Prometheus

The light, the shadow and the livers...

Martin Chodúr

FUSAKLA

Project not a company

prometheus/prometheus

fabxc

juliusv

brian-brazil

beorn7

Incubating in Cloud Native Computing Foundation

Prometheus

CNCF Incubating

Data model

- Multidimensional model
- Every time-serie identified by name and labels
- Metric name is just specifically handled label
- Sample consists of millisecond precision timestamp and float64
- 4 basic metric types: Counter, Gauge, Summary, Histogram

Counter

```
http_requests_total{status=”200”, endpoint=”/metrics”} 10
```

- Always rising, cannot decrease at any point
- Should have the `_total` suffix in metric name
- Engine needs to deal with counter resets (application restart)
- Should be used with the `rate` , `irate` and `increase` functions

Gauge

```
node_memory_usage_bytes{node=”fusakla.cz”}  23005456
```

- Represents actual value of some phenomenon
- Accuracy relies on the scrape interval you use
- Can’t be used with the `rate` , `irate` and `increase` functions

Histogram

```
http_request_duration_seconds_count{endpoint=”/metrics”}  5
http_request_duration_seconds_sum{endpoint=”/metrics”}  1
http_request_duration_seconds_bucket{endpoint=”/metrics”,  le=”0.1”} 1
http_request_duration_seconds_bucket{endpoint=”/metrics”,  le=”0.5”} 3
                                   ...
http_request_duration_seconds_bucket{endpoint=”/metrics”,  le=”+Inf”} 5
```

- Metric type consisting of set of Counters
- Describes count, total sum and distribution in given buckets
- Using the `histogram_quantile` it’s possible to compute quantiles

Summary

```
http_request_duration_seconds_count{endpoint=”/metrics”}  5
http_request_duration_seconds_sum{endpoint=”/metrics”}  1
http_request_duration_seconds{endpoint=”/metrics”,  quantile=”0.5”} 1
http_request_duration_seconds{endpoint=”/metrics”,  quantile=”0.9”} 3
                                          ...
http_request_duration_seconds{endpoint=”/metrics”,  quantile=”0.99”} 5
```

- Same as Histogram only evaluated on exporter side
- No need to compute the quantile on the Prometheus side
- Hides bucket distribution (Histogram is more used)

Gathering the data

- Prometheus uses pull based model
- Application exposes metrics on HTTP endpoint
- Periodically scrapes data from targets
- Variety of client libraries
- Exporters. A lots of them...

Exposition format

```
# HELP grafana_info Information about the Grafana
# TYPE grafana_info gauge
grafana_info{version="5.1.3"}  1
# HELP grafana_instance_start_total counter for started instances
# TYPE grafana_instance_start_total counter
grafana_instance_start_total  1
# HELP grafana_page_response_status_total page http response status
# TYPE grafana_page_response_status_total counter
grafana_page_response_status_total{code="200"}  11808
grafana_page_response_status_total{code="unknown"}  89
# HELP grafana_proxy_response_status_total proxy http response status
# TYPE grafana_proxy_response_status_total counter
grafana_proxy_response_status_total{code="200"}  981338
grafana_proxy_response_status_total{code="400"}  3
grafana_proxy_response_status_total{code="500"}  49672
```

Open Metrics

OpenMetrics is a working group to determine a standard for exposing metrics data, influenced by the Prometheus exposition format.

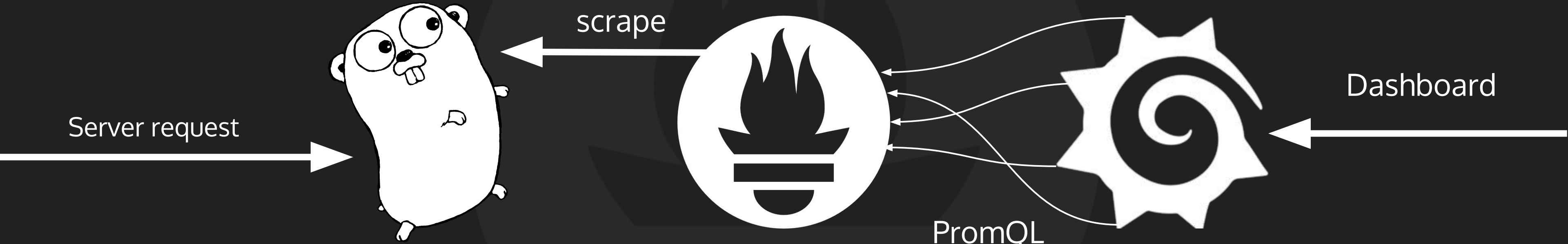
RichiH/OpenMetrics

Running and operating it

- Prometheus is single binary with no dependencies written in Go
- Configured by YAML file
- Prometheus does NOT have any configuration features...

  That leaves to the operator
- Can be CPU and memory intensive
- Needs persistent filesystem (do not use NFS!)

Demo time



Go instrumentation

```
var httpRequestDurationHistogram  = prometheus.NewHistogramVec(
                                   prometheus.HistogramOpts{
                                        Name: "http_request_duration_seconds",
                                        Help: "HTTP request latency distribution",
                                    },
                                    []string{"status", "endpoint"},
                              )

prometheus.MustRegister(httpRequestDurationHistogram)

httpRequestDurationHistogram.WithLabelValues("200","/demo").Observe(duration.Seconds())

http.Handle("/demo/metrics", promhttp.Handler())
```

Prometheus configuration

```yaml
global:
  scrape_interval:    5s
  evaluation_interval: 10s


scrape_configs:
  - job_name: 'prometheus'
   static_configs:
        - targets: ['localhost:9090']
  - job_name: 'error-server'
   metrics_path: "/demo/metrics"
   scheme: https
   static_configs:
        - targets: ['fusakla.cz']
```

TSDB

- Prometheus uses own database   prometheus/tsdb
- Uses 2h WAL and than stores data in blocks on filesystem
- Blocks are compacted to longer blocks (6h, 12h)
- Not meant for longer retentions
- Prometheus allows backups using API

Service discovery

- Main feature for pull based model
- Implements cca 12 different SD allowing dynamic target load
- Well integrated with Kubernetes coreos/prometheus-operator

PromQL

- Powerful query language with variety of operators and functions
- Results in one of these: Instant vector, Range vector, Scalar, String
- Allows aggregating on all the labels
- Cannot create Range vector for function result
- Turing complete

```
rate(http_requests_total{job="api-server"}[5m])
```

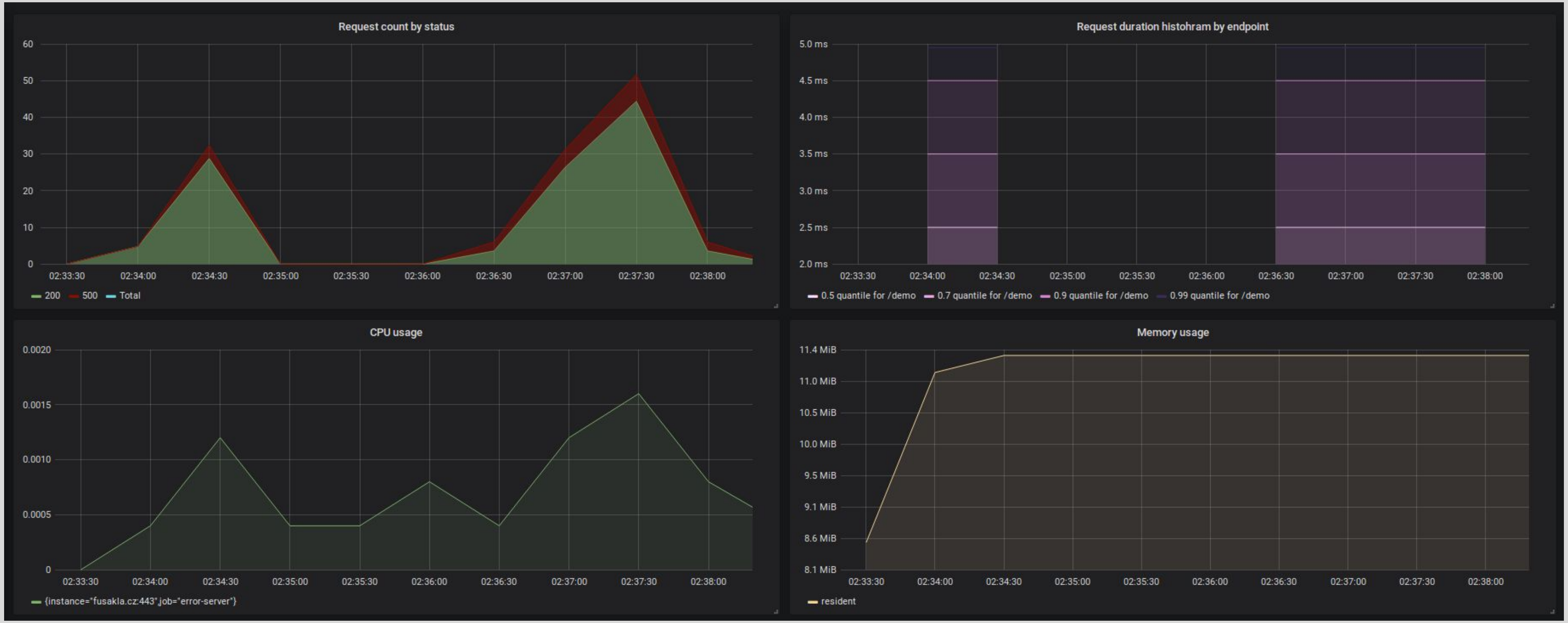
Data and PromQL

```
                    increase(http_requests_total{endpoint="/metrics"}[1m])

                  2  4   5   6    7   8  15  1   2   4    5  5   7   8
Raw data         |-+--+---+---+----+---+--+---+---+---+----+--+---+---+--|
                 |  1m  |      |      |      |      |      |      |      |
Range vector     |------+------+------+------+------+------+------+------|
                  (4-2)  (6-5)    --   (15-8)  (2-1)  (5-4)  (7-5)  (8-7)
                   /60    /60           /60     /60    /60    /60    /60


Result             0.03   0.01    --     0.1    0.01   0.01  0.03    0.01
```

Grafana dashboard



What’s next

- Using recording rules to avoid repetitive query evaluation
- Explore Prometheus HTTP api and remote read and write API
- HA Alerting using alert rules and Alertmanger
- Hierarchical federation
- Relabeling configs
- Long term storage

Closing remarks

Every time when Prometheus exits, last thing it does is writing to the log:

```
level=info caller=main.go:599 msg="See you next time!"
```

Wait when you mess up configuration few times in a row... spiteful little troll

HEY EAGLE, HERE TO EAT MY LIVER?
GEE, PROMETHEUS, HOW'D YOU GUESS?
EVERYTHING OK?
YEAH, I LOVE EATING THE SAME DAMN LIVER EVERY DAMN DAY!
I'M SORRY...
WHY AM I BEING PUNISHED?! THIS WAS ALL YOUR FAULT!
I KNOW...
WHATEVER... LET'S GET THIS OVER WITH...

Ask questions to

Me:

FUSAKLA

m.chodur@seznam.cz

Martin Chodur

Community:

IRC #prometheus

prometheus-users

@PrometheusIO

...it's always better to waste someone else's time than yours