

Intro to Prometheus

With a dash of operations & observability

Richard Hartmann & Ben Kochie,
@TwitchiH

2018-11-14

Who are we?

- Richard "RichiH" Hartmann
 - Swiss army chainsaw at SpaceNet
 - Project lead for building one of the most modern datacenters in Europe
 - Debian Developer
 - FOSDEM, DebConf, DENOGx, PromCon staff
- Ben "SuperQ" Kochie
 - Staff engineer at GitLab
 - Bit plumber
 - Retired SRE
 - FOSDEM infrastructure, PromCon staff

Show of hands

- Who has heard of Prometheus?
- Who is considering to use Prometheus?
- Who is POCing Prometheus?
- Who uses Prometheus in production?

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

Working assumptions & concepts

- 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: Instrumentation 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
- No built-in TLS yet, use reverse proxies for now

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
 - Access rates to a webserver (counter)
 - Temperatures in a datacenter (gauge)

Efficiency

- 1,000,000+ samples/second no problem on current hardware
- 200,000 samples/second/core
- 16 bytes/sample compressed to 1.36 bit/sample
- Cheap ingestion & storage means more data for you

Exposition format

```
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 vs SQL

```
avg by(city) (temperature_celsius{country="germany"})
```

```
SELECT city, AVG(value) FROM temperature_celsius WHERE \
  country="germany" GROUP BY city
```

```
rate(errors{job="foo"}[5m]) / rate(total{job="foo"}[5m])
```

```
SELECT errors.job, errors.instance, [...more labels...], \
  rate(errors.value, 5m) / rate(total.value, 5m) \
FROM errors JOIN total ON [...all label equalities...] \
WHERE errors.job="foo" AND total.job="foo"
```

Grafana

- Supports dozens of data sources
- Modern UI
- Allows for complex data manipulation and visualization
- Native Prometheus support
- New feature: Interactive exploration of Prometheus data

Toil

“Toil is manual, repeated work with no lasting benefit which scales linearly with your service”

- If teams are busy firefighting, they don't have time to engineer
- Keep legacy systems working, but have clear path forward
- Keep extra effort on the team low, if possible
- Strive for immediate benefits
- Focus on removing repeated, manual tasks of no lasting benefit
- Show that you free up time and reduce toil

Sanity & sleep

- If it's not actionable, it's not an alert
- If it's not urgent, it's not an alert
- Important but non-urgent incidents are handled during business hours
- Predict your usage so you add capacity during business hours
- If there's no playbook, it does not go into production
- If a service does not have proper SLOs and alerts, it does not go into production

Perspective & Incentives

”An engineer can talk for hours about source code; try that with the CEO”

- Managers: revenue, process execution
- Architects: clean design, process definition
- Product/Service owners: Powerful dashboards
- Team leads: morale, quick execution
- Operators: reduce toil, increase sleep

Tell everyone what they need to hear (but never lie)

Post-Mortems

- Mistakes happen
- It is important to learn from mistakes so not to repeat them
- To write a good incident report, there must be no fear of retribution
- Blame-free post-mortems allow everyone to document exactly what went wrong and in what order
- It is important to build trust among the teams and management

Leverage

- One combined system allows for correlation and combination
- Power usage against service load
- Optical networks against outside temperature
- Datacenter power feed load against new deployments
- ...and lots more
- Metrics are the starting point of most observability stories

Oracle

- One source of truth for
 - Tactical overview for current state
 - Dashboards for drill-down
 - Auto-generated PDFs for customers
 - Global SLO statements for sales
 - Usage exports for accounting
- If all you have is a hammer... choose your hammer well

Thanks!

Thanks for listening!

Questions?