Prometheus Introduction

Matt Layher
Fastly, @mdlayher

Ganesh Vernekar
Grafana Labs, @_codesome

Slides: Julius Volz
Co-founder, Prometheus, @juliusvolz

Prometheus team members

What is Prometheus?

Metrics-based monitoring & alerting stack.

- Instrumentation for applications and systems
- Metrics collection and storage
- Querying, alerting, dashboarding
- For all levels of the stack!

Made for dynamic cloud environments.

What is it not?

Prometheus does not do:

- Logging or tracing
- Automatic anomaly detection
- Scalable or durable storage

History

- Started 2012 at SoundCloud
- Fully publicised in 2015
- Joined CNCF, Prometheus v1.0.0 released in 2016
- Prometheus v2.0.0 released in 2017

Architecture

Architecture

Targets

web app

API
server

Architecture

Targets

web app	clientlib
API server	clientlib

Architecture

Targets

web app   clientlib

API
server   clientlib

Linux VM

mysqld

cgroups

Instrumentation & Exposition

Architecture

Targets

web app — clientlib

API server — clientlib

Linux VM — exporter

mysqld — exporter

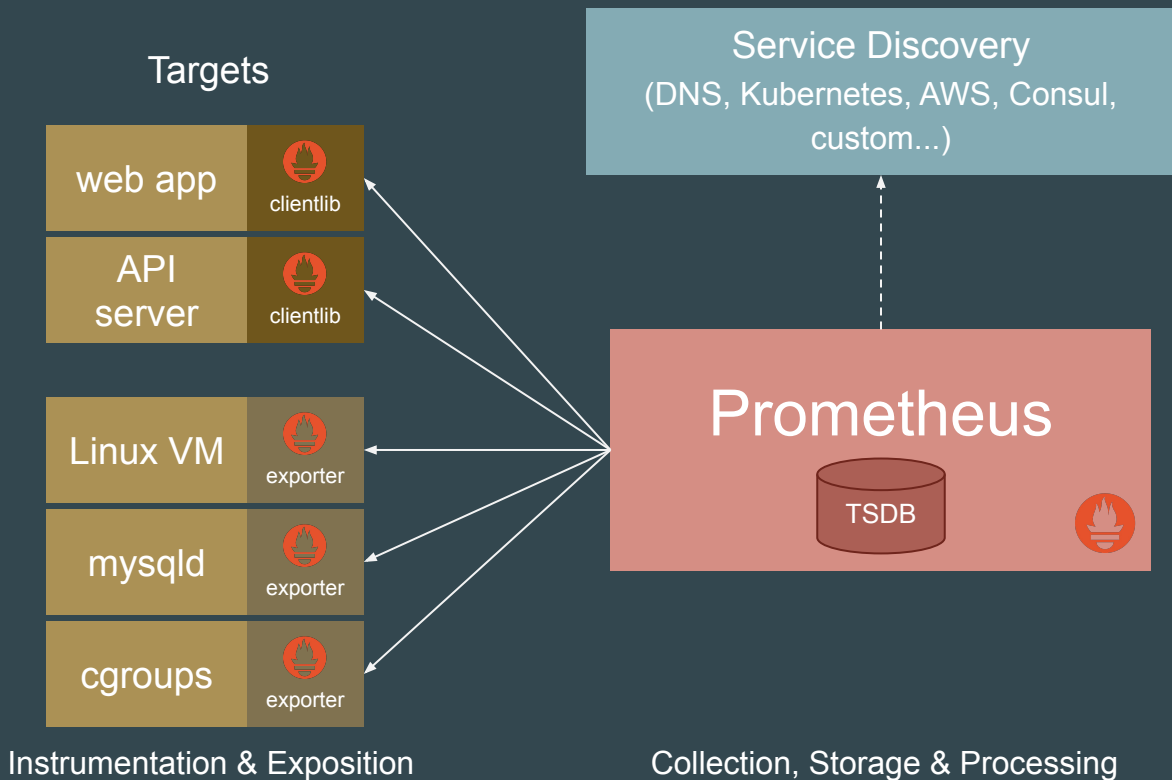cgroups — exporter

Instrumentation & Exposition

Interlude: Exposition Format

```
# HELP http_requests_total Total number of HTTP requests.
# TYPE http_requests_total counter
http_requests_total{method="post",code="200"} 1027
http_requests_total{method="post",code="400"} 3
```

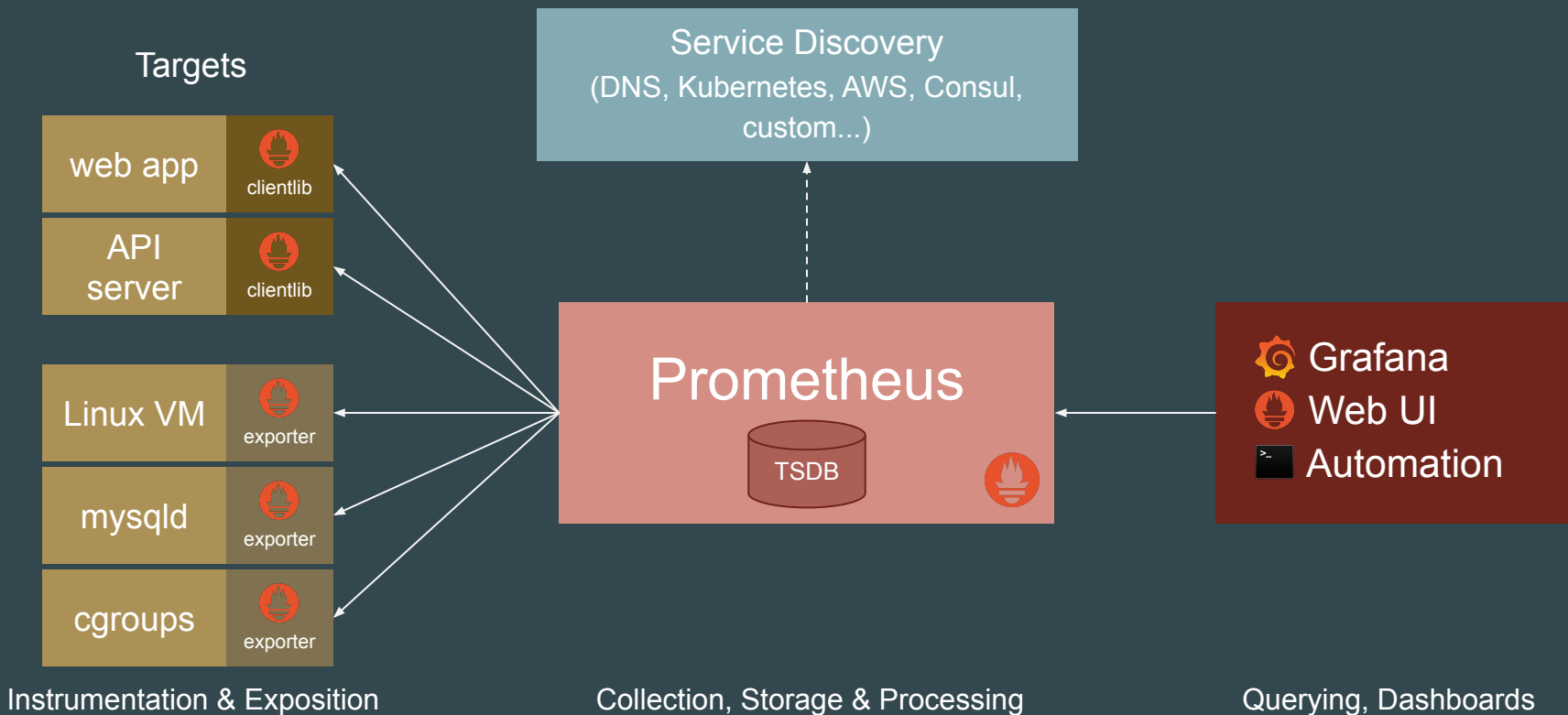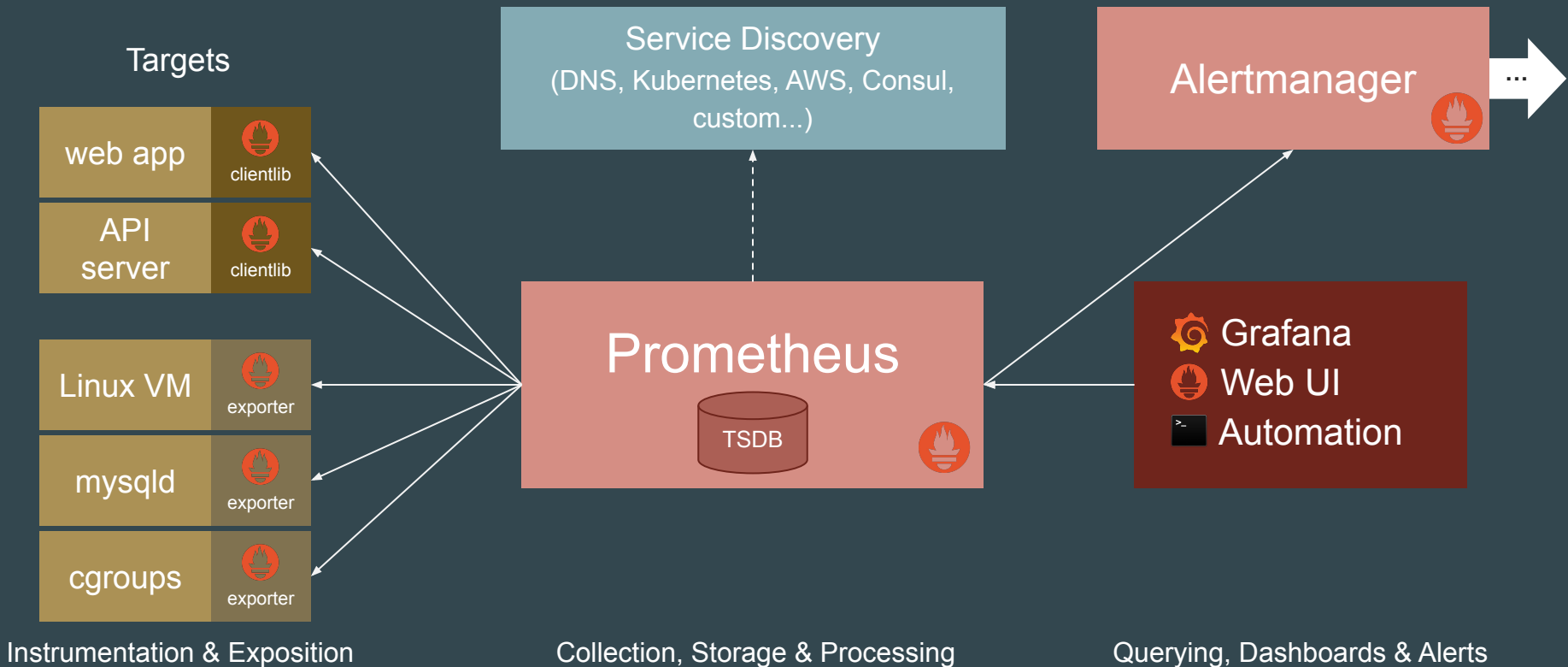# Architecture

Targets

web app   clientlib

API server   clientlib

Linux VM   exporter

mysqld   exporter

cgroups   exporter

Prometheus

TSDB

Instrumentation & Exposition   Collection, Storage & Processing

Architecture

Architecture

Targets

web app	clientlib

API server	clientlib

Linux VM	exporter

mysqld	exporter

cgroups	exporter

Service Discovery
(DNS, Kubernetes, AWS, Consul, custom...)

Prometheus
TSDB

Grafana
Web UI
Automation

Instrumentation & Exposition   Collection, Storage & Processing   Querying, Dashboards

Architecture

Selling Points

- Dimensional data model
- Powerful query language (PromQL)
- Simple & efficient server
- Service discovery integration

Data Model

What is a time series?

`<identifier>` → `[ (t0, v0), (t1, v1), … ]`

string   int64   float64

Data Model

What identifies a time series?

```
http_requests_total{job="nginx",instance="1.2.3.4:80",status="200"} 28
```

metric name   labels

- Flexible
- No hierarchy
- Explicit dimensions

Querying

Querying

PromQL

- New query language

- Great for time series computations

- **Not** SQL-style

Querying

All partitions in my entire infrastructure with more than 100GB capacity that are not mounted on root?

```
node_filesystem_bytes_total{mountpoint!="/"} / 1e9 > 100
```

```
{device="sda1", mountpoint="/home", instance="10.0.0.1"}      118.8

{device="sda1", mountpoint="/home", instance="10.0.0.2"}      118.8

{device="sdb1", mountpoint="/data", instance="10.0.0.2"}      451.2

{device="xdvc", mountpoint="/mnt", instance="10.0.0.3"}       320.0
```

Querying

What's the ratio of request errors across all service instances?

```
    sum(rate(http_requests_total{status="500"}[5m]))
 / sum(rate(http_requests_total[5m]))
```

```
{}                                0.029
```

Querying

What's the ratio of request errors across all service instances?

```
    sum by(path) (rate(http_requests_total{status="500"}[5m]))
  / sum by(path) (rate(http_requests_total[5m]))
```

{path="/status"}	0.0039
{path="/"}	0.0011
{path="/api/v1/topics/:topic"}	0.087
{path="/api/v1/topics}	0.0342

Querying

99th percentile request latency across all instances?

```
histogram_quantile(0.99,
    sum without(instance) (rate(request_latency_seconds_bucket[5m]))
)
```

```
{path="/status", method="GET"}                    0.012
{path="/", method="GET"}                           0.43
{path="/api/v1/topics/:topic", method="POST"}      1.31
{path="/api/v1/topics, method="GET"}               0.192
```

Expression browser
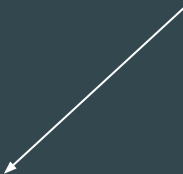
Built-in graphing

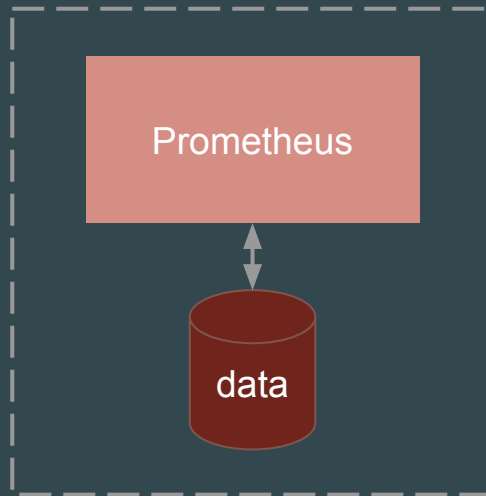# Dashboarding

Alerting

generate an alert for each
path with an error rate of >5%

```
alert: Many500Errors
expr: |
  (
      sum by(path) (rate(http_requests_total{status="500"}[5m]))
    /
      sum by(path) (rate(http_requests_total[5m]))
  ) * 100 > 5
for: 5m
labels:
  severity: "critical"
annotations:
  summary: "Many 500 errors for path {{$labels.path}} ({{$value}}%)"
```

Operational Simplicity

- Local storage, no clustering
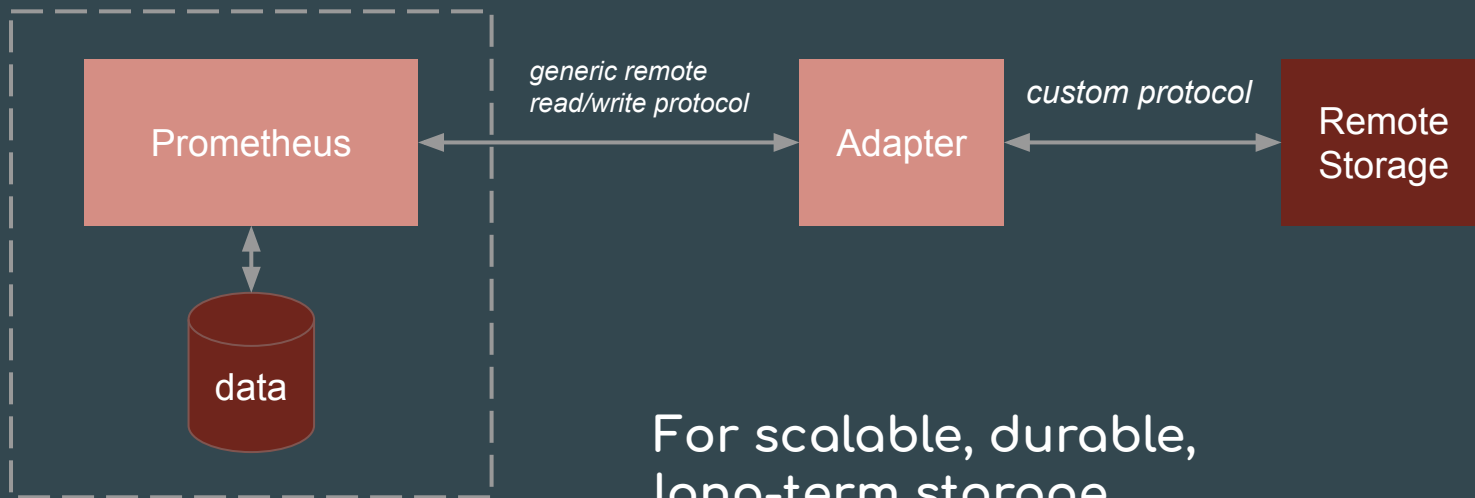- HA by running two
- Go: static binary

Efficiency

Local storage is scalable enough for many orgs:

- 1 million+ samples/s
- Millions of series
- 1-2 bytes per sample

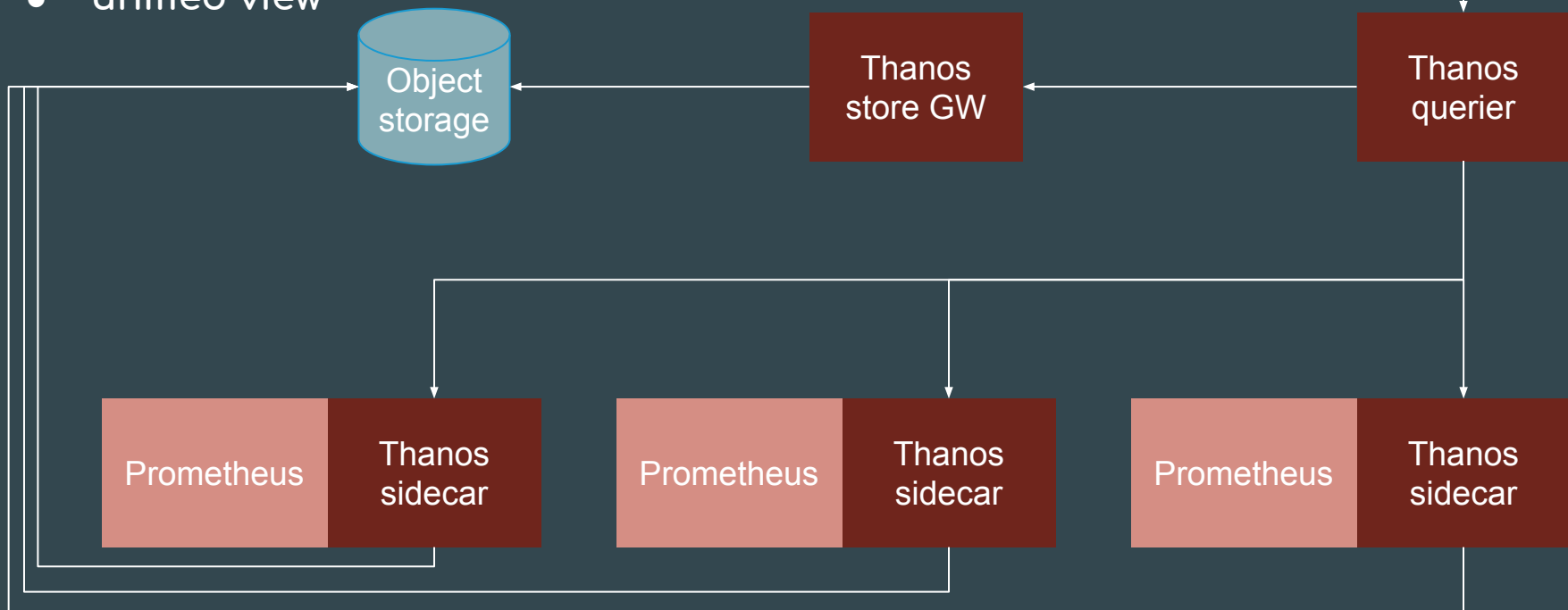Good for keeping a few weeks or months of data.

Decoupled Remote Storage

Prometheus ←generic remote read/write protocol→ Adapter ←custom protocol→ Remote Storage

data

For scalable, durable, long-term storage.

E.g.: Cortex, InfluxDB

Or: thanos.io

Thanos

- long-term storage
- durability
- unified view

Dynamic Environments

...pose new challenges:

- Dynamic VMs
- Cluster schedulers
- Microservices

→ many services, dynamic hosts, and ports

How to make sense of this all?

Service Discovery

Use service discovery to:

- ...know what *should* be there
- ...decide where to pull from
- ...add dimensional metadata to series

Service Discovery

Prometheus has built-in support for:

- VM providers (AWS, Azure, Google, …)
- Cluster managers (Kubernetes, Marathon, …)
- Generic mechanisms (DNS, Consul, Zookeeper, custom, …)

Conclusion

Prometheus helps you make sense of complex dynamic environments via its:

- Dimensional data model
- Powerful query language
- Simplicity + efficiency
- Service discovery integration

Thanks!