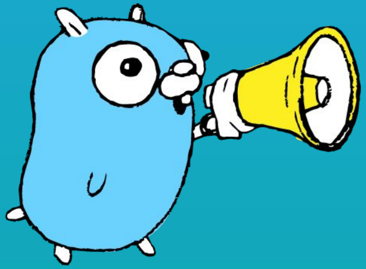
Helsinki, OCT 30 2019

Monitoring and log aggregation with Go tools

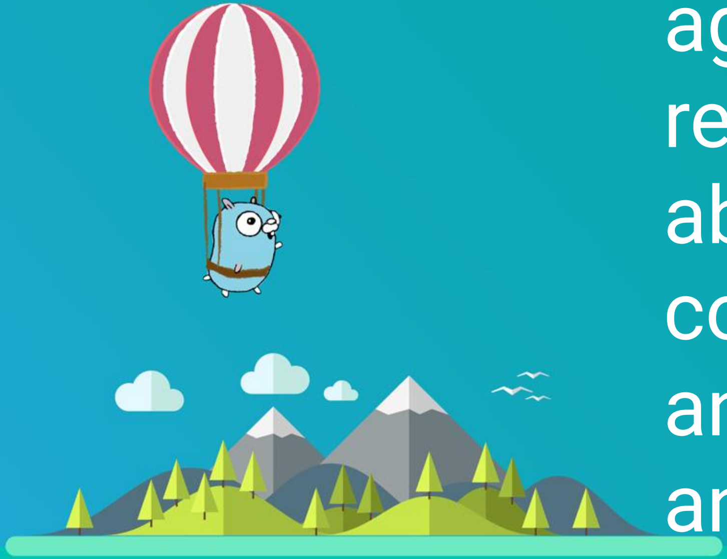
Valério Valério

valeriovalerio

@VDVsx



Monitoring



“Collecting, processing, aggregating, and displaying real-time quantitative data about a system, such as query counts and types, error counts and types, processing times, and server lifetimes.”

Site Reliability Engineering: How Google Runs Production Systems (O'Reilly Media)

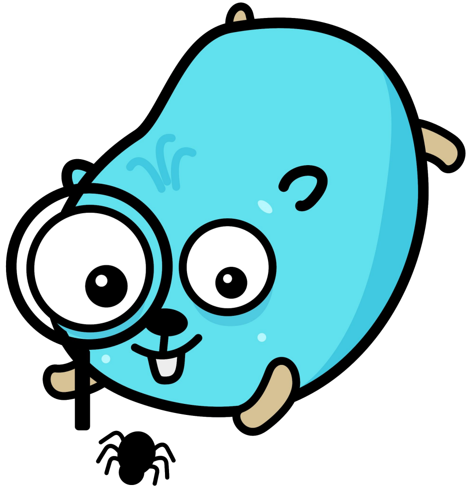
“

Go best practices, six years in - Peter Bourgon

You should be instrumenting every significant component of your codebase. If it’s a resource, like a queue, instrument it according to Brendan Gregg’s USE method: utilization, saturation, and error count (rate). If it’s something like an endpoint, instrument it according to Tom Wilkie’s RED method: request count (rate), error count (rate), and duration.

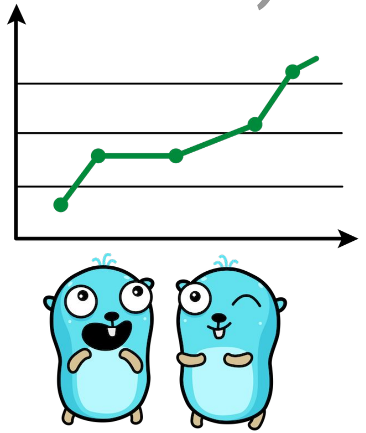
Why Monitor?

- Available/fast/correct/efficient
- Alerts
- Analyzing trends
- Comparing over time
- Ad hoc debugging/post-mortem



Types of monitoring

- Check-based monitoring (e.g nagios)
- Logs/Events (e.g elastic, influxDB)
- Request tracing (e.g Jaeger)
- Metrics/Time Series (e.g statsD, Prometheus)



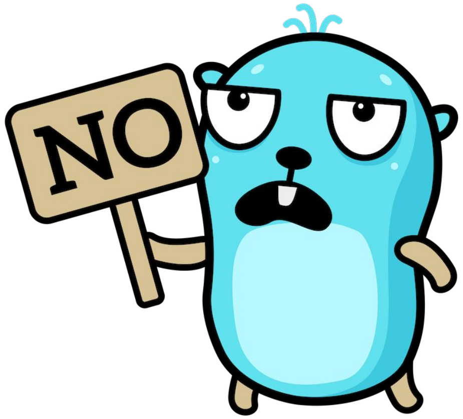
Prometheus

- Metrics-based monitoring and alerting stack
- Instrumentation
- Metrics collection and storage
- Querying, alerting, dashboarding
- For all levels of the stack

Source: [An introduction to Systems and Service Monitoring with Prometheus]()

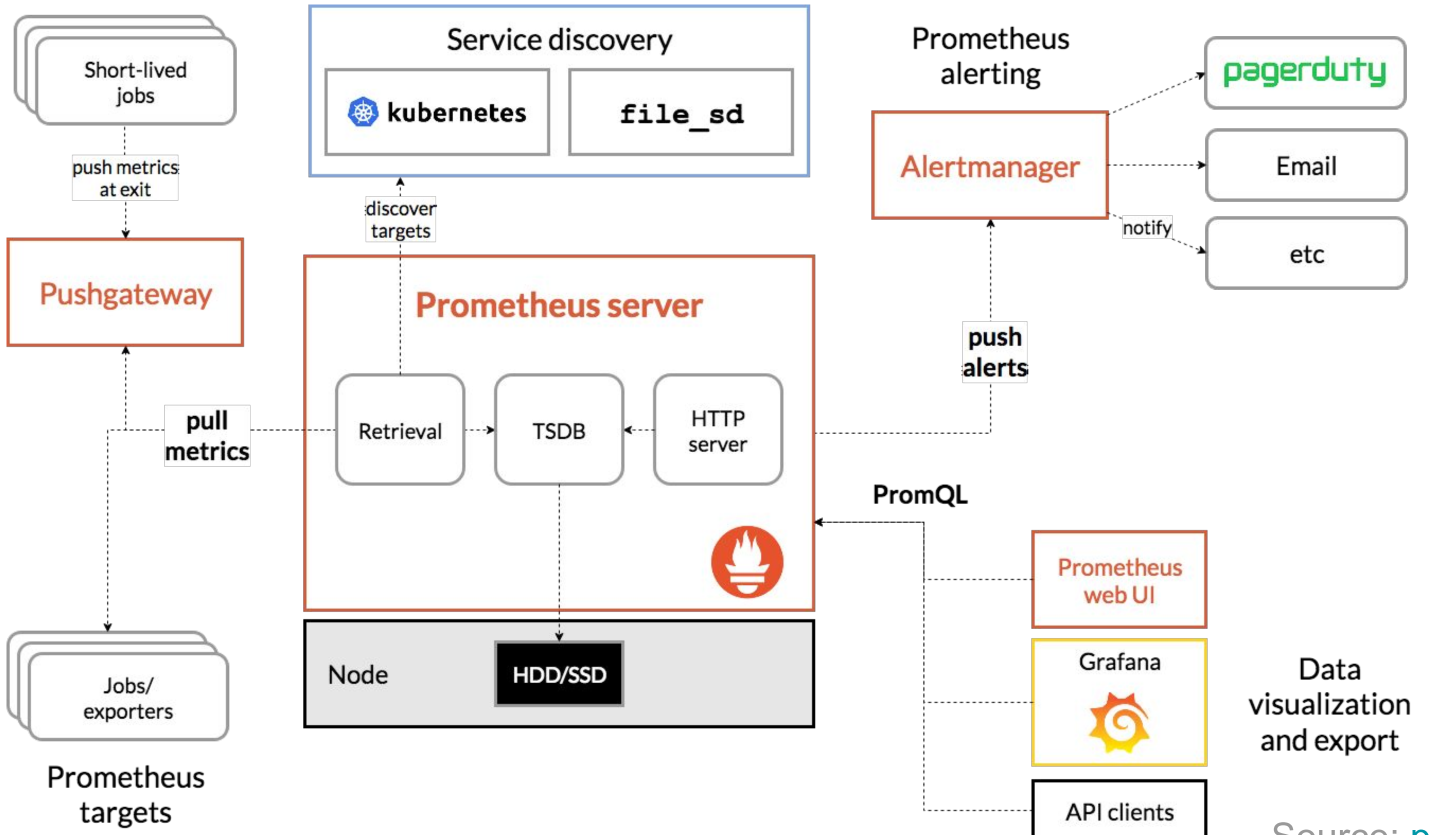
What Prometheus is not for?

- Logging or tracing
- Automatic anomaly detection
- Scalable or durable storage (Thanos/Cortex)



Source: An introduction to Systems and Service Monitoring with Prometheus

Prometheus architecture



Source: prometheus.io

Prometheus 💙 Go

```go
// instrumentCounter instruments the handler with a request counter
// grouped by method and status code
func instrumentCounter(endpoint string, hFunc http.HandlerFunc) http.HandlerFunc {
    return promhttp.InstrumentHandlerCounter(
        promauto.NewCounterVec(
            prometheus.CounterOpts{
                Name: fmt.Sprintf("%v_requests_total", endpoint),
                Help: "A counter for requests to the wrapped handler.",
            },
            []string{"code", "method"},
        ),
        hFunc,
    )
}
```

```go
mux := http.NewServeMux()
mux.Handle("/", instrumentCounter("root", myHandle))
mux.Handle("/metrics", promhttp.Handler())
log.Fatal(http.ListenAndServe(":8000", mux))
```

Logging

Logging?

“Logging is the process of cutting trees, processing them, and moving them to a location for transport.” - Wikipedia

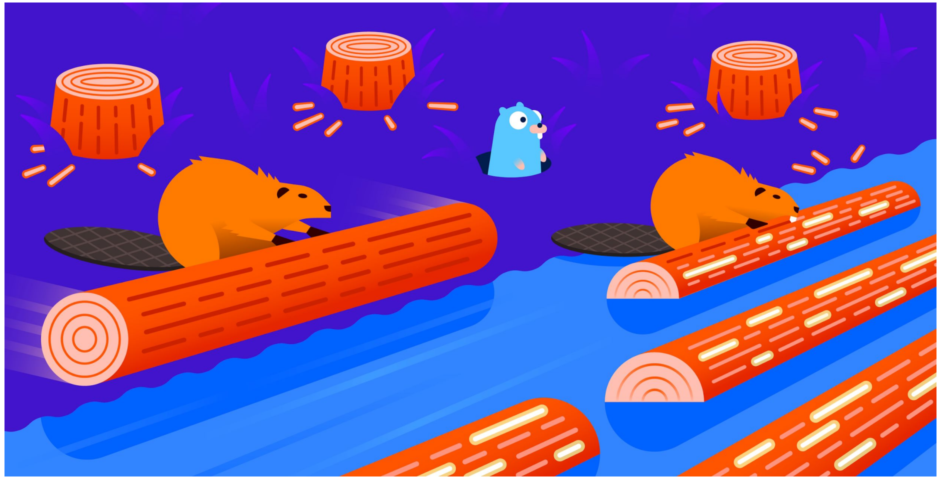


Image source: datadoghq.com

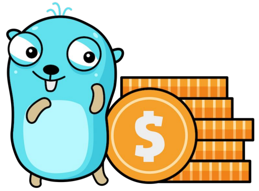
Logging?

“In computing, a log file is a file that records either events that occur in an operating system or other software runs, or messages between different users of a communication software. Logging is the act of keeping a log. In the simplest case, messages are written to a single log file.” - Wikipedia

What should I log?

- Actionable information - That's it!
- Avoid fine-grained log levels
- Use structured logging
- Log what makes sense for your use case

Logs are expensive!



[The Log: What every software engineer should know about real-time data's unifying abstraction - Jay Kreps]()

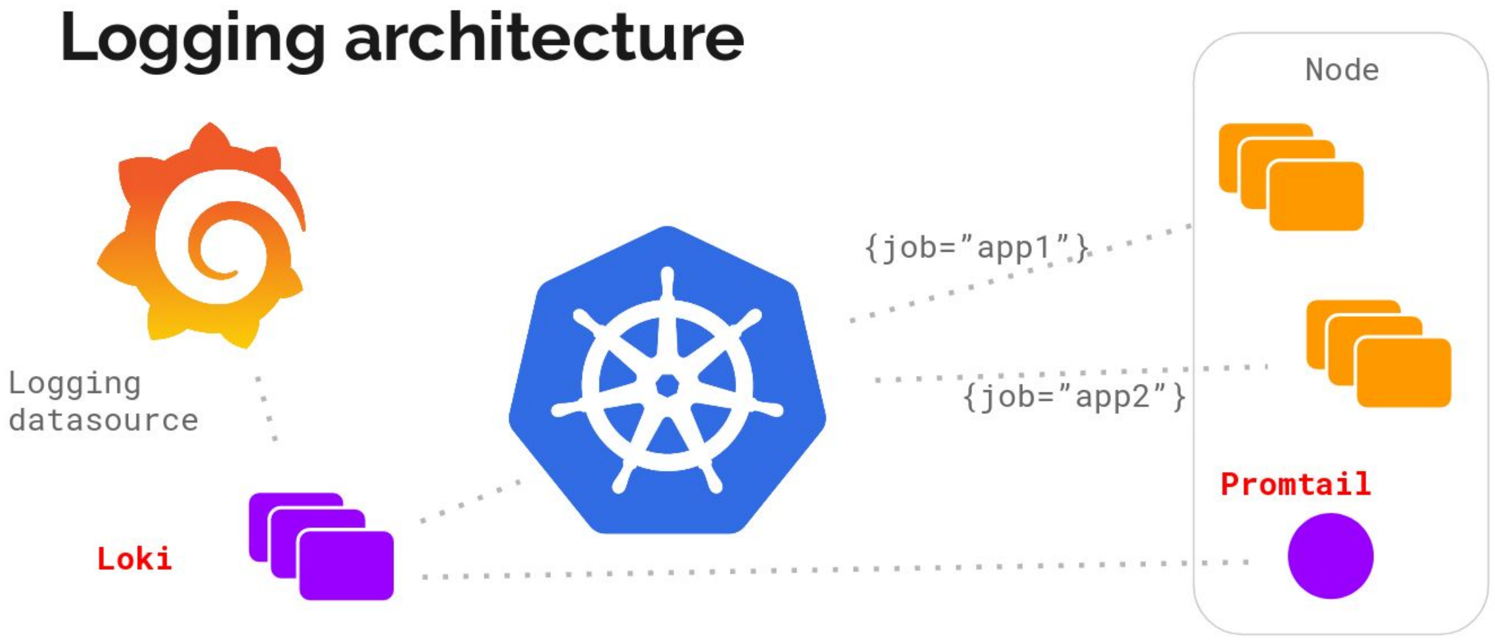
Grafana Loki: like Prometheus, but for logs

"Loki is a horizontally-scalable, highly-available, multi-tenant log aggregation system inspired by Prometheus. It is designed to be very cost effective and easy to operate. It does not index the contents of the logs, but rather a set of labels for each log stream." - Loki website

Loki: Comparing to other log aggregation systems

- Does not do full text indexing on logs. By storing compressed, unstructured logs and only indexing metadata, Loki is simpler to operate and cheaper to run.
- Indexes and groups log streams using the same labels you’re already using with Prometheus.
- Is an especially good fit for storing Kubernetes Pod logs. Metadata such as Pod labels is automatically scraped and indexed.
- Not a replacement for ELK stack BI features
- Still in beta

Loki Architecture



Source: Loki github

Loki query language: LogQL

Can be considered a distributed **grep** with labels for filtering

```
{type="db",name="db-primary"}

=: exactly equal.
!=: not equal.
=~: regex matches.
!~: regex does not match.

Example:

{name=~"db.+"}
{name!~"db.+"}
```

Log Stream selector

Loki query language: LogQL

Search expressions can be just text of regex

```
{job="worker"} |= "error"


|=: Log line contains string.
!=: Log line does not contain string.
|~: Log line matches regular expression.
!~: Log line does not match regular
expression.


// Chained, will satisfy every filter
{job="mysql"} |= "error" != "timeout"
```

Filter expression

Loki query language: LogQL

Same as PromQL but for logs

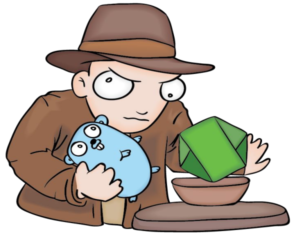
```
// number of log lines in last 5m
count_over_time({job="worker"}[5m])

// per second rate in the last 10s
rate(({job="worker"} |= "error"[10s]))

// Aggregation
sum, min, max, avg, count
stddev,stdvar
bottomk,topk

// rate of HTTP GET requests from NGINX logs
avg(rate(({job="nginx"} |= "GET")[10s])) by (region)

// top 10 applications by the highest log throughput
topk(10,sum(rate({region="us-east1"}[5m]) by (name))
```

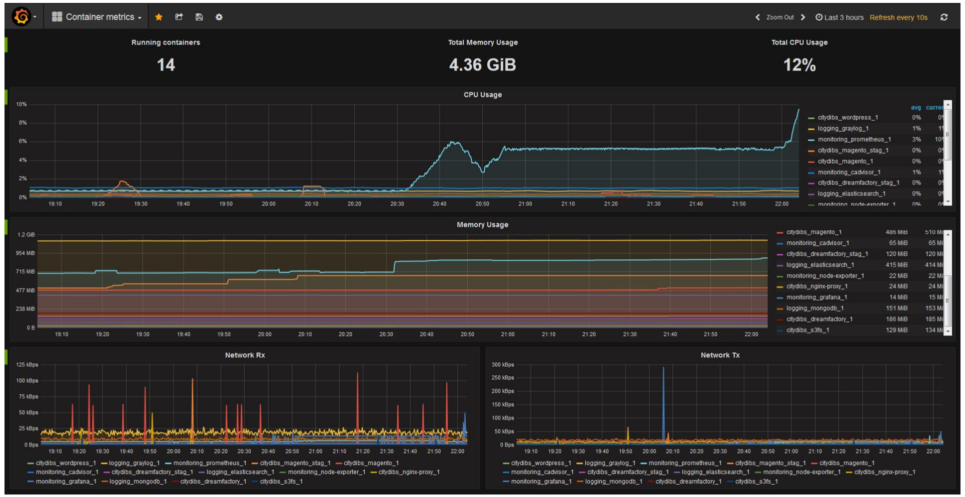
Counting/
Aggregating
logs

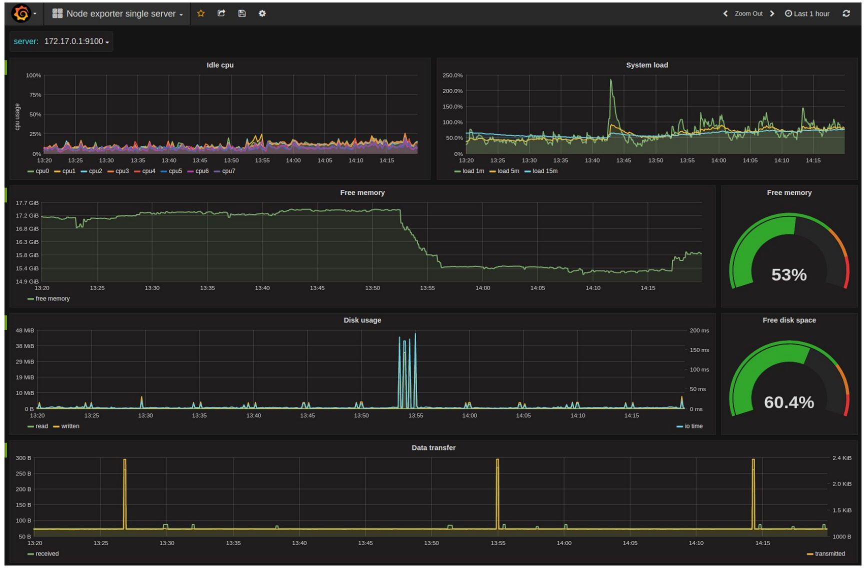
Legacy systems

Can I use this tools for legacy systems?

- Fluentd/fluentbit
- Node exporter
- Cadvisor

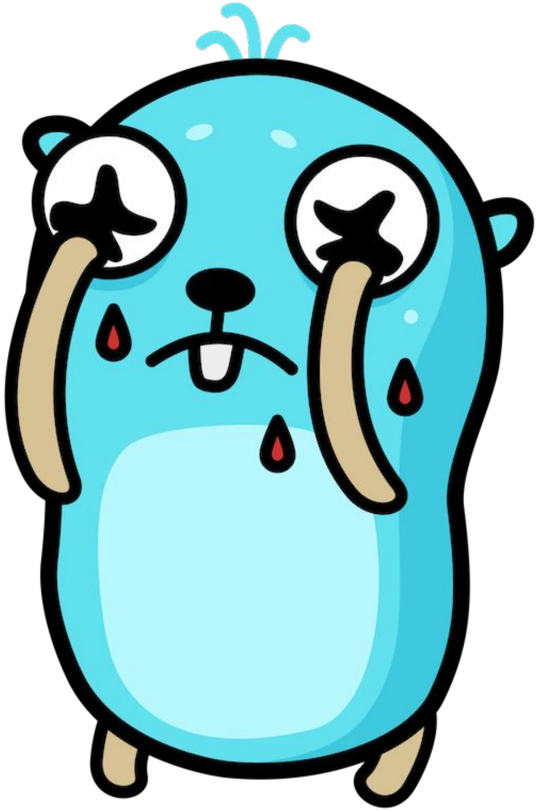






Demo

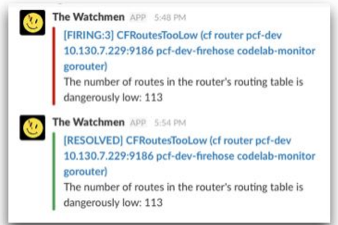
Demo gods....
Please!



Future

Explorer workflow

1. Alert



The Watchmen APP 5:48 PM

[FIRING:3] CFRoutesTooLow (cf router pcf-dev 10.130.7.229:9186 pcf-dev-firehose codelab-monitor gorouter)

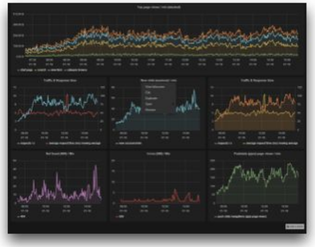
The number of routes in the router's routing table is dangerously low: 113

The Watchmen APP 5:54 PM

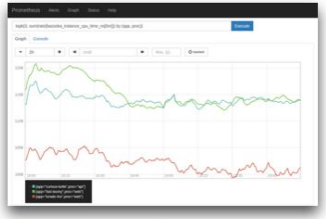
[RESOLVED] CFRoutesTooLow (cf router pcf-dev 10.130.7.229:9186 pcf-dev-firehose codelab-monitor gorouter)

The number of routes in the router's routing table is dangerously low: 113

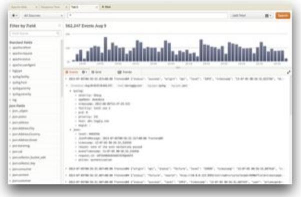
2. Dashboard



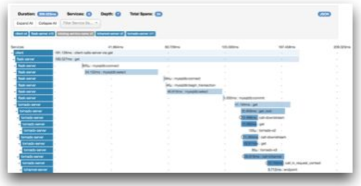
3. Adhoc Query



4. Log Aggregation



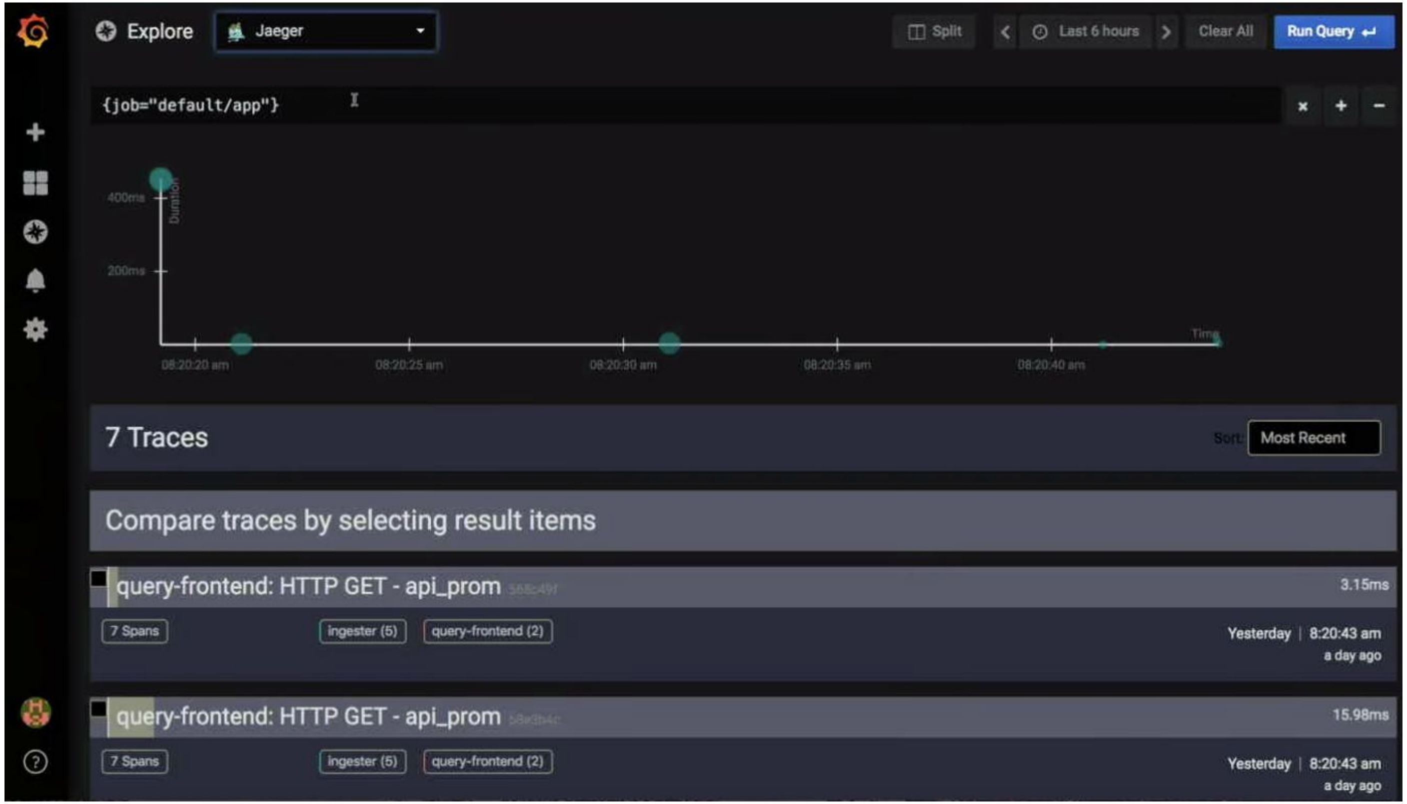
5. Distributed Tracing



Fix!

Source: Grafana.com

Explorer workflow future



```
{job="default/app"}
```

7 Traces

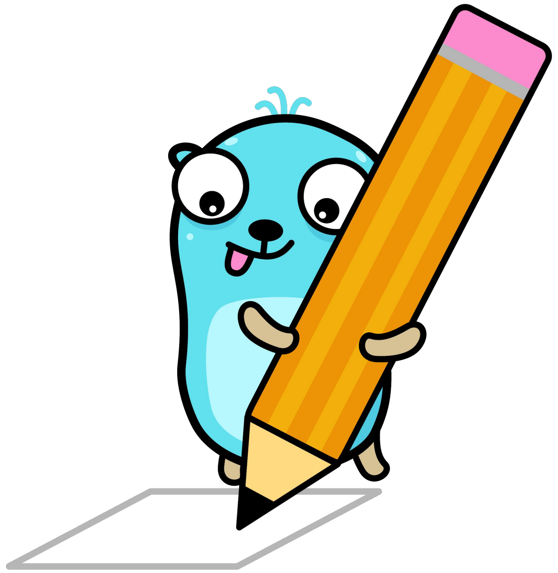
Compare traces by selecting result items

query-frontend: HTTP GET - api_prom — 3.15ms
7 Spans | ingester (5) | query-frontend (2) | Yesterday | 8:20:43 am, a day ago

query-frontend: HTTP GET - api_prom — 15.98ms
7 Spans | ingester (5) | query-frontend (2) | Yesterday | 8:20:43 am, a day ago

Takeaways

- Instrument your code
- Log wisely
- Metrics are cheaper than logs
- Consider OSS tools



Resources

- An introduction to Systems and Service Monitoring with Prometheus
- Site Reliability Engineering: How Google Runs Production Systems(free)
- Go best practices, six years in - Peter Bourgon
- Logging v. instrumentation - Peter Bourgon
- The Log: What every software engineer should know about real-time data's unifying abstraction
- The Explore Workflow and Troubleshooting with Loki (video)
- How to Export Prometheus Metrics from Just About Anything (video)
- Loki docs (github)

Artwork

- Go presentation theme
- Free Gophers pack by Maria Letta
- Gopher stickers by Takuya Ueda

Questions?

https://www.linkedin.com/in/valeriovalerio/

https://twitter.com/vdvsx