What makes Prometheus a “next generation” monitoring system?

Björn “Beorn” Rabenstein, Production Engineer, SoundCloud Ltd.
Sales pitch.
It’s how “they” did it...
Site Reliability Engineering
How Google Runs Production Systems

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

#GIFEE
Google Infrastructure For Everybody Else
Flow of inspiration – from 1st to 2nd systems

Borg → 

Borgmon →

2005 → 2015
It’s an “industry standard”...
Consolidate monitoring and alerting
Popular against all odds...
It’s a whole ecosystem!
Best 15m introduction talk: **Brian Brazil at FOSDEM 2016**
Best talk about Promethean philosophy: 
Julius Volz at PromCon 2016

“Prometheus is an opinionated monitoring system that chooses to do a lot of things differently from traditional monitoring systems. This leads to a culture clash for those used to other approaches, and raises questions as to why we didn't take a seemingly better approach.”
Blackbox vs. Whitebox.
Host vs. Service.
Dimensions of monitoring

- **white-box** (needs instrumentation)
- **black-box** (no changes required)

**host-based** “traditional”

**service-based** “modern”

**Nagios**

**STATSD** & Graphite
Challenges of service-based monitoring

Scale and semantics

➢ Way more targets to monitor.
  ○ A service has many instances.
  ○ A hosts can run many instances of different services.

➢ And they constantly change.
  ○ Deploys, rescheduling unhealthy instances ...

➢ Need a fleet-wide 3,048m view.
  ○ What’s my overall 99th percentile latency?

➢ Still need to be able to drill down for troubleshooting.
  ○ Which instance causes those errors I’m seeing?

➢ Meaningful alerting.
  ○ Symptom-based alerting for pages, cause-based alerting for warnings.
  ○ See Rob Ewaschuk’s *My philosophy on alerting* [https://goo.gl/2vrpSO](https://goo.gl/2vrpSO)
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.
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 [...]
Time series.
Static disk-full alert (*e.g.* Nagios)

Alert!!!

This is fine!?!
Time-series based disk-full alert (*e.g.* Prometheus)

This is actually fine!

Alert here… not there
Counter vs Gauge.
Why gauges suck and counters rule

Graphs shamelessly stolen from Jamie Wilkinson

Look up “Nyquist-Shannon sampling theorem”.
Labels are the new hierarchies.
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*
ampelmann.ams3.ip-10-12-11-17.api-mobile.5xx.requests

sum(ampelmann.ams3.ip-10-12-11-17.*.*.requests)

sum(ampelmann.*.*.api-mobile.5xx.requests)

ampelmann_requests_total{pool="ams3",instance="10.12.11.17:80", backend="api-mobile",code="5xx"}

sum(ampelmann_requests_total{pool="ams3",instance="10.12.11.17:80"})

sum(ampelmann_requests_total{backend="api-mobile",code="5xx"})
app_env_zone:request_duration:99perc_5m{app="api",zone="ams",env="production"}  0.243
app_env_zone:request_duration:99perc_5m{app="api",zone="ams",env="canary"}  0.219

[FIRING:1] APILatencyHigh (production api-team critical)
api app in env production has 99th percentile latency of 243ms, SLO is 240ms. Runbook