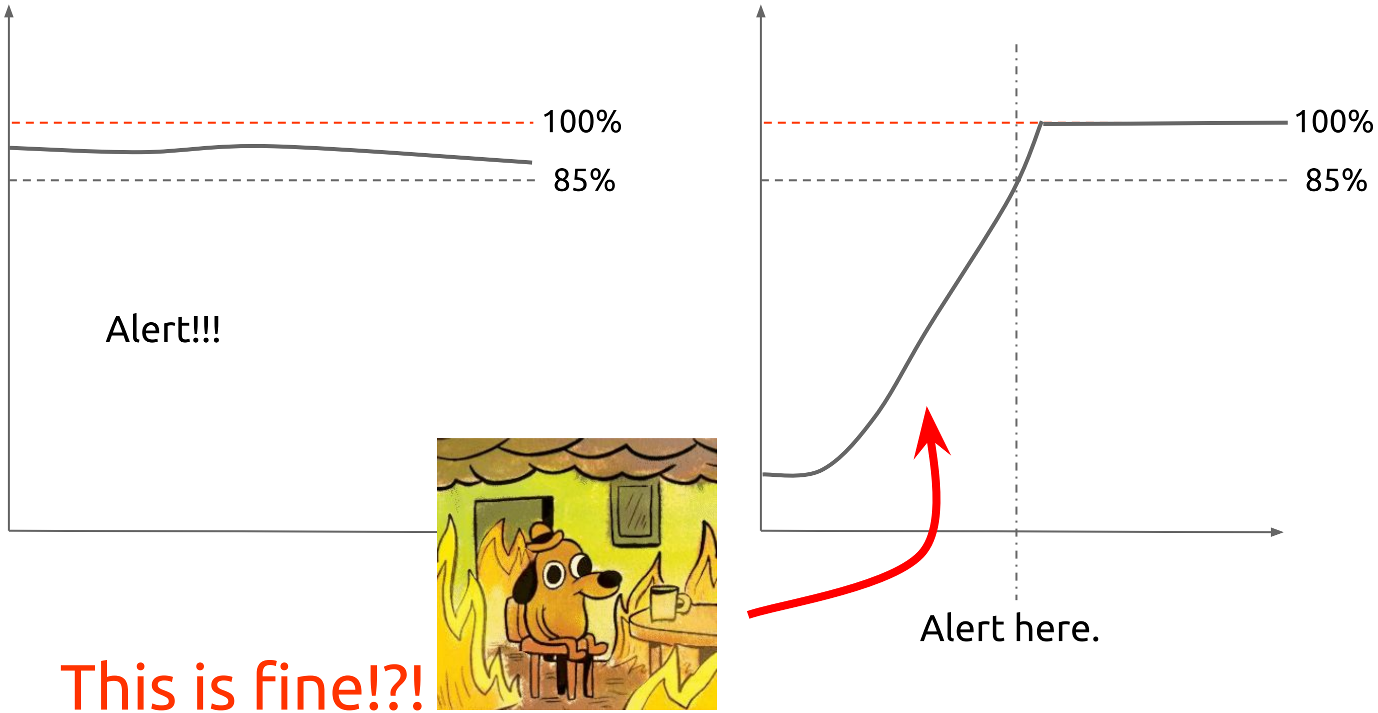
White-box and time-series based monitoring FTW.

- Loss of redundancy (going from N+1 to N+0).
- More complex reasoning based on insights into a system.
- “Nearly full” scenarios.

[...] the idea of treating time-series data as a data source for generating alerts is now accessible to everyone through those open source tools like Prometheus, Riemann, Heka, and Bosun [...]

Chapter 10: Practical Alerting from Time-Series Data

Static disk-full alert *(e.g. Nagios)*

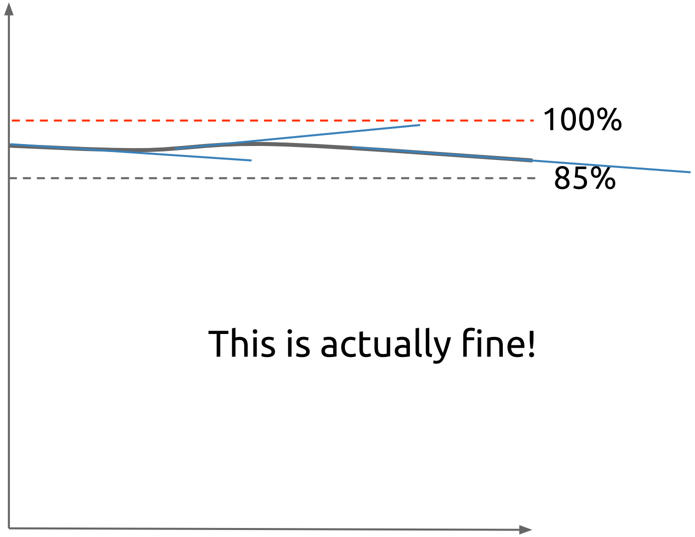


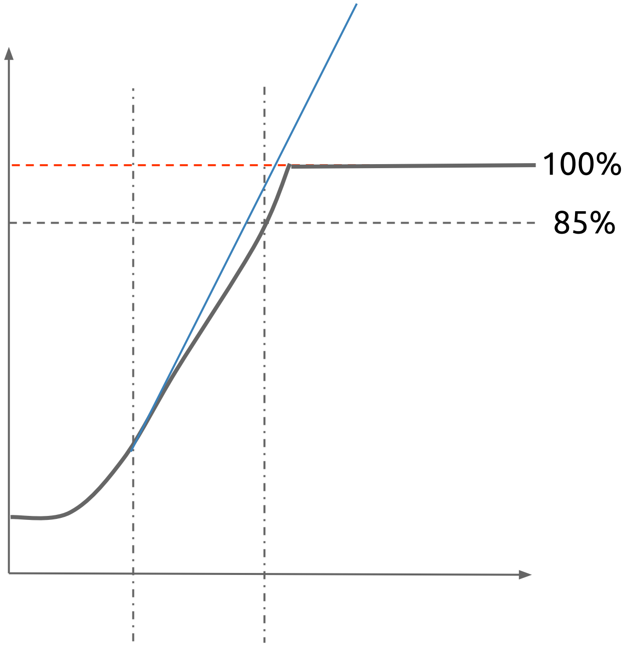
Alert!!!

This is fine!?!

Alert here.

Time-series based disk-full alert *(e.g. Prometheus)*





Alert here… not there

Causes are important, too.

True story: Google’s stats processing pipeline.

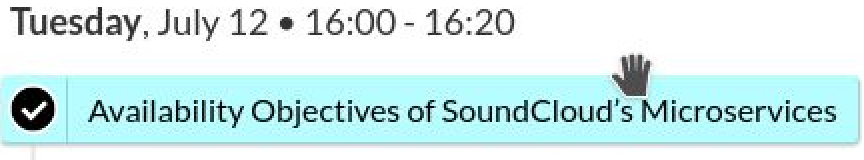
Informational alerts and sometimes tickets are great for causes.

Symptom or cause?

Note that in a multilayered system, one person’s symptom is another person’s cause.

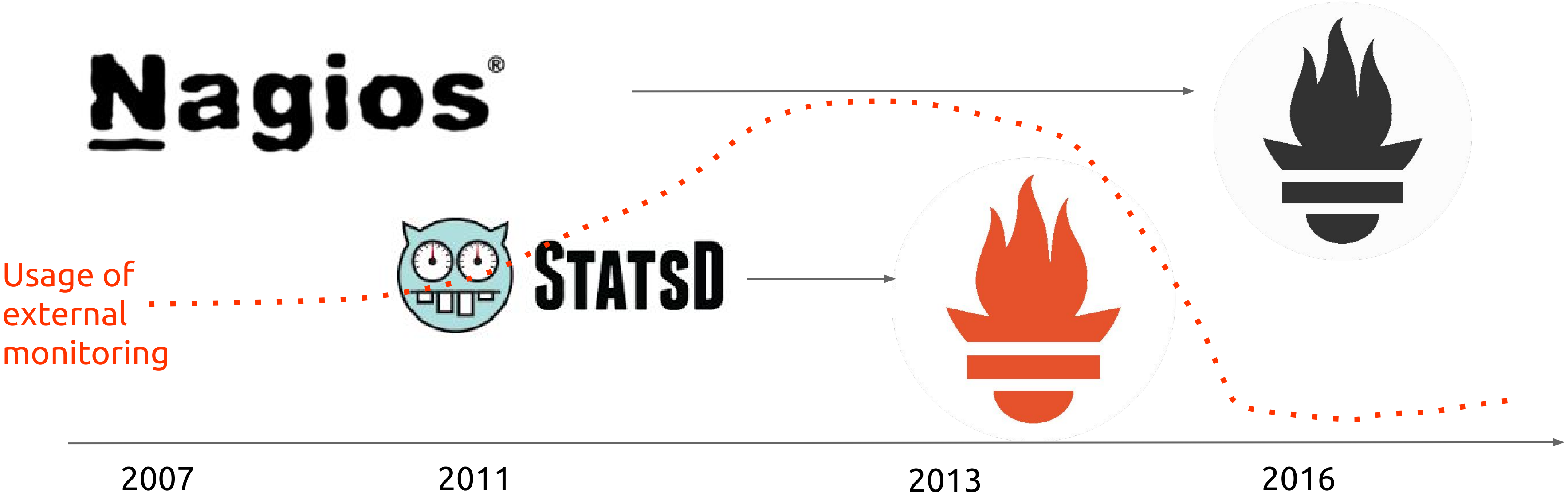
Chapter 6: Monitoring Distributed Systems

To achieve the decoupling desired in a microservice architecture, teams become users of each other (in addition to the “real” user in the big picture).



Tuesday, July 12 • 16:00 - 16:20

✓ Availability Objectives of SoundCloud's Microservices



We need monitoring systems that allow us to alert for high-level service objectives, but retain the granularity to inspect individual components as needed.

Chapter 10: Practical Alerting from Time-Series Data

But what about anomaly detection?

Neither symptom nor cause.

In general, Google has trended toward *simpler and faster* monitoring systems, with better tools for post hoc analysis. We avoid "*magic*" systems that try to *learn thresholds or automatically detect causality*. [...]

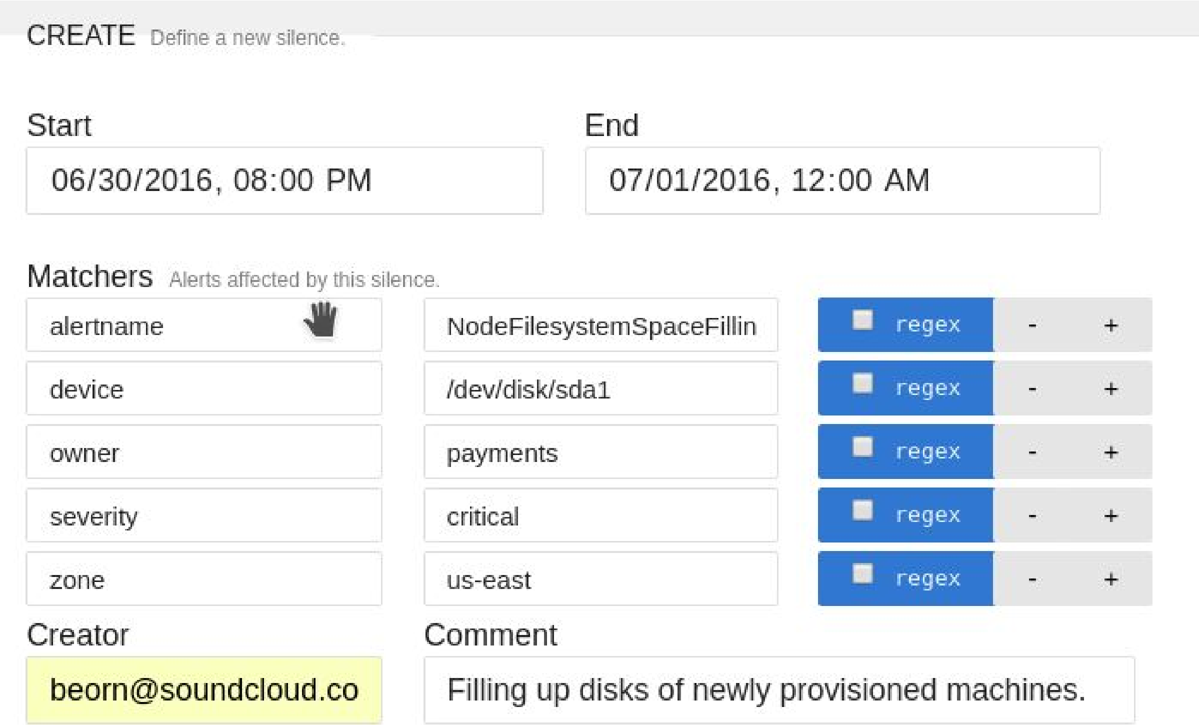
Similarly, to keep noise low and signal high, the elements of your monitoring system that direct to a pager need to be *very simple and robust*. Rules that generate alerts for humans should be simple to *understand* and represent a *clear failure*.

Chapter 6: Monitoring Distributed Systems

Anomaly detection for pages should be simple and robust.

More complex systems can be great under circumstances, but not for pages.

Silencing for humans



CREATE Define a new silence.

Start

06/30/2016, 08:00 PM

End

07/01/2016, 12:00 AM

Matchers Alerts affected by this silence.

alertname	NodeFilesystemSpaceFillin	□ regex	-	+
device	/dev/disk/sda1	□ regex	-	+
owner	payments	□ regex	-	+
severity	critical	□ regex	-	+
zone	us-east	□ regex	-	+

Creator

beorn@soundcloud.co

Comment

Filling up disks of newly provisioned machines.

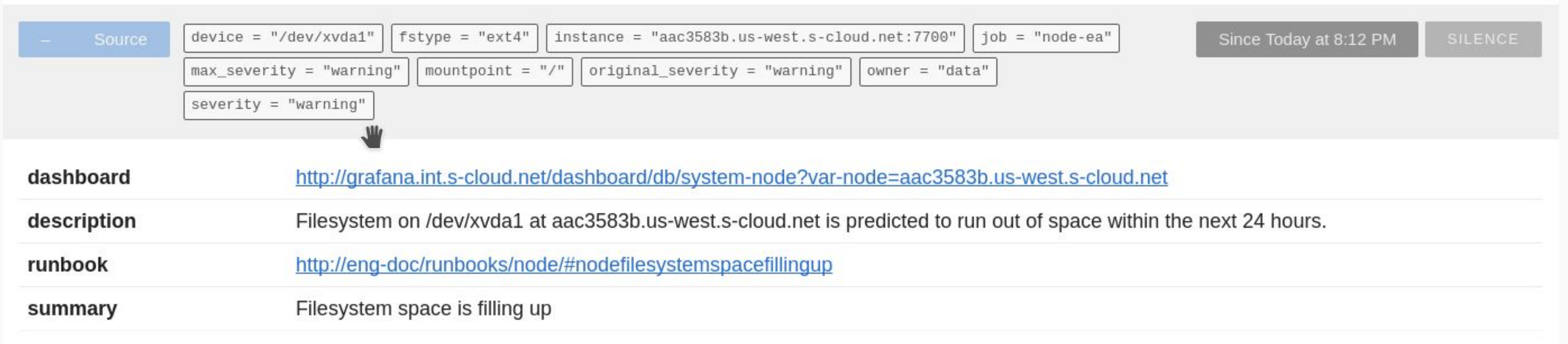
Runbooks and robotic responses

Google SRE relies on on-call playbooks, in addition to exercises such as the “Wheel of Misfortune,” to prepare engineers to react to on-call events.

Chapter 1: Introduction

Every page response should require intelligence. If a page merely merits a robotic response, it shouldn’t be a page.

Chapter 6: Monitoring Distributed Systems



device = "/dev/xvda1" fstype = "ext4" instance = "aac3583b.us-west.s-cloud.net:7700" job = "node-ea"
max_severity = "warning" mountpoint = "/" original_severity = "warning" owner = "data"
severity = "warning"

dashboard	http://grafana.int.s-cloud.net/dashboard/db/system-node?var-node=aac3583b.us-west.s-cloud.net
description	Filesystem on /dev/xvda1 at aac3583b.us-west.s-cloud.net is predicted to run out of space within the next 24 hours.
runbook	http://eng-doc/runbooks/node/#nodefilesystemspacefillingup
summary	Filesystem space is filling up

Perfectly self-healing systems?

Caveats of automation

Being on-call for a quiet system is blissful, but what happens if the system is too quiet or when SREs are not on-call often enough? *An operational underload is undesirable for an SRE team.*

Chapter 11: Being On-Call

Do gamedays, DiRT, “Wheel of Misfortune”, whatever you call it…

The End

May the queries flow,
and the pager stay silent.

Chapter 10: Practical Alerting from Time-Series Data

Bonus Slides

Muting for machines.

Google SRE has experienced only limited success with complex dependency hierarchies. [...] Dependency-reliant rules usually pertain to very stable parts of our system, such as our system for draining user traffic away from a datacenter. For example, “If a datacenter is drained, then don’t alert me on its latency” is one common datacenter alerting rule. Few teams at Google maintain complex dependency hierarchies because our infrastructure has a steady rate of continuous refactoring.

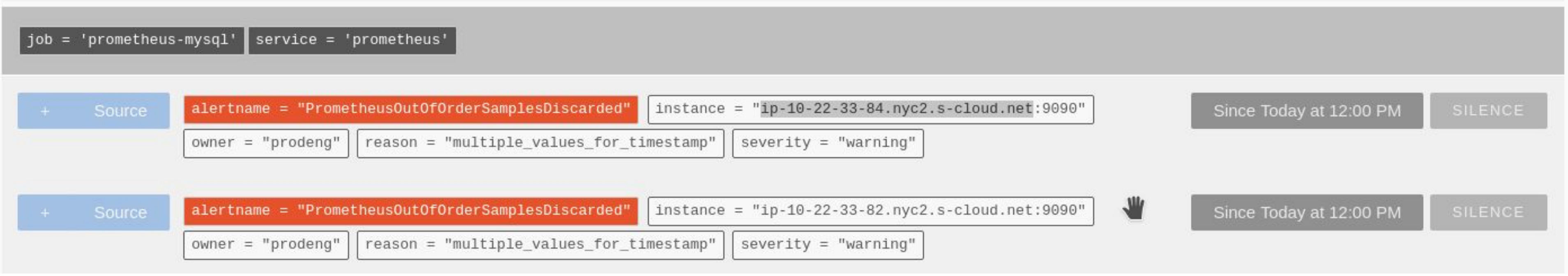
Chapter 6: Monitoring Distributed Systems

Alert grouping

Thousands of nodes suddenly cried out in terror…

Noisy alerts that systematically generate more than one alert per incident should be tweaked to approach a 1:1 alert/incident ratio. Doing so allows the on-call engineer to focus on the incident instead of triaging duplicate alerts.

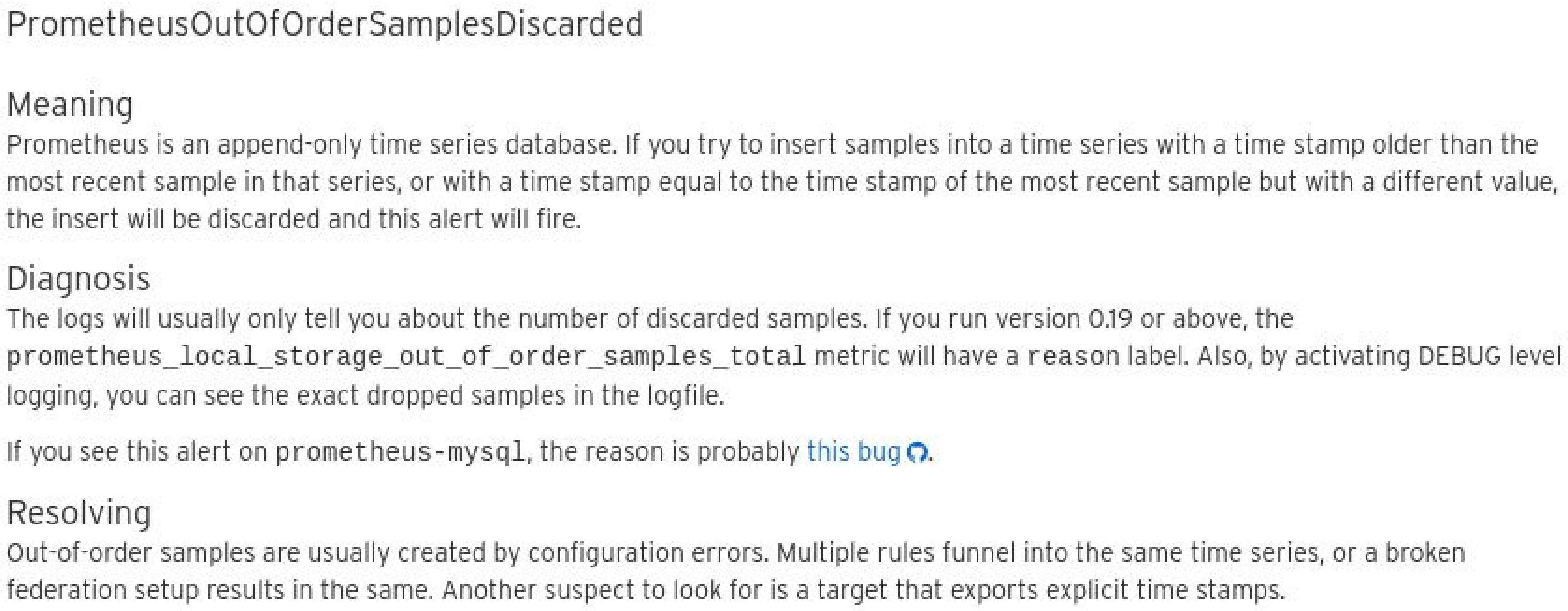
Chapter 11: Being On-Call



```
job = 'prometheus-mysql'   service = 'prometheus'

alertname = "PrometheusOutOfOrderSamplesDiscarded"   instance = "ip-10-22-33-84.nyc2.s-cloud.net:9090"
owner = "prodeng"   reason = "multiple_values_for_timestamp"   severity = "warning"

alertname = "PrometheusOutOfOrderSamplesDiscarded"   instance = "ip-10-22-33-82.nyc2.s-cloud.net:9090"
owner = "prodeng"   reason = "multiple_values_for_timestamp"   severity = "warning"
```



PrometheusOutOfOrderSamplesDiscarded

Meaning

Prometheus is an append-only time series database. If you try to insert samples into a time series with a time stamp older than the most recent sample in that series, or with a time stamp equal to the time stamp of the most recent sample but with a different value, the insert will be discarded and this alert will fire.

Diagnosis

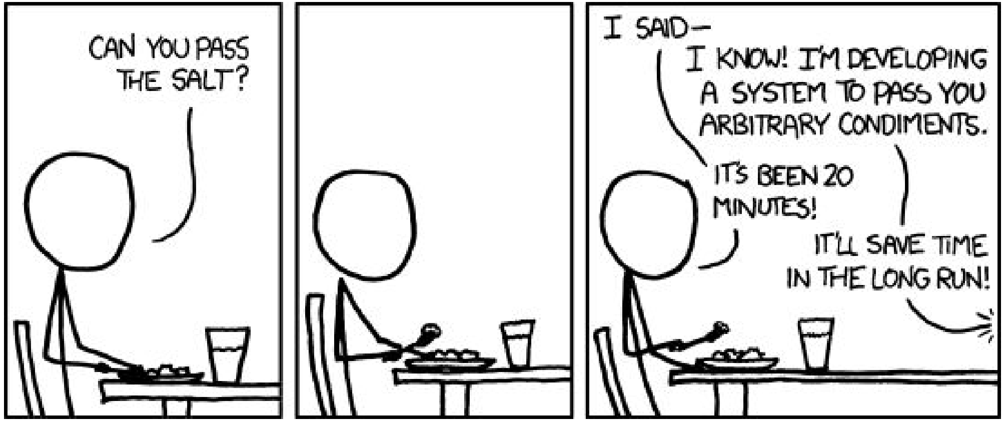
The logs will usually only tell you about the number of discarded samples. If you run version 0.19 or above, the `prometheus_local_storage_out_of_order_samples_total` metric will have a `reason` label. Also, by activating DEBUG level logging, you can see the exact dropped samples in the logfile.

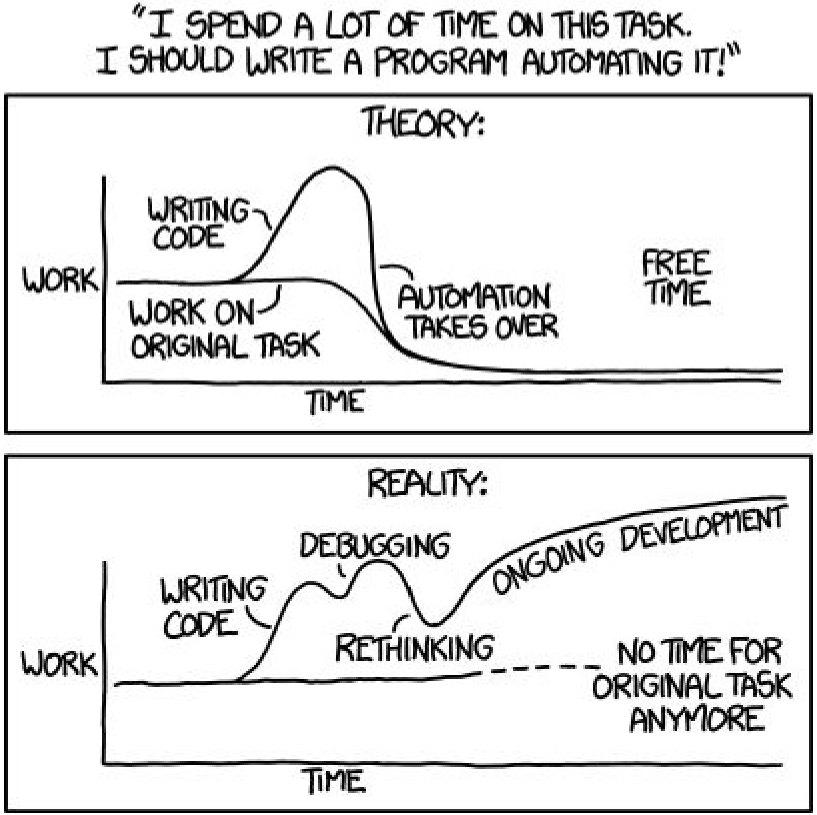
If you see this alert on `prometheus-mysql`, the reason is probably this bug.

Resolving

Out-of-order samples are usually created by configuration errors. Multiple rules funnel into the same time series, or a broken federation setup results in the same. Another suspect to look for is a target that exports explicit time stamps.

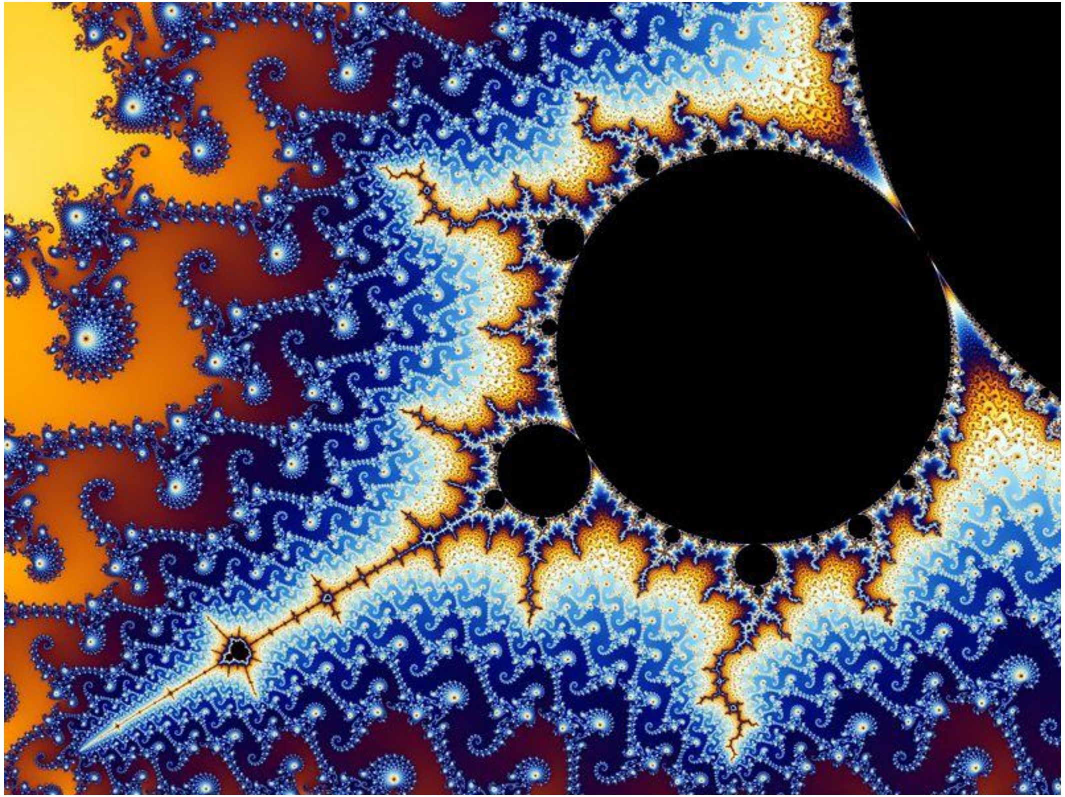
Automation might not pay off.


CAN YOU PASS THE SALT?
I SAID–
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!



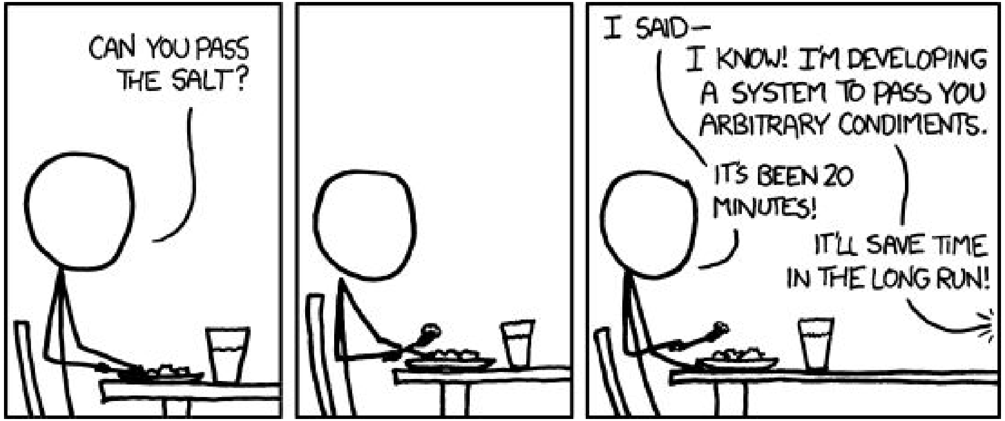
Source: https://xkcd.com/974/ https://xkcd.com/1319/

Automation introduces a feedback loop…

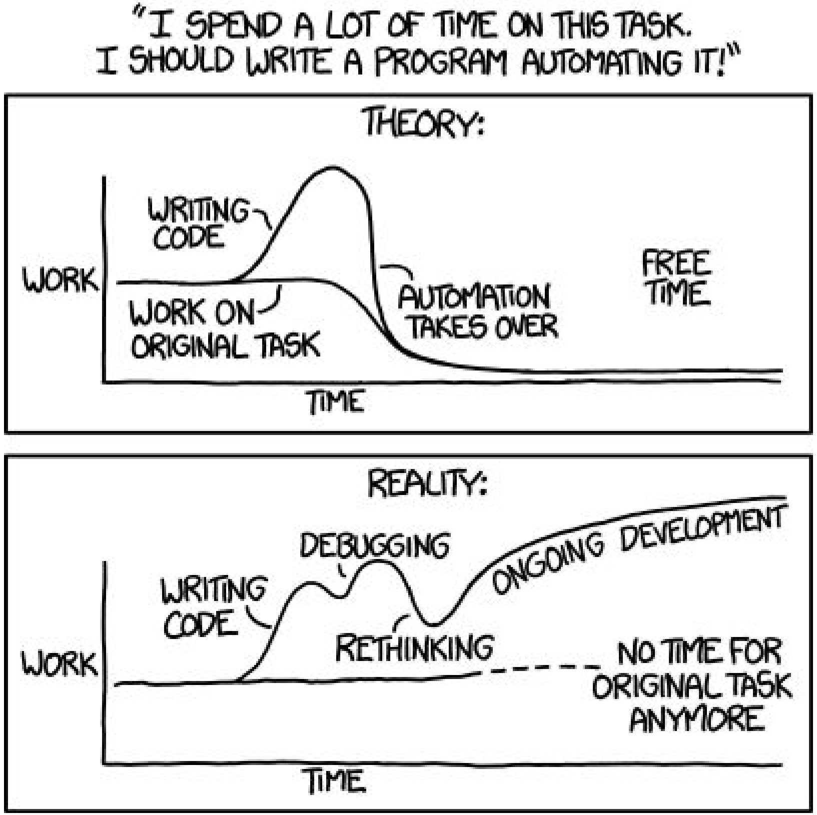


Source: Wolfgang Beyer, https://commons.wikimedia.org/wiki/File:Mandel_zoom_08_satellite_antenna.jpg

Automation might not pay off.



CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!



Source: https://xkcd.com/974/ https://xkcd.com/1319/