Inclusive Monitoring with Rancher and Prometheus

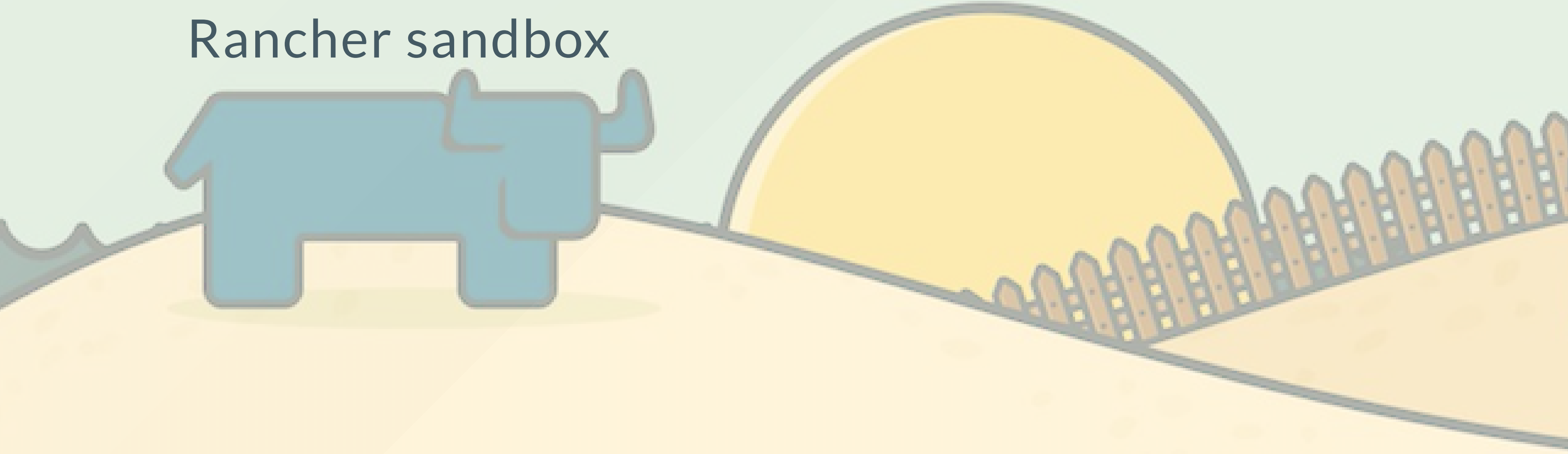
Sydney Rancher Meetup June 2017

Martin Baillie (@martinbaillie / IAG)

But first...

Tonight's Demo Environment:

- **RancherOS**: Fast, ultra-lightweight container OS
- **GCP**: 3 Sydney zones as of last week.. $400 credit!
- **try.rancher.com**: Join hosts to your own free Rancher sandbox



"Inclusive Monitoring"?

Monitoring ALL the things

Inclusive Monitoring

(I've seen this also called "whitebox monitoring")

Is about not just monitoring at the edge:
- CPU, Memory, Threads, Swap, Net, `containerd`

But also instrumenting the code **within**.

Both technology metrics 😎
- success rate, latency, saturation, pool size, db calls

And equally important... **business** metrics! 😱
- *e.g. insurance context:* self-service logins, policies bought, quotes made, claims lodged, refunds given

Meaning...

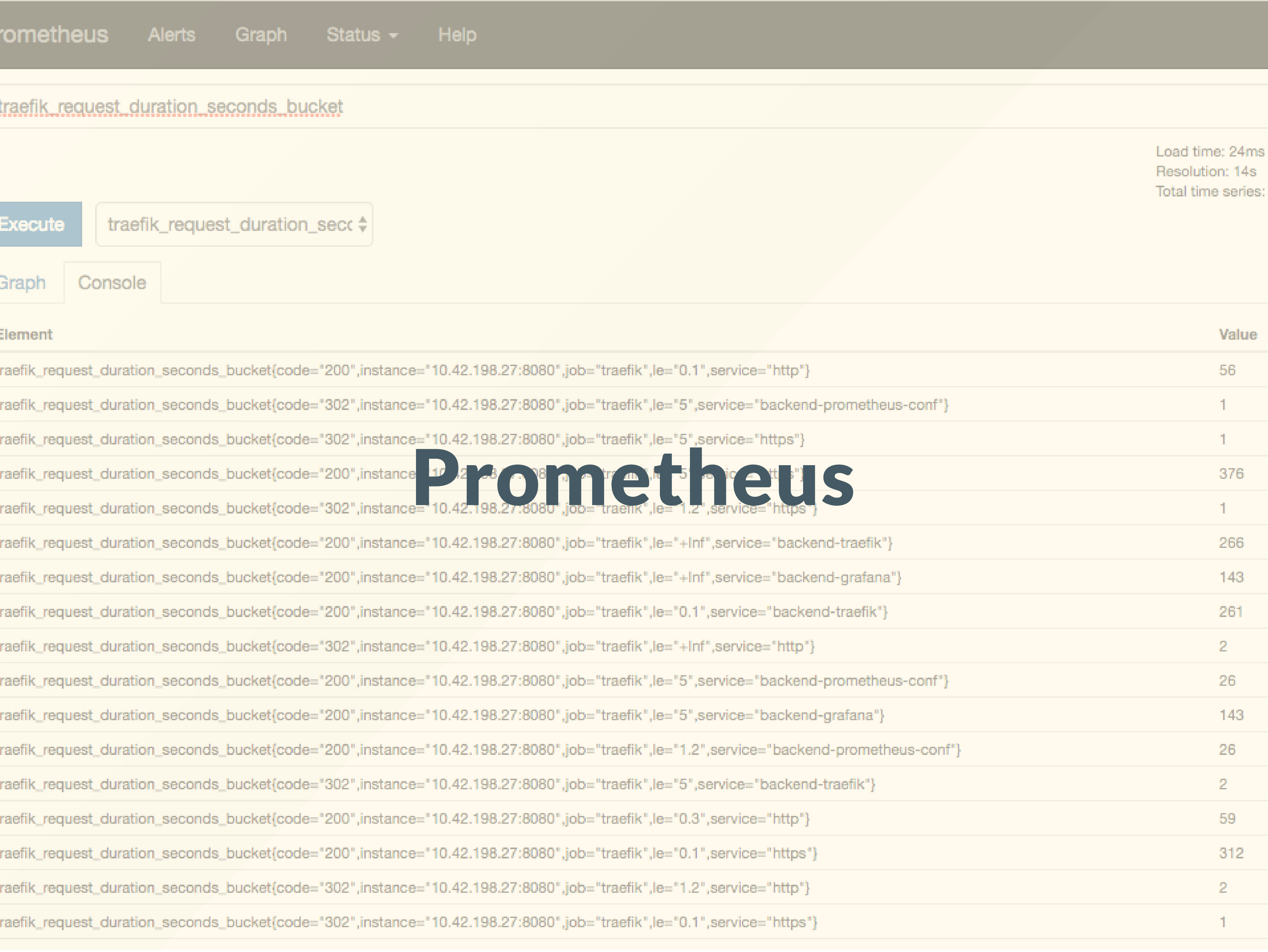
Metric instrumentation needs to become a core part of your engineering culture

Rancher and the Prometheus ecosystem can help with that

The demo will show these tools:

- Allowing developers to ship metrics, alerts, and dashboards alongside their code artefacts
- Having them auto-discovered (zero conf!)
- Achieving automatic monitoring of infrastructure, UIs and a microservice architecture as it changes
- Stored as code, shippable to multiple environments immutably

Prometheus



traefik_request_duration_seconds_bucket

Load time: 24ms
Resolution: 14s
Total time series:

traefik_request_duration_secc

Element	Value
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="0.1",service="http"}	56
raefik_request_duration_seconds_bucket{code="302",instance="10.42.198.27:8080",job="traefik",le="5",service="backend-prometheus-conf"}	1
raefik_request_duration_seconds_bucket{code="302",instance="10.42.198.27:8080",job="traefik",le="5",service="https"}	1
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="5",service="https"}	376
raefik_request_duration_seconds_bucket{code="302",instance="10.42.198.27:8080",job="traefik",le="1.2",service="https"}	1
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="+Inf",service="backend-traefik"}	266
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="+Inf",service="backend-grafana"}	143
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="0.1",service="backend-traefik"}	261
raefik_request_duration_seconds_bucket{code="302",instance="10.42.198.27:8080",job="traefik",le="+Inf",service="http"}	2
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="5",service="backend-prometheus-conf"}	26
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="5",service="backend-grafana"}	143
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="1.2",service="backend-prometheus-conf"}	26
raefik_request_duration_seconds_bucket{code="302",instance="10.42.198.27:8080",job="traefik",le="5",service="backend-traefik"}	2
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="0.3",service="http"}	59
raefik_request_duration_seconds_bucket{code="200",instance="10.42.198.27:8080",job="traefik",le="0.1",service="https"}	312
raefik_request_duration_seconds_bucket{code="302",instance="10.42.198.27:8080",job="traefik",le="1.2",service="http"}	2
raefik_request_duration_seconds_bucket{code="302",instance="10.42.198.27:8080",job="traefik",le="0.1",service="https"}	1

Prometheus

Is a monitoring [eco]system and time-series database

- Originally written by ex-Googlers @ Soundcloud
- Inspired by Google's Borgmon monitoring system

> “ *Even though Borgmon remains internal to Google, the idea of treating time-series data as a data source for generating alerts is now accessible to everyone*[SRE book on Prometheus] ”

- Prometheus is to Borgmon what Kubernetes is to Borg...*I guess*

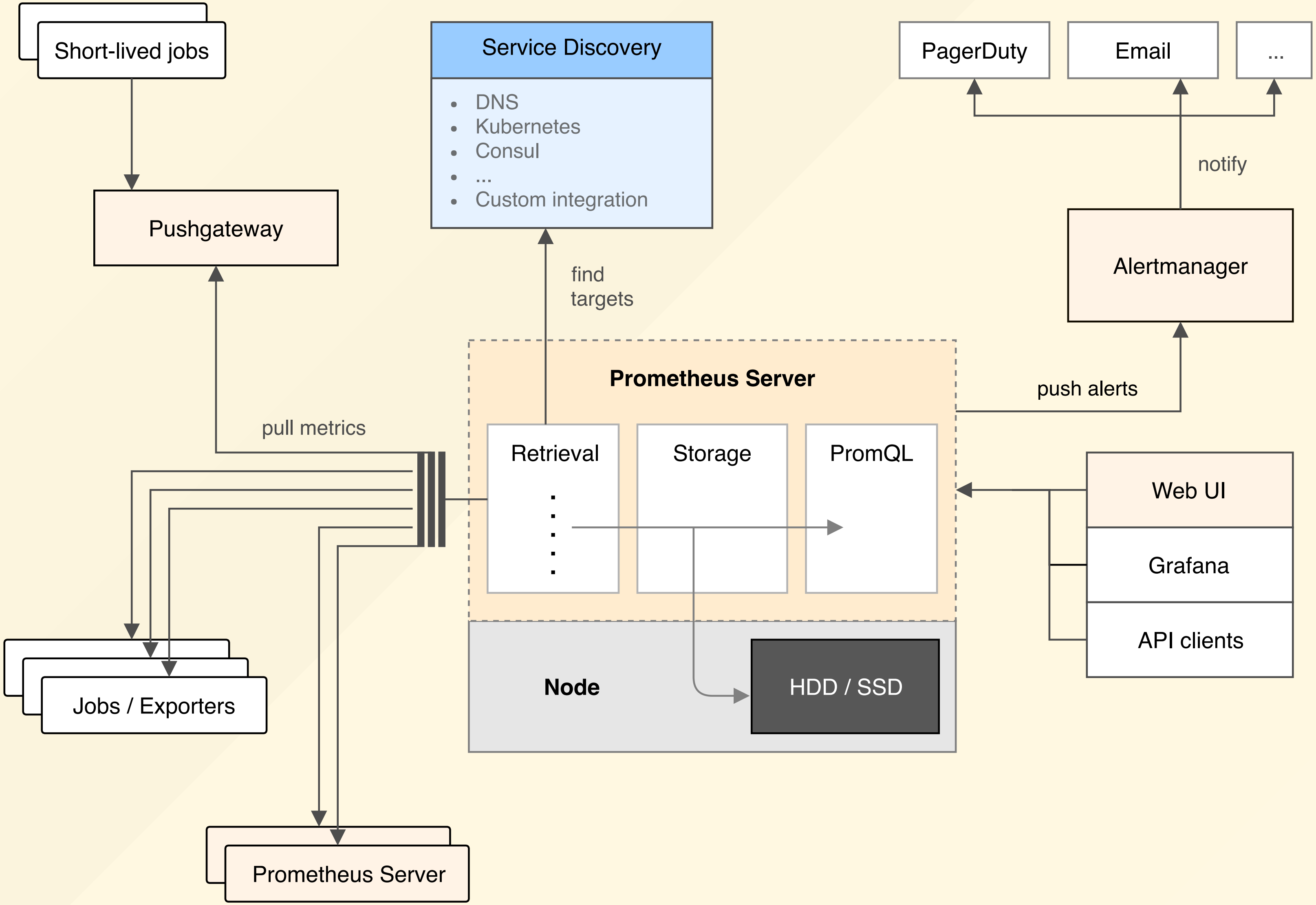
Prometheus

- A community OSS project (no single company)
  - With clear goals
  - Measured acceptance of PRs
  - And a careful eye on potential scope creep
- Second accepted project to the CNCF (after K8s)
- Enterprise support by RobustPerception.io
- Written (mostly) in Golang
  - One of the most well-architected Go codebases I've studied </opinion>

Key Features

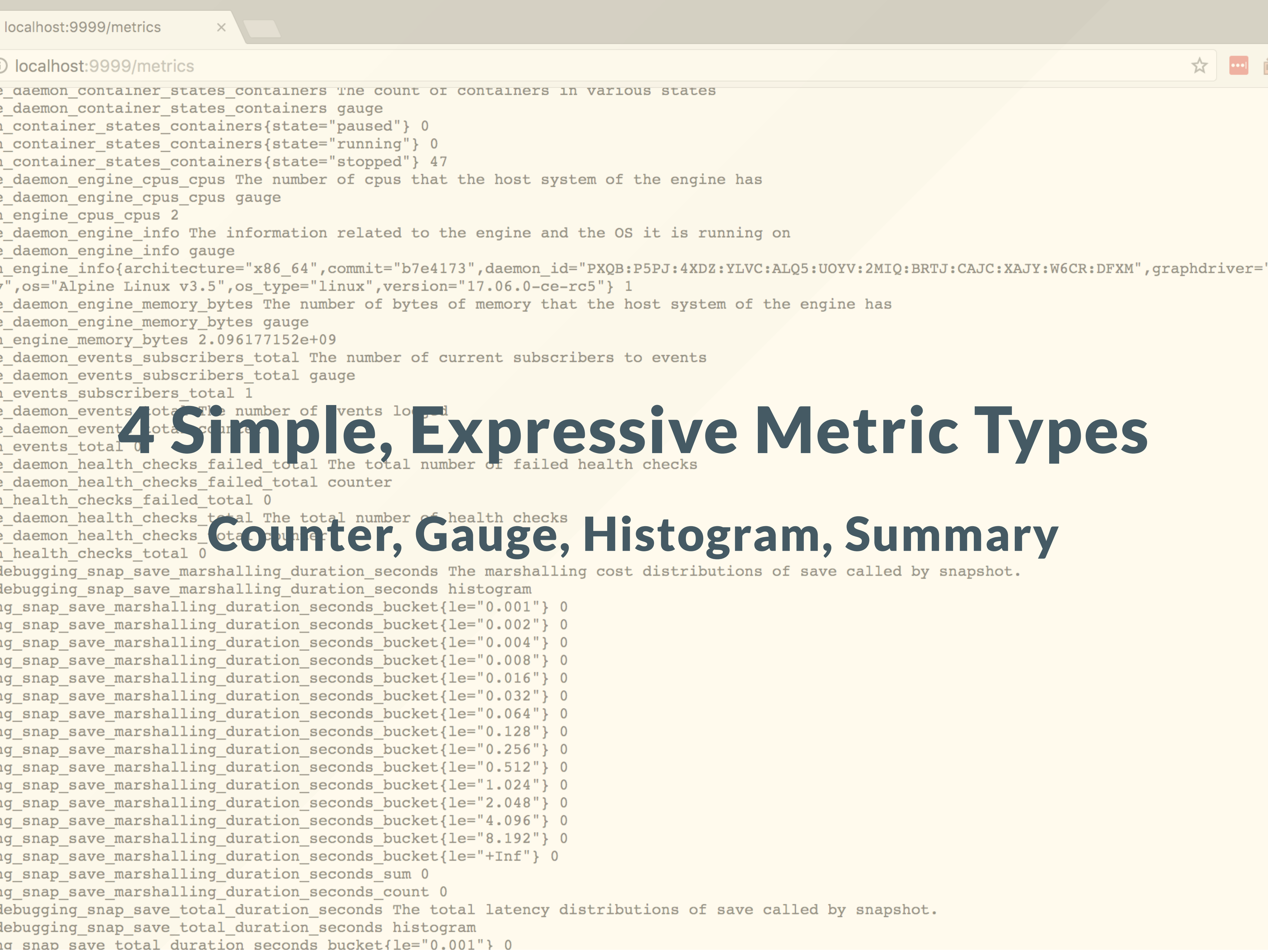
- A powerful query language (Turing complete! 😱)
- Efficient storage and dimensional data model
- Scalable telemetry (pull-based) monitoring
- Metric instrumenting libraries in many languages
- Tons of pre-canned exporters for existing systems
- Industry-leading visualisation by way of Grafana
- Alerting with many integrations via Alertmanager
- Simple APIs, easy deployment (static Golang binaries, Docker) and all configuration as code

Pull-based Architecture



4 Simple, Expressive Metric Types

Counter, Gauge, Histogram, Summary



```
e_daemon_container_states_containers The count of containers in various states
e_daemon_container_states_containers gauge
n_container_states_containers{state="paused"} 0
n_container_states_containers{state="running"} 0
n_container_states_containers{state="stopped"} 47
e_daemon_engine_cpus_cpus The number of cpus that the host system of the engine has
e_daemon_engine_cpus_cpus gauge
n_engine_cpus_cpus 2
e_daemon_engine_info The information related to the engine and the OS it is running on
e_daemon_engine_info gauge
n_engine_info{architecture="x86_64",commit="b7e4173",daemon_id="PXQB:P5PJ:4XDZ:YLVC:ALQ5:UOYV:2MIQ:BRTJ:CAJC:XAJY:W6CR:DFXM",graphdriver="
y",os="Alpine Linux v3.5",os_type="linux",version="17.06.0-ce-rc5"} 1
e_daemon_engine_memory_bytes The number of bytes of memory that the host system of the engine has
e_daemon_engine_memory_bytes gauge
n_engine_memory_bytes 2.096177152e+09
e_daemon_events_subscribers_total The number of current subscribers to events
e_daemon_events_subscribers_total gauge
n_events_subscribers_total 1
e_daemon_events_total The number of events logged
e_daemon_events_total counter
n_events_total 0
e_daemon_health_checks_failed_total The total number of failed health checks
e_daemon_health_checks_failed_total counter
n_health_checks_failed_total 0
e_daemon_health_checks_total The total number of health checks
e_daemon_health_checks_total counter
n_health_checks_total 0
debugging_snap_save_marshalling_duration_seconds The marshalling cost distributions of save called by snapshot.
debugging_snap_save_marshalling_duration_seconds histogram
ng_snap_save_marshalling_duration_seconds_bucket{le="0.001"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.002"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.004"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.008"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.016"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.032"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.064"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.128"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.256"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="0.512"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="1.024"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="2.048"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="4.096"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="8.192"} 0
ng_snap_save_marshalling_duration_seconds_bucket{le="+Inf"} 0
ng_snap_save_marshalling_duration_seconds_sum 0
ng_snap_save_marshalling_duration_seconds_count 0
debugging_snap_save_total_duration_seconds The total latency distributions of save called by snapshot.
debugging_snap_save_total_duration_seconds histogram
ng_snap_save_total_duration_seconds_bucket{le="0.001"} 0
```

As an aside: Metric != Log

Metrics are not a panacea. You will need multiple **complementary** tools for successful debugging.

Metrics	cheap, low cardinality	store lots
Logs	expensive, high cardinality	store few

Metrics for *which service* in a distributed system issue is. Log for digging deeper e.g. *which request*.

Also, **Metric != Trace**
You will still likely need distributed tracing in your microservice architecture (see OpenTracing, Zipkin)

Metric Exporters and Client Libraries *(not exhaustive)*

- Server, SNMP, Dovecot, Kubernetes, Rancher, Mesos, Graphite, StatsD, Collectd, Expvar, JMX, Spring, uWSGI, Cloudflare, AWS, VMWare, Solr, Apache, Traefik HAProxy, Nginx, CouchDB, ElasticSearch, MongoDB, MySQL, Oracle, Redis, Memcached, OpenTSDB, RabbitMQ, IBM MQ, Kafka, Ceph, GlusterFS, Docker, Jenkins...
- Go, Java, Scala, Python, Ruby, Bash, C++, Common Lisp, Elixir, Erlang, Lua, .NET, Node.js, PHP, Rust...

Metric Instrumentation

Example: time taken to service a HTTP request?

Golang

```go
var requestDuration = prometheus.NewSummaryVec(
prometheus.SummaryOpts{
        Name: "request_duration_seconds",
        Help: "Request duration in seconds",
}, []string{})

func my_handler(w http.ResponseWriter, r *http.Request) {
        defer func(begin time.Time) {
                requestDuration.With(nil).Observe(
                time.Since(begin).Seconds())
        }(time.Now())
        // Your code here
```

Even less LOC in other langs

Python Decorators

```
REQUEST_DURATION = Summary('request_duration_seconds',
'Request duration in seconds')

@REQUEST_DURATION.time()
def my_handler(request):
    pass # Your code here
```

Java Annotations

```
@RequestMapping
@PrometheusTimeMethod(name = "request_duration_seconds",
help="Request duration in seconds")
public myHandler() { // Your code here
```

Eggs In One Basket

Or: ***How I don't like hedging my bets in this industry***

1. Just like how using Rancher as my container management does not preclude me from using:
   - Kubernetes, Mesos, Swarm as my orchestrator
2. Or how annotating my microservice code with OpenTracing does not preclude me from using:
   - Zipkin, AppDash, Jaegar as my tracer

Prometheus libraries are open too! Instrument code using them; export to Graphite, Collectd, Nagios etc.

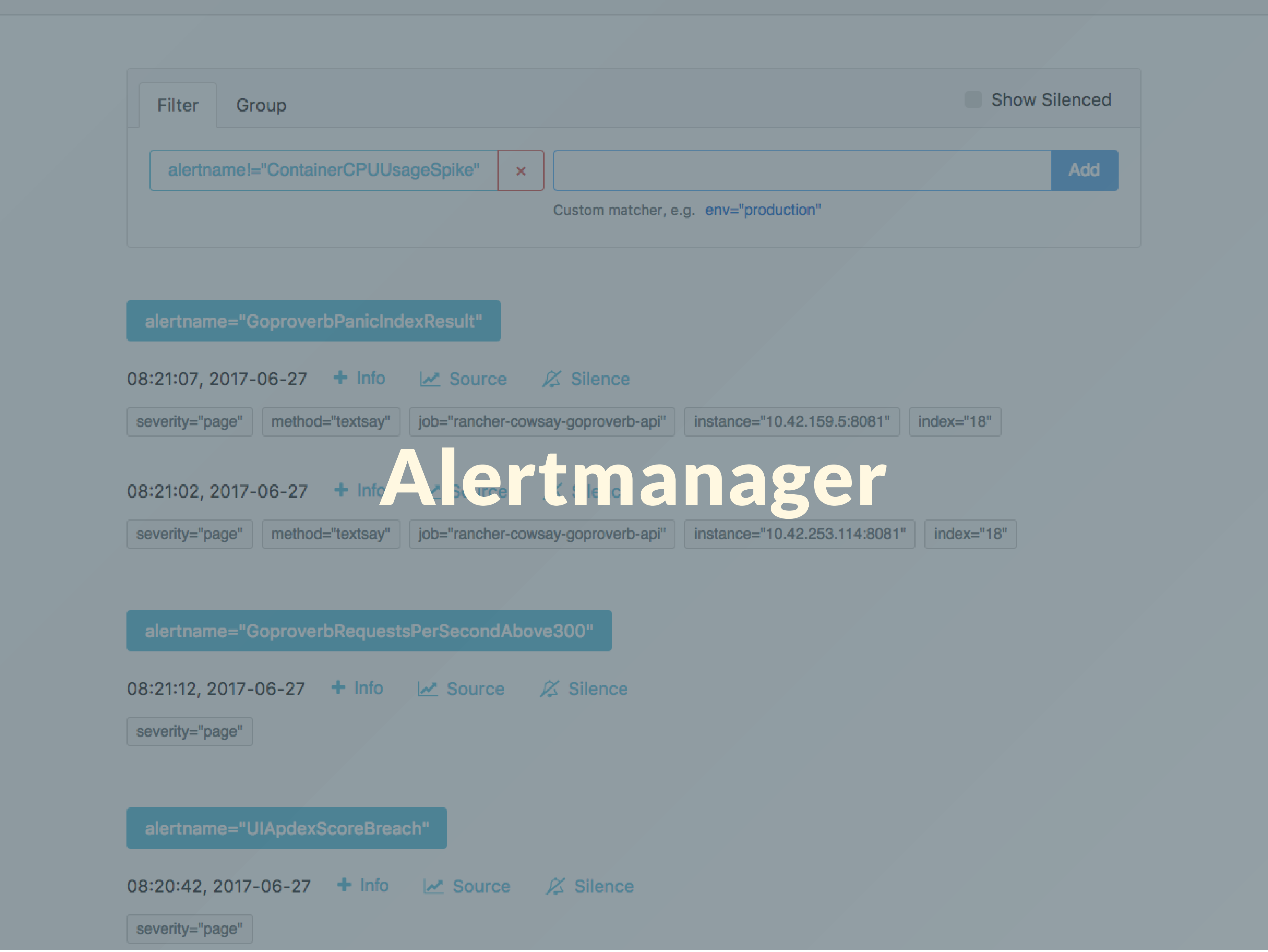
Alert On What Matters

```
ALERT HostDiskWillFillIn2Hours
  IF sum(predict_linear(node_filesystem_free[30m], 2*3600)
  LABELS { severity = "page" }
  ANNOTATIONS {
    summary="{{$labels.instance}} disk will fill in 2 hrs"

ALERT RancherContainerInstanceUnhealthy
  IF rancher_service_health_status{health_state !=
        "healthy"} == 1
  FOR 5m
  LABELS { severity="notify", method="slack" }

ALERT AbnormalSelfServicePortalLoginRate
# Outside its Holt-Winters exponentially smoothed forecast
  IF abs(job:portal_logins:rate1m -
    job:portal_logins:holt_winters_rate5m)
    > abs(0.6 * job:portal_logins:holt_winters_rate5m)
```

Alertmanager



alertname!="ContainerCPUUsageSpike"

Custom matcher, e.g. env="production"

alertname="GoproverbPanicIndexResult"

08:21:07, 2017-06-27 + Info Source Silence

severity="page" method="textsay" job="rancher-cowsay-goproverb-api" instance="10.42.159.5:8081" index="18"

08:21:02, 2017-06-27 + Info Source Silence

severity="page" method="textsay" job="rancher-cowsay-goproverb-api" instance="10.42.253.114:8081" index="18"

alertname="GoproverbRequestsPerSecondAbove300"

08:21:12, 2017-06-27 + Info Source Silence

severity="page"

alertname="UIApdexScoreBreach"

08:20:42, 2017-06-27 + Info Source Silence

severity="page"

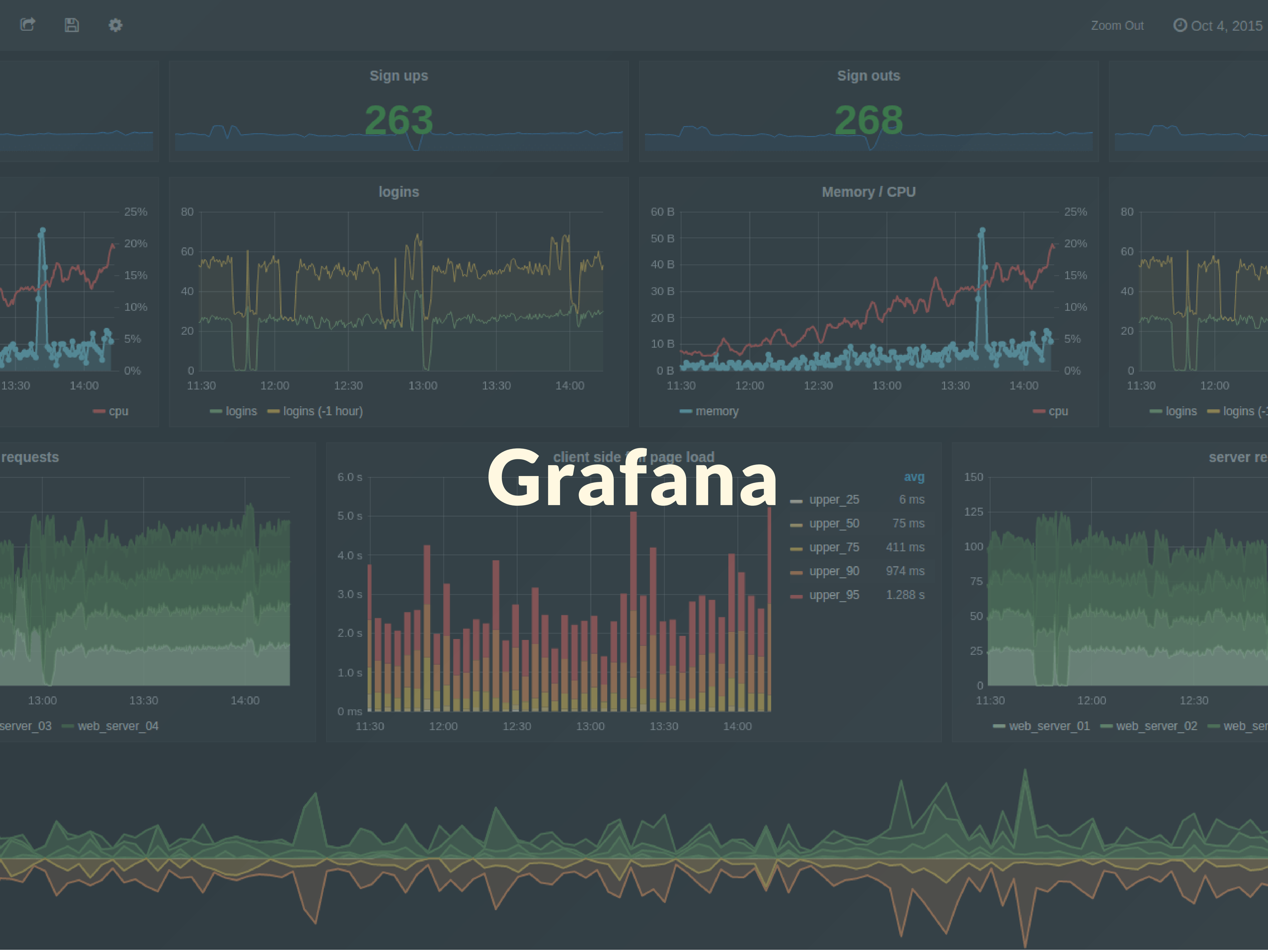
Alertmanager

Handles alerts sent by Prometheus (or other clients)

Takes care of:

- Grouping alerts of similar nature by category
- De-duplication of the same alerts
- Silencing alerts. Keep signal to noise ratio low!
- Routing alerts to receivers
  - Email, SMS, Slack, HipChat, PagerDuty, OpsGenie, VictorOps, Webhooks

Grafana



	avg
upper_25	6 ms
upper_50	75 ms
upper_75	411 ms
upper_90	974 ms
upper_95	1.288 s

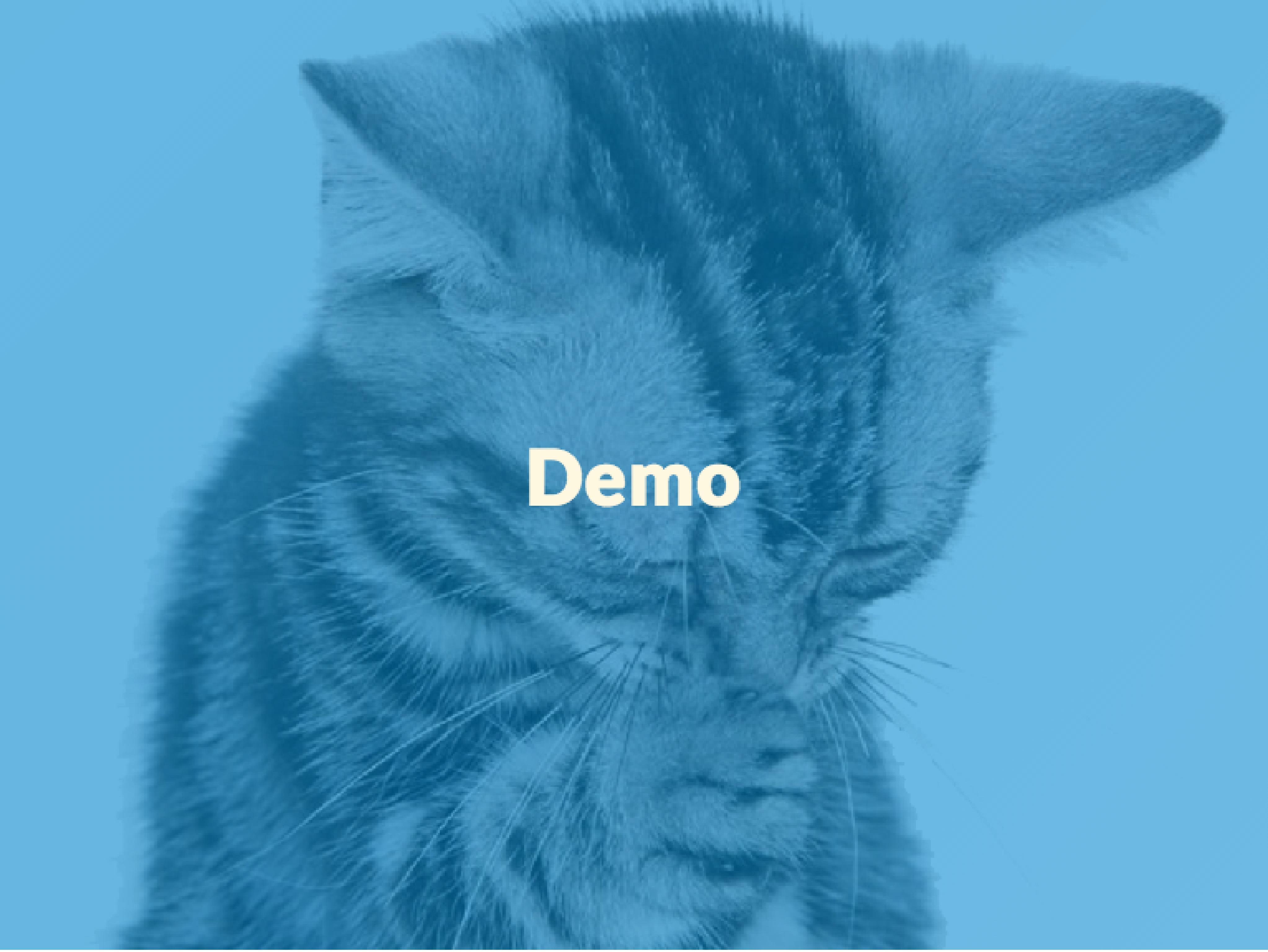
Grafana

Leading open-source platform for beautifully visualising time-series analytics and monitoring

Takes care of:

- Querying Prometheus as a datasource
- Building dashboards on the exact queries you're using in Prometheus for alerts, reporting

Also has hundreds of pre-canned dashboards and other datasources e.g. Graphite, ElasticSearch, CloudWatch, InfluxDB, Splunk, DataDog, OpenTSDB



Demo