An Introduction to Prometheus

Cemal Ünal

- Software Engineer @ Havelsan Inc.

 cemalunal

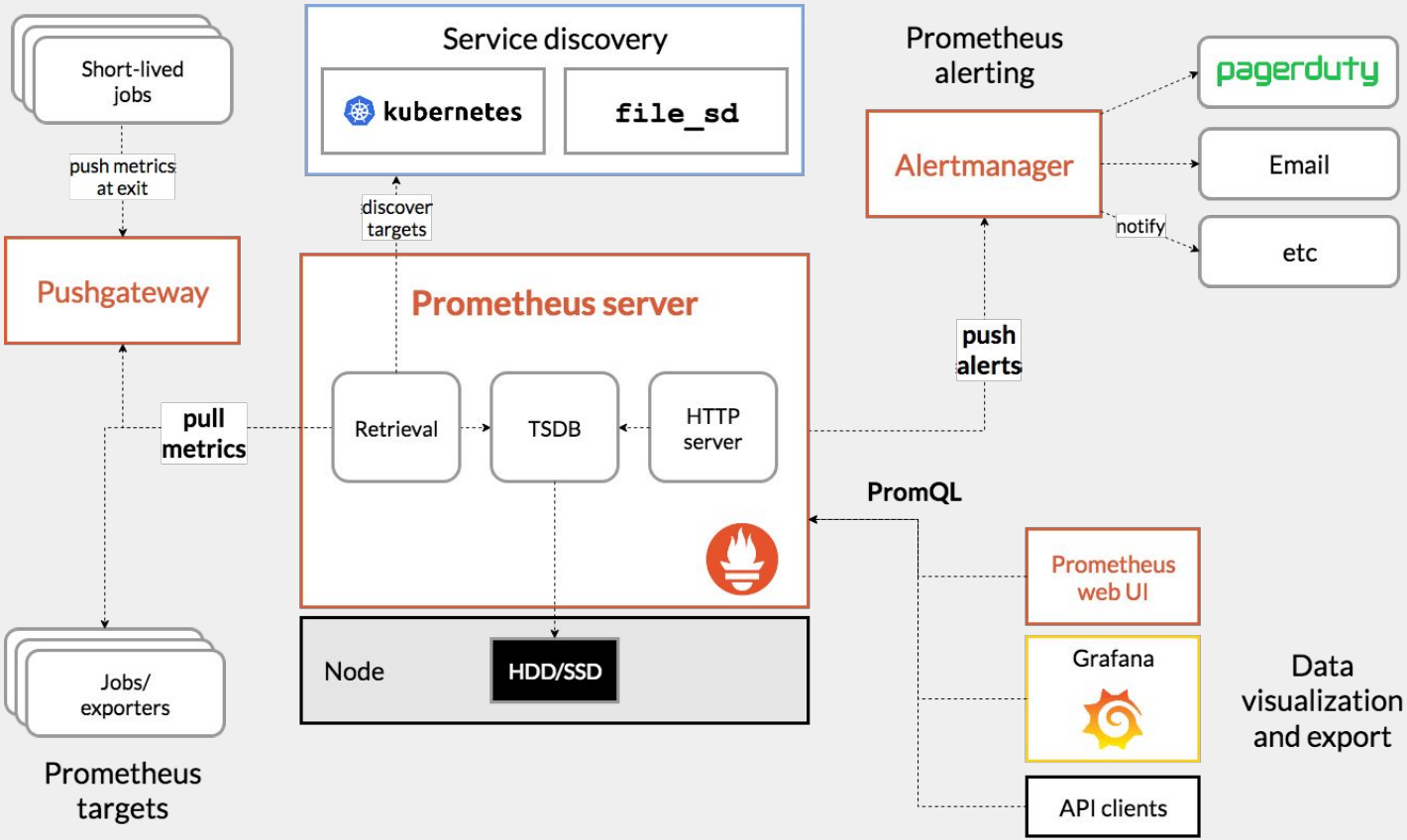
 cemalunal

 cunal@havelsan.com.tr

Prometheus

- Open-source systems monitoring and alerting toolkit
- Records real-time metrics in a time series database built using a HTTP pull model
- Metric collection, processing and storage
- Powerful query language
- Metric Visualization
- Real time alerting

Ecosystem and Architecture



Components

The ecosystem consists of multiple components. Some of them are:

- The main **Prometheus server** which scrapes and stores time series data
- **Special-purpose exporters** for services like HAProxy, StatsD, Graphite, etc.
- **AlertManager** to handle alerts
- **Push Gateway** supporting short living jobs

Data Model

- Fundamentally stores all data as time series (measurement/metric with a timestamp).
- Each time series is uniquely identified by a metric name and a set of key-value pairs (labels).
- <metric name>{<label name>=<label value>, ...}

Metric Types

There are four different types of metrics:

- **Counter:** A cumulative metric that only ever increases. (E.g. requests served, tasks completed, errors occurred)
- **Gauge:** A metric that can arbitrarily go up or down. (E.g. temperature, current memory usage)
- **Histogram:** Binned measurement of a continuous variable. (E.g. latency, request duration)
- **Summary:** Similar to a histogram, except the bins are converted into an aggregate (e.g. 99% percentile) immediately.

Metric Example

```
http_requests_total{service="webserver", region="EU"} 136
```

- Metric name is **http_requests_total**
- Two labels, **service** and **region** with values **"webserver"** and **"EU"**
- Metric value is **136**

This webserver service has received 136 http requests since its started.

PromQL Examples

Filter:

```
http_requests_total{service="webserver"}
```

Total HTTP requests that webserver receives

Aggregate:

```
sum(http_requests_total)
```

total HTTP requests

Average rate of increase:

```
rate(http_requests_total[5m])
```

the per-second rate of HTTP requests as measured over the last 5 minutes

Alerting

Prometheus allows defining rules based on PromQL in the following format:

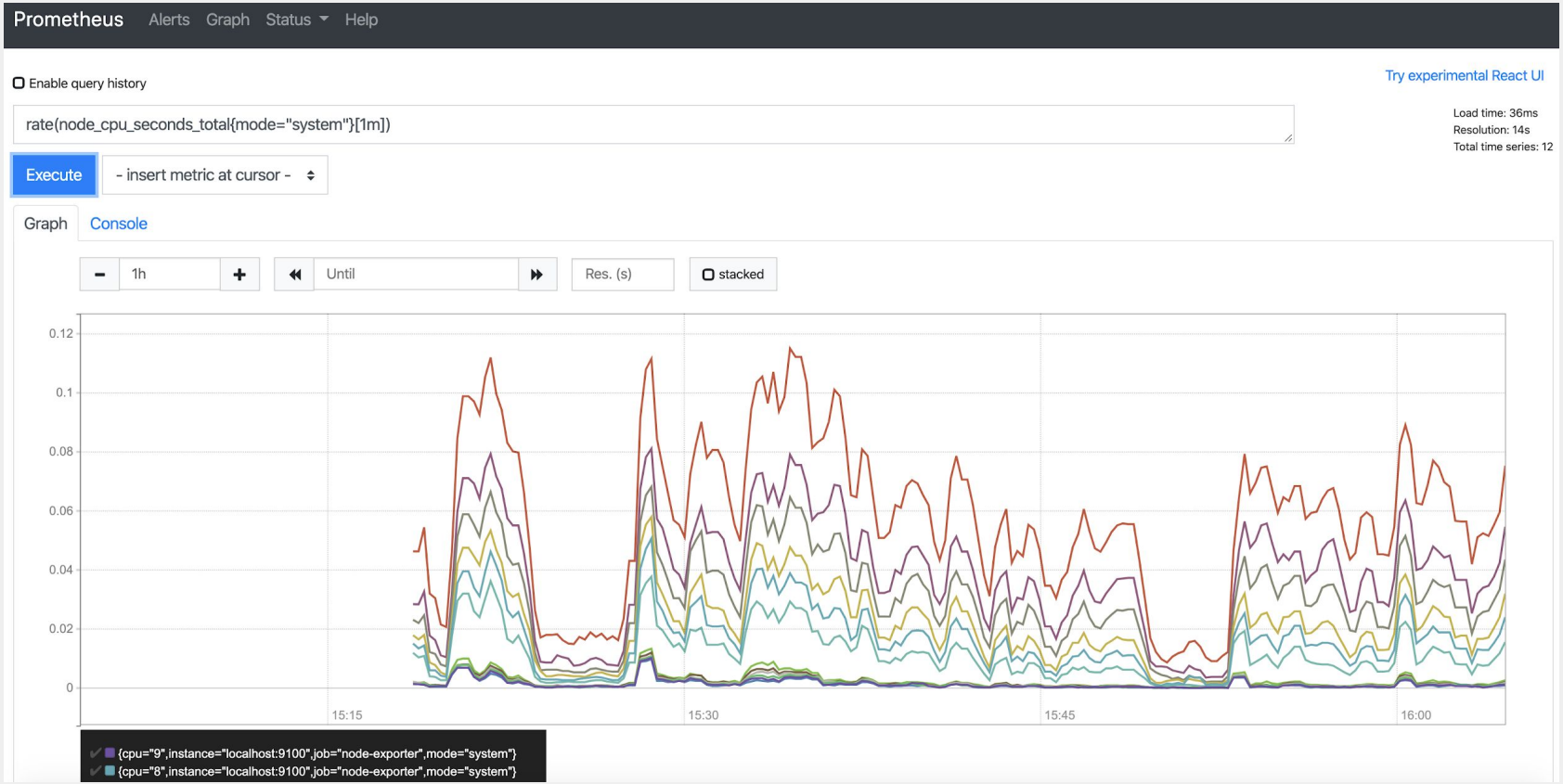
```
- alert: LowRequests
  expr: rate(http_requests_total{service="webserver"}[1m]) < 0.1
  for: 5m
```

Prometheus will execute this regularly.

AlertManager will send a notification via email, webhook, etc.

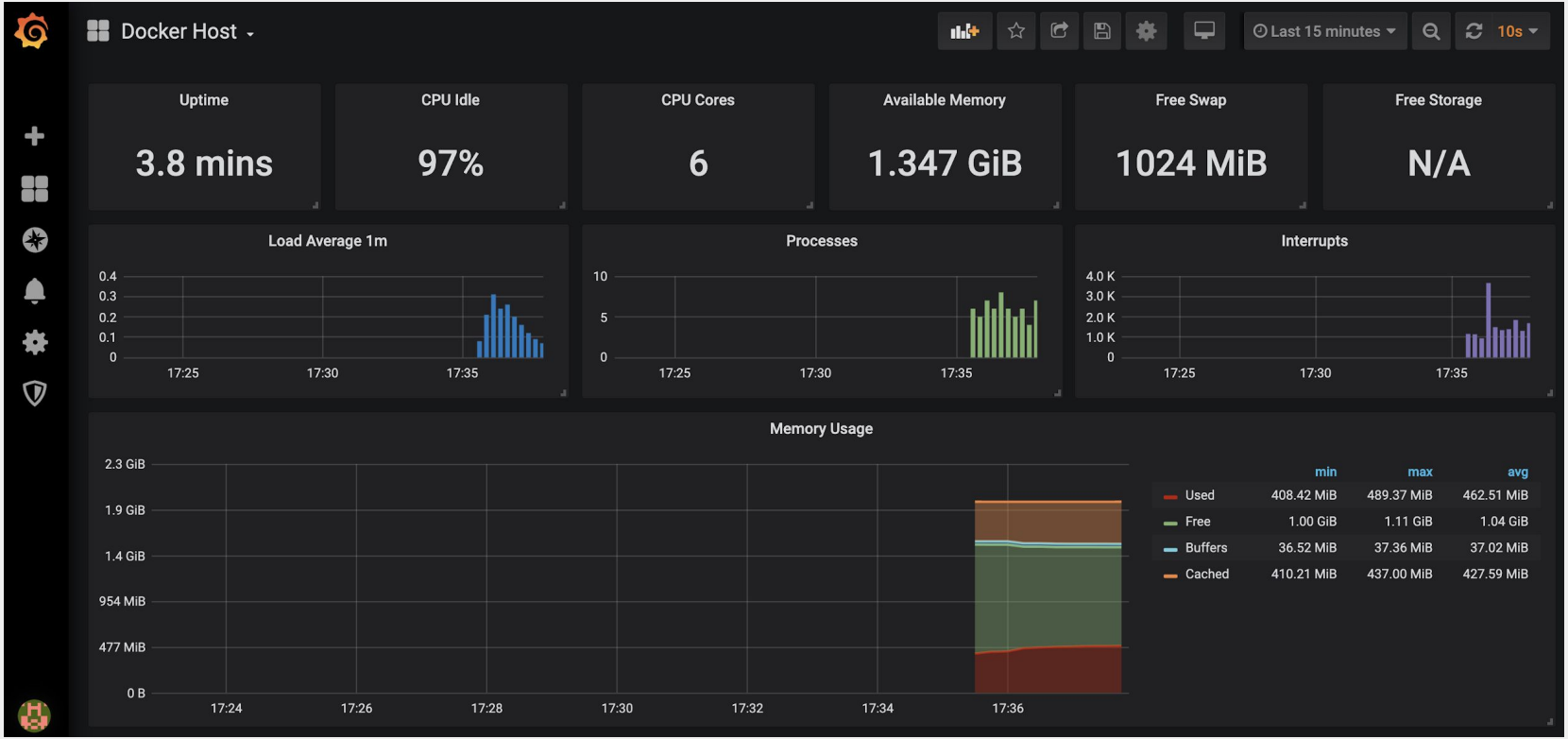
Visualization - Built-in

- Allows us to enter any expression and see its result either in a table or graphed over time.
- Useful for ad-hoc queries and debugging purposes
- Available at /graph endpoint


Prometheus
Alerts
Graph
Status
Help
Enable query history
Try experimental React UI
rate(node_cpu_seconds_total{mode="system"}[1m])
Load time: 36ms
Resolution: 14s
Total time series: 12
Execute
- insert metric at cursor -
Graph
Console
1h
Until
Res. (s)
stacked
0.12
0.1
0.08
0.06
0.04
0.02
0
15:15
15:30
15:45
16:00
{cpu="9",instance="localhost:9100",job="node-exporter",mode="system"}
{cpu="8",instance="localhost:9100",job="node-exporter",mode="system"}

Visualization - Grafana

- Open source analytics and monitoring solution
- Supports querying Prometheus metrics


Docker Host
Last 15 minutes
10s
Uptime
3.8 mins
CPU Idle
97%
CPU Cores
6
Available Memory
1.347 GiB
Free Swap
1024 MiB
Free Storage
N/A
Load Average 1m
0.4
0.3
0.2
0.1
0
17:25
17:30
17:35
Processes
10
5
0
17:25
17:30
17:35
Interrupts
4.0 K
3.0 K
2.0 K
1.0 K
0
17:25
17:30
17:35
Memory Usage
2.3 GiB
1.9 GiB
1.4 GiB
954 MiB
477 MiB
0 B
17:24
17:26
17:28
17:30
17:32
17:34
17:36
min
max
avg
Used 408.42 MiB 489.37 MiB 462.51 MiB
Free 1.00 GiB 1.11 GiB 1.04 GiB
Buffers 36.52 MiB 37.36 MiB 37.02 MiB
Cached 410.21 MiB 437.00 MiB 427.59 MiB

DEMO

THANKS!