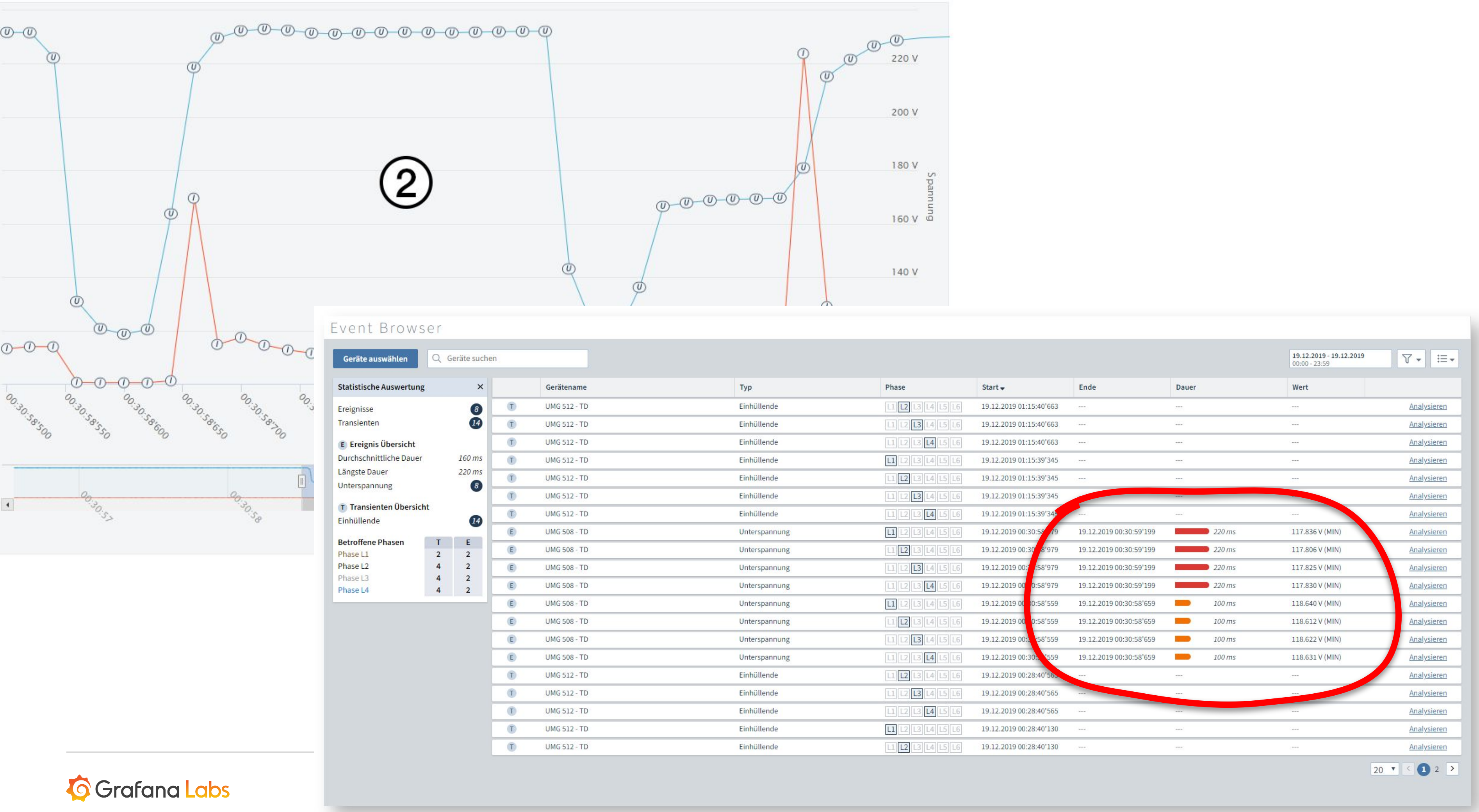
Cloud native observability with Prometheus and beyond

Or: philosophy of Observability

Richard “RichiH” Hartmann

Grafana Labs

How humans deal with data



Event Browser

Geräte auswählen | Geräte suchen | 19.12.2019 - 19.12.2019 00:00 - 23:59

Statistische Auswertung

Ereignisse	8
Transienten	14

E Ereignis Übersicht

Durchschnittliche Dauer	*160 ms*
Längste Dauer	*220 ms*
Unterspannung	8

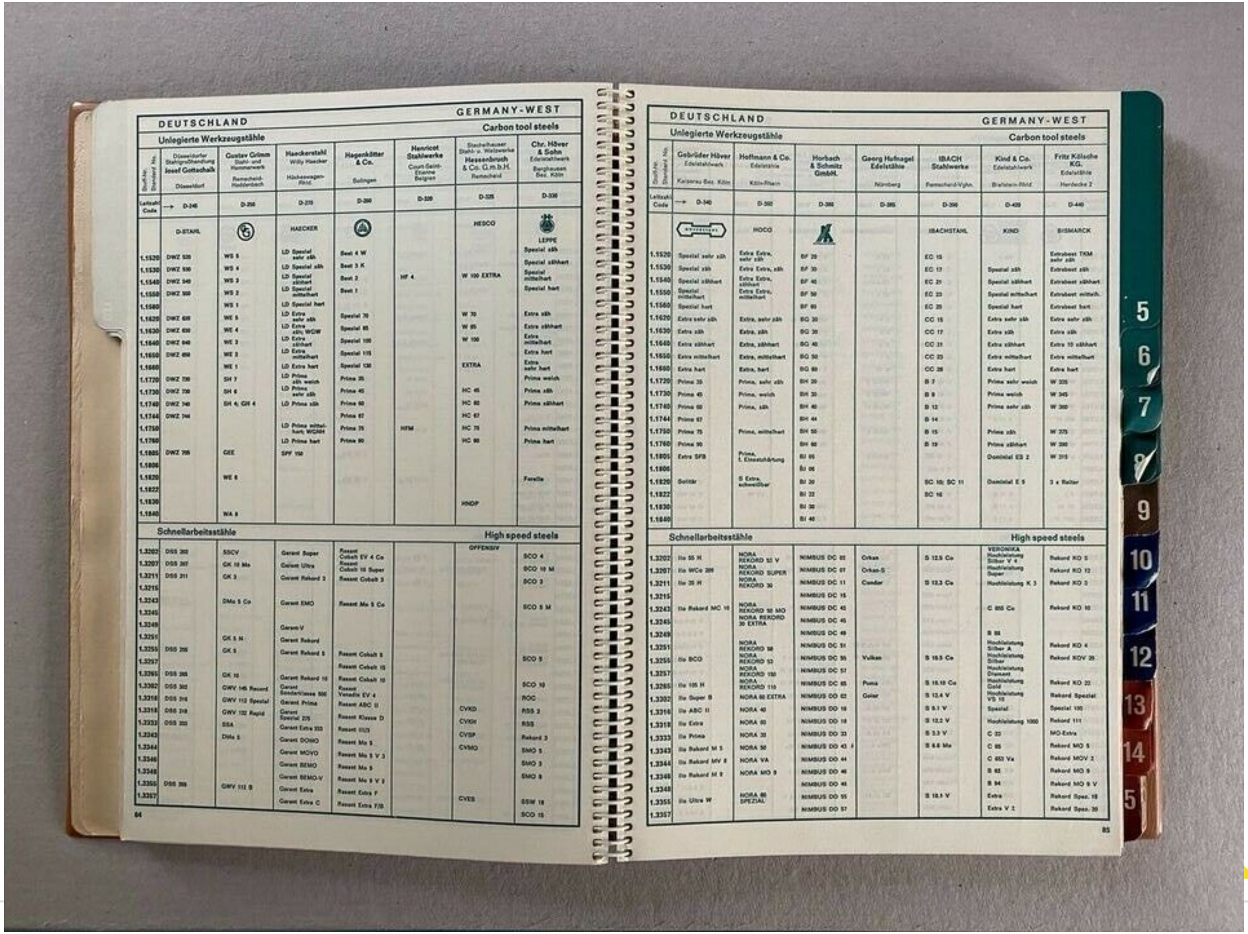
T Transienten Übersicht

Einhüllende	14

Betroffene Phasen	T	E
Phase L1	2	2
Phase L2	4	2
Phase L3	4	2
Phase L4	4	2

	Gerätename	Typ	Phase	Start	Ende	Dauer	Wert	
T	UMG 512 - TD	Einhüllende	L1 **L2** L3 L4 L5 L6	19.12.2019 01:15:40'663	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	L1 L2 **L3** L4 L5 L6	19.12.2019 01:15:40'663	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	L1 L2 L3 **L4** L5 L6	19.12.2019 01:15:40'663	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	**L1** L2 L3 L4 L5 L6	19.12.2019 01:15:39'345	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	L1 **L2** L3 L4 L5 L6	19.12.2019 01:15:39'345	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	L1 L2 **L3** L4 L5 L6	19.12.2019 01:15:39'345	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	L1 L2 L3 **L4** L5 L6	19.12.2019 01:15:39'345	---	---	---	Analysieren
E	UMG 508 - TD	Unterspannung	**L1** L2 L3 L4 L5 L6	19.12.2019 00:30:58'979	19.12.2019 00:30:59'199	🟥 *220 ms*	117.836 V (MIN)	Analysieren
E	UMG 508 - TD	Unterspannung	L1 **L2** L3 L4 L5 L6	19.12.2019 00:30:58'979	19.12.2019 00:30:59'199	🟥 *220 ms*	117.806 V (MIN)	Analysieren
E	UMG 508 - TD	Unterspannung	L1 L2 **L3** L4 L5 L6	19.12.2019 00:30:58'979	19.12.2019 00:30:59'199	🟥 *220 ms*	117.825 V (MIN)	Analysieren
E	UMG 508 - TD	Unterspannung	L1 L2 L3 **L4** L5 L6	19.12.2019 00:30:58'979	19.12.2019 00:30:59'199	🟥 *220 ms*	117.830 V (MIN)	Analysieren
E	UMG 508 - TD	Unterspannung	**L1** L2 L3 L4 L5 L6	19.12.2019 00:30:58'559	19.12.2019 00:30:58'659	🟧 *100 ms*	118.640 V (MIN)	Analysieren
E	UMG 508 - TD	Unterspannung	L1 **L2** L3 L4 L5 L6	19.12.2019 00:30:58'559	19.12.2019 00:30:58'659	🟧 *100 ms*	118.612 V (MIN)	Analysieren
E	UMG 508 - TD	Unterspannung	L1 L2 **L3** L4 L5 L6	19.12.2019 00:30:58'559	19.12.2019 00:30:58'659	🟧 *100 ms*	118.622 V (MIN)	Analysieren
E	UMG 508 - TD	Unterspannung	L1 L2 L3 **L4** L5 L6	19.12.2019 00:30:58'559	19.12.2019 00:30:58'659	🟧 *100 ms*	118.631 V (MIN)	Analysieren
T	UMG 512 - TD	Einhüllende	L1 **L2** L3 L4 L5 L6	19.12.2019 00:28:40'565	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	L1 L2 **L3** L4 L5 L6	19.12.2019 00:28:40'565	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	L1 L2 L3 **L4** L5 L6	19.12.2019 00:28:40'565	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	**L1** L2 L3 L4 L5 L6	19.12.2019 00:28:40'130	---	---	---	Analysieren
T	UMG 512 - TD	Einhüllende	L1 **L2** L3 L4 L5 L6	19.12.2019 00:28:40'130	---	---	---	Analysieren

20 ▾ ‹ 1 2 ›



DEUTSCHLAND GERMANY-WEST

Unlegierte Werkzeugstähle Carbon tool steels

Stoff-Nr. Standard No.	Düsseldorfer Stahlgroßhandlung Josef Gottschalk Düsseldorf	Gustav Grimm Stahl- und Hammerwerk Remscheid-Haddenbach	Haeckerstahl Willy Haecker Hückeswagen-Rhld.	Hagenkötter & Co. Solingen	Henricot Stahlwerke Court-Saint-Etienne Belgien	Stachelhauser Stahl- u. Walzwerke Hessenbruch & Co. G.m.b.H. Remscheid	Chr. Höver & Sohn Edelstahlwerk Berghausen Bez. Köln
Leitzahl Code	→ D-240	D-250	D-270	D-280	D-320	D-325	D-330
	D-STAHL		HAECKER			HESCO	LEPPE
1.1520	DWZ 520	WS 5	LD Spezial sehr zäh	Best 4 W			Spezial zäh
1.1530	DWZ 530	WS 4	LD Spezial zäh	Best 3 K			Spezial zähhart
1.1540	DWZ 540	WS 3	LD Spezial zähhart	Best 2	HF 4	W 100 EXTRA	Spezial mittelhart
1.1550	DWZ 550	WS 2	LD Spezial mittelhart	Best 1			Spezial hart
1.1580		WS 1	LD Spezial hart				
1.1620	DWZ 620	WE 5	LD Extra sehr zäh	Spezial 70		W 70	Extra zäh
1.1630	DWZ 630	WE 4	LD Extra zäh; WOW	Spezial 85		W 85	Extra zähhart
1.1640	DWZ 640	WE 3	LD Extra zähhart	Spezial 100		W 100	Extra mittelhart
1.1650	DWZ 650	WE 2	LD Extra mittelhart	Spezial 115			Extra hart
1.1660		WE 1	LD Extra hart	Spezial 130		EXTRA	Extra sehr hart
1.1720	DWZ 720	SH 7	LD Prima zäh weich	Prima 35			Prima weich
1.1730	DWZ 730	SH 6	LD Prima sehr zäh	Prima 45		HC 45	Prima zäh
1.1740	DWZ 740	SH 4; GH 4	LD Prima zäh	Prima 60		HC 60	Prima zähhart
1.1744	DWZ 744			Prima 67		HC 67	
1.1750			LD Prima mittelhart; WOMH	Prima 75	HFM	HC 75	Prima mittelhart
1.1760			LD Prima hart	Prima 90		HC 90	Prima hart
1.1805	DWZ 705	GEE	SPF 150				
1.1806							
1.1820		WE 6					Forelle
1.1822							
1.1830						HNDP	
1.1840		WA 8					

Schnellarbeitsstähle High speed steels

Stoff-Nr. Standard No.	Josef Gottschalk	Gustav Grimm	Haeckerstahl	Hagenkötter & Co.	Henricot Stahlwerke	Hessenbruch & Co.	Chr. Höver & Sohn
						OFFENSIV	
1.3202	DSS 202	SSCV	Garant Super	Rasant Cobalt EV 4 Co			SCO 4
1.3207	DSS 207	GK 10 Mo	Garant Ultra	Rasant Cobalt 10 Super			SCO 10 M
1.3211	DSS 211	GK 3	Garant Rekord 3	Rasant Cobalt 3			SCO 3
1.3215							
1.3243		DMo 5 Co	Garant EMO	Rasant Mo 5 Co			SCO 5 M
1.3245							
1.3249			Garant V				
1.3251		GK 5 N	Garant Rekord				
1.3255	DSS 255	GK 5	Garant Rekord 5	Rasant Cobalt 5			
1.3257				Rasant Cobalt 15			SCO 5
1.3265	DSS 265	GK 10	Garant Rekord 10	Rasant Cobalt 10			
1.3302	DSS 302	GWV 145 Record	Garant Sonderklasse 500	Rasant Vanadin EV 4			SCO 10
1.3316	DSS 316	GWV 112 Spezial	Garant Prima	Rasant ABC II			ROC
1.3318	DSS 318	GWV 132 Rapid	Garant Spezial 275	Rasant Klasse D		CVKD	RSS 3
1.3333	DSS 333	SSA	Garant Extra 333	Rasant III/3		CVKH	RSS
1.3343		DMo 5	Garant DOMO	Rasant Mo 5		CVSP	Rekord 3
1.3344			Garant MOVO	Rasant Mo 5 V 3		CVMO	SMO 5
1.3346			Garant BEMO	Rasant Mo 9			SMO 3
1.3348			Garant BEMO-V	Rasant Mo 9 V 2			SMO 9
1.3355	DSS 355	GWV 112 B	Garant Extra	Rasant Extra F			
1.3357			Garant Extra C	Rasant Extra F/3		CVES	SSW 18 SCO 15

84

DEUTSCHLAND GERMANY-WEST

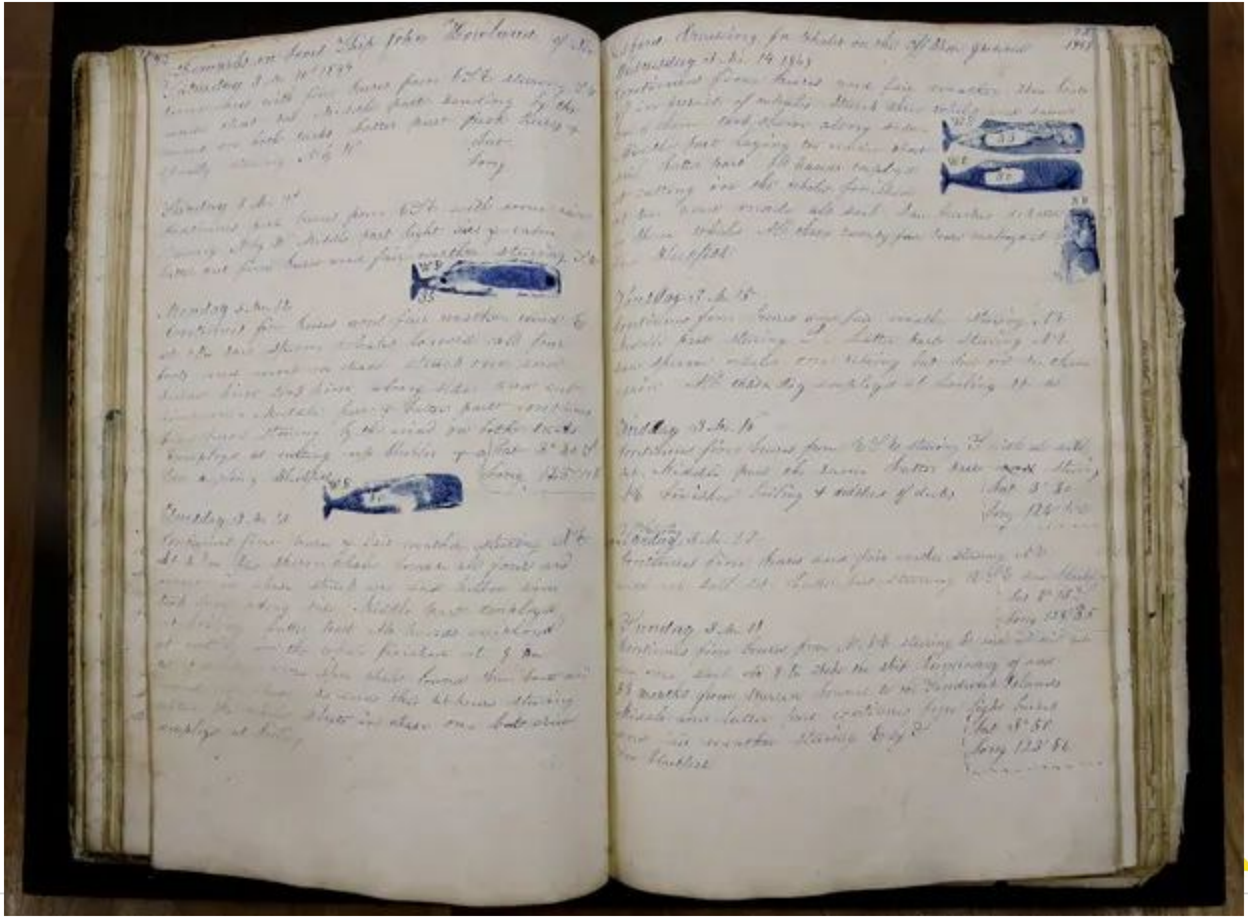
Unlegierte Werkzeugstähle Carbon tool steels

Stoff-Nr. Standard No.	Gebrüder Höver Edelstahlwerk Kaiserau Bez. Köln	Hoffmann & Co. Edelstähle Köln-Rhein	Horbach & Schmitz GmbH.	Georg Hufnagel Edelstähle Nürnberg	IBACH Stahlwerke Remscheid-Vghn.	Kind & Co. Edelstahlwerk Bielstein-Rhld.	Fritz Kölsche KG. Edelstähle Herdecke 2
Leitzahl Code	→ D-340	D-350	D-380	D-385	D-390	D-400	D-440
		HOCO			IBACHSTAHL	KIND	BISMARCK
1.1520	Spezial sehr zäh	Extra Extra, sehr zäh	BF 20		EC 15		Extrabest TKM sehr zäh
1.1530	Spezial zäh	Extra Extra, zäh	BF 30		EC 17	Spezial zäh	Extrabest zäh
1.1540	Spezial zähhart	Extra Extra, zähhart	BF 40		EC 21	Spezial zähhart	Extrabest zähhart
1.1550	Spezial mittelhart	Extra Extra, mittelhart	BF 50		EC 23	Spezial mittelhart	Extrabest mitteh.
1.1560	Spezial hart		BF 60		EC 25	Spezial hart	Extrabest hart
1.1620	Extra sehr zäh	Extra, sehr zäh	BG 20		CC 15	Extra sehr zäh	Extra sehr zäh
1.1630	Extra zäh	Extra, zäh	BG 30		CC 17	Extra zäh	Extra zäh
1.1640	Extra zähhart	Extra, zähhart	BG 40		CC 21	Extra zähhart	Extra 10 zähhart
1.1650	Extra mittelhart	Extra, mittelhart	BG 50		CC 23	Extra mittelhart	Extra mittelhart
1.1660	Extra hart	Extra, hart	BG 60		CC 26	Extra hart	Extra hart
1.1720	Prima 35	Prima, sehr zäh	BH 20		B 7	Prima sehr weich	W 325
1.1730	Prima 45	Prima, weich	BH 30		B 9	Prima weich	W 345
1.1740	Prima 60	Prima, zäh	BH 40		B 12	Prima sehr zäh	W 360
1.1744	Prima 67		BH 44		B 14		
1.1750	Prima 75	Prima, mittelhart	BH 50		B 15	Prima zäh	W 375
1.1760	Prima 90		BH 60		B 19	Prima zähhart	W 390
1.1805	Extra SFB	Prima, f. Einsatzhärtung	BI 05			Dominial ES 2	W 315
1.1806			BI 06				
1.1820	Solitär	S Extra, schweißbar	BI 20		SC 10; SC 11	Dominial E 5	3 x Reiter
1.1822			BI 22		SC 16		
1.1830			BI 30				
1.1840			BI 40				

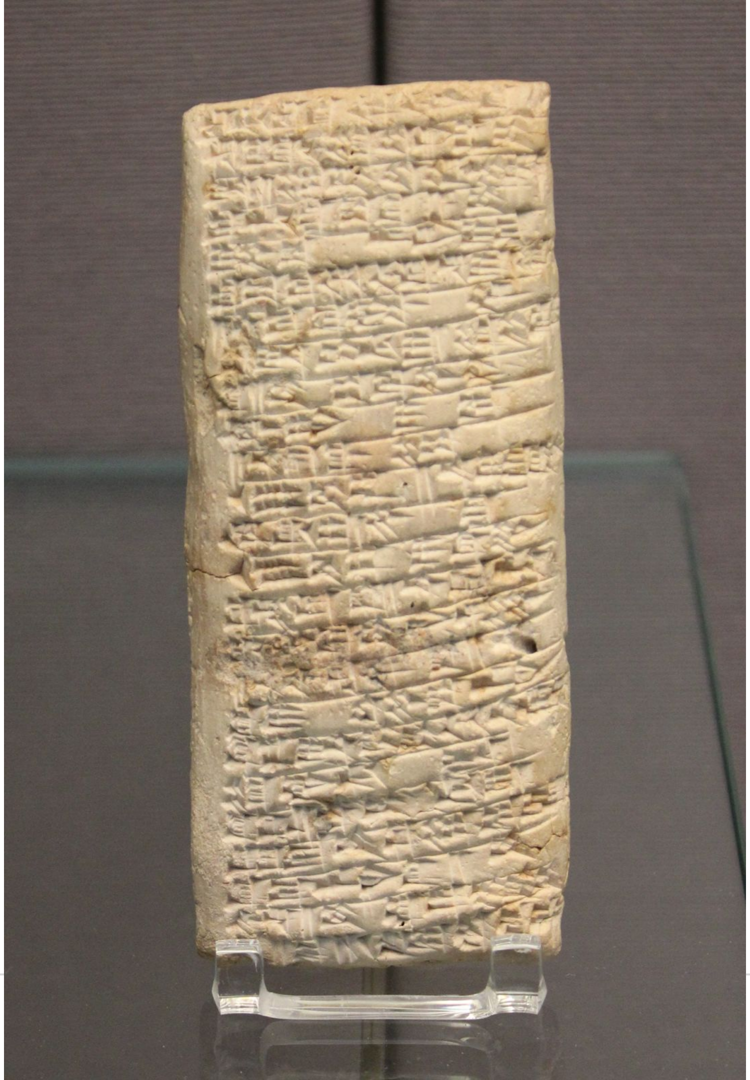
Schnellarbeitsstähle High speed steels

Stoff-Nr. Standard No.	Gebrüder Höver	Hoffmann & Co.	Horbach & Schmitz	Georg Hufnagel	IBACH Stahlwerke	Kind & Co.	Fritz Kölsche KG.
1.3202	Ilo 55 H	NORA REKORD 53 V	NIMBUS DC 02	Orkan	S 12.5 Co	VERONIKA Hochleistung Silber V 4	Rekord KO 5
1.3207	Ilo WCo 209	NORA REKORD SUPER	NIMBUS DC 07	Orkan-S		Hochleistung Super	Rekord KO 12
1.3211	Ilo 35 H	NORA REKORD 30	NIMBUS DC 11	Condor	S 12.3 Co	Hochleistung K 3	Rekord KO 3
1.3215			NIMBUS DC 15				
1.3243	Ilo Rekord MC 10	NORA REKORD 50 MO	NIMBUS DC 43			C 605 Co	Rekord KO 10
1.3245		NORA REKORD 30 EXTRA	NIMBUS DC 45				
1.3249			NIMBUS DC 49			B 88	
1.3251		NORA REKORD 58	NIMBUS DC 51			Hochleistung Silber A	Rekord KO 4
1.3255	Ilo BCO	NORA REKORD 53	NIMBUS DC 55	Vulkan	S 18.5 Co	Hochleistung Silber	Rekord KOV 20
1.3257		NORA REKORD 150	NIMBUS DC 57			Hochleistung Diamant	
1.3265	Ilo 105 H	NORA REKORD 110	NIMBUS DC 65	Puma	S 18.10 Co	Hochleistung Gold	Rekord KO 22
1.3302	Ilo Super B	NORA 80 EXTRA	NIMBUS DO 02	Geier	S 12.4 V	Hochleistung VS 10	Rekord Spezial
1.3316	Ilo ABC II	NORA 40	NIMBUS DO 16		S 8.1 V	Spezial	Spezial 100
1.3318	Ilo Extra	NORA 60	NIMBUS DO 18		S 12.2 V	Hochleistung 1000	Rekord 111
1.3333	Ilo Prima	NORA 33	NIMBUS DO 33		S 3.3 V	C 33	MO-Extra
1.3343	Ilo Rekord M 5	NORA 50	NIMBUS DO 43		S 6.6 Mo	C 65	Rekord MO 5
1.3344	Ilo Rekord MV 8	NORA VA	NIMBUS DO 44			C 653 Va	Rekord MOV 2
1.3346	Ilo Rekord M 9	NORA MO 9	NIMBUS DO 46			B 92	Rekord MO 9
1.3348			NIMBUS DO 48			B 94	Rekord MO 9 V
1.3355	Ilo Ultra W	NORA 80 SPEZIAL	NIMBUS DO 55		S 18.1 V	Extra	Rekord Spez. 18
1.3357			NIMBUS DO 57			Extra V 2	Rekord Spez. 20

85



Remarks on board Ship John Howland of New Bedford Cruising for whales on the Off Shore ground



“

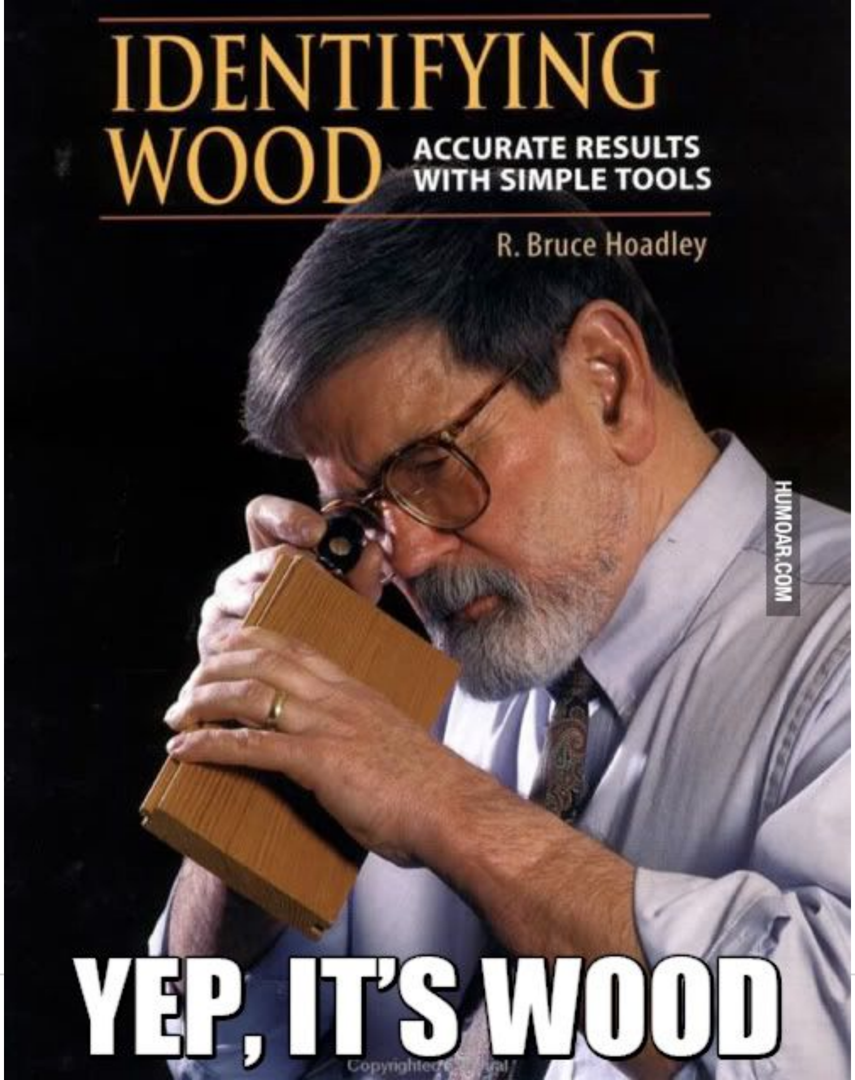
Humanity has optimized detailed accounts into key events into numbers for millenia, again and again

Observability & SRE

Or: Buzzwords, and their useful parts

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**

Services

- What's a 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

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

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

Concepts & guarantuees

- 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
- New release candidate every six weeks

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
```

PromQL

All partitions in my entire infrastructure with more than 100GB capacity that are not mounted on root?

```
node_filesystem_bytes_total{mountpoint!=”/”} / 1e9 > 100

 {device="sda1", mountpoint="/home”, instance=”10.0.0.1”} 118.8
 {device="sda1", mountpoint="/home”, instance=”10.0.0.2”} 118.8
 {device="sdb1", mountpoint="/data”, instance=”10.0.0.2”} 451.2
 {device="xdvc", mountpoint="/mnt”, instance=”10.0.0.3”} 320.0
```

PromQL

What’s the ratio of request errors across all service instances?

```
sum by(path) (rate(http_requests_total{status="500"}[5m])) /
 sum by(path) (rate(http_requests_total[5m]))

 {path="/status"} 0.0039
 {path="/"} 0.0011
 {path="/api/v1/topics/:topic"} 0.087
 {path="/api/v1/topics} 0.0342
```

New features

- Remote Write Receiver (v2.25 (feature flag) v2.33 (stable))
- Trigonometric functions (v2.31)
- Agent mode (v2.32)
- Long term support versions (v2.27)
- Out of order ingestion (v2.39)

Next highlight feature: Native histograms

Cloud native defaults

- 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
- Reliable into the tens of millions of active series

Mimir

Mimir

- For **M**etrics
- Prometheus -> Cortex -> Grafana Enterprise Metrics -> Mimir
- Scales to more than 1,000,000,000 Active Series
- Blazingly fast query performance
- Hard multi-tenancy, access control, and three-way replication
- Can ingest native OpenTelemetry, DataDog, Graphite, and Influx

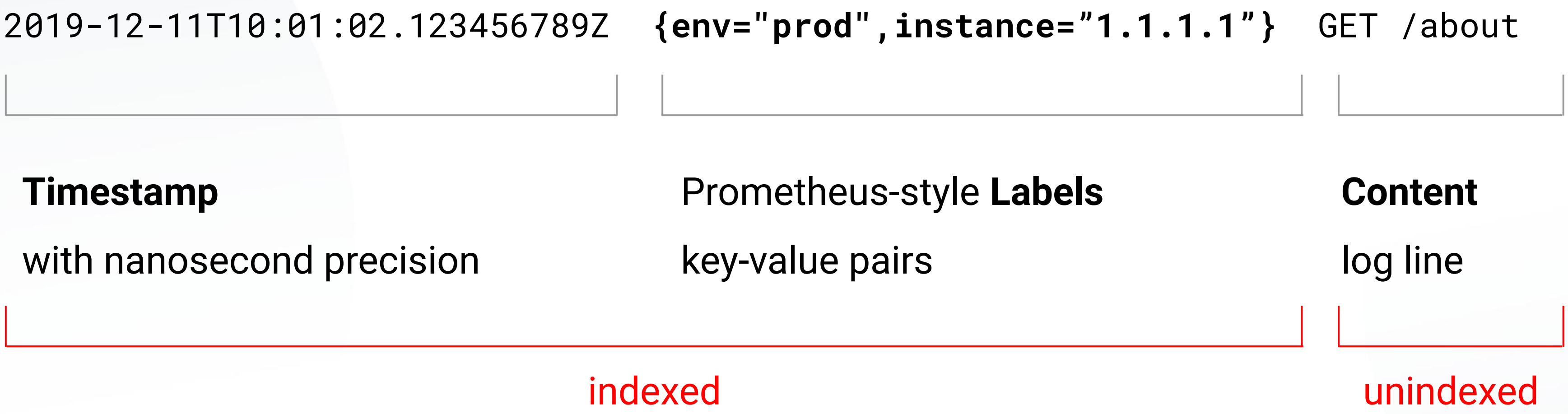
Mimir @ Grafana

- 1,000,000,000 Active Series - in one cluster
- 1,500 machines
- 7,000 CPU cores
- 30 TiB RAM

Loki

Loki 101

- For **L**ogs
- Following the same label-based system as Prometheus
  - Only index what you need often, query the rest
  - “Index the labels, query the data”
- Work with logs at scale, without the massive cost
  - Scalable low latency write path
  - Flexible schema on read
- Access logs with the same label sets as metrics
  - Turn logs into metrics, to make it easier & cheaper to work with them 📊



```
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

- Largest user cluster (as of 2022-09): 180 TiB per day
- Queries regularly see 80 GiB/s
- Query terabytes of data in under a minute
    - Including complex processing of result sets

Tempo

Tempo

- For **T**races
- 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 (2022-09)

- 2,200,000 samples per second @ 350 MiB/s
  - 5,000,000 samples second peak
- 14-day retention @ 3 copies stored
- Latencies:
  - p99 - 2.5s
  - p90 - 2.3s
  - p50 - 1.6s

Phlare

Phlare

- For **P**rofiling
- Pronounced “Flare”
- Profiles
  - “How much CPU & RAM am I spending in what areas of the code?”
  - “...and how does this change over time?”
- Go: pprof
- Java: https://github.com/grafana/JPProf

It’s a numbers game

Logs to metrics, the savings

- Full text indexing: 10 TiB logs -> ~20 TiB index
- Loki: 10 TiB logs -> ~200 MiB index

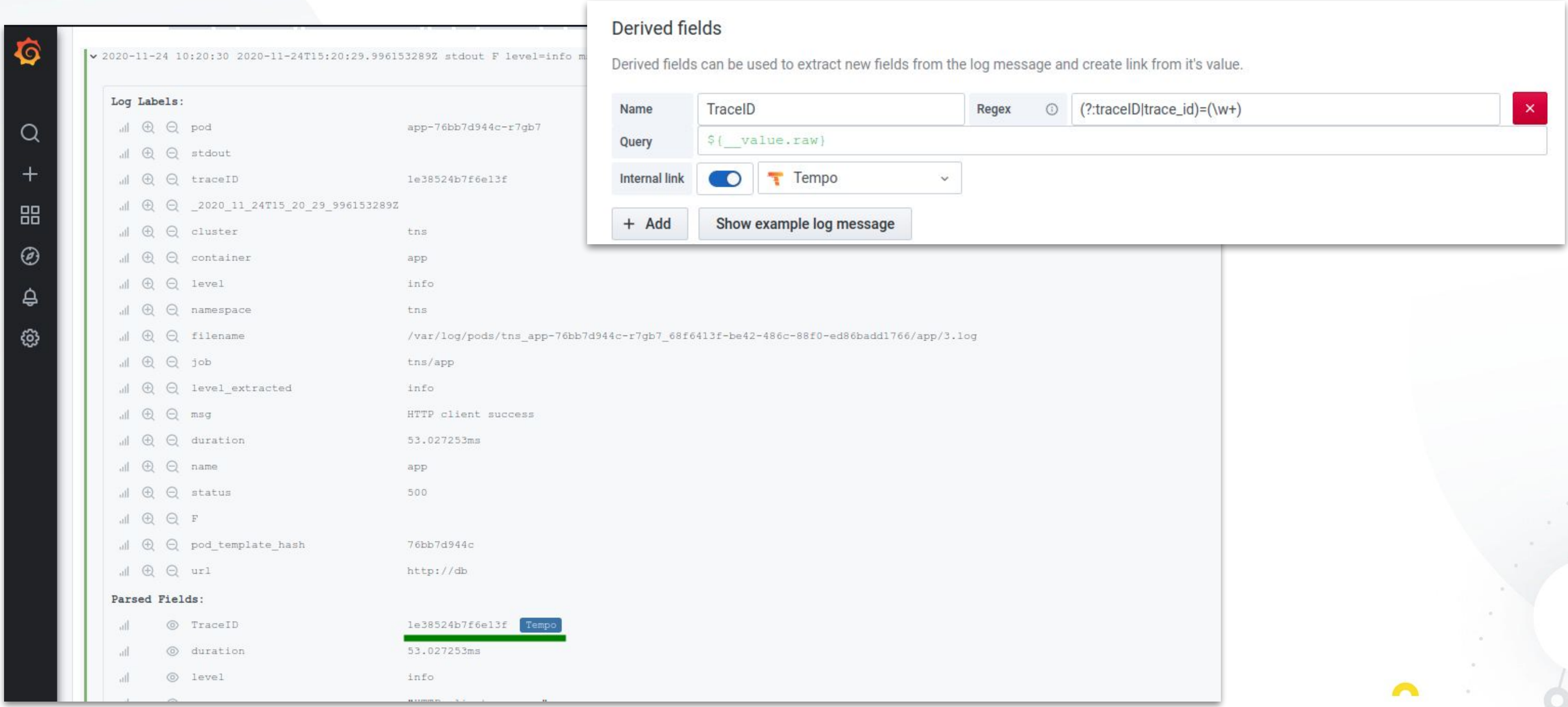
- Logs @ Grafana ~600 B average per line
- Metrics ~1.36 byte per metric sample

-> 99.8% reduction in storage size for first log line

~100% for every follow-up log line

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:

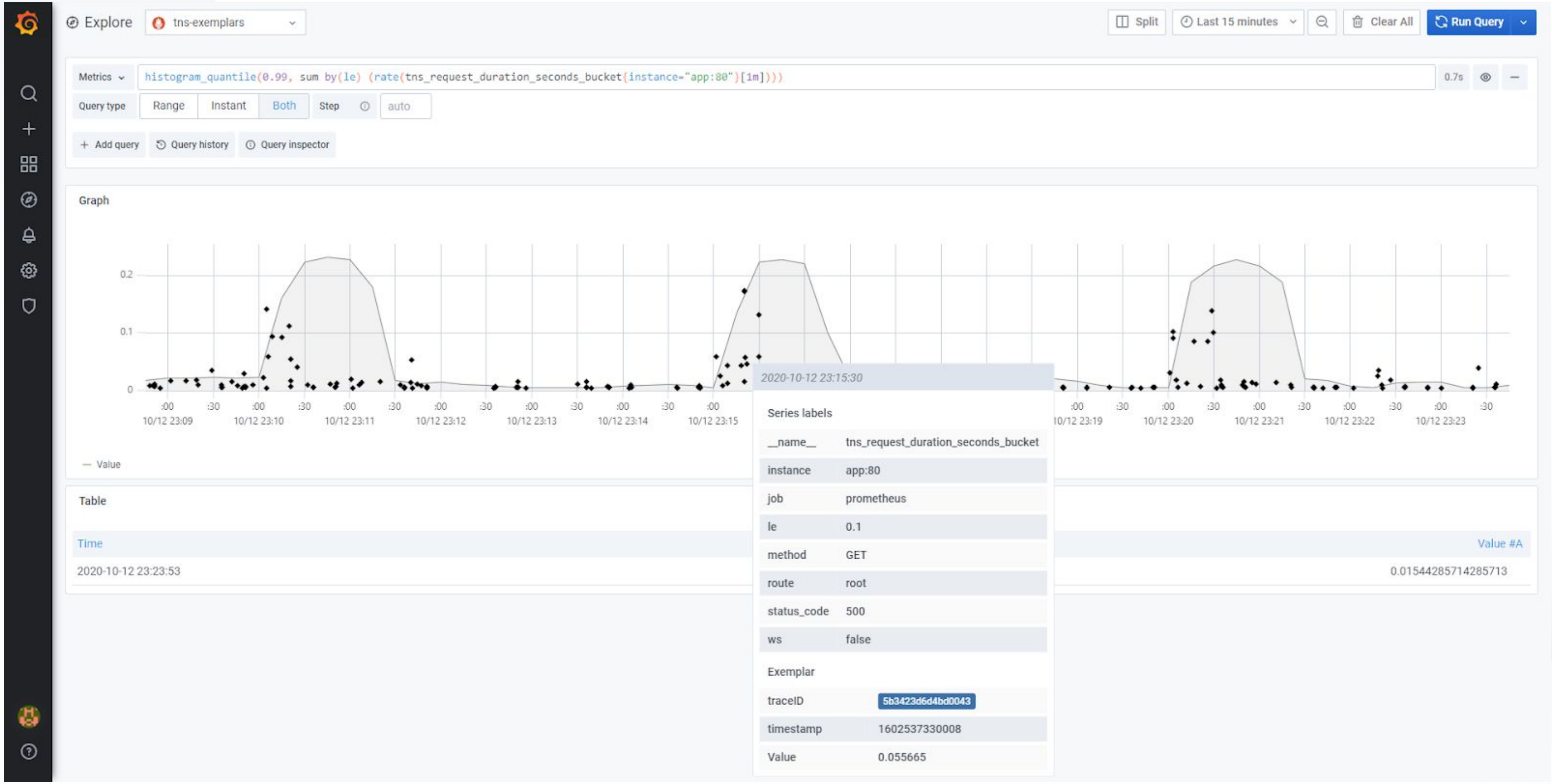
TraceID	1e38524b7f6e13f Tempo
duration	53.027253ms
level	info

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	●	Tempo	

From metrics to traces



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

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

2020-10-12 23:15:30

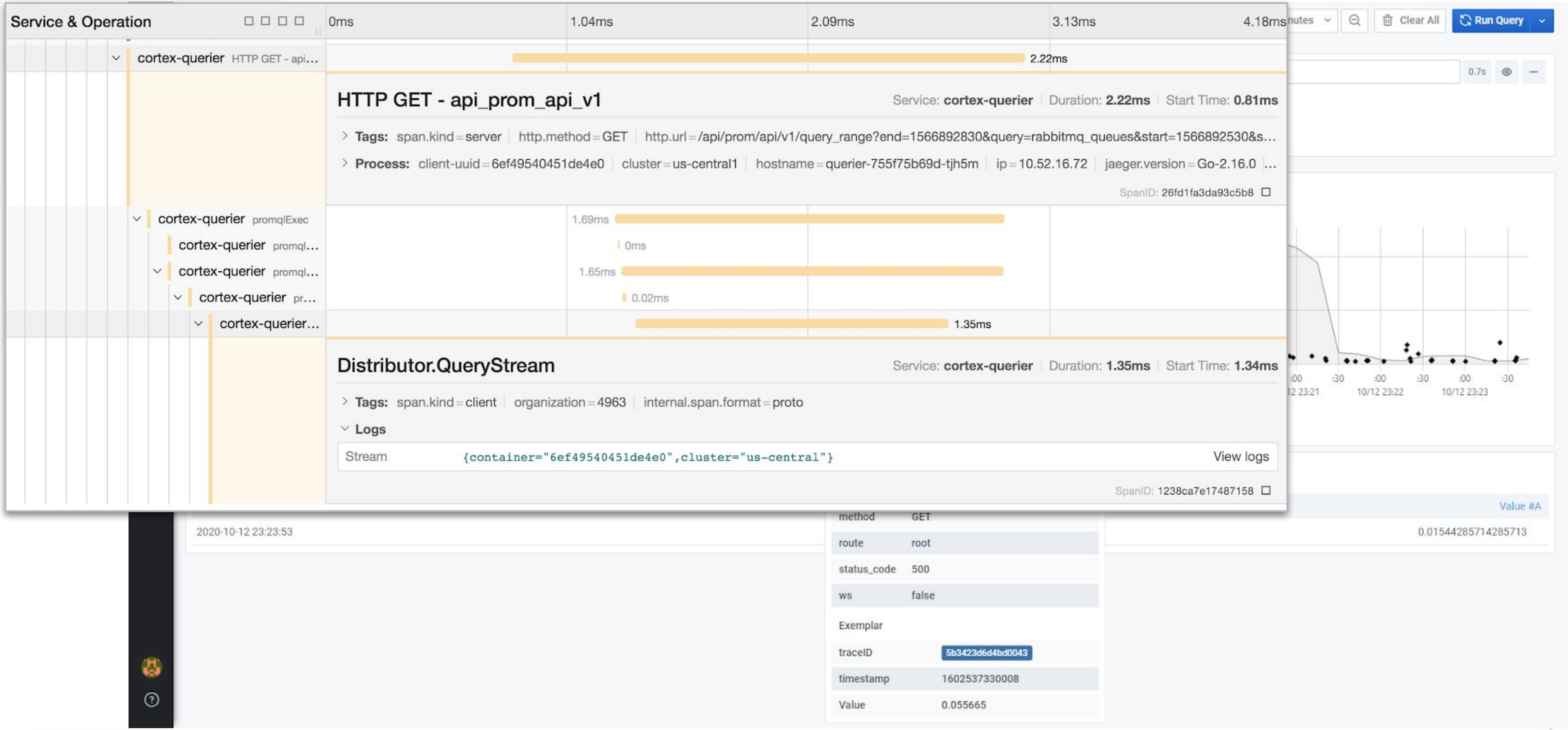
Series labels

__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



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

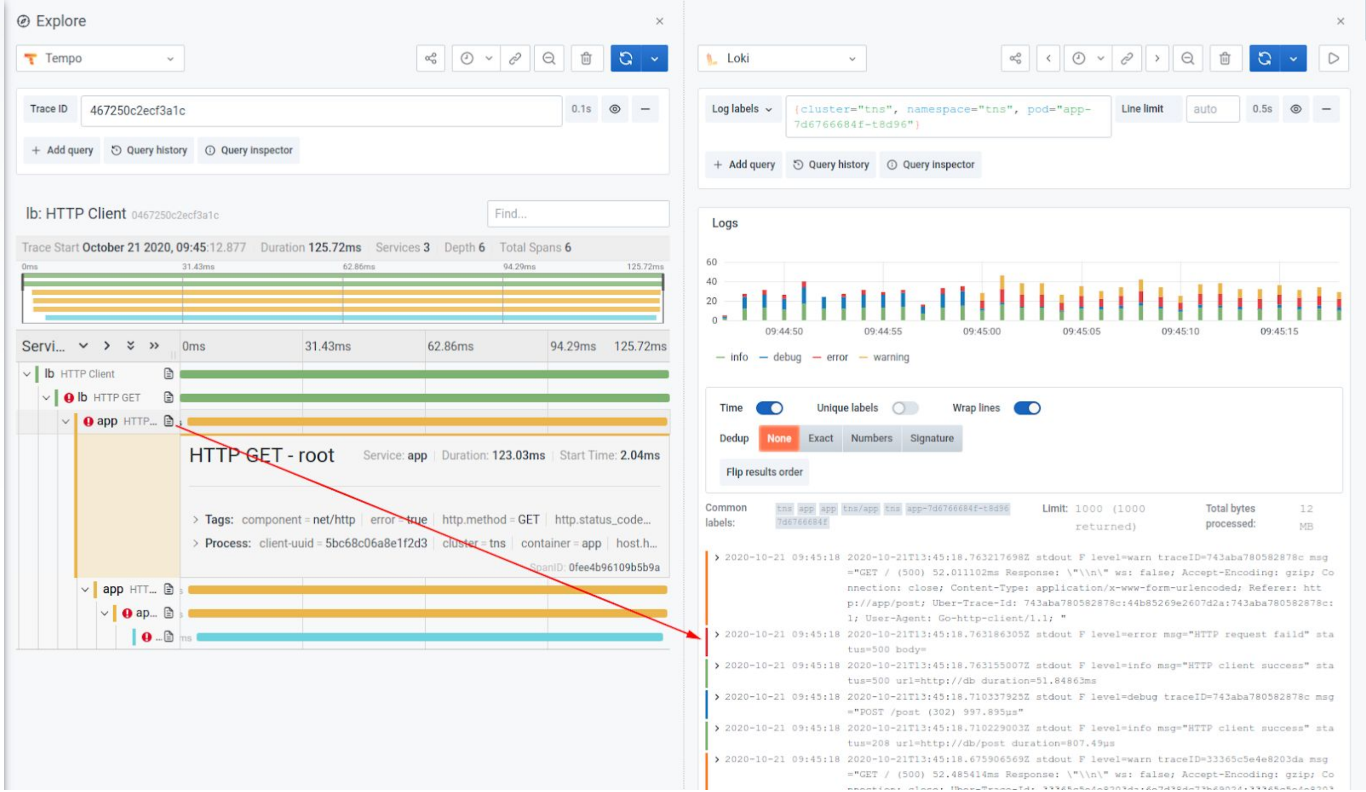
method	GET
route	root
status_code	500
ws	false

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**

- lb HTTP Client
  - lb HTTP GET
    - app HTTP...

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

- app HTT...
  - ap...
    - ...

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

Time ☑ Unique labels ☐ Wrap lines ☑

Dedup: [✓ None] Exact Numbers Signature

Common labels: tns app app tns/app tns app-7d6766684f-t8d96 7d6766684f

Limit: 1000 (1000 returned)

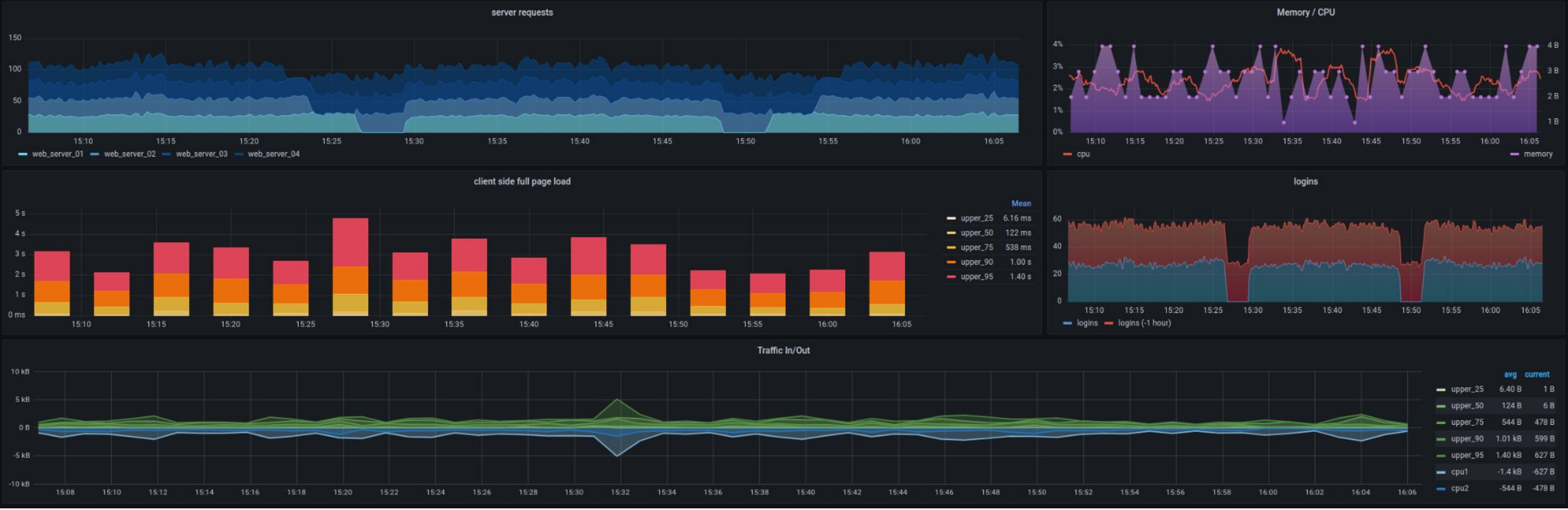
Total bytes processed: 12 MB

```
> 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
```

“

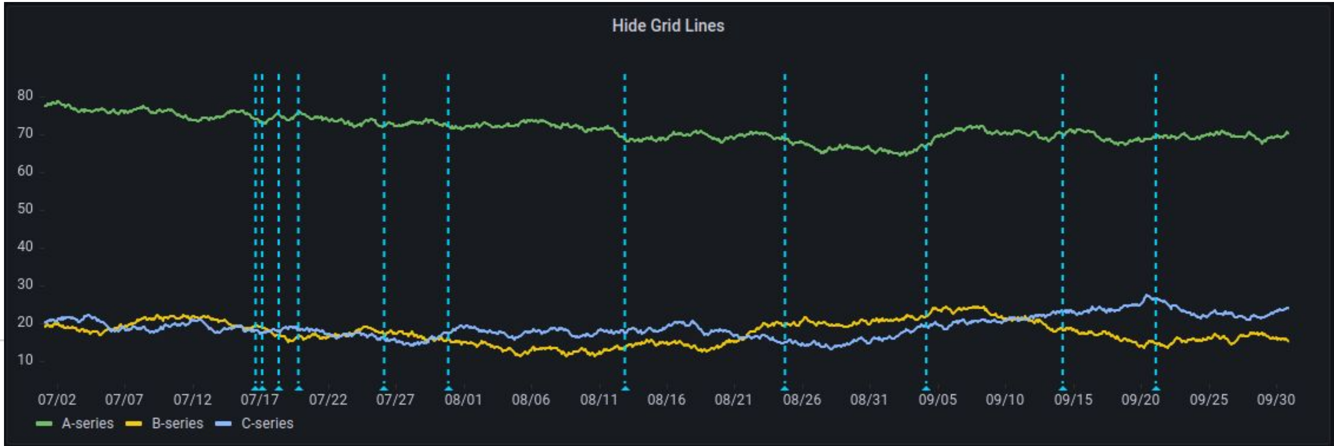
All of this is Open Source and you can run it yourself

(But we will also sell it to you happily)

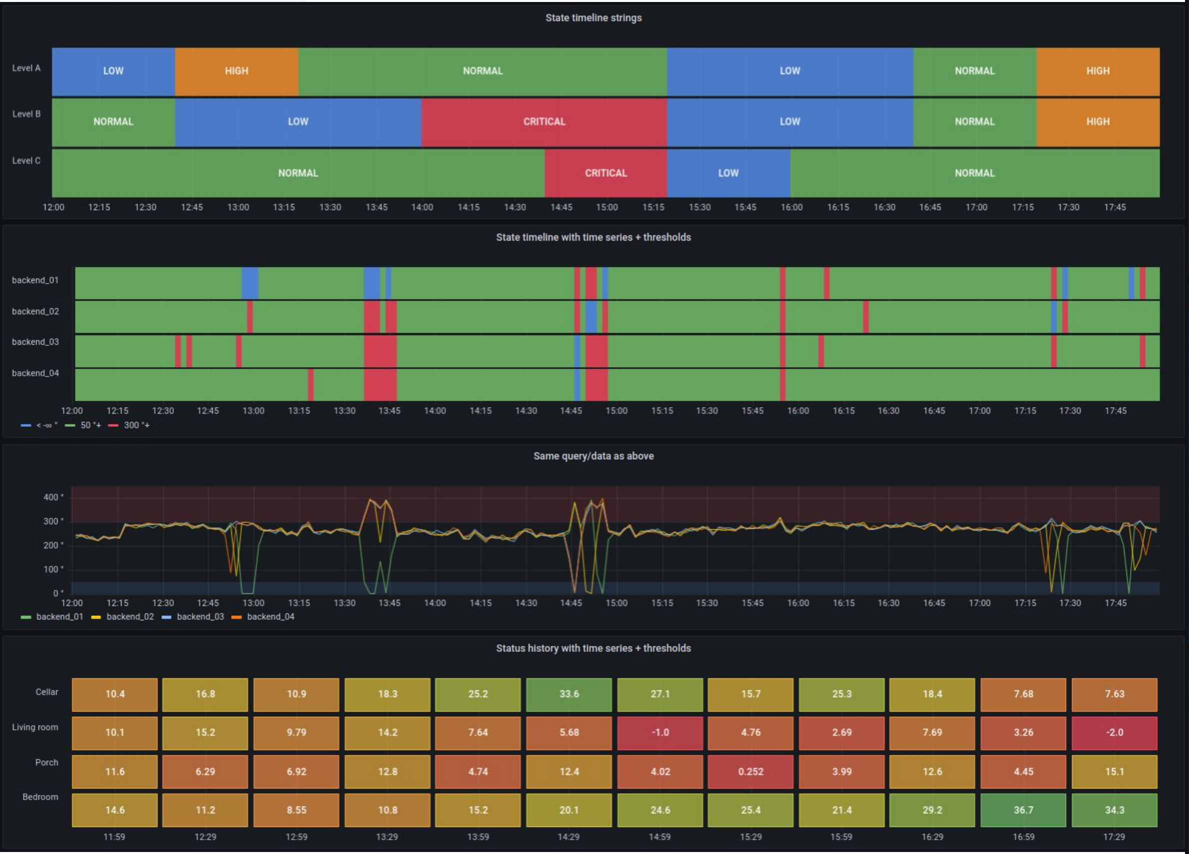


	Mean
upper_25	6.16 ms
upper_50	122 ms
upper_75	538 ms
upper_90	1.00 s
upper_95	1.40 s

	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

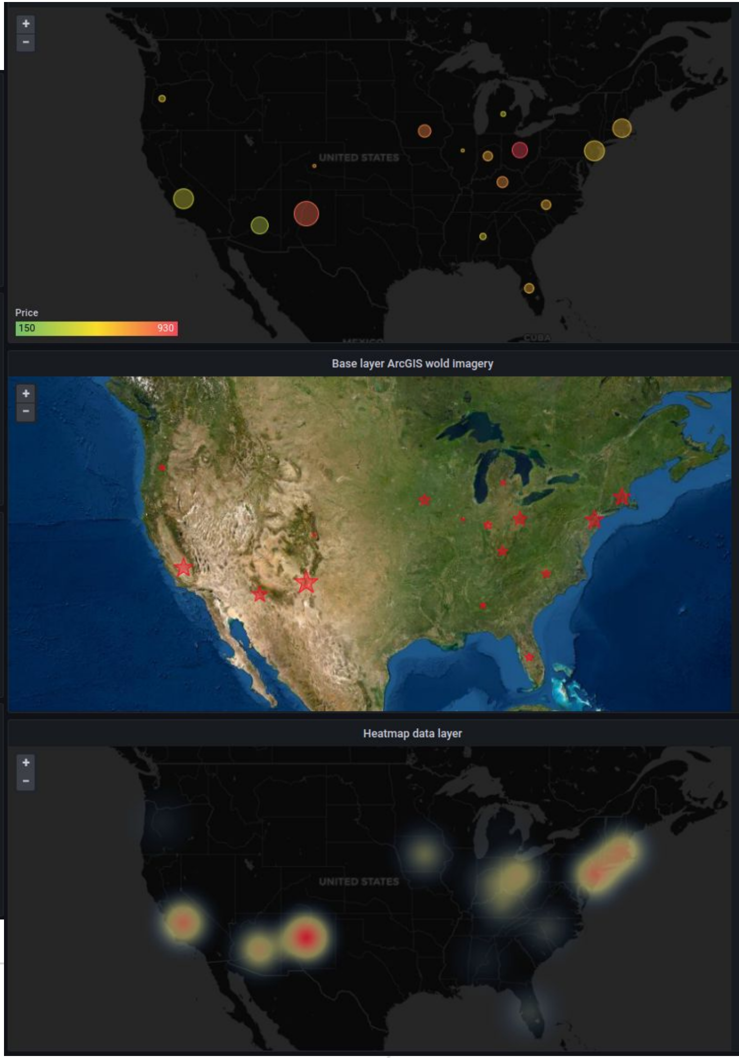


Grafana Labs



Status history with time series + thresholds

	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



Thank you!

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