Instrumenting Java with Prometheus
Alpes JUG Meet-up

Simon Pasquier, December 19th 2018

Who am I?

- Software engineer working at Red Hat and Prometheus maintainer.
- Java n00b :-)

Definition

“Prometheus is a systems and service monitoring system”

A bit of history

- Originated at SoundCloud by Xooglers.
- Open-sourced in 2012.
- First “public” release and hosted by the CNCF in 2015.
- Graduation from the CNCF in 2018.

Who uses it?

- SoundCloud
- GitLab
- Digital Ocean
- CloudFlare
- Fastly
- ~~CoreOS~~ Red Hat
- And many more

Focus

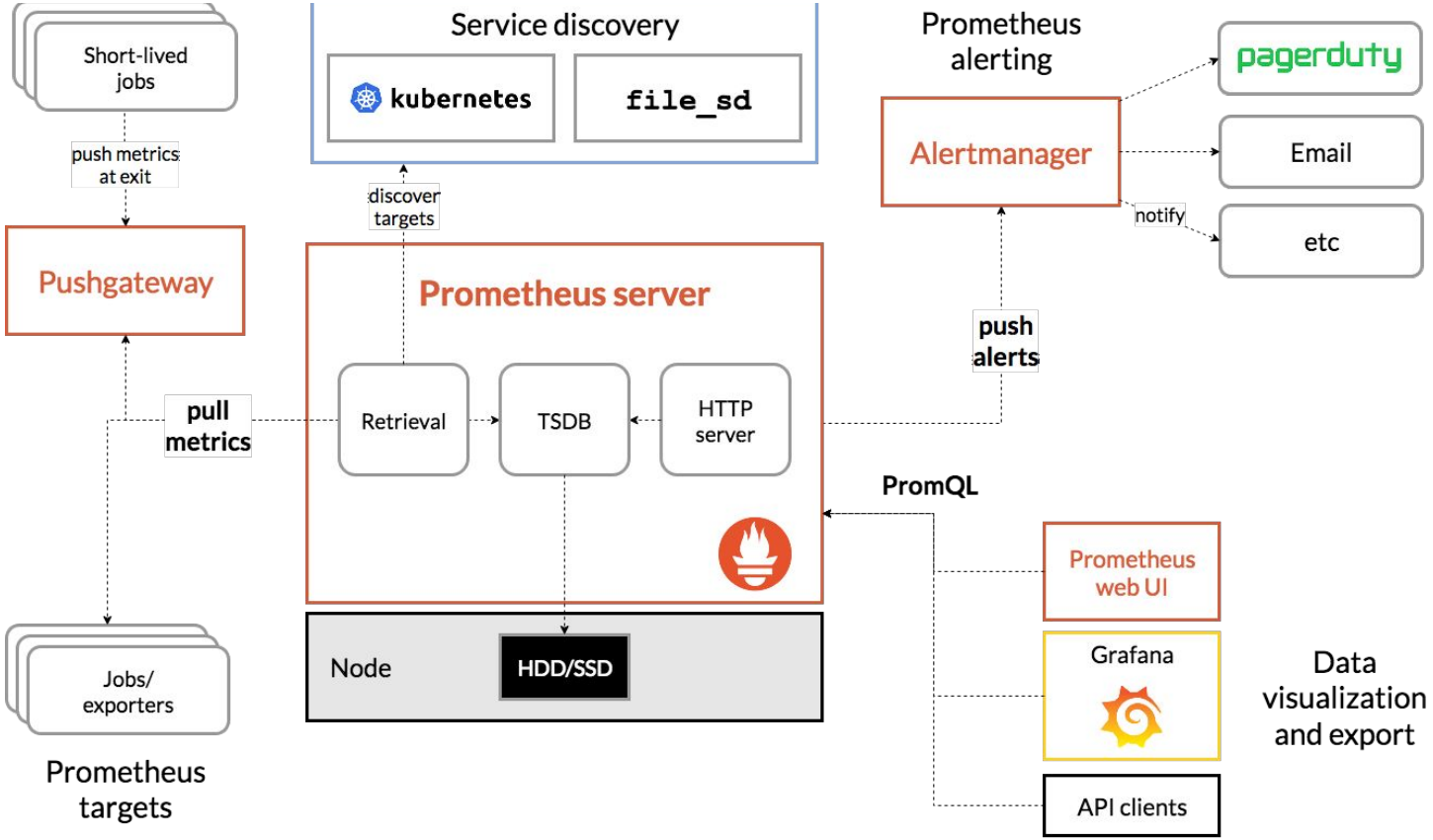
- Metrics and alerting
- Pull model
- Whitebox
- Simple to operate

Non-goals

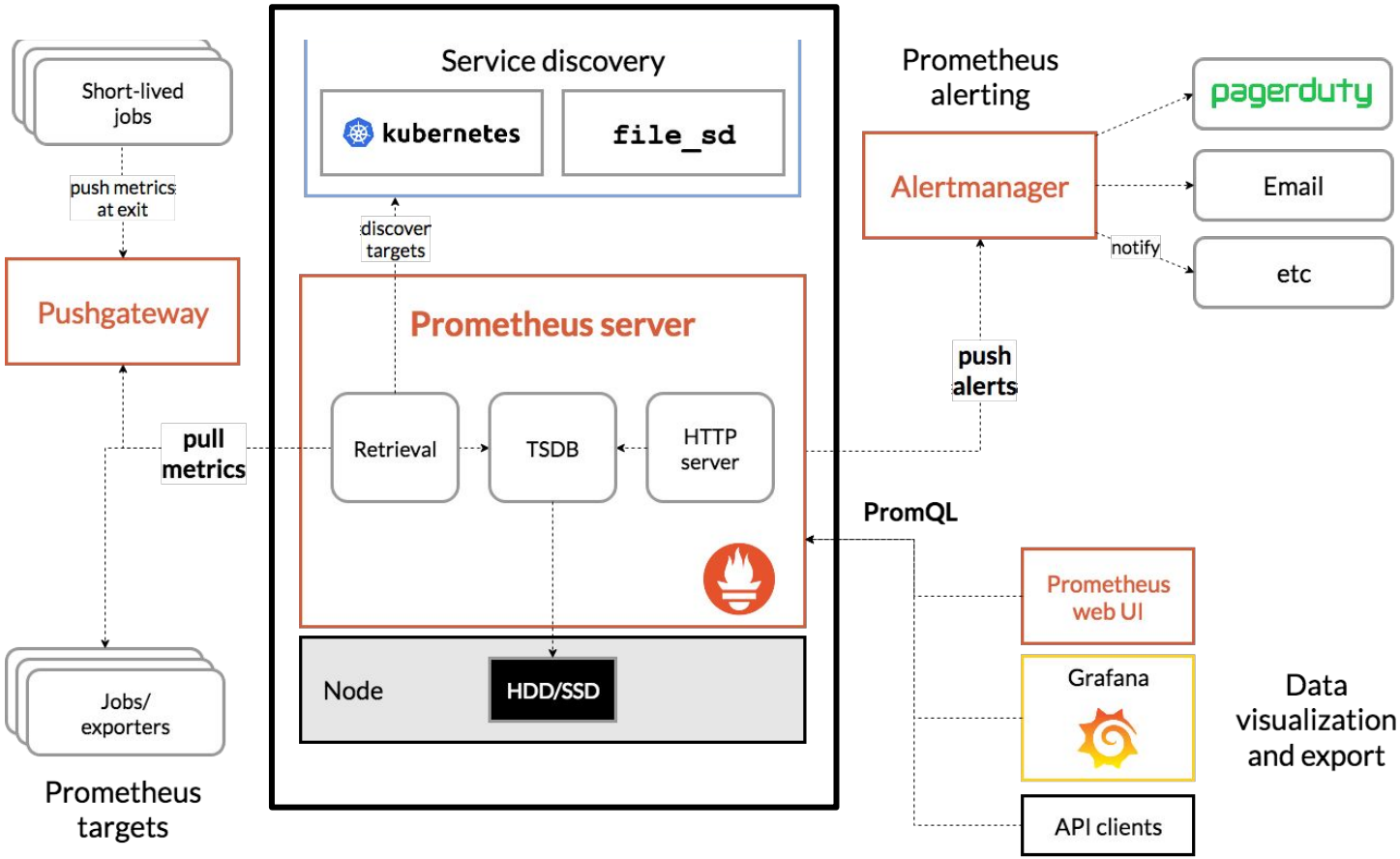
- Event/logging/billing
- Push model
- Long term storage

Ecosystem

- Prometheus
- AlertManager
- Exporters
- Client libraries



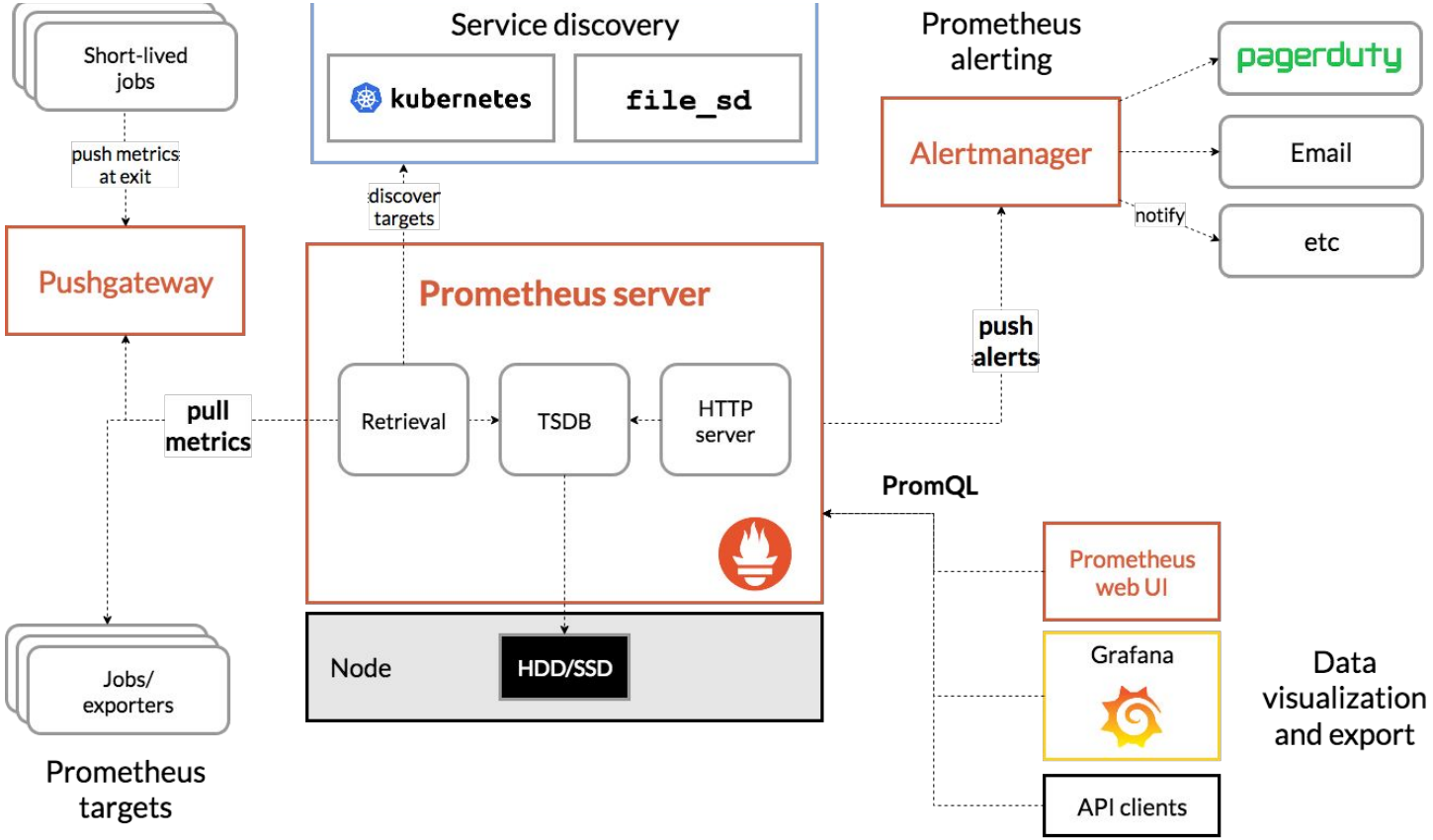
source: https://prometheus.io/docs/introduction/overview/#architecture



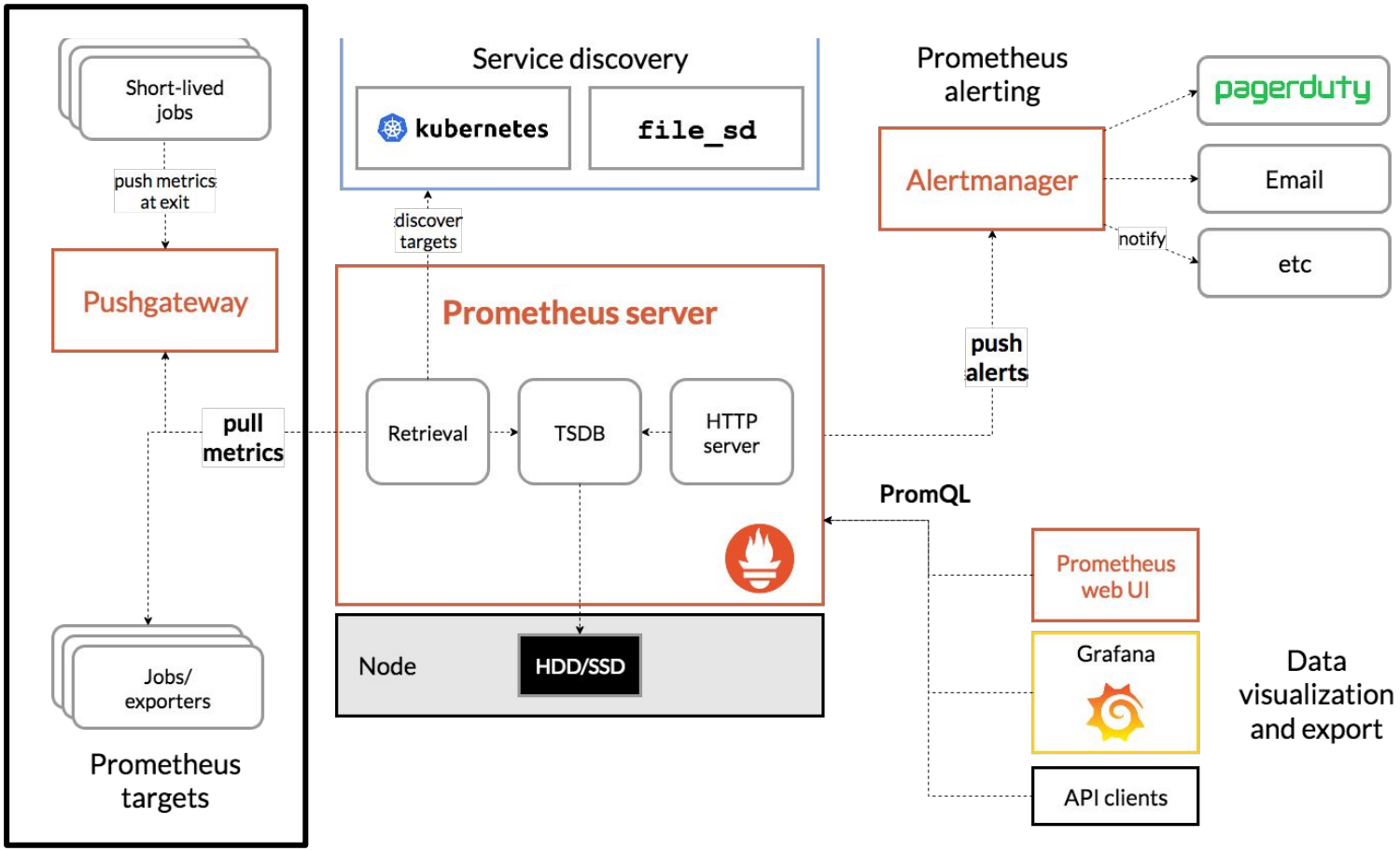
source: https://prometheus.io/docs/introduction/overview/#architecture

Prometheus

- Metrics collection (scraping) and storage
- Query engine (PromQL)
- Alerting
- REST API
- User interface



source: https://prometheus.io/docs/introduction/overview/#architecture



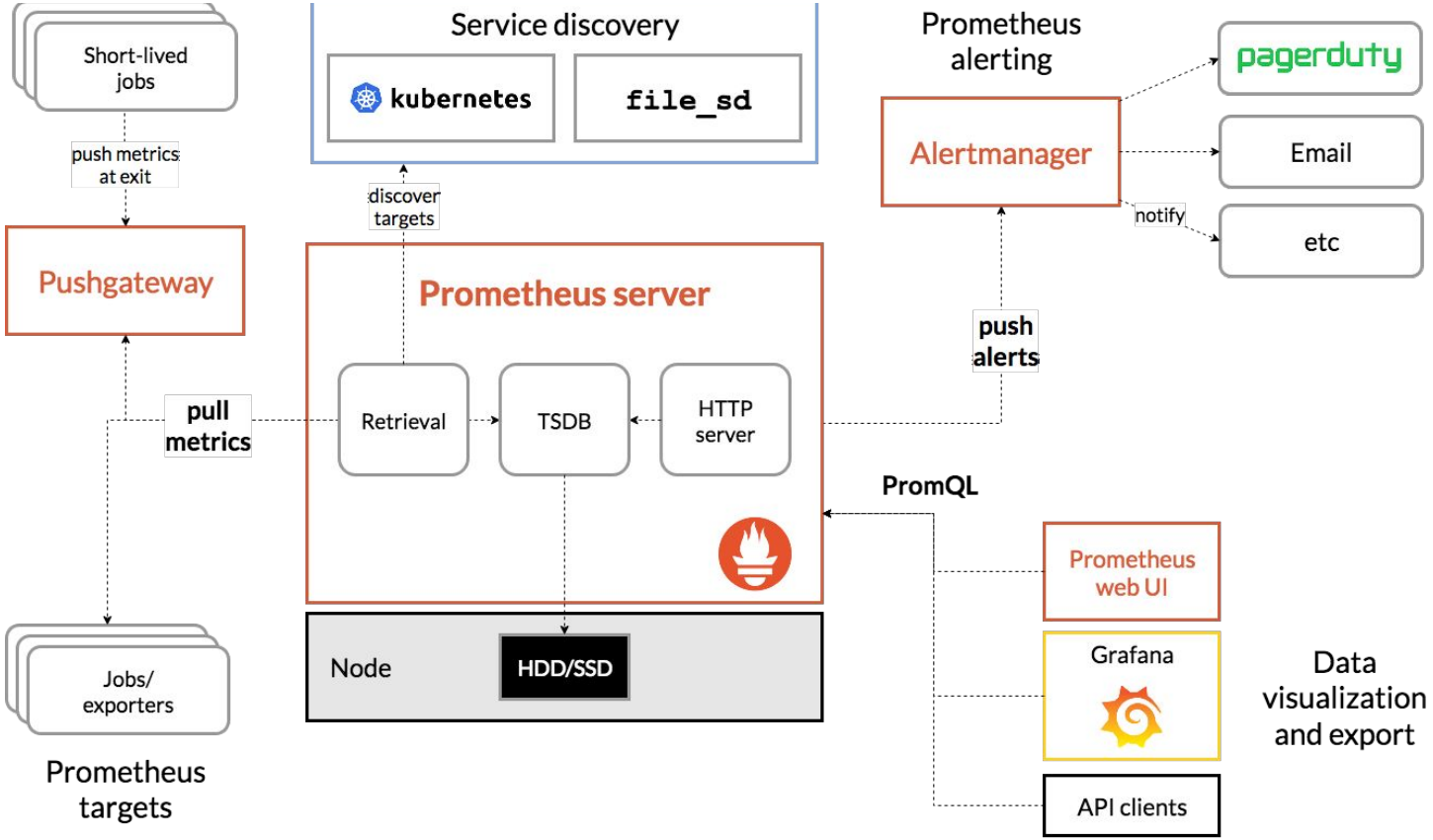
source: https://prometheus.io/docs/introduction/overview/#architecture

Exporters

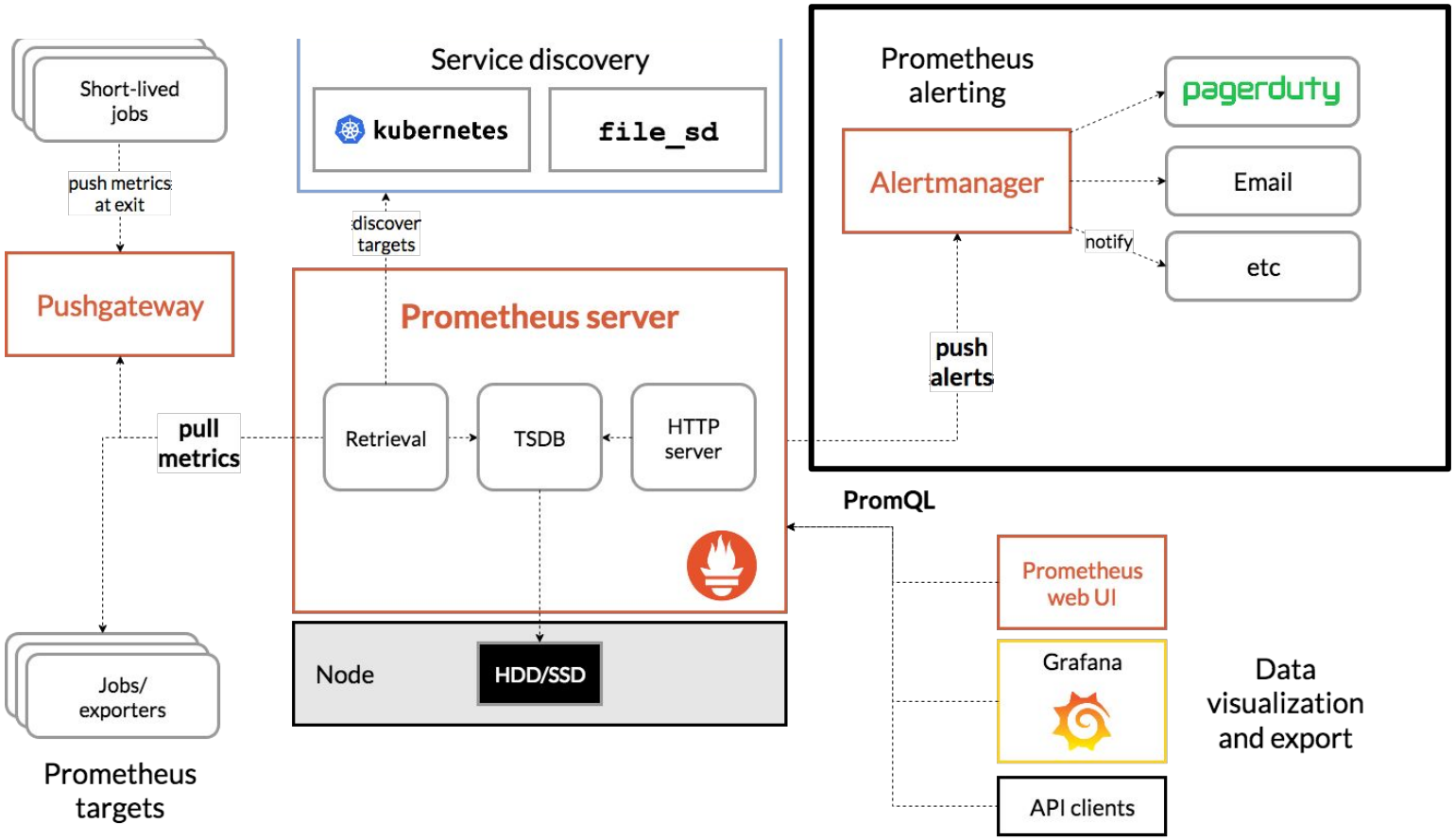
- node_exporter
- mysql_exporter
- haproxy_exporter
- blackbox_exporter
- ...

Client libraries

- Java
- Go
- Python
- Ruby
- ...



source: https://prometheus.io/docs/introduction/overview/#architecture



source: https://prometheus.io/docs/introduction/overview/#architecture

AlertManager

- Group, inhibit and deduplicate alerts received from Prometheus
- Route alerts to external systems

Instrumentation

- Why?
- What?
- How?

Why instrument?

- Alert when something wrong happens.
- Help troubleshooting/debugging.
- Know how your apps are performing.
- Capacity planning.

Why Prometheus?

- Prometheus format widely supported.
- Low overhead.
- Relevant for legacy and cloud-based solutions.

What?

- Business and user-facing metrics.
- Services, not machines.

RED method

- Rate
- Errors
- Duration

https://grafana.com/blog/2018/08/02/the-red-method-how-to-instrument-your-services/

USE method

- Utilization
- Saturation
- Errors

http://www.brendangregg.com/usemethod.html

How

To take full advantage of Prometheus, it is critical to understand its model first.

Data model

Prometheus only deals with time series.

Data model

```
room_temperature_celsius{site="hq1",floor="2",room="meeting_1"}
```

Data model

```
room_temperature_celsius{site="hq1",floor="2",room="meeting_1"}
```

Data model

```
room_temperature_celsius{site="hq1",floor="2",room="meeting_1"}
```

Metric types

- Gauge
- Counter
- Histogram
- Summary

Data model

	t	t+1m	t+2m	t+3m	t+4m
http_requests_total{job="web",instance="foo:8080",method="GET"}	0	1	2	3	4
http_requests_total{job="web",instance="foo:8080",method="POST"}	0	0	2	3	3
http_requests_failed_total{job="web",instance="foo:8080",method="GET"}	0	0	0	0	0
http_requests_failed_total{job="web",instance="foo:8080",method="POST"}	0	0	1	2	2

Time-series = metric name + labels

Samples = time-series + timestamp (milliseconds) + float value

PromQL

	t	t+1m	t+2m	t+3m	t+4m
http_requests_total{job="web",instance="foo:8080",method="GET"}	0	1	2	3	4
http_requests_total{job="web",instance="foo:8080",method="POST"}	0	0	2	3	3
http_requests_failed_total{job="web",instance="foo:8080",method="GET"}	0	0	0	0	0
http_requests_failed_total{job="web",instance="foo:8080",method="POST"}	0	0	1	2	2

```
http_requests_total

http_requests_failed_total{method="GET"}

http_requests_failed_total{method="POST"}[5m]

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

Instrumentation for Java apps

Many options!

JMX exporter

Runs as a Java agent (no separate process) and exposes JMX beans as metrics.

https://github.com/prometheus/jmx_exporter

JMX exporter

- Pros
  - Useful when you can’t modify the code.
- Cons
  - Tedious to setup with lots of metrics.
  - Not the most efficient resource-wise.

client_java

Officially part of the Prometheus organization

https://github.com/prometheus/client_java

client_java

- Pros
  - Aligned with the Prometheus way.
  - Nice additions like Log4J wrapper and HTTP filters.
- Cons
  - May require some wiring.

MicroProfile Metrics

Open specification supported by multiple vendors defining a unified Java API for instrumentation.

https://github.com/eclipse/microprofile-metrics

MicroProfile Metrics

- Pros
  - Prometheus as a first-class citizen.
  - Rich metadata model + API.
  - JVM/runtime metrics out of the box.
  - Portability across runtimes.
- Cons
  - Some inconsistencies with the Prometheus model (addressed in the upcoming 2.0 version).

Micrometer

The official instrumentation library for Spring.

https://micrometer.io/

Micrometer

- Pros
  - Out of the box instrumentation.
  - Supports more than just Prometheus (Graphite, SaaS vendors).
- Cons
  - Trade-offs regarding Prometheus best practices.

Other alternatives

- Write your own implementation
- Convert logs to metrics (mtail, grok_exporter)
- OpenCensus

Wrapping up

- Easy to use
- Start small
- Vibrant community

Thanks!

Simon Pasquier
pasquier.simon@gmail.com
@SimonHiker

Resources

https://prometheus.io/

https://www.youtube.com/channel/UC4pLFely0-Odea4B2NL1nWA

https://www.robustperception.io/blog

https://landing.google.com/sre/books/

https://github.com/simonpasquier/instrumenting-java-for-prometheus