Intro to open source observability with Grafana, Prometheus, Loki, and Tempo

Richard “RichiH” Hartmann

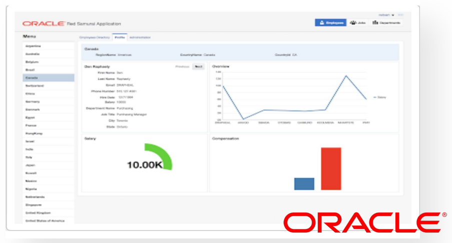
Grafana Labs

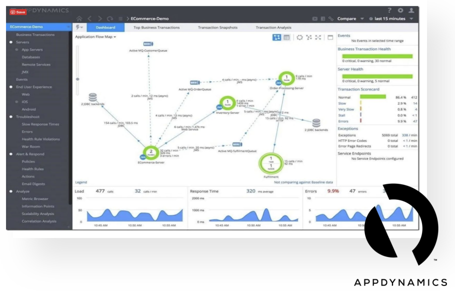
“

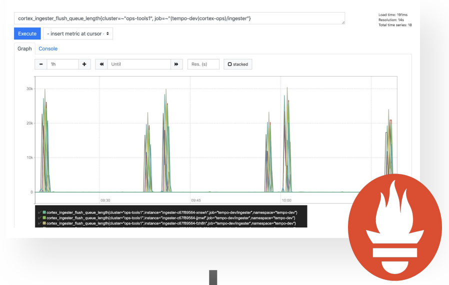
I come from the trenches of tech

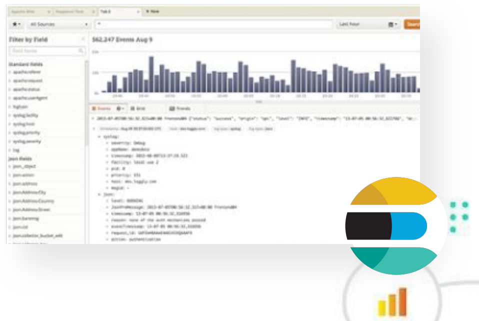
Bad tools are worse than no tools

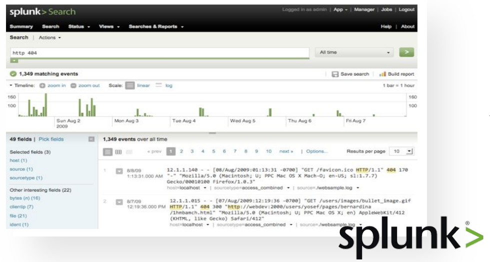
Today’s reality: Disparate systems. Disparate data.











“

Back to the basics

Let’s rethink this

Observability & SRE

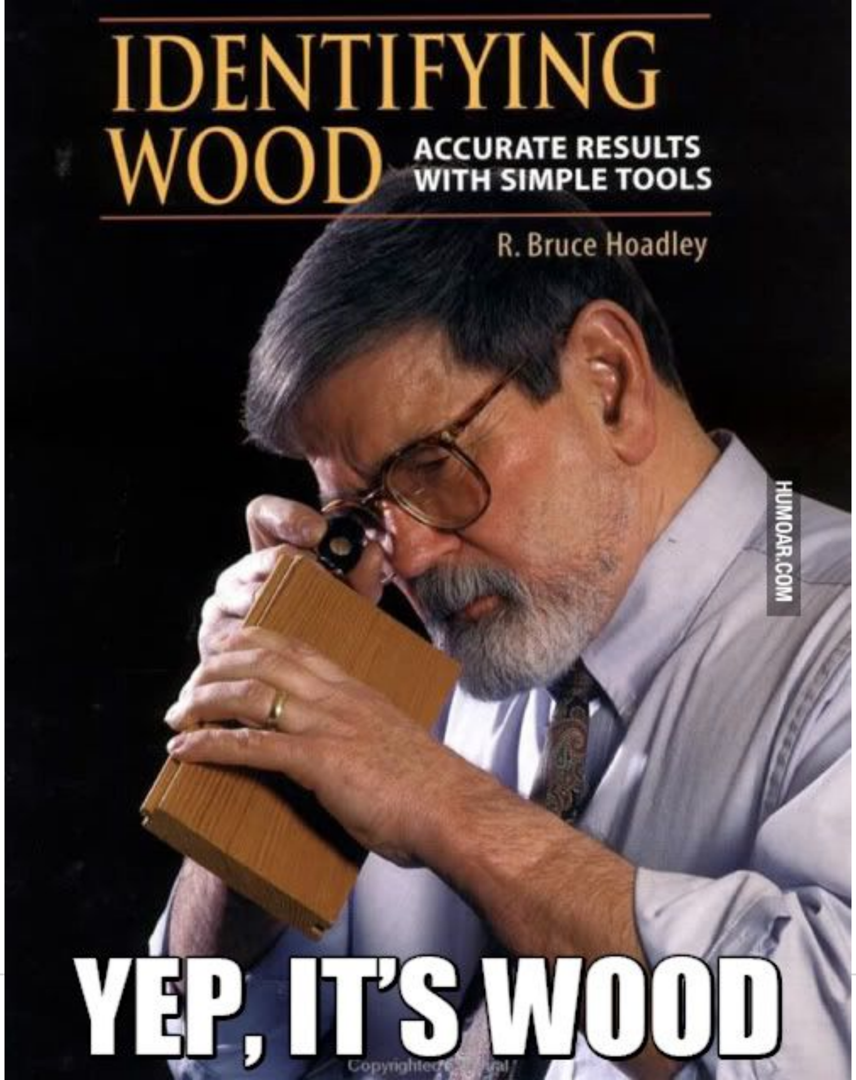
Or: Buzzwords, and their useful parts

Dirty secret

- What is “cloud native scale” is “Internet scale” of networks two decades ago
- The combination of metrics and events has been standard in power measurement for half a century
- Modern tools with good engineering practices map very well onto brownfield technology

Buzzword alert!

- Cool new term, almost meaningless by now, what does it mean?
  - Pitfall alert: Cargo culting
  - It's about changing the behaviour, not about changing the name
- "Monitoring" has taken on a meaning of collecting, not using data
  - One extreme: Full text indexing
  - Other extreme: Data lake
- "Observability" is about enabling humans to understand complex systems
  - Ask **why** it's not working instead of just knowing that it's not


IDENTIFYING WOOD
ACCURATE RESULTS WITH SIMPLE TOOLS
R. Bruce Hoadley
HUMOAR.COM
YEP, IT'S WOOD

“

If you can’t ask new questions on the fly, it’s not observability

Complexity

- Fake complexity, a.k.a. bad design
  - Can be **reduced**
- Real, system-inherent complexity
  - Can be **moved** (monolith vs client-server vs microservices)
  - Must be **compartmentalized** (service boundaries)
  - Should be **distilled meaningfully**

SRE, an instantiation of DevOps

- At its core: Align incentives across the org
    - Error budgets allow devs, ops, PMs, etc. to optimize for shared benefits
- Measure it!
    - SLI: Service Level Indicator: What you measure
    - SLO: Service Level Objective: What you need to hit
    - SLA: Service Level Agreement: When you need to pay

Shared understanding

- Everyone uses the same tools & dashboards
  - Shared incentive to invest into tooling
  - Pooling of institutional system knowledge
  - Shared language & understanding of services

Services

- Service?
  - Compartmentalized complexity, with an interface
  - Different owners/teams
  - Contracts define interfaces
- Why “contract”: Shared agreement which MUST NOT be broken
  - Internal and external customers rely on what you build and maintain
- Other common term: layer
  - The Internet would not exist without network layering
  - Enables innovation, parallelizes human engineering
- Other examples: CPUs, harddisk, compute nodes, your lunch

Alerting

- Customers care about services being up, not about individual components
- Discern between different SLIs
  - Primary: service-relevant, for alerting
  - Secondary: informational, debugging, might be underlying's primary

Anything currently or imminently impacting customer service must be alerted upon

But nothing(!) else

Prometheus

Prometheus 101

- Inspired by Google's Borgmon
- Time series database
- unit64 millisecond timestamp, float64 value
- Instrumentation & exporters
- Not for event logging
- Dashboarding via Grafana

Main selling points

- Highly dynamic, built-in service discovery
- No hierarchical model, n-dimensional label set
- PromQL: for processing, graphing, alerting, and export
- Simple operation
- Highly efficient

Main selling points

- Prometheus is a pull-based system
- Black-box monitoring: Looking at a service from the outside (Does the server answer to HTTP requests?)
- White-box monitoring: Instrumenting code from the inside (How much time does this subroutine take?)
- Every service should have its own metrics endpoint
- Hard API commitments within major versions

Time series

- Time series are recorded values which change over time
- Individual events are usually merged into counters and/or histograms
- Changing values are recorded as gauges
- Typical examples
    - Requests to a webserver (counter)
    - Temperatures in a datacenter (gauge)
    - Service latency (histograms)

Super easy to emit, parse & read

```
http_requests_total{env="prod",method="post",code="200"} 1027
http_requests_total{env="prod",method="post",code="400"} 3
http_requests_total{env="prod",method="post",code="500"} 12
http_requests_total{env="prod",method="get",code="200"} 20
http_requests_total{env="test",method="post",code="200"} 372
http_requests_total{env="test",method="post",code="400"} 75
```

Scale

- Kubernetes is Borg
- Prometheus is Borgmon
- Google couldn't have run Borg without Borgmon (plus Omega and Monarch)
- Kubernetes & Prometheus are designed and written with each other in mind

Prometheus scale

- 1,000,000+ samples/second no problem on current hardware
- ~200,000 samples/second/core
- 16 bytes/sample compressed to 1.36 bytes/sample

The highest we saw in production on a single Prometheus instance were 15 million active times series at once!

Long term storage

- Two long term storage solutions have Prometheus-team members working on them
  - Thanos
    - Historically easier to run, but slower
    - Scales storage horizontally
  - Cortex
    - Easy to run these days
    - Scales both storage, ingester, and querier horizontally

Cortex @ Grafana (largest cluster, 2021-09)

- ~65 million active series (just the one cluster)
- 668 CPU cores
- 3,349 GiB RAM

One customer running at 3 billion active series

Loki

Loki 101

- Following the same label-based system like Prometheus
- No full text index needed, incredible speed
- Work with logs at scale, without the massive cost
- Access logs with the same label sets as metrics
- Turn logs into metrics, to make it easier to work with them
- Make direct use of syslog data, via promtail

```
2019-12-11T10:01:02.123456789Z  {env="prod",instance="1.1.1.1"}  GET /about
```

Timestamp

with nanosecond precision

Prometheus-style **Labels**

key-value pairs

Content

log line

indexed

unindexed

Loki @ Grafana Labs

- Queries regularly see 40 GiB/s
- Query terabytes of data in under a minute
  - Including complex processing of result sets

Tempo

Tempo

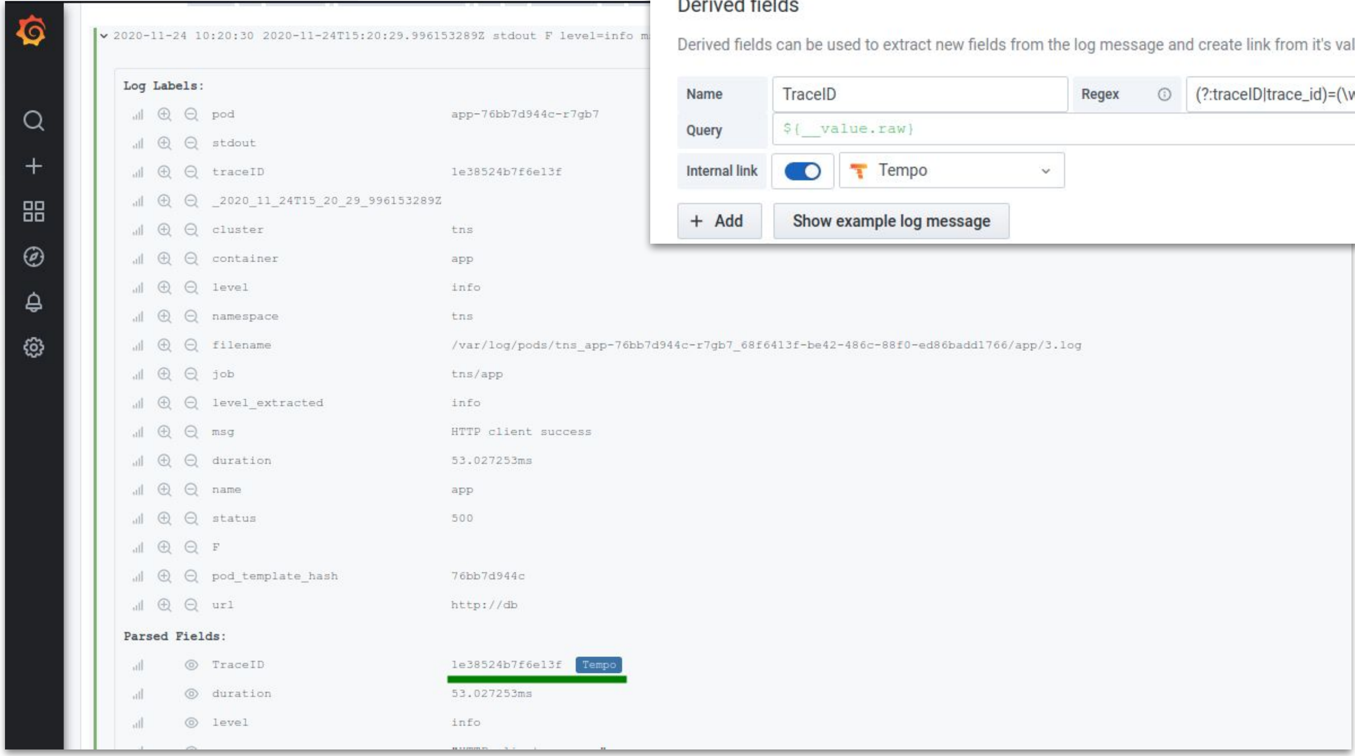
- Exemplars: Jump from relevant logs & metrics
  - Native to Prometheus, Cortex, Thanos, and Loki
  - Exemplars work at Google scale, with the ease of Grafana
- Index and search by labelsets available for those who need it
- Object store only: No Cassandra, Elastic, etc.
- 100% compatible with OpenTelemetry Tracing, Zipkin, Jaeger
- 100% of your traces, no sampling

Tempo @ Grafana Labs (2021-07)

- 2,200,000 samples per second @ 350 MiB/s
- 14-day retention @ 3 copies stored
- ~240 CPU cores (includes compression cost)
- ~450 GiB RAM
- 132 TiB object storage
- Latencies:
    - p99 - 2.5s
    - p90 - 2.3s
    - p50 - 1.6s

Bringing it together

From logs to traces



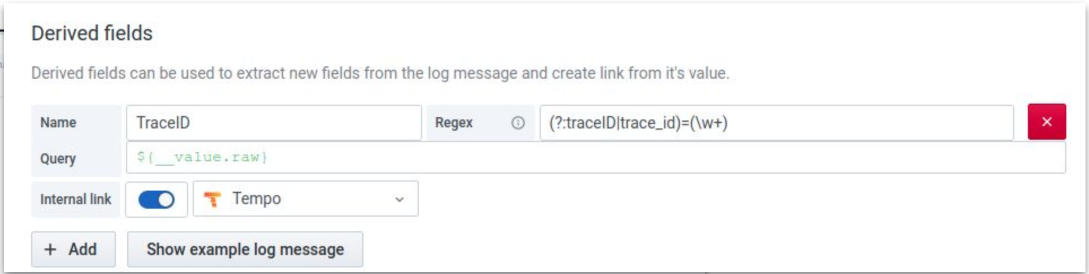
```
2020-11-24 10:20:30 2020-11-24T15:20:29.996153289Z stdout F level=info m
```

Log Labels:

pod	app-76bb7d944c-r7gb7
stdout	
traceID	1e38524b7f6e13f
_2020_11_24T15_20_29_996153289Z	
cluster	tns
container	app
level	info
namespace	tns
filename	/var/log/pods/tns_app-76bb7d944c-r7gb7_68f6413f-be42-486c-88f0-ed86badd1766/app/3.log
job	tns/app
level_extracted	info
msg	HTTP client success
duration	53.027253ms
name	app
status	500
F	
pod_template_hash	76bb7d944c
url	http://db

Parsed Fields:

TraceID	1e38524b7f6e13f Tempo
duration	53.027253ms
level	info

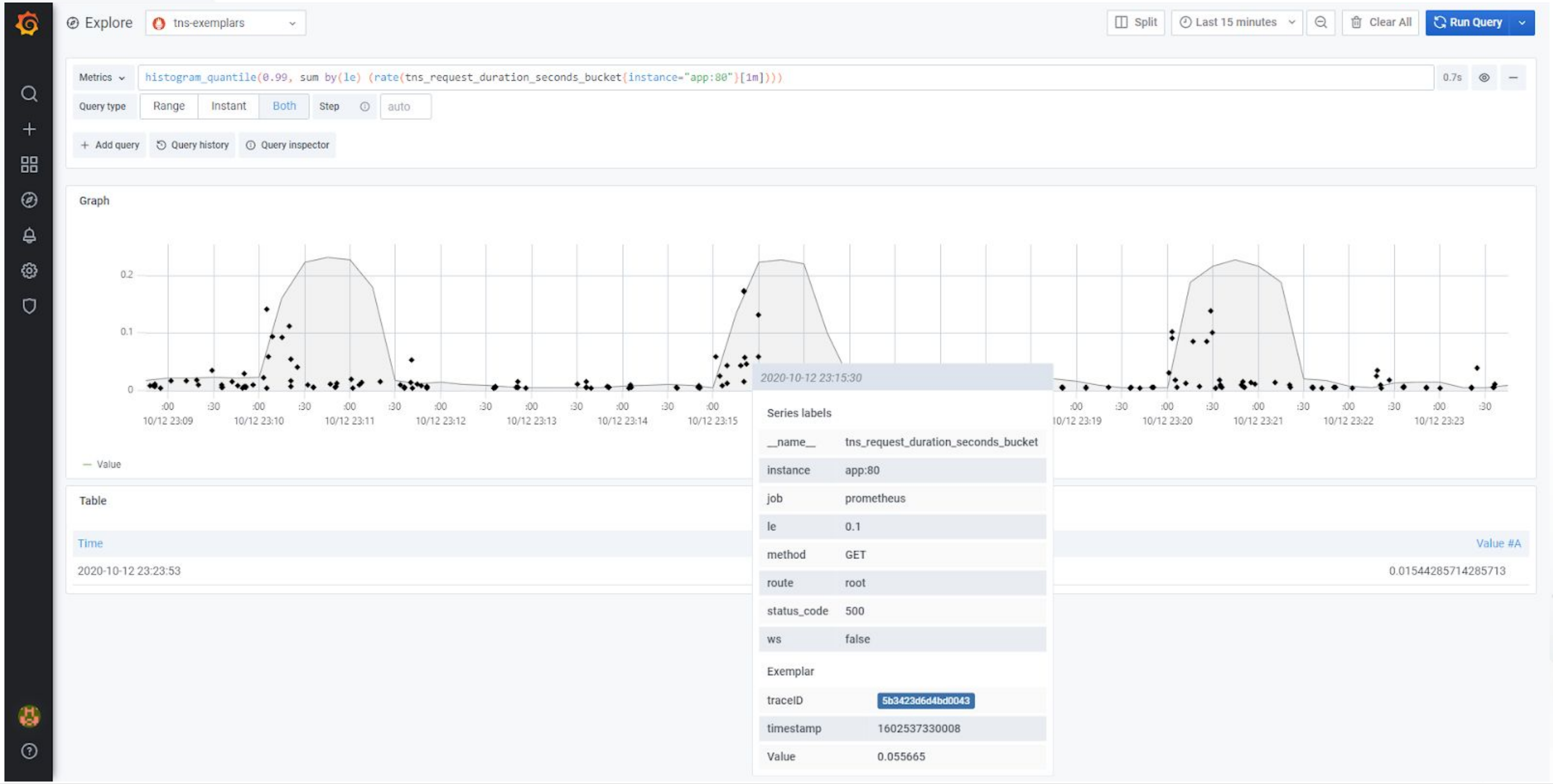


Derived fields

Derived fields can be used to extract new fields from the log message and create link from it's value.

Name	TraceID	Regex	(?:traceID|trace_id)=(\w+)
Query	${__value.raw}		
Internal link	on	Tempo	

From metrics to traces



Metrics `histogram_quantile(0.99, sum by(le) (rate(tns_request_duration_seconds_bucket{instance="app:80"}[1m])))` 0.7s

Query type: Range | Instant | [✓ Both] — Step: auto

Table

Time	Value #A
2020-10-12 23:23:53	0.01544285714285713

2020-10-12 23:15:30

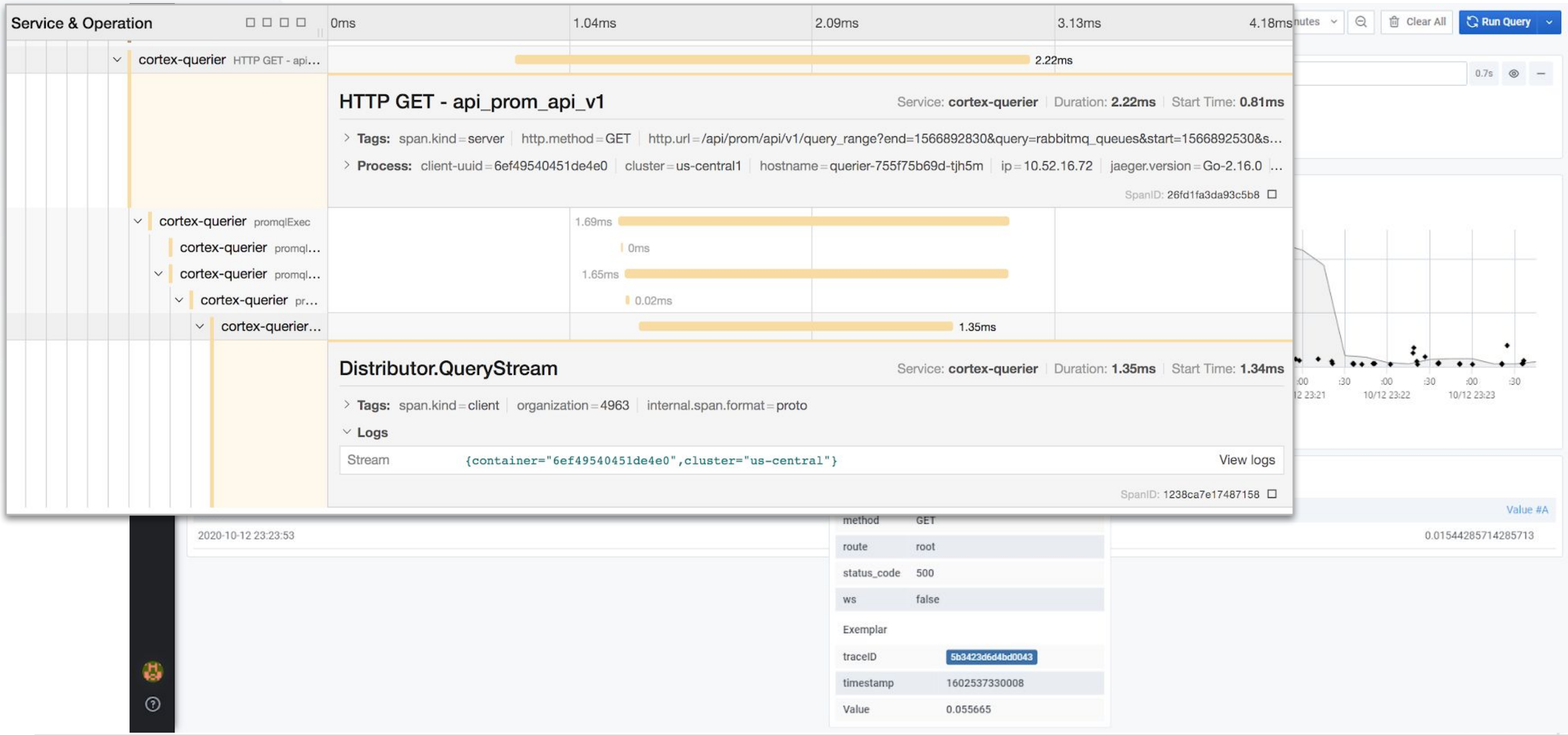
Series labels

__name__	tns_request_duration_seconds_bucket
instance	app:80
job	prometheus
le	0.1
method	GET
route	root
status_code	500
ws	false

Exemplar

traceID	5b3423d6d4bd0043
timestamp	1602537330008
Value	0.055665

From metrics to traces



Service & Operation | 0ms | 1.04ms | 2.09ms | 3.13ms | 4.18ms

cortex-querier HTTP GET - api… 2.22ms

HTTP GET - api_prom_api_v1 Service: **cortex-querier** | Duration: **2.22ms** | Start Time: **0.81ms**

> **Tags:** span.kind = server | http.method = GET | http.url = /api/prom/api/v1/query_range?end=1566892830&query=rabbitmq_queues&start=1566892530&s…

> **Process:** client-uuid = 6ef49540451de4e0 | cluster = us-central1 | hostname = querier-755f75b69d-tjh5m | ip = 10.52.16.72 | jaeger.version = Go-2.16.0 | …

SpanID: 26fd1fa3da93c5b8

cortex-querier promqlExec 1.69ms
cortex-querier promql… 0ms
cortex-querier promql… 1.65ms
cortex-querier pr… 0.02ms
cortex-querier… 1.35ms

Distributor.QueryStream Service: **cortex-querier** | Duration: **1.35ms** | Start Time: **1.34ms**

> **Tags:** span.kind = client | organization = 4963 | internal.span.format = proto

Logs

```
Stream   {container="6ef49540451de4e0",cluster="us-central"}   View logs
```

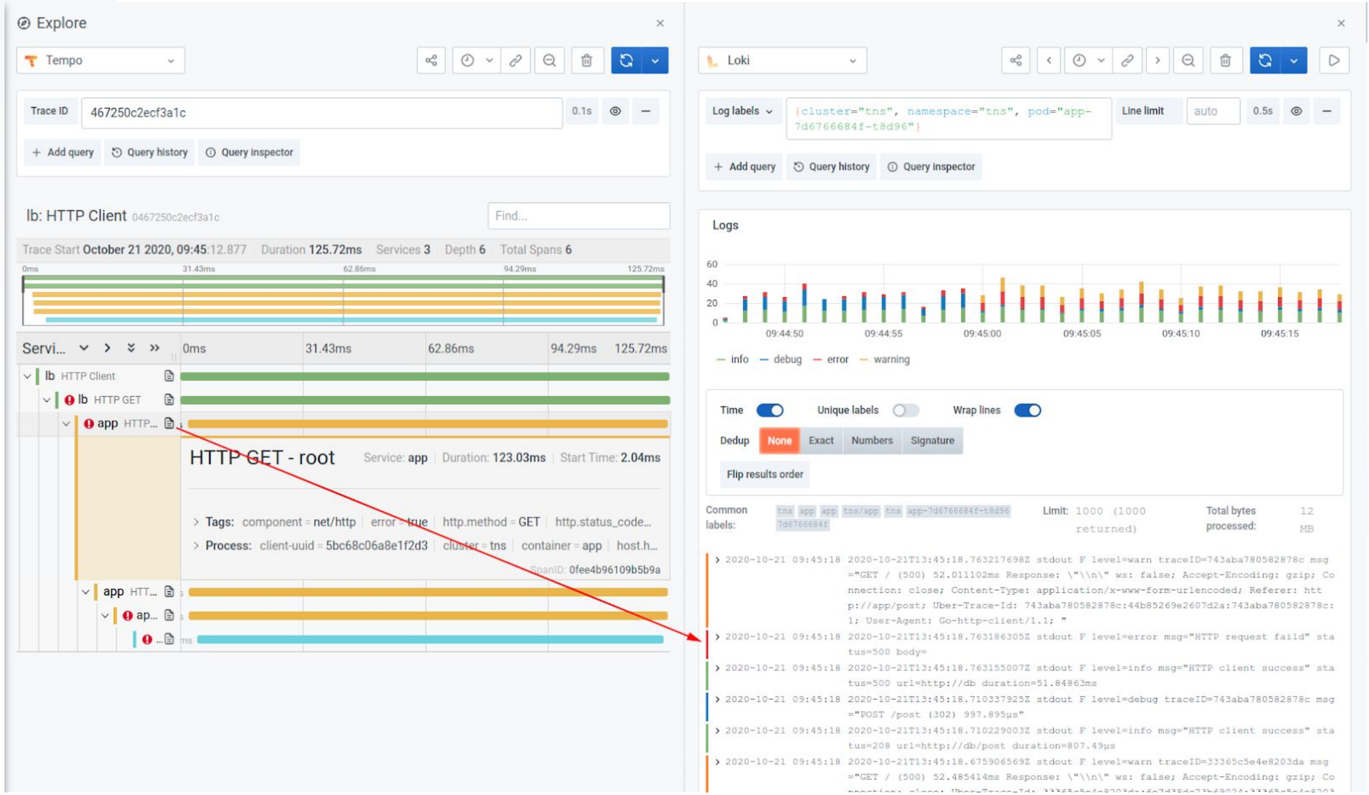
SpanID: 1238ca7e17487158

2020-10-12 23:23:53 | 0.01544285714285713

method	GET
route	root
status_code	500
ws	false
Exemplar	
traceID	5b3423d6d4bd0043
timestamp	1602537330008
Value	0.055665

Grafana Labs

...and from traces to logs



Trace ID: 467250c2ecf3a1c — 0.1s

lb: HTTP Client 0467250c2ecf3a1c

Trace Start October 21 2020, 09:45:12.877 Duration 125.72ms Services 3 Depth 6 Total Spans 6

Service	
lb HTTP Client	
lb HTTP GET	
app HTTP...	
app HTT...	
ap...	

HTTP GET - root — Service: app | Duration: 123.03ms | Start Time: 2.04ms

Tags: component = net/http | error = true | http.method = GET | http.status_code...

Process: client-uuid = 5bc68c06a8e1f2d3 | cluster = tns | container = app | host.h...

SpanID: 0fee4b96109b5b9a

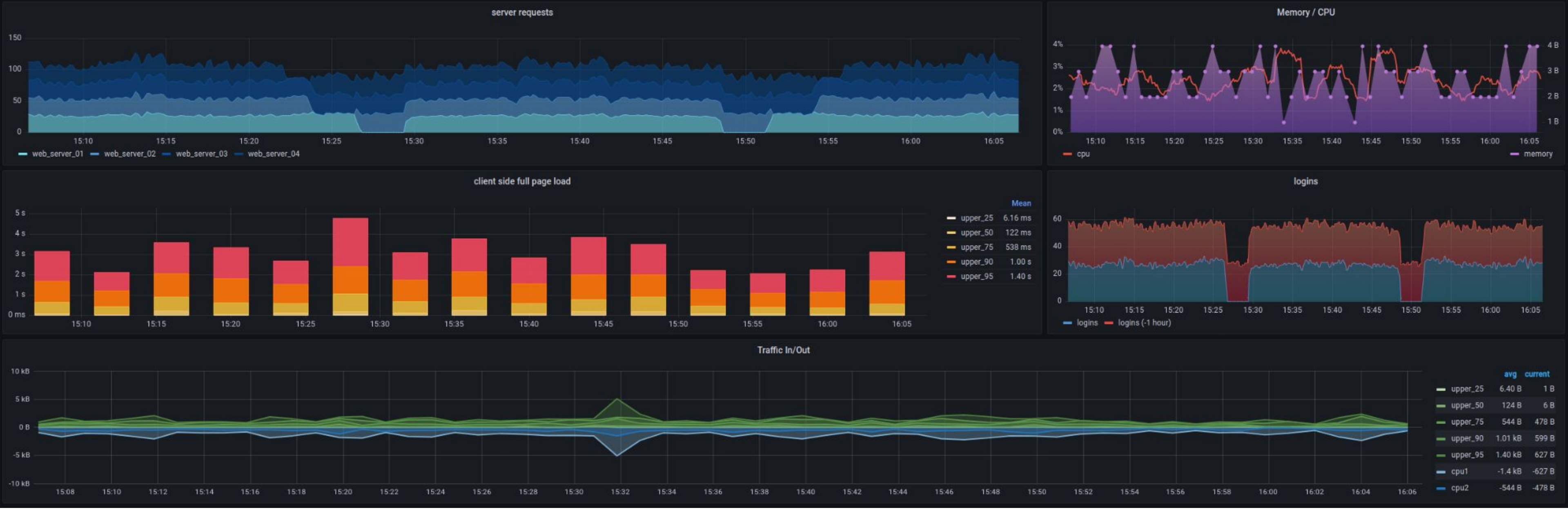
```
{cluster="tns", namespace="tns", pod="app-7d6766684f-t8d96"}
```

Common labels: tns app app tns/app tns app-7d6766684f-t8d96 7d6766684f — Limit: 1000 (1000 returned) — Total bytes processed: 12 MB

```
> 2020-10-21 09:45:18 2020-10-21T13:45:18.763217698Z stdout F level=warn traceID=743aba780582878c msg
                      ="GET / (500) 52.011102ms Response: \"\\n\" ws: false; Accept-Encoding: gzip; Co
                      nnection: close; Content-Type: application/x-www-form-urlencoded; Referer: htt
                      p://app/post; Uber-Trace-Id: 743aba780582878c:44b85269e2607d2a:743aba780582878c:
                      1; User-Agent: Go-http-client/1.1; "
> 2020-10-21 09:45:18 2020-10-21T13:45:18.763186305Z stdout F level=error msg="HTTP request faild" sta
                      tus=500 body=
> 2020-10-21 09:45:18 2020-10-21T13:45:18.763155007Z stdout F level=info msg="HTTP client success" sta
                      tus=500 url=http://db duration=51.84863ms
> 2020-10-21 09:45:18 2020-10-21T13:45:18.710337925Z stdout F level=debug traceID=743aba780582878c msg
                      ="POST /post (302) 997.895µs"
> 2020-10-21 09:45:18 2020-10-21T13:45:18.710229003Z stdout F level=info msg="HTTP client success" sta
                      tus=208 url=http://db/post duration=807.49µs
> 2020-10-21 09:45:18 2020-10-21T13:45:18.675906569Z stdout F level=warn traceID=33365c5e4e8203da msg
                      ="GET / (500) 52.485414ms Response: \"\\n\" ws: false; Accept-Encoding: gzip; Co
```

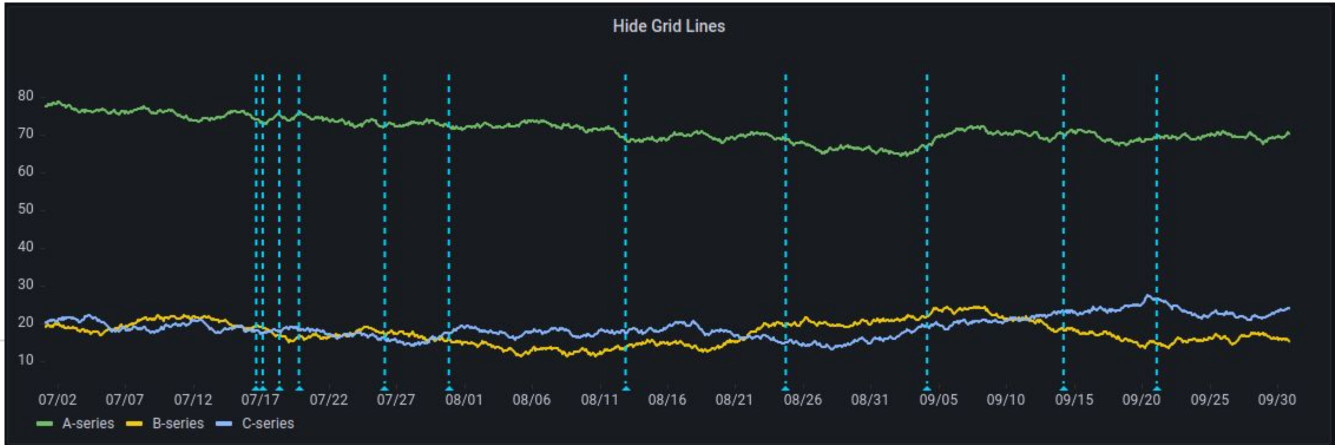
“

All of this is Open Source and you can run it yourself

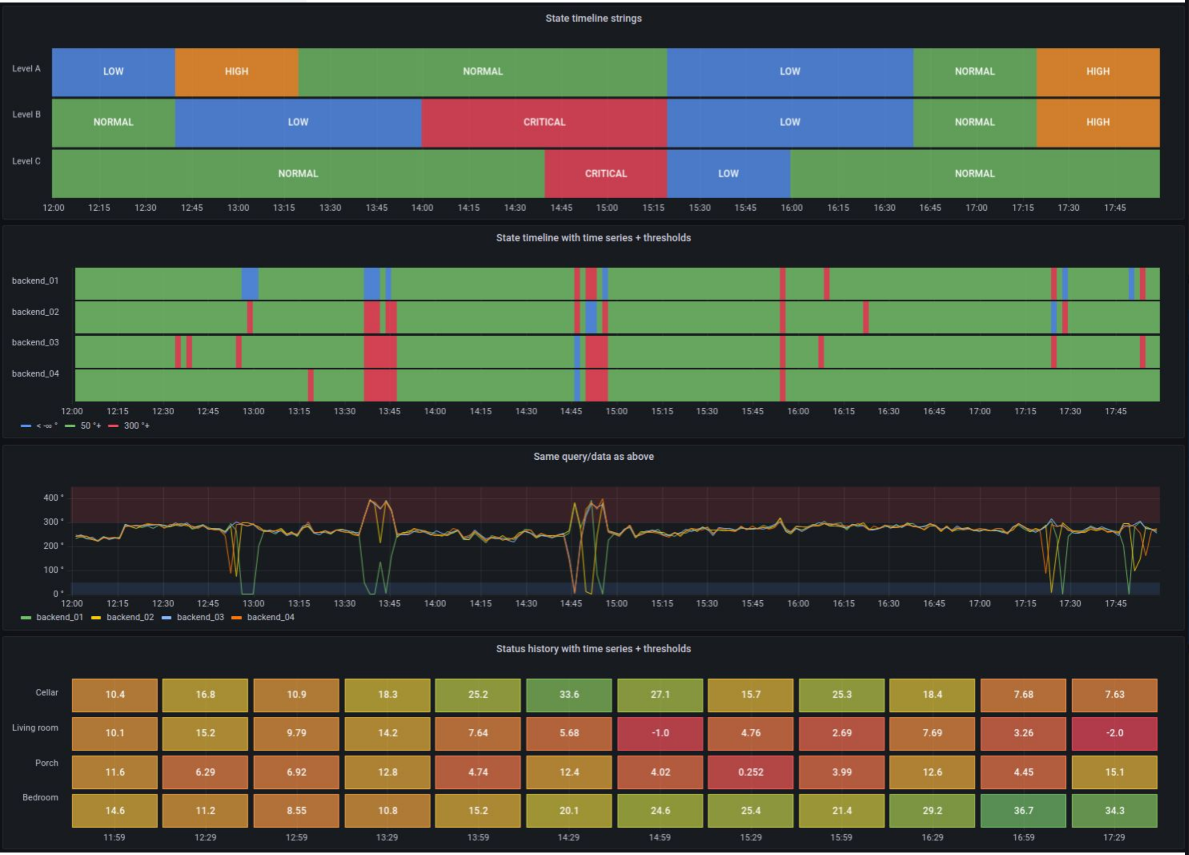


	Mean
upper_25	6.16 ms
upper_50	122 ms
upper_75	538 ms
upper_90	1.00 s
upper_95	1.40 s

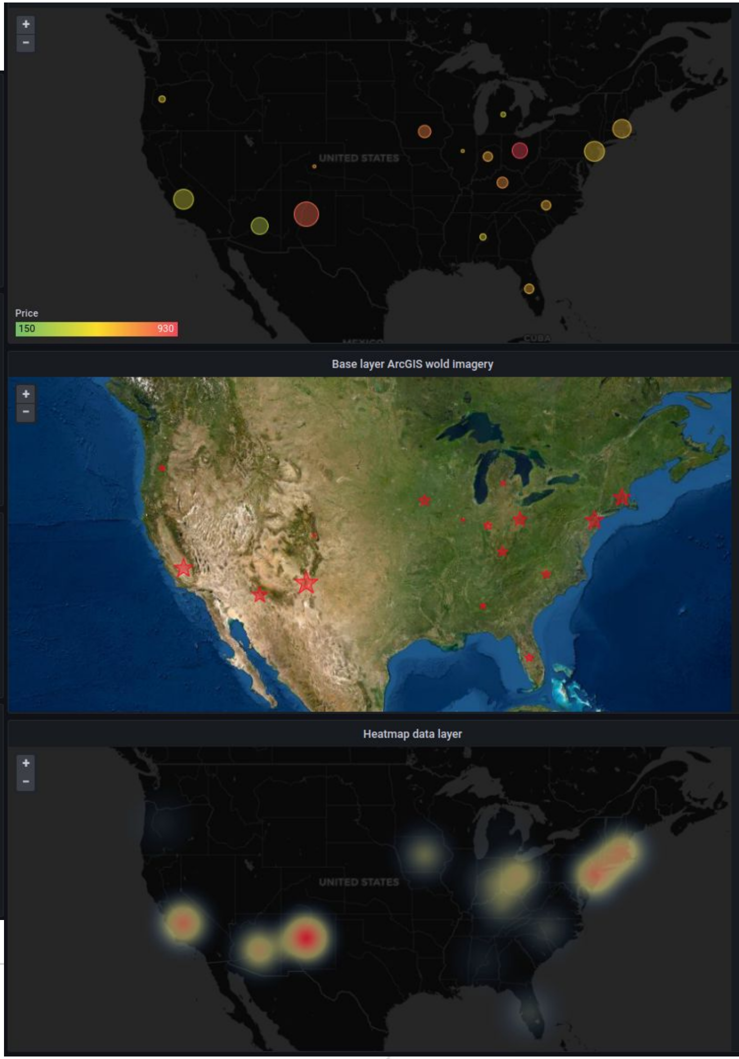
	avg	current
upper_25	6.40 B	1 B
upper_50	124 B	6 B
upper_75	544 B	478 B
upper_90	1.01 kB	599 B
upper_95	1.40 kB	627 B
cpu1	-1.4 kB	-627 B
cpu2	-544 B	-478 B



Grafana Labs



	11:59	12:29	12:59	13:29	13:59	14:29	14:59	15:29	15:59	16:29	16:59	17:29
Cellar	10.4	16.8	10.9	18.3	25.2	33.6	27.1	15.7	25.3	18.4	7.68	7.63
Living room	10.1	15.2	9.79	14.2	7.64	5.68	-1.0	4.76	2.69	7.69	3.26	-2.0
Porch	11.6	6.29	6.92	12.8	4.74	12.4	4.02	0.252	3.99	12.6	4.45	15.1
Bedroom	14.6	11.2	8.55	10.8	15.2	20.1	24.6	25.4	21.4	29.2	36.7	34.3



For a deeper dive into
Open Source Observability, go to:

grafana.com/go/obscon2021

Thank you!

richih@richih.org
twitter.com/TwitchiH
github.com/RichiH/talks