Systems and Service Monitoring with Prometheus

Julius Volz

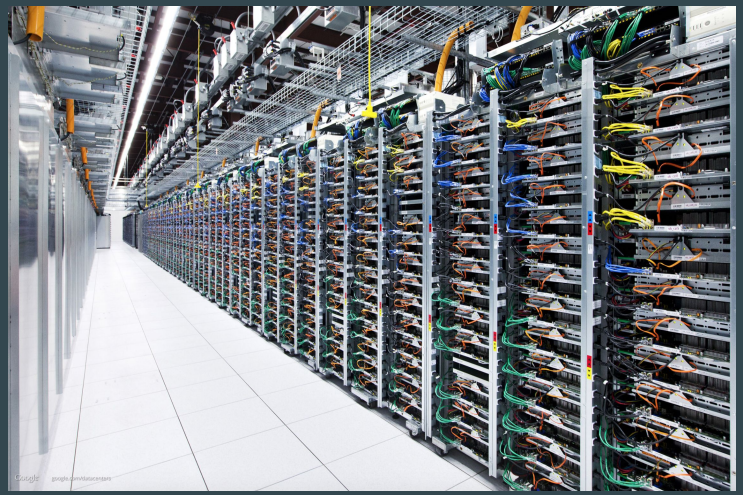
Co-founder,
Prometheus

@juliusvolz

Monitoring: A Primer

Systems/Service Goals

Want to ensure that a system or service is:



Google datacenter

- Available
- Fast
- Correct
- Efficient
- ...

Reality: systems are complicated and break a lot.

Potential Problems

Examples:

- **Disk full** □ no new data stored
- **Software bug** □ request errors
- **High temperature** □ hardware failure
- **Network outage** □ services cannot communicate
- **Low memory utilization** □ money wasted

Monitoring

Need to **observe** your systems to get insight into:

- Request / event rates
- Latency
- Errors
- Resource usage
- ...Temperature, humidity, ...

...and then **react** when something looks bad.

Types of Monitoring

Check-Based Monitoring

Run scripts periodically to check health right now

Host	Service	Status	Last Check	Duration	Attempt	Status Information
equallogic01	Disk Status 	CRITICAL	11-20-2009 08:50:12	0d 15h 47m 43s	3/3	DISK CRITICAL 1 disk(s) in critical state
	Disk Usage	OK	11-20-2009 08:50:11	0d 15h 47m 43s	1/3	DISKUSAGE OK Total 8230125 Used 0 (0%)
	Ethernet Interfaces	OK	11-20-2009 08:50:11	0d 15h 32m 13s	1/3	2 controller(s) present eth0 (0:9:8a:2:e5:b8) is up eth1 (0:9:8a:2:e5:b9) is down eth2 (0:9:8a:2:e5:ba) is down
	General Health	OK	11-20-2009 08:50:11	0d 15h 47m 43s	1/3	OVERALL HEALTH OK
	General Information	OK	11-20-2009 08:50:11	0d 15h 47m 43s	1/3	Equallogic Model "70-0115" Serial No "SHM94552C010314" 2 controller(s) running FW V4.1.6 16 disks
	ISCSI Connections	OK	11-20-2009 08:50:11	0d 15h 44m 35s	1/3	CONNECTIONS OK 0 ISCSI Connections
	PING	OK	11-20-2009 08:50:11	0d 7h 14m 45s	1/2	PING OK - Packet loss = 0%, RTA = 5.24 ms
	Power Supply	OK	11-20-2009 08:50:11	0d 15h 22m 50s	1/3	PS OK "Power Cooling Module 0" on and operating "Power Cooling Module 1" on and operating
	Raid Status	OK	11-20-2009 08:50:11	0d 15h 47m 43s	1/3	RAID OK
	Uptime	OK	11-20-2009 08:50:11	0d 15h 22m 50s	1/3	Timeticks: (16151131) 1 day, 20:51:51.31

Examples: Nagios, Icinga

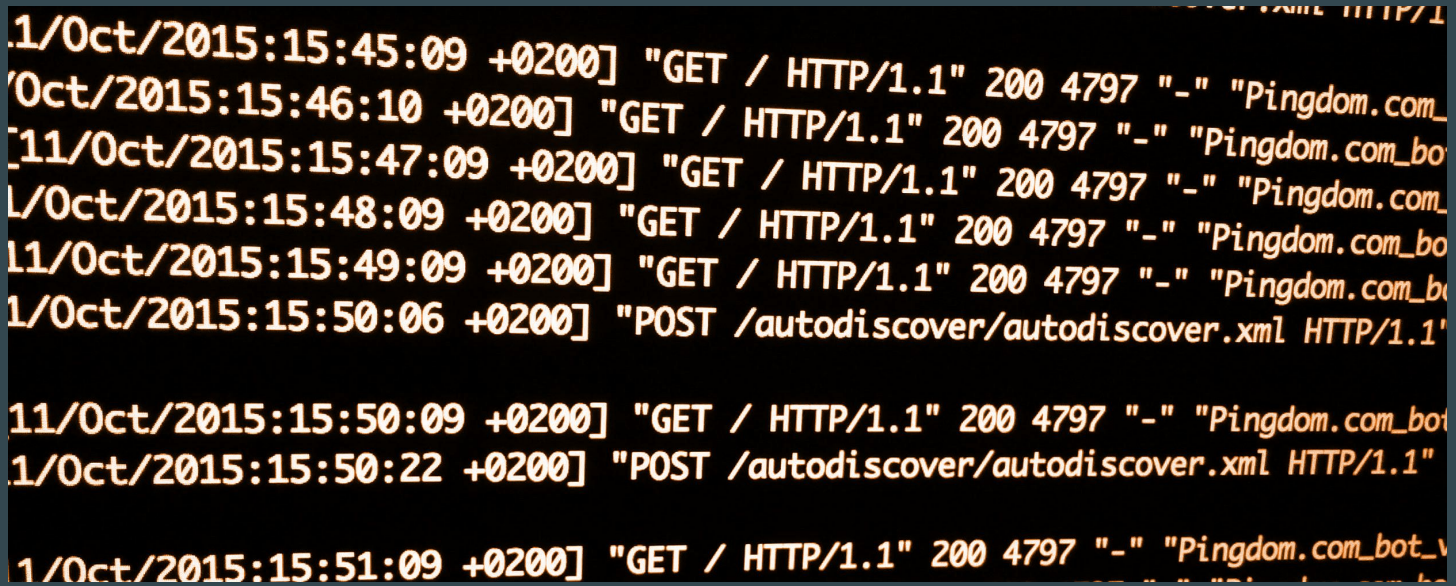
Check-Based Monitoring

+ Better than nothing!

- Focus on blackbox monitoring
- Lack of global and temporal context
- Breaks down in complex dynamic environments

Logs / Events

Record full details about each event (unstructured / structured)



Examples: local disk logging, Elasticsearch, Loki, InfluxDB

Logs / Events

+ High detail
+ Simple to produce

- Potentially very expensive
- Lack of inter-service correlation

Metrics / Time Series

Numeric values, sampled over time

Examples: StatsD, OpenTSDB, Ganglia, InfluxDB, Prometheus

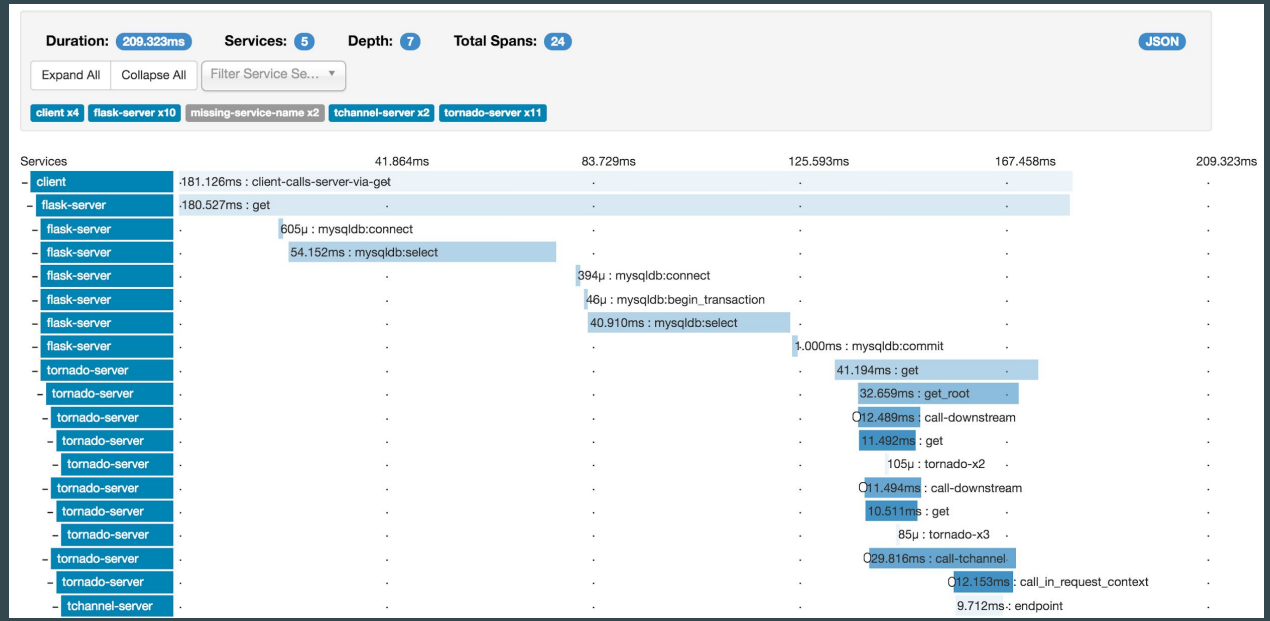
Metrics / Time Series

+ Cheap
+ Good for aggregate health monitoring
+ Simple to produce

- Less detail for debugging
- Lack of inter-service correlation

Request Tracing

Track single requests through entire stack



Examples: Zipkin, Jaeger

Request Tracing

+ Provides inter-service correlation
+ Detailed request insight

- Expensive (sampling helps)
- Requires cooperation of full path
- Only suitable for request-style info

Prometheus

What is Prometheus?

Metrics-based monitoring & alerting stack.

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

Made for dynamic cloud environments.

What is it not?

We don't do:

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

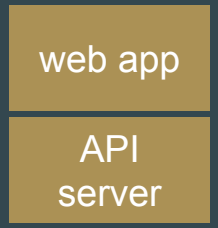
History

- Started 2012 at SoundCloud
- Motivation: Lack of suitable OSS tools
- Part of CNCF since 2016
- Not a company!
- Find us at: https://prometheus.io/

Architecture

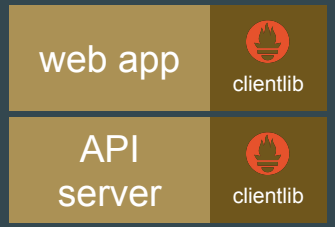
Architecture

Targets



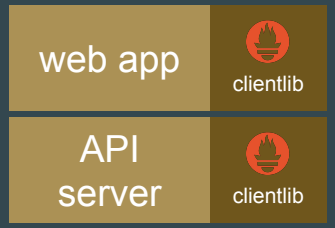
Architecture

Targets



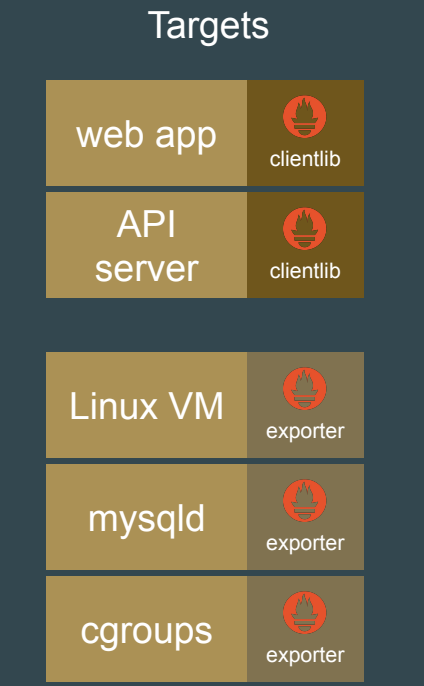
Architecture

Targets

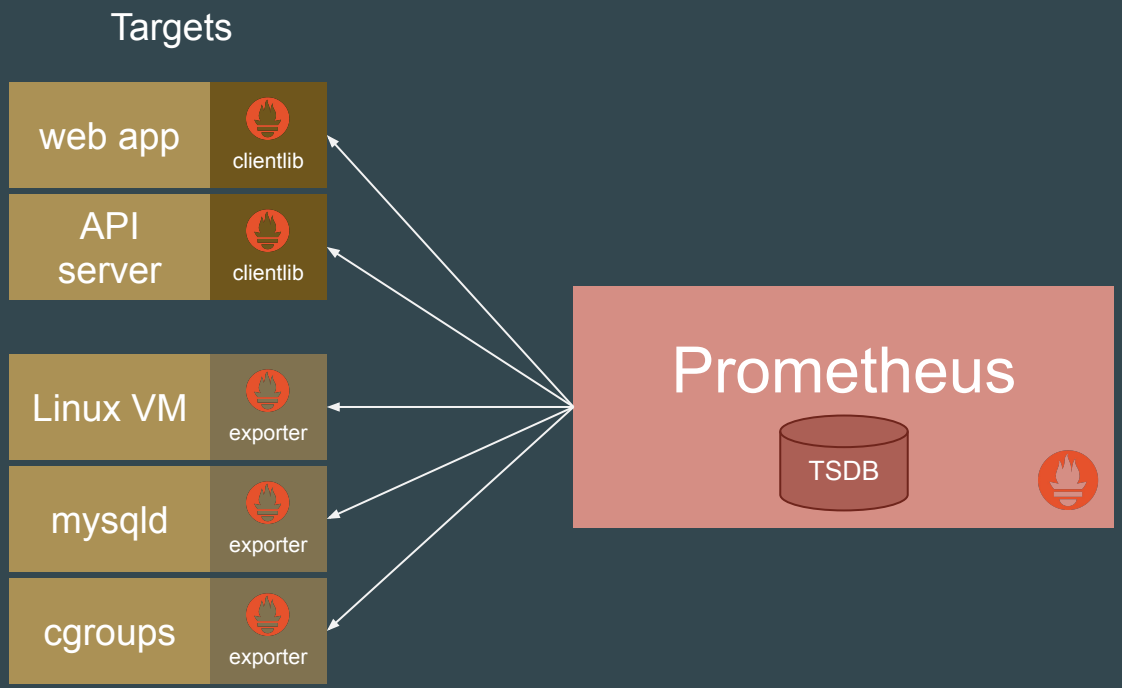


Instrumentation & Exposition

Architecture



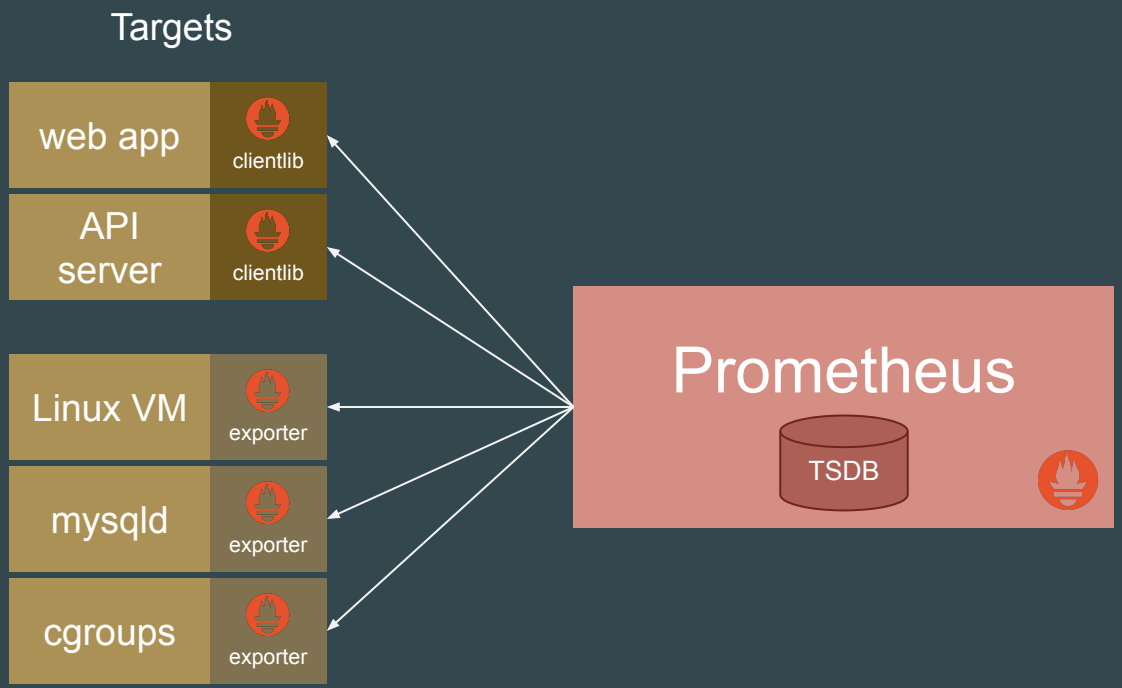
Architecture



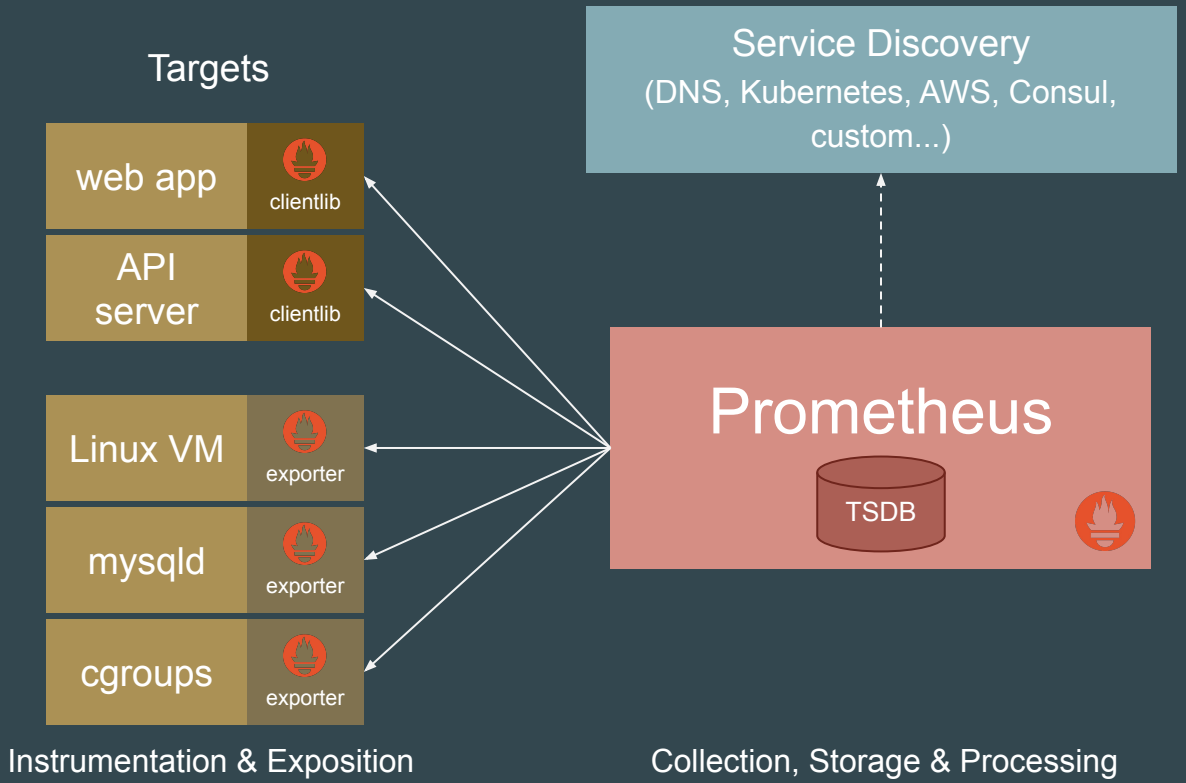
Interlude: Exposition Format

```
# HELP http_requests_total The 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

# HELP http_request_duration_seconds A histogram of the request duration.
# TYPE http_request_duration_seconds histogram
http_request_duration_seconds_bucket{le="0.05"} 24054
http_request_duration_seconds_bucket{le="0.1"} 33444
http_request_duration_seconds_bucket{le="0.2"} 100392
http_request_duration_seconds_bucket{le="0.5"} 129389
http_request_duration_seconds_bucket{le="1"} 133988
http_request_duration_seconds_bucket{le="+Inf"} 144320
http_request_duration_seconds_sum53423
http_request_duration_seconds_count144320

# HELP rpc_duration_seconds A summary of the RPC duration in seconds.
# TYPE rpc_duration_seconds summary
rpc_duration_seconds{quantile="0.01"} 3102
rpc_duration_seconds{quantile="0.05"} 3272
rpc_duration_seconds{quantile="0.5"} 4773
rpc_duration_seconds{quantile="0.9"} 9001
rpc_duration_seconds{quantile="0.99"} 76656
rpc_duration_seconds_sum 1.7560473e+07
rpc_duration_seconds_count 2693
```

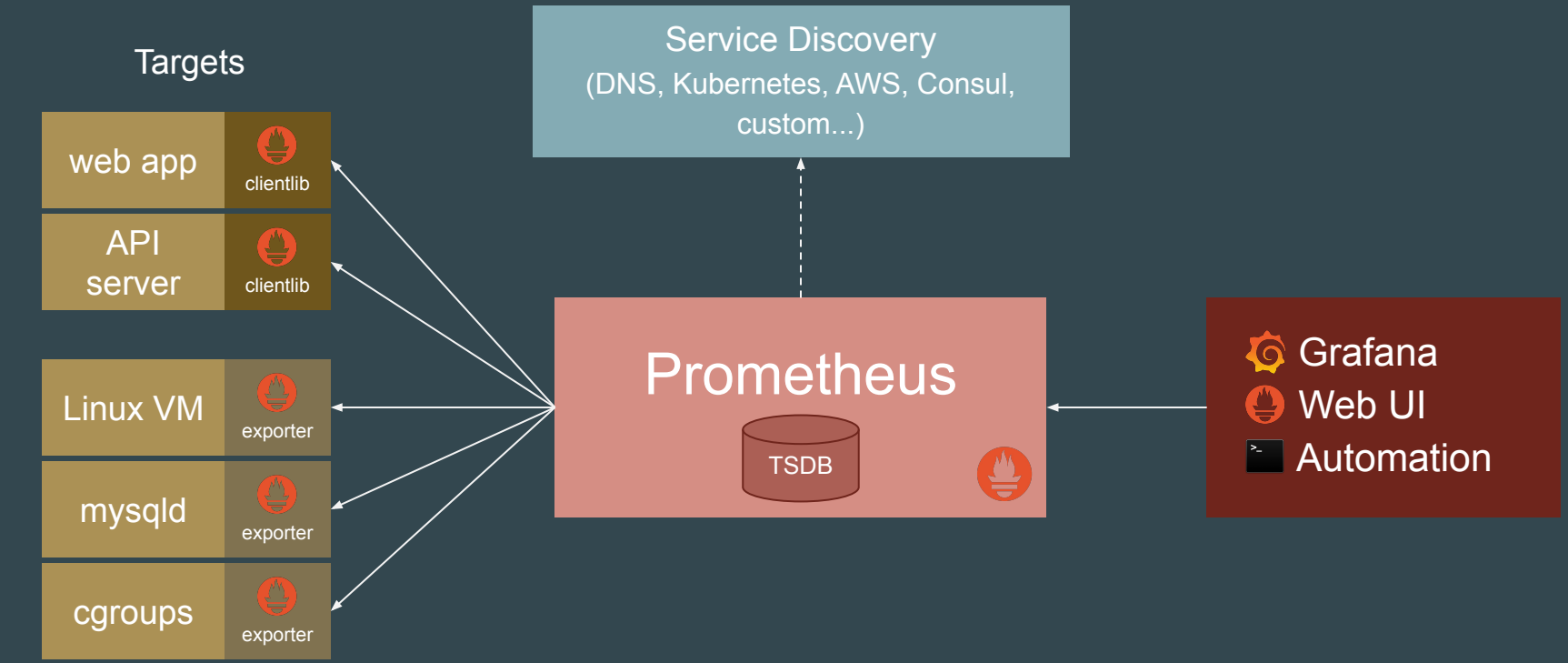
Architecture



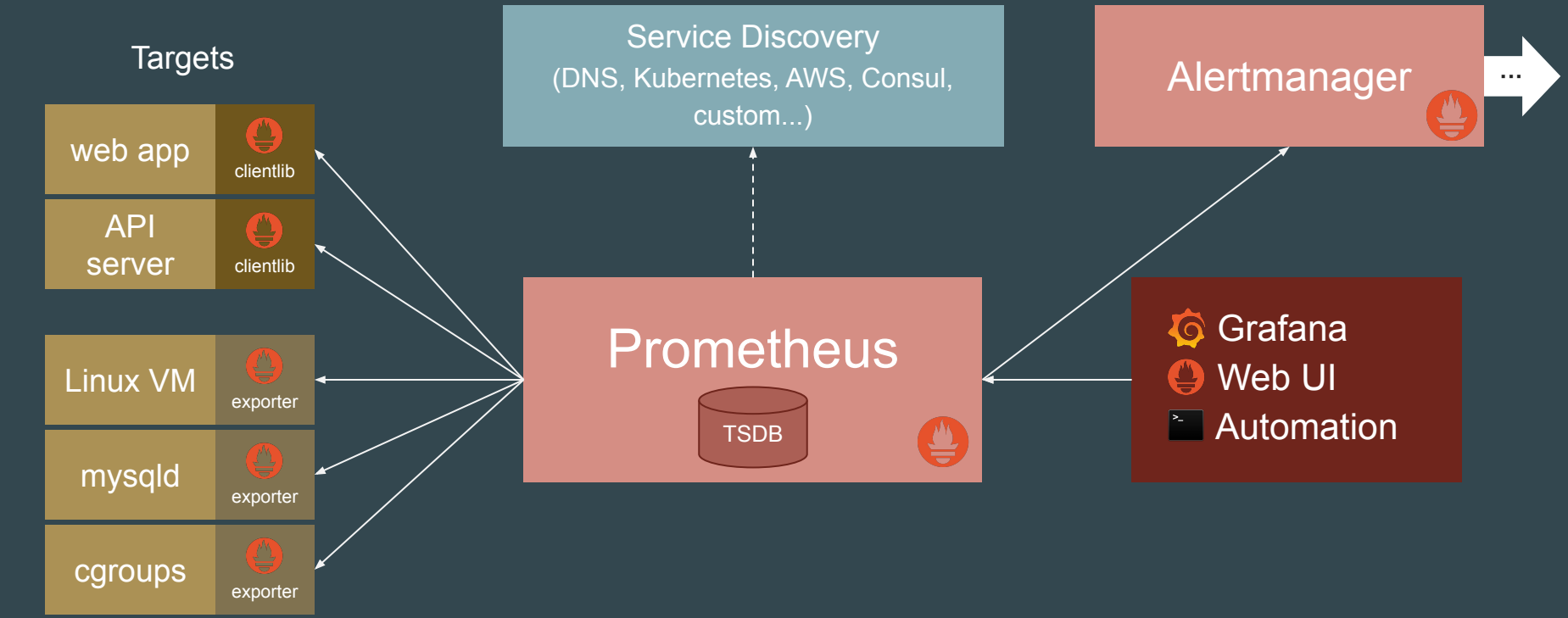
Architecture



Architecture



Architecture



Favorite Features

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

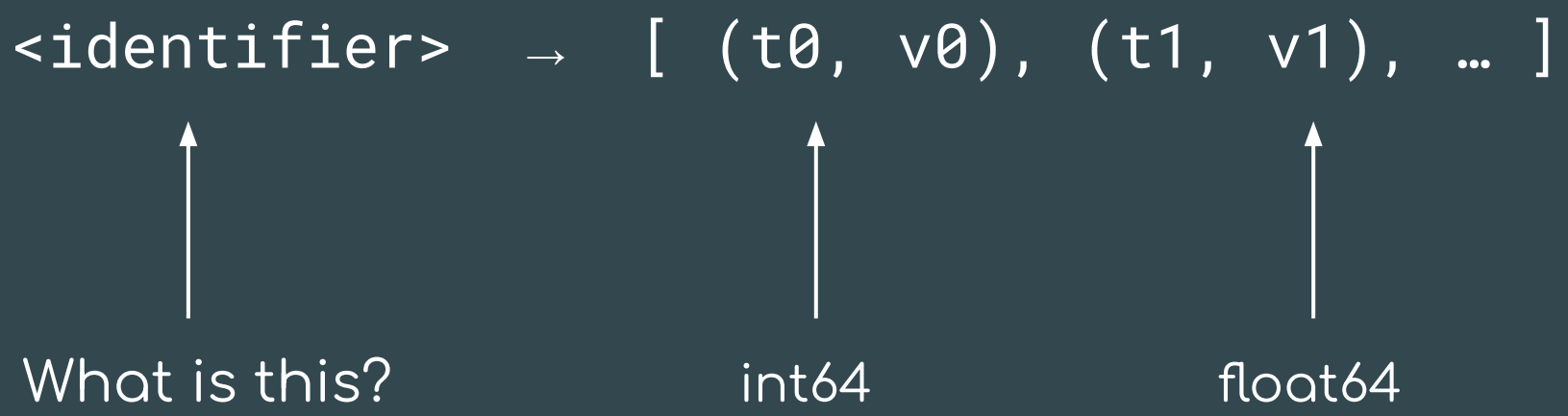
Data Model

What is a time series?

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

Data Model

What is a time series?



Data Model

What identifies a time series?

```
http_requests_total{job="nginx",instance="1.2.3.4:80",path="/home",status="200"}
```

Data Model

What identifies a time series?

```
http_requests_total{job="nginx",instance="1.2.3.4:80",path="/home",status="200"}
```

metric name

labels

Data Model

What identifies a time series?

```
http_requests_total{job="nginx",instance="1.2.3.4:80",path="/home",status="200"}
```

metric name

labels

- Flexible
- No hierarchy
- Explicit dimensions

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

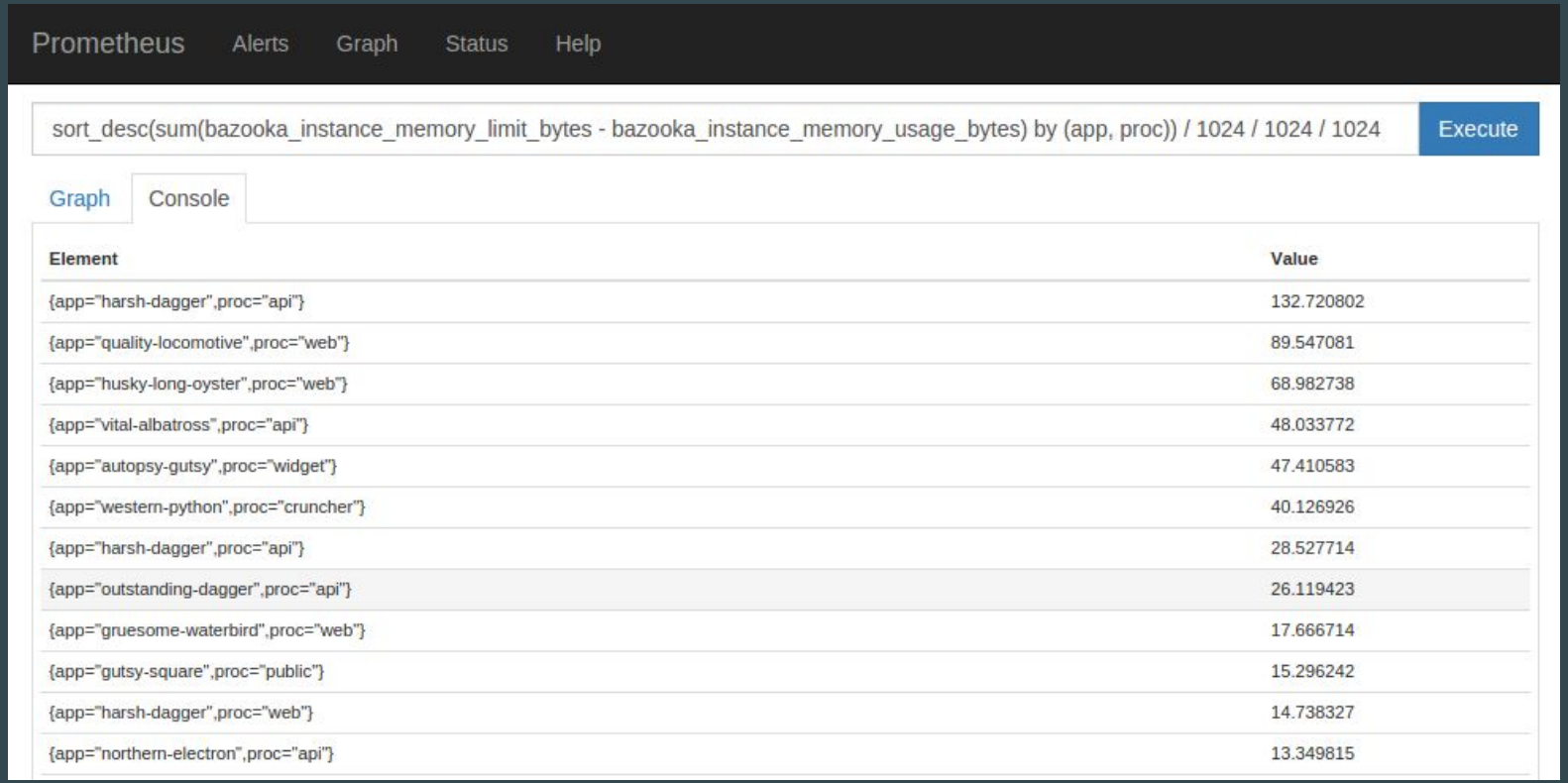
Querying

99th percentile request latency across all instances?

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

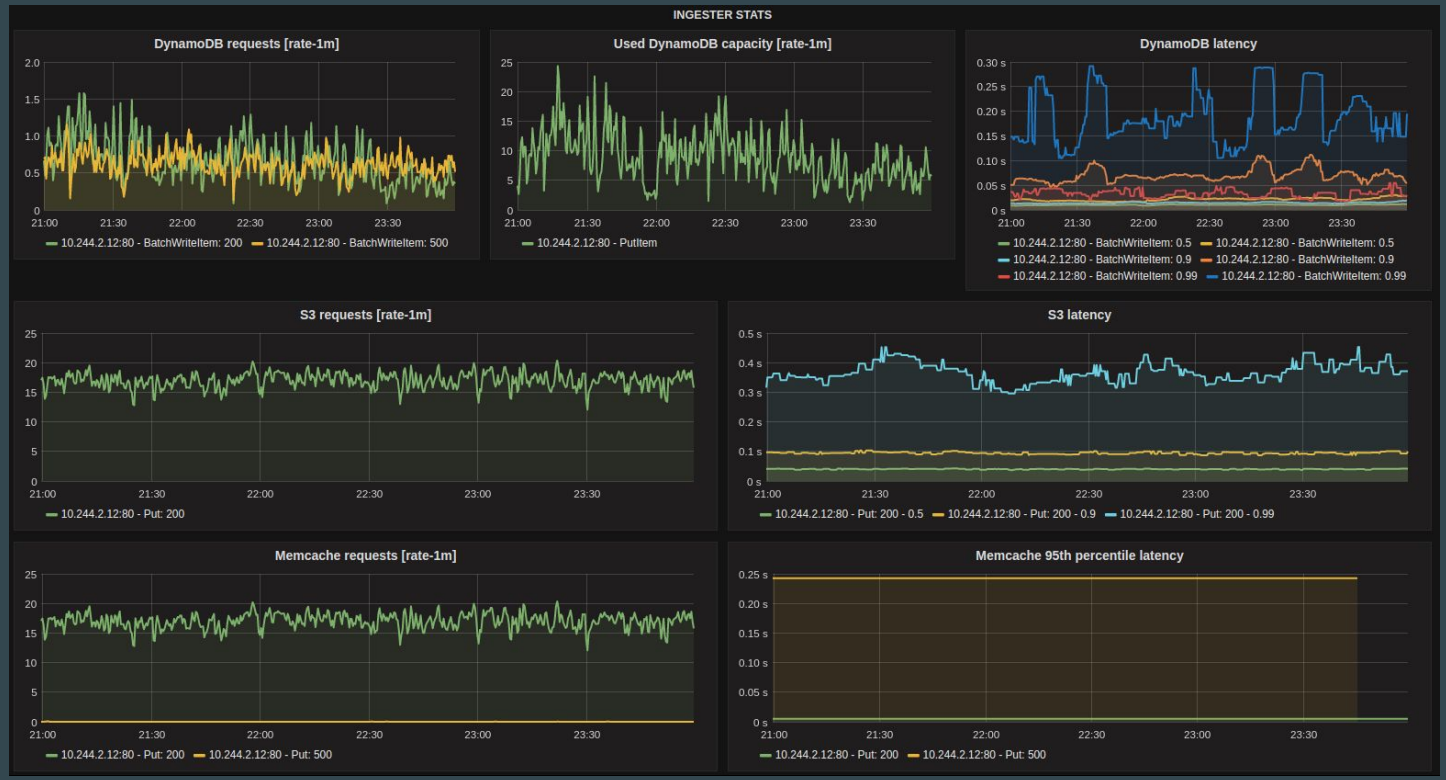
```
{path="/status", method="GET"}                          0.012
{path="/", method="GET"}                                0.43
{path="/api/v1/topics/:topic", method="POST"}           1.31
{path="/api/v1/topics, method="GET"}                    0.192
```

Expression browser



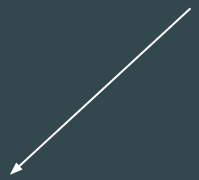
Built-in graphing

Dashboarding via Grafana



Alerting

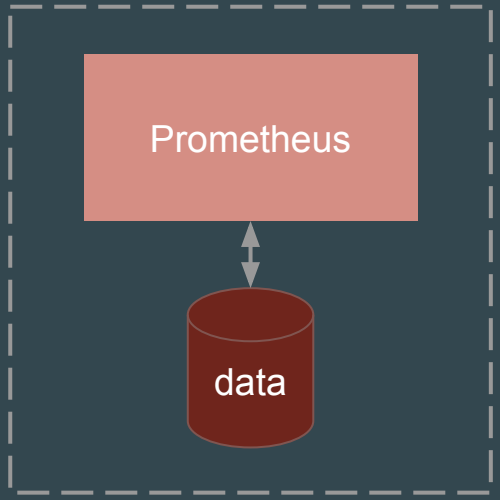
generate an alert for each
path with an error rate of >5%



```
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



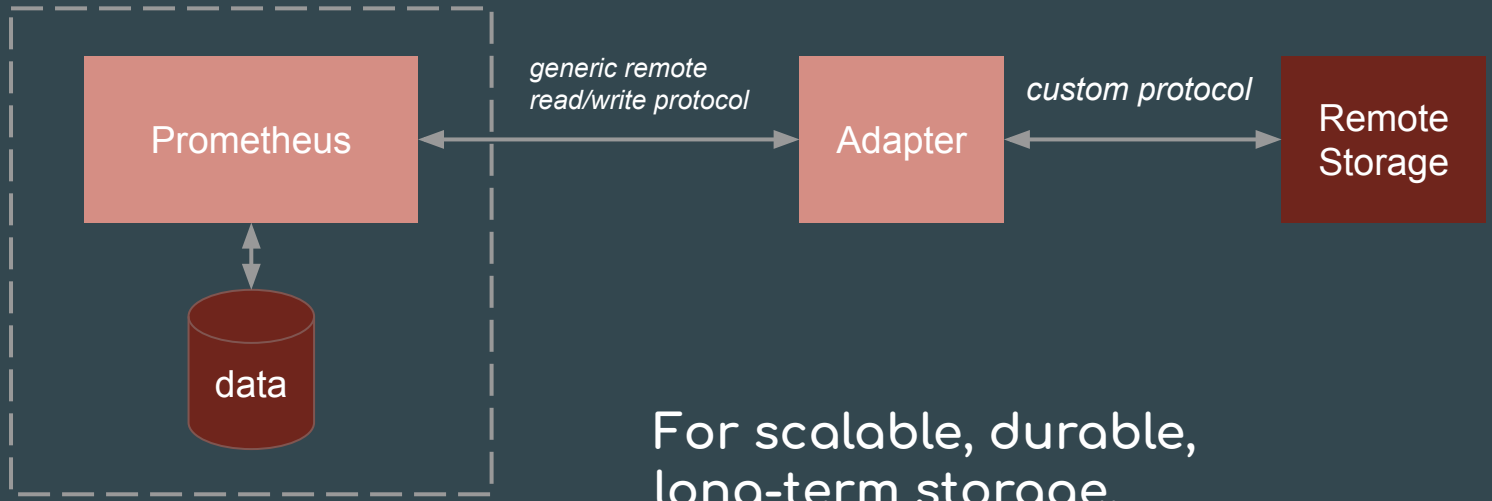
Efficiency

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

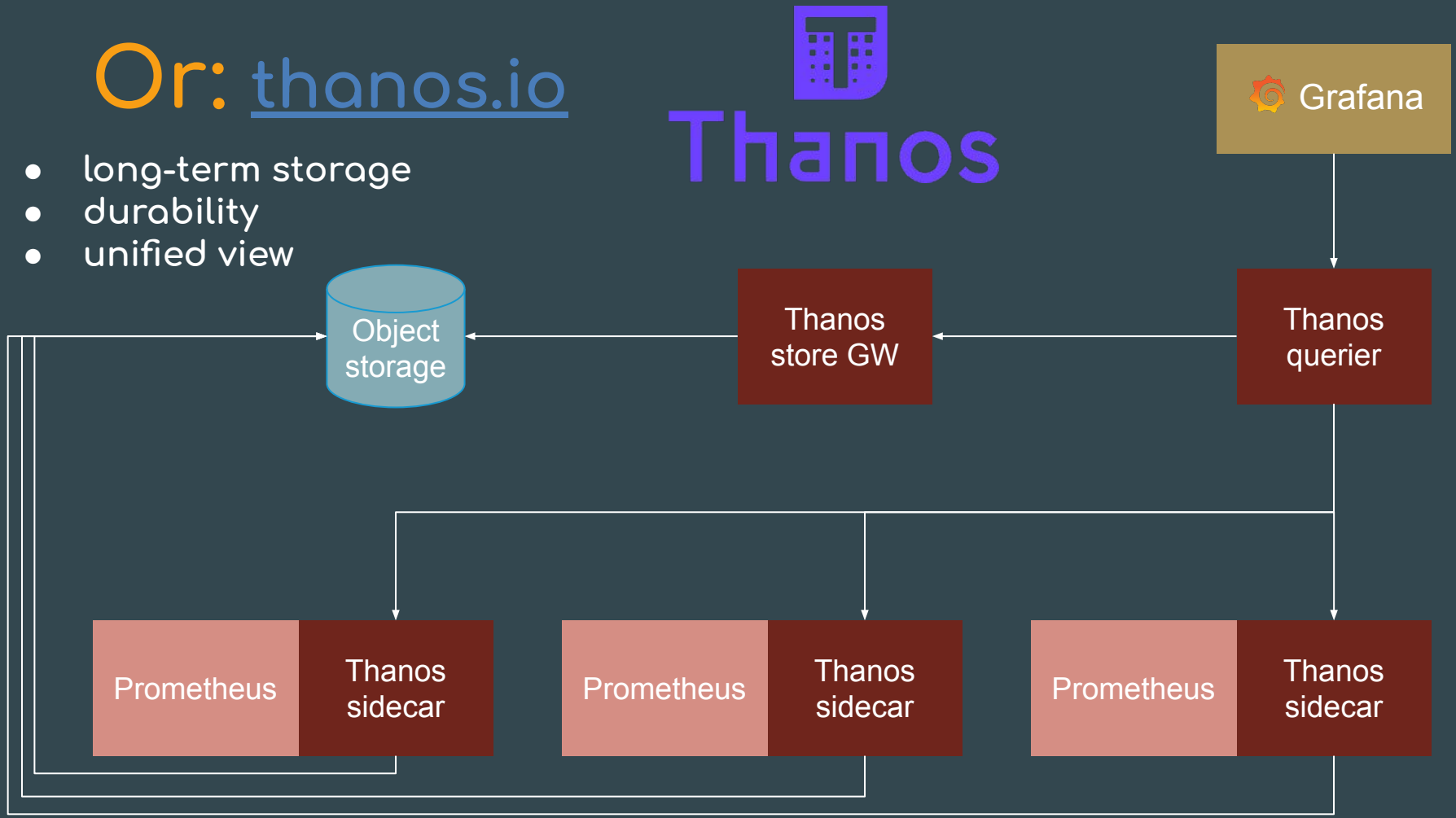


For scalable, durable,
long-term storage.

E.g.: Cortex, InfluxDB

Or: thanos.io

- long-term storage
- durability
- unified view



Dynamic Environments

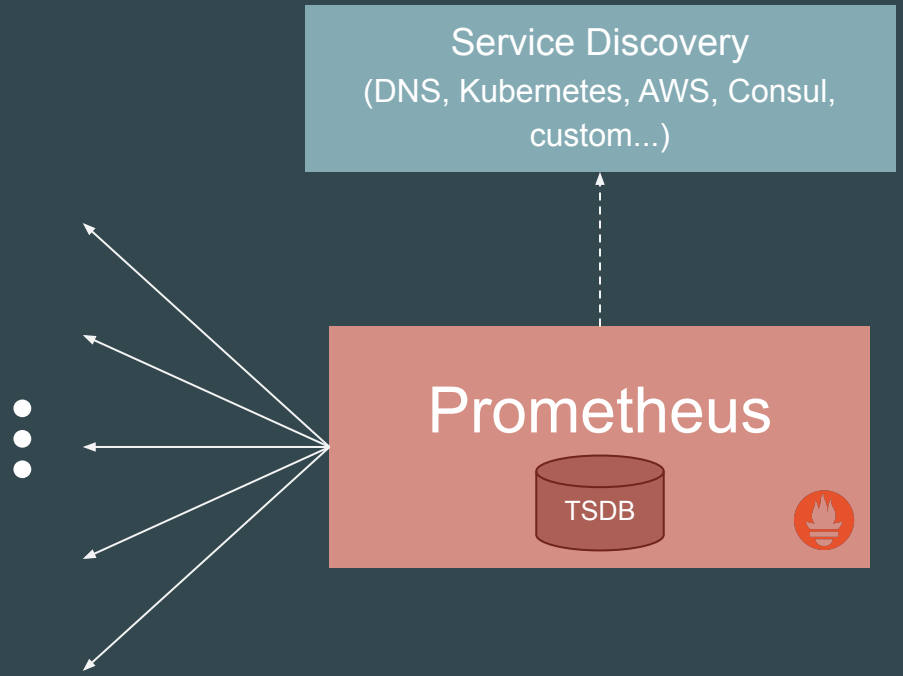
...pose new challenges:

- Dynamic VMs
- Cluster schedulers
- Microservices

→ many services, dynamic hosts, and ports

How to make sense of this all?

Service Discovery Integration



Answers three questions:

- what **should** be there?
- how do I reach it?
- what is it? (metadata)

Conclusion

Prometheus helps you make sense of complex dynamic environments thanks to its:

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

Thanks!