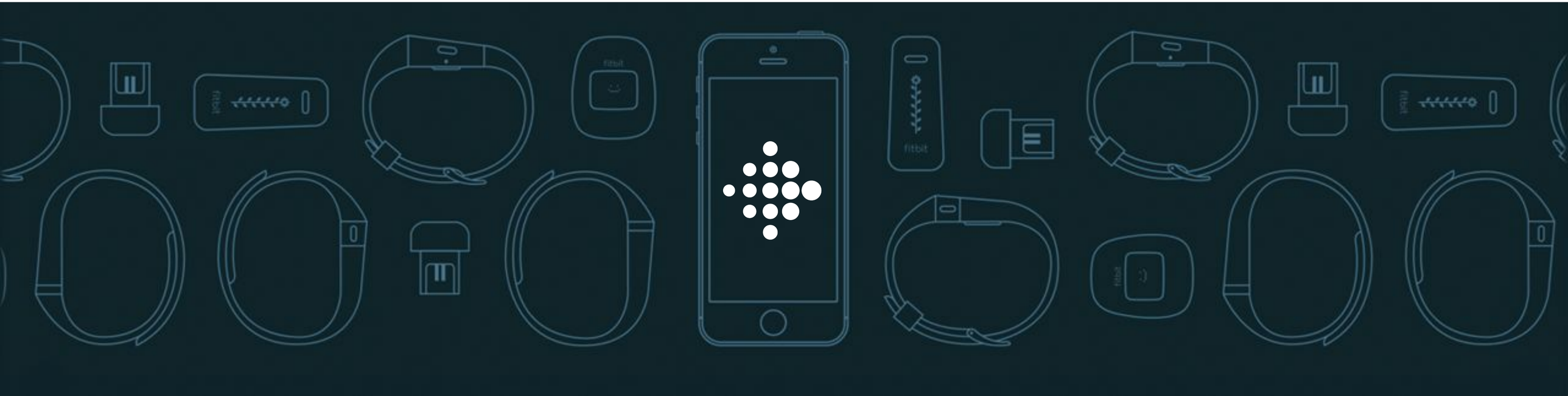
MONITORING AT FITBIT

Anton Kholodok



TRADE-OFF

granularity / resources

complexity / costs

pull / push

etc

PART I

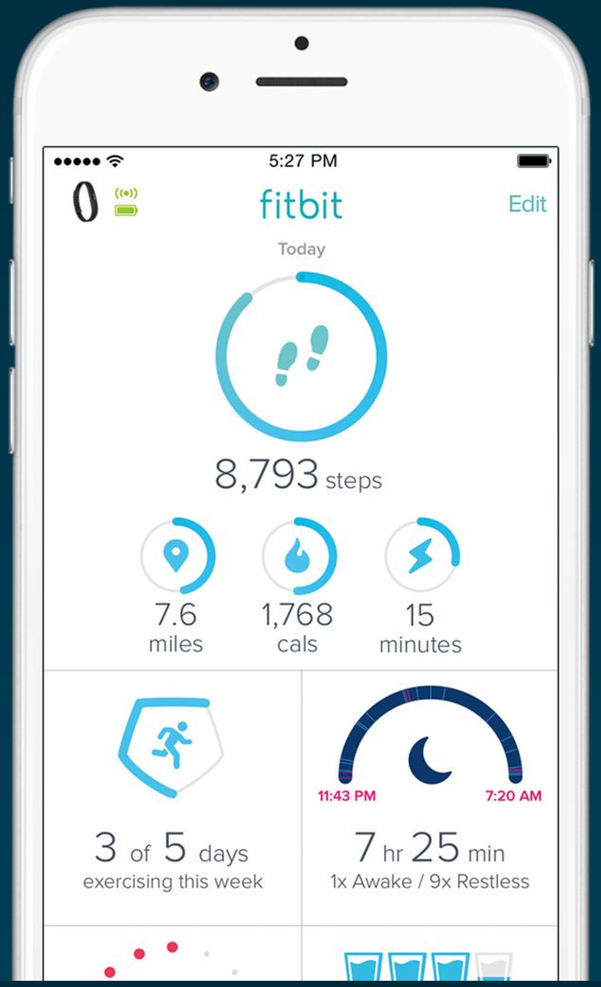
monitoring basics

BUSINESS VALUE

CODE

features

improvements



refactoring

bug fixes

CODE

BUSINESS VALUE

DECISION

reality NOT assumptions

MEASUREMENT

production

MONITORING

collecting, processing, aggregating, and displaying real-time data about a system

METRICS

LOGS

VISIBILITY

throughput / latency / errors / etc

ANALYSIS

ad-hoc retrospective debugging and correlations

short & long term

PATTERN

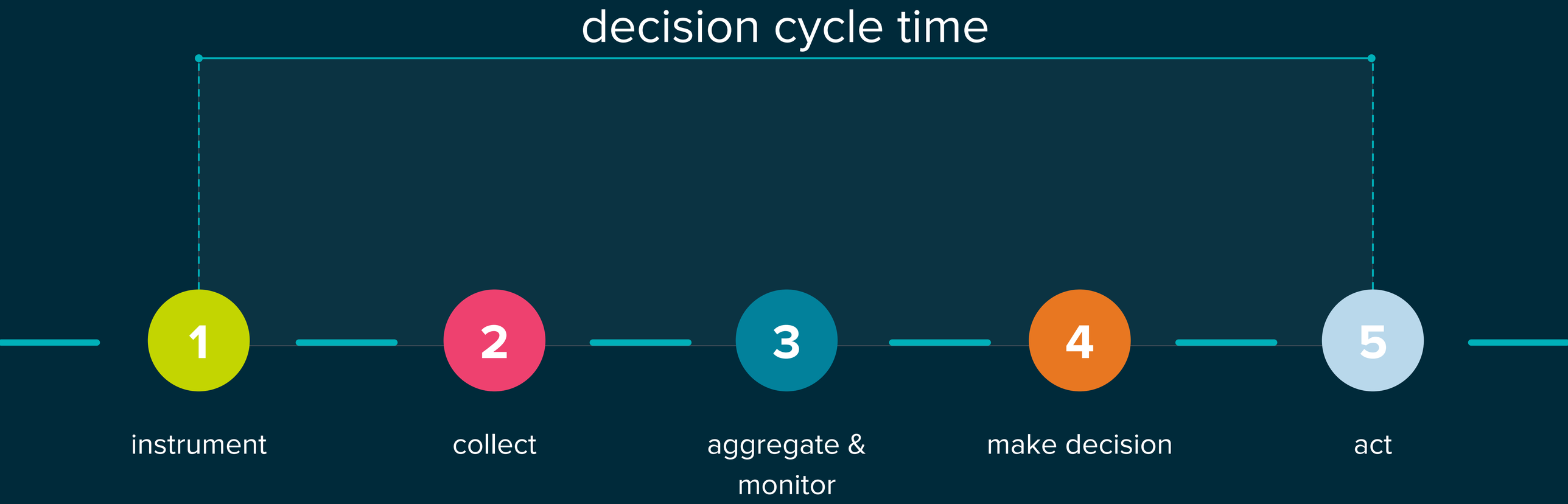
that could be analyzed

DECISION CYCLE

“sequence of steps used by an entity on a repeated basis to reach and implement decisions and to learn from the results”

Wikipedia

DECISION CYCLE



DECISION CYCLE

LESS decision cycle time → **FASTER** code delivery → **MORE** business value

WHAT TO MEASURE?

what may negatively affect business value

SLI

SERVICE LEVEL INDICATORS

latency
throughput
error rate
...

SLO

SERVICE LEVEL OBJECTIVES

SLO (upper bound)

SLI

SLO (lower bound)

SLA

SERVICE LEVEL AGREEMENTS

$

GET /user/{id}/stats

```
{
  “steps”: 7000,
  “floors”: 10,
  “distance”: 6500
  ...
}
```

SLI

response time in ms

SLO

“99% or more of the requests occuring on a 24-hour calendar day (UTC) shall be serviced in 100ms or less time”

SYMPTOMS

VS

CAUSES

METHODOLOGIES

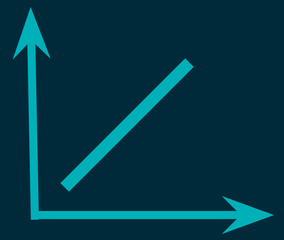
USE (Brendan Gregg)	SRE Golden Signals (Google SRE Book)	RED (Tom Wilkie)
utilization saturation errors	traffic error latency saturation	rate errors duration
resources & infrastructure oriented	distributed systems oriented	microservices oriented
causes	symptoms	symptoms

BLACKBOX

VS

WHITEBOX

METRIC TYPES



COUNTER

a cumulative value that can only go up

GAUGE

instantaneous value that can arbitrarily go up / down

TIMER

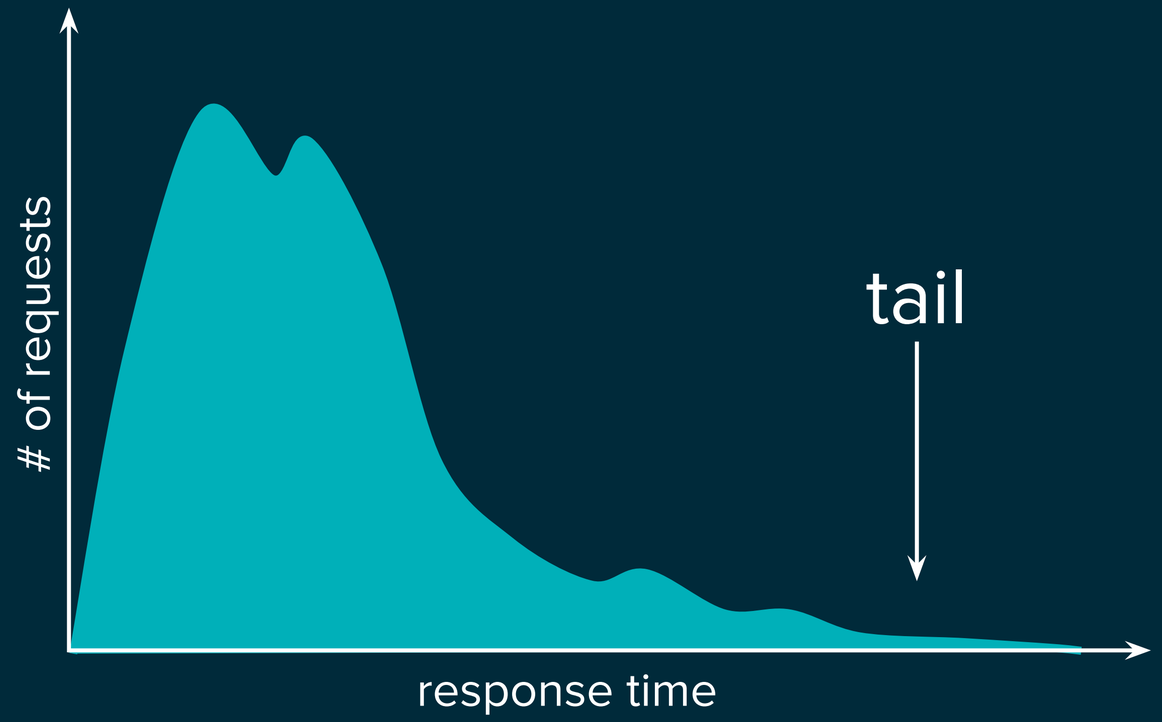
event duration in predefined time unit

HISTOGRAM

statistical distribution of values in a stream of data

OUTLIERS

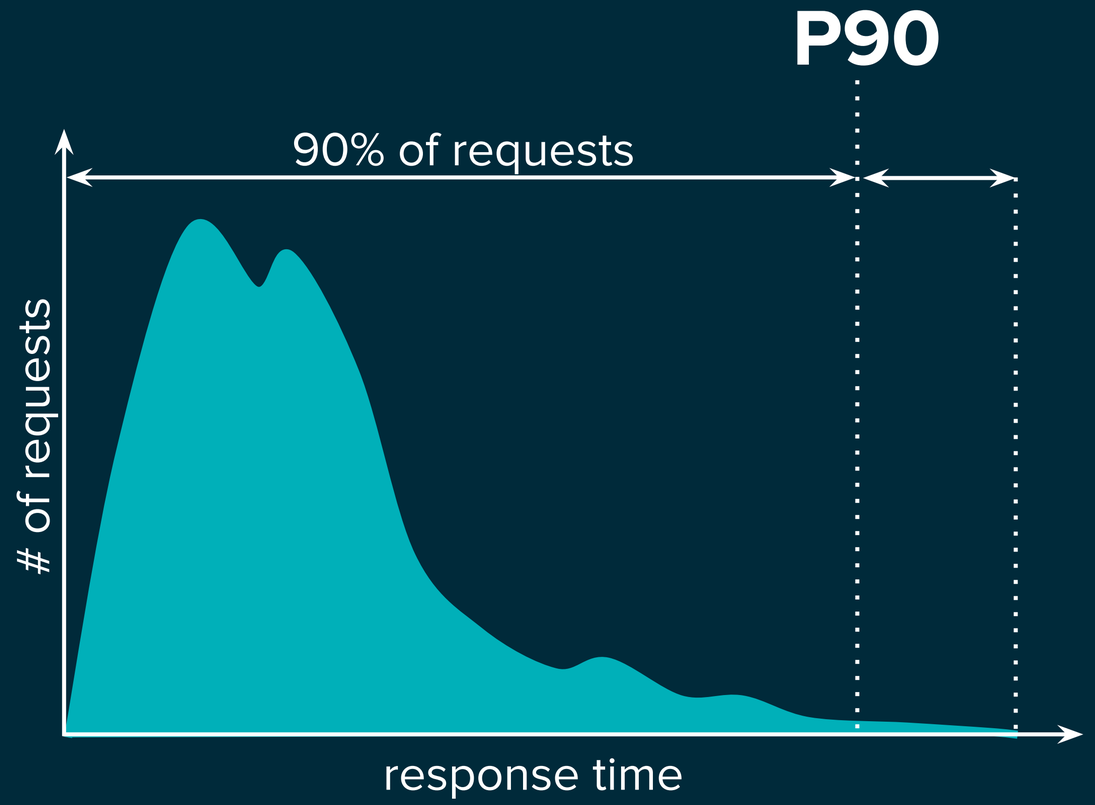
nor typical not usual behavior



PERCENTILES

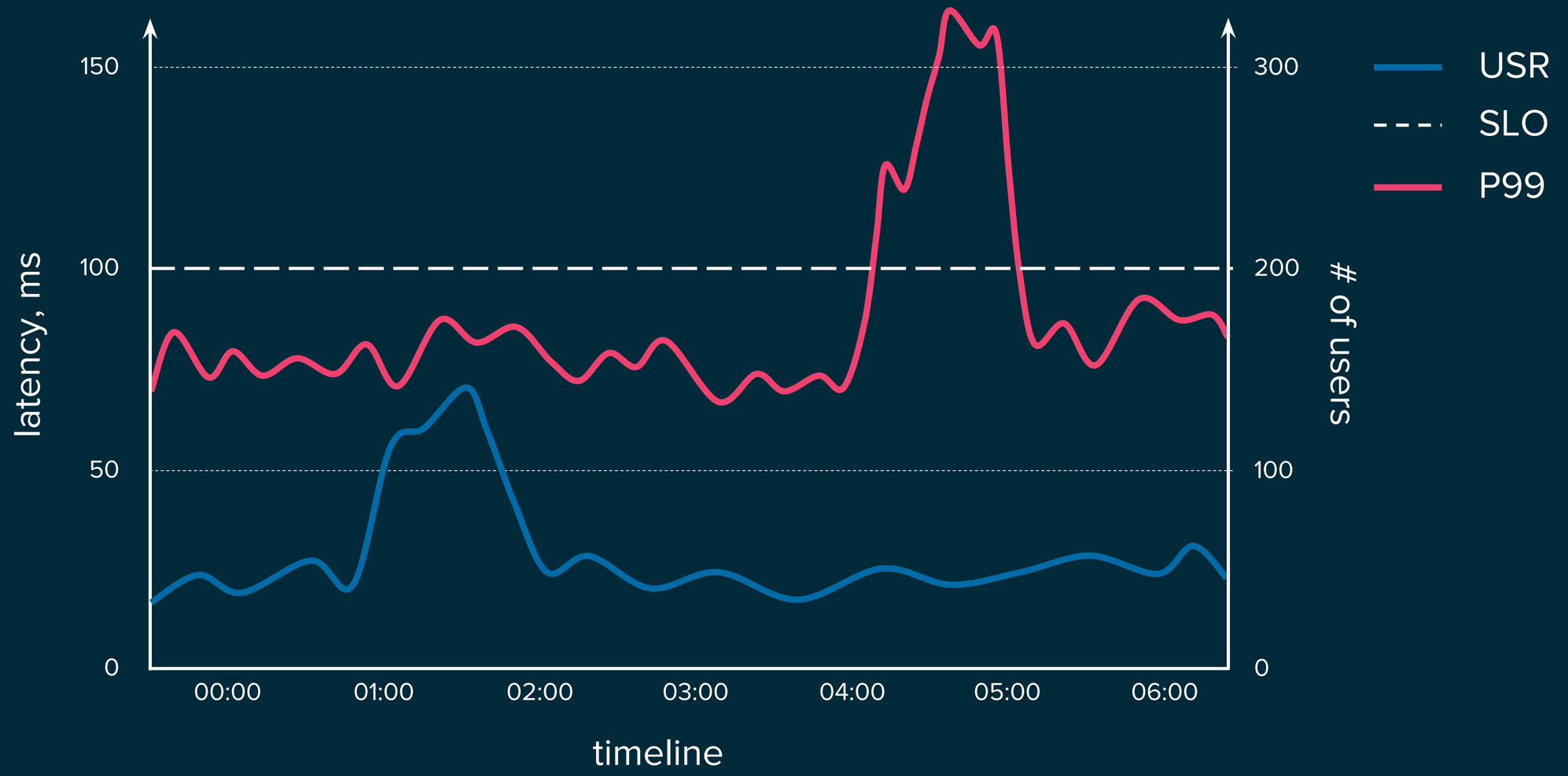
“measure used in statistics indicating the value below which a given percentage of observations in a group of observations fall”

Wikipedia



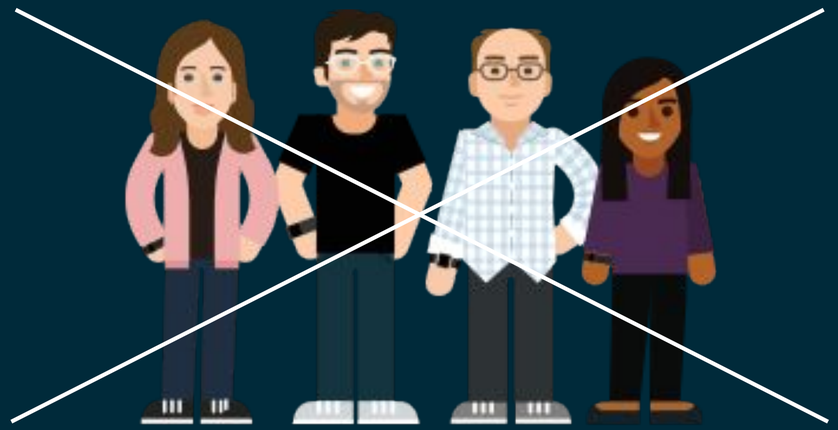
AVERAGE vs 99-PERCENTILE LATENCY

99-PERCENTILE AND USERS



PERCENTILES

aren’t people

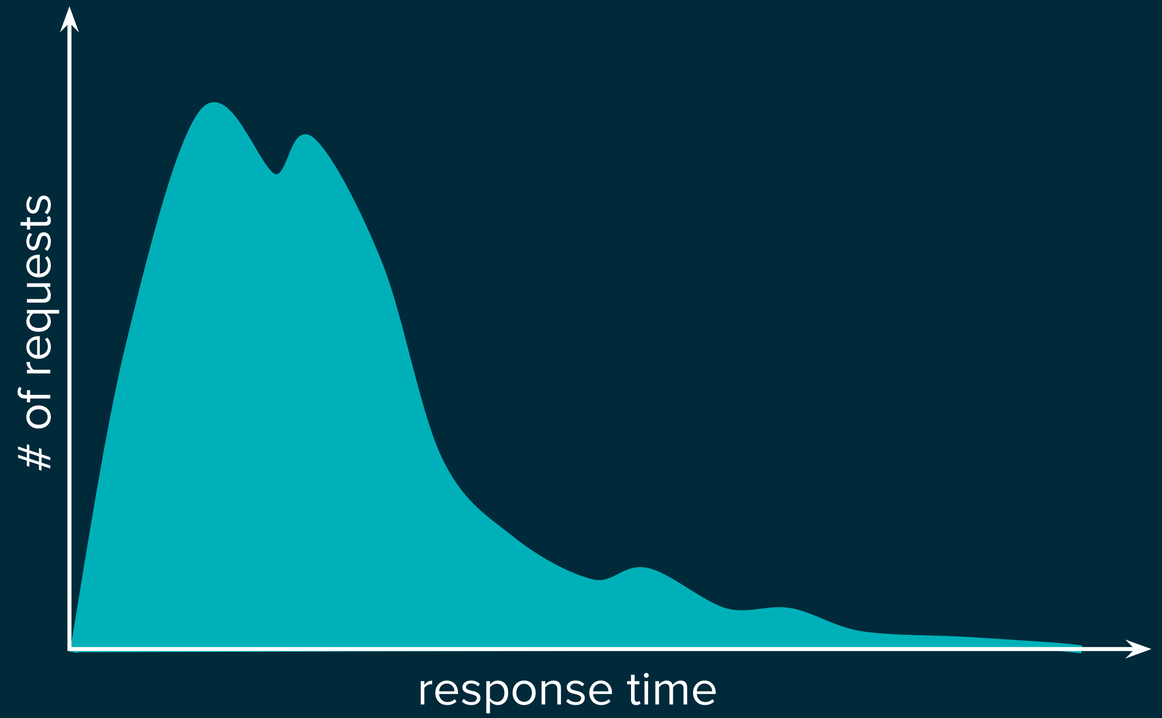


PERCENTILES

aren’t people

BUT

INDICATORS



percentiles (pN)

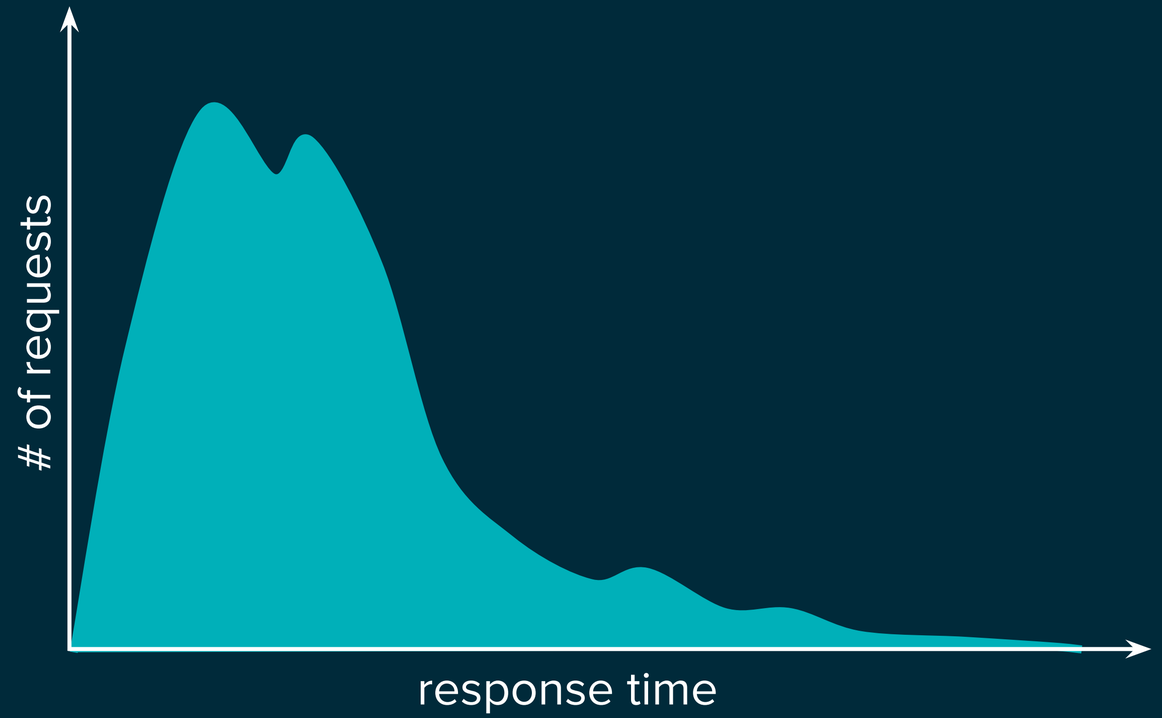
INDICATORS

percentiles (pN)

min

max (p100)

INDICATORS



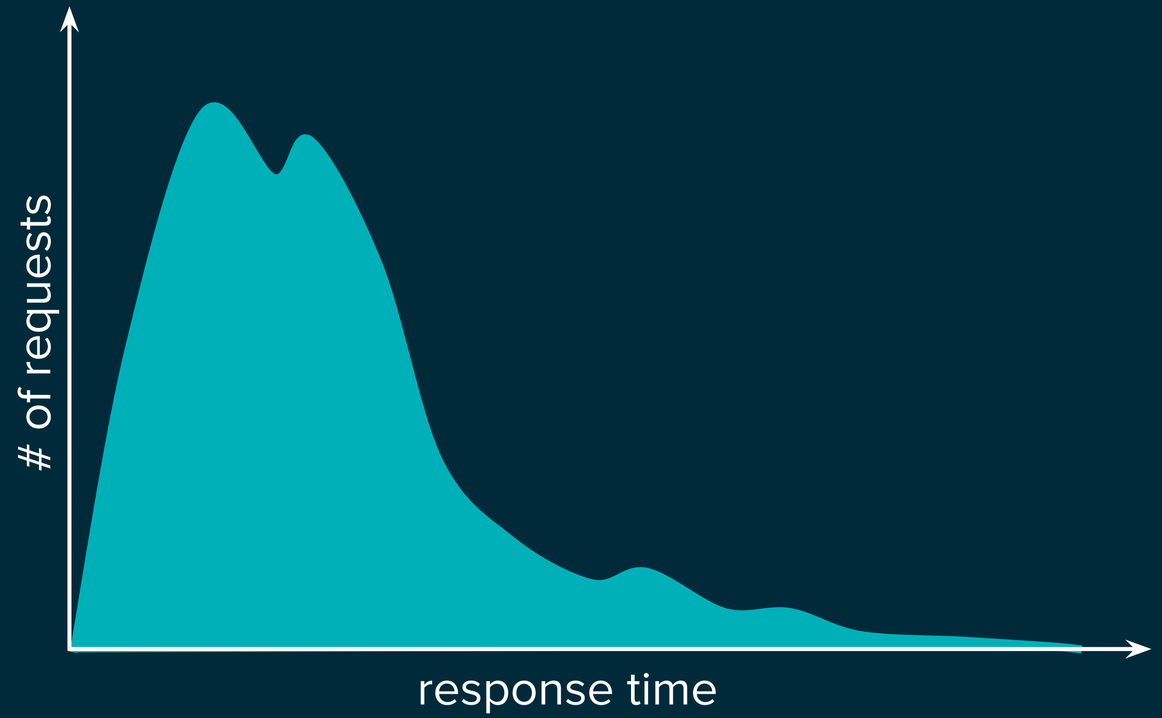
percentiles (pN)

min

max (p100)

count = sum (# of requests)

INDICATORS



percentiles (pN)

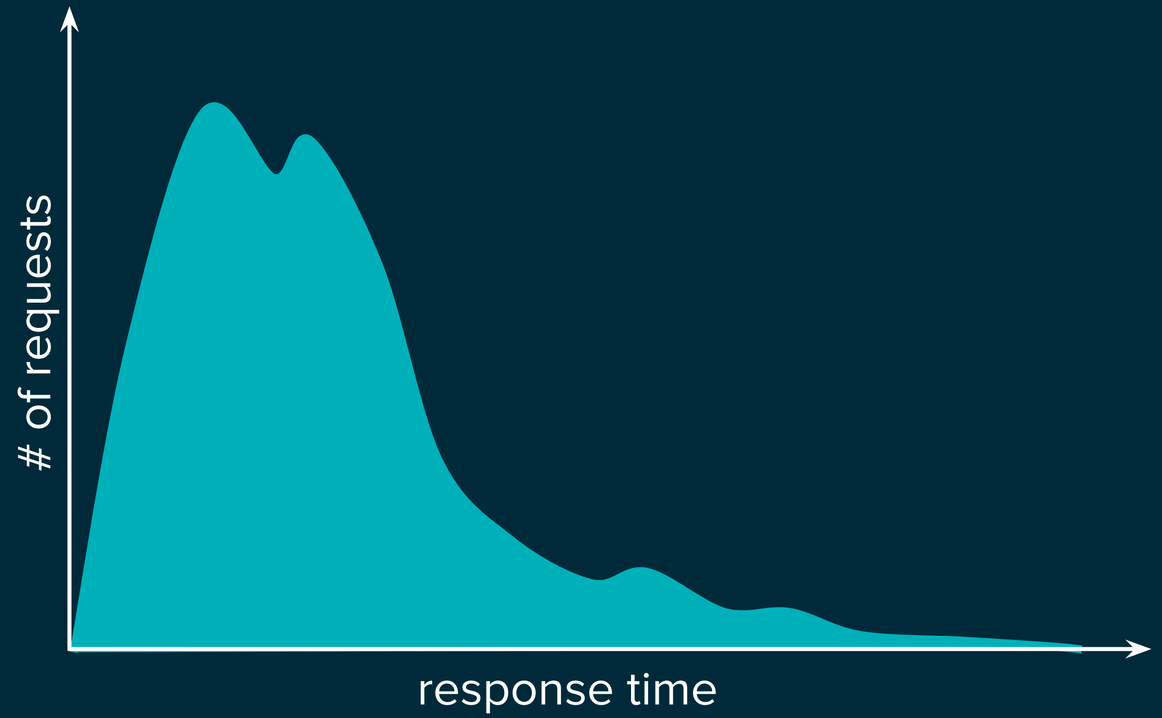
min

max (p100)

count

sum = sum (response time)

INDICATORS



percentiles (pN)

min

max (p100)

count

sum

avg = sum / count

INDICATORS

percentiles (pN)

min

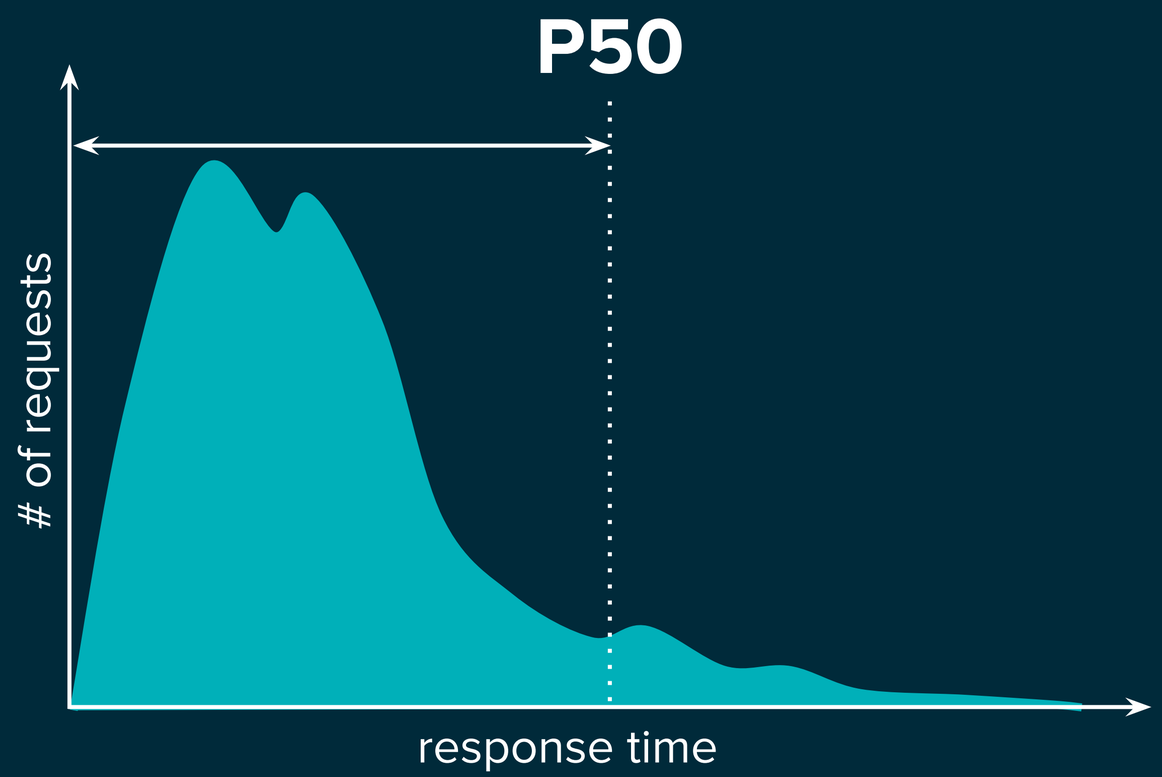
max (p100)

count

sum

~~**avg = sum / count**~~

INDICATORS



percentiles (pN)

min

max (p100)

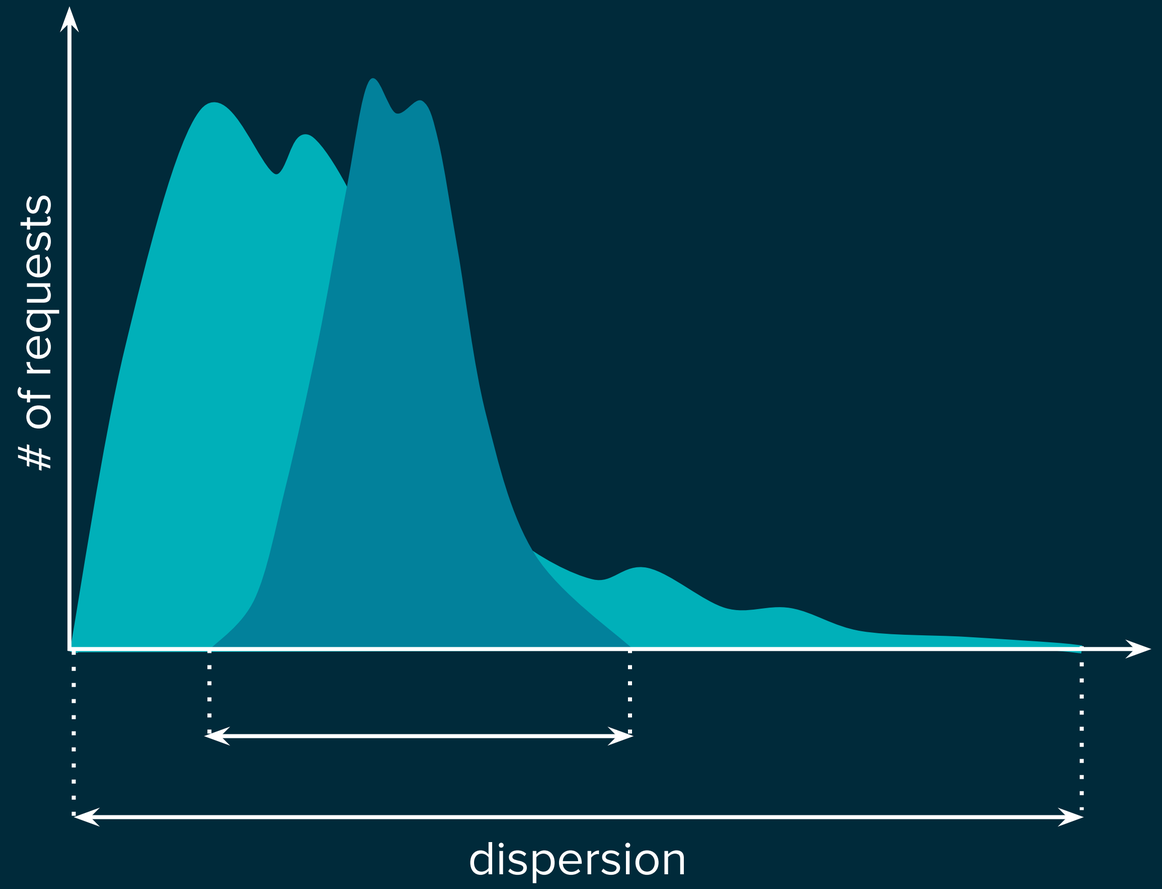
count

sum

~~avg~~

median (p50)

INDICATORS



percentiles (pN)

min

max (p100)

count

sum

~~avg~~

median (p50)

standard deviation

MATH IS PERFECT

don’t break it, imperfect human

P99(A ∪ B) != AVG [P99(A) ∪ P99(B)]

MATH IS PERFECT

don’t break it, imperfect human

P99(A ∪ B) != P99 [P99(A) ∪ P99(B)]

GOOD TRY, BUT NO

MATH IS PERFECT

don’t break it, imperfect human

percentiles == estimation

MATH IS PERFECT

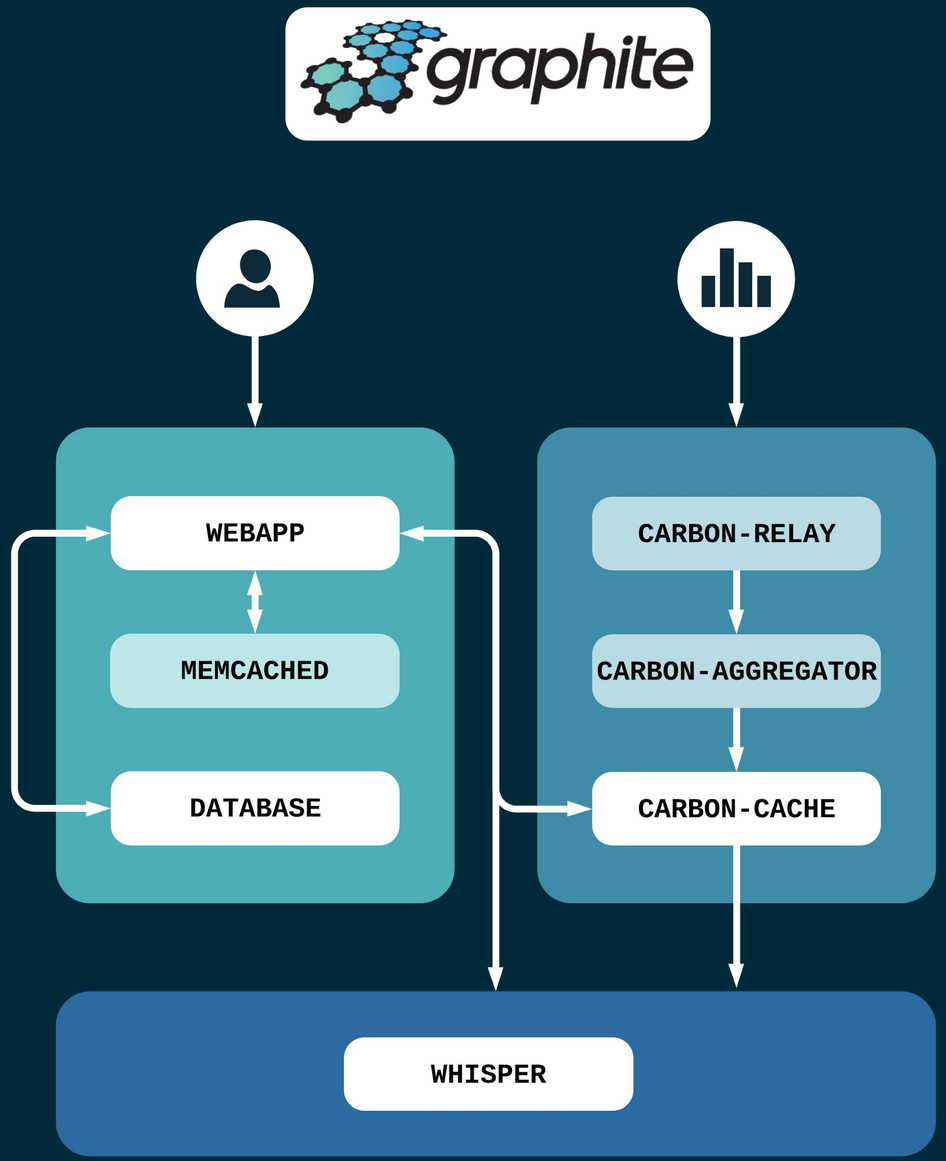
don’t break it, imperfect human

use **seconds** and **float64**, Luke!

PART II

how we do it

GRAPHITE



ABOUT

- numeric time-series data store
- written in Python
- open-source

COMPONENTS

- graphite webapp
- carbon
- whisper

1min.prod.host-001.service-a.metric-a
1min.prod.host-001.service-a.metric-b
1min.prod.host-001.service-b.metric-a

```
|-- 1min
  |-- prod
    |-- host-001
      |-- service-a
        |-- metric-a.wsp
        |-- metric-b.wsp
      |-- service-b
        |-- metric-a.wsp
```

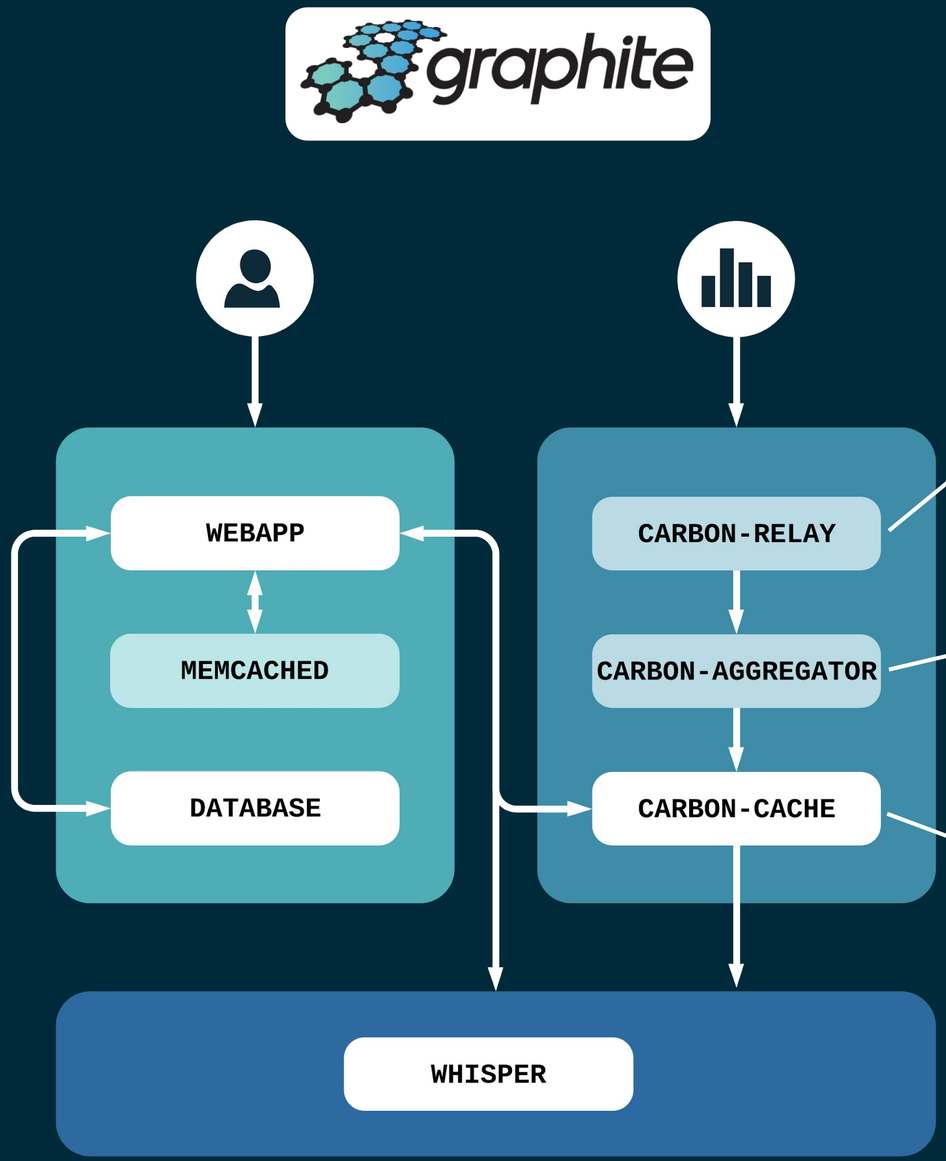
WHISPER

FIXED-SIZE DATABASE

RRD (round-robin database) *
file per metric

RETENTION & PRECISION

rollup aggregations



CARBON

CARBON-RELAY

- replication & sharding
- rules & consistent-hashing

CARBON-AGGREGATOR

- metrics buffer & aggregator
- I/O reduction

CARBON-CACHE

- cache & flush

STATSD

STATSD

STATSITE

NETWORK DAEMON

UDP / TCP

counter / timer / gauge / set

NODE.JS

C

AGGREGATIONS

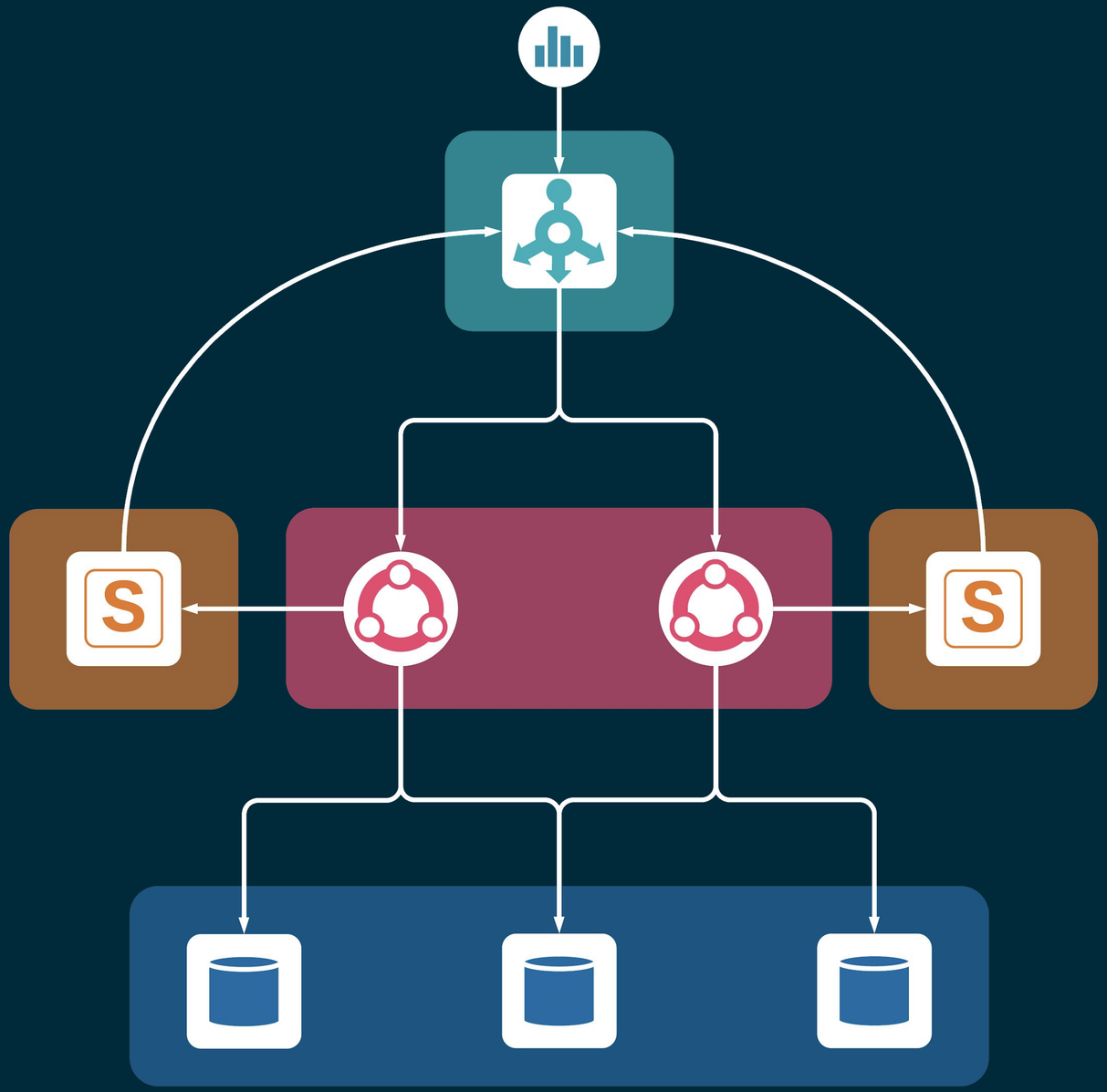
percentiles

sampling

pluggable backend

40K MPS / GB RAM

500K MPS / MB RAM



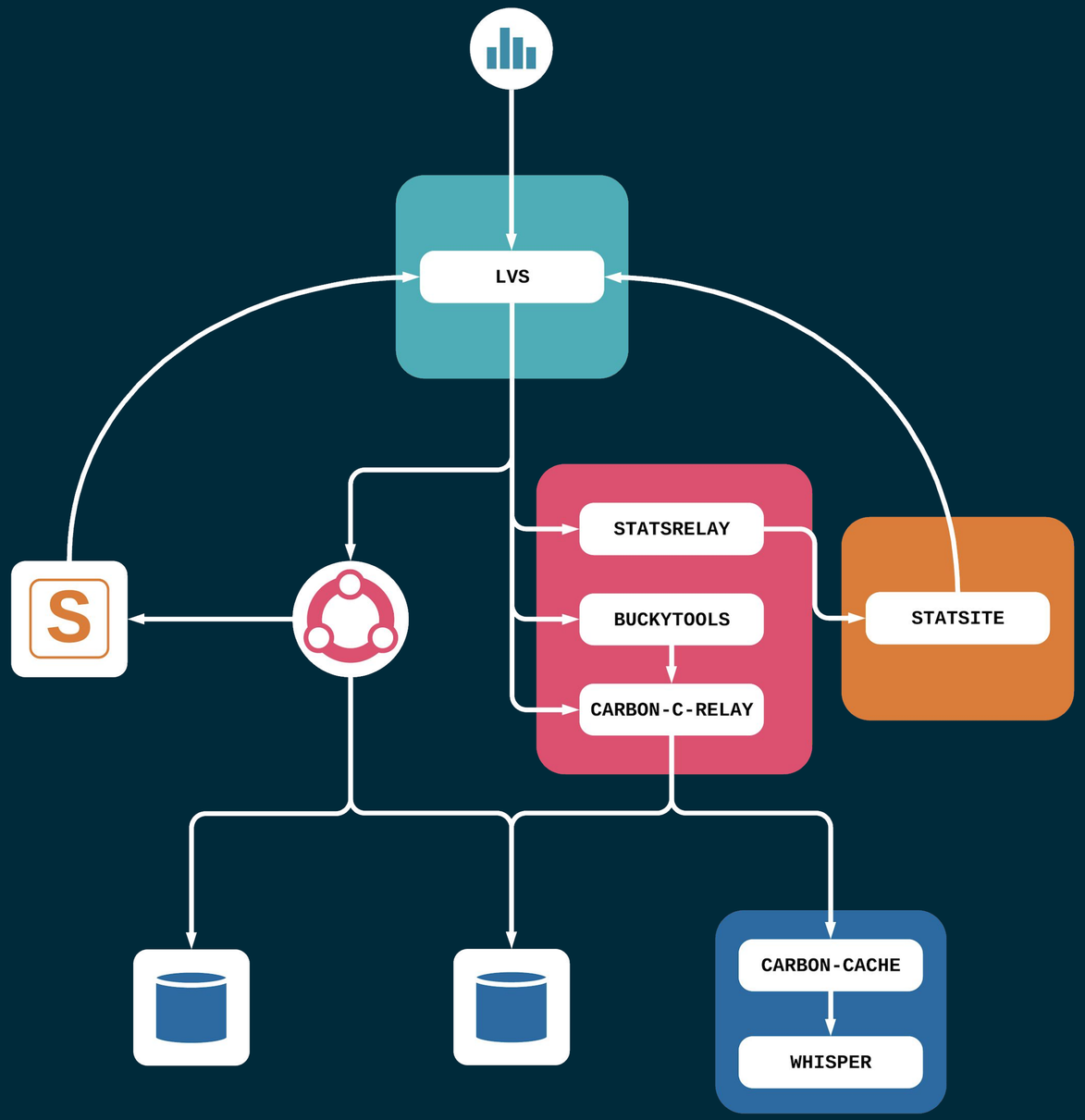
TIERS

RELAY load-balancing

HASH consistent-hashing

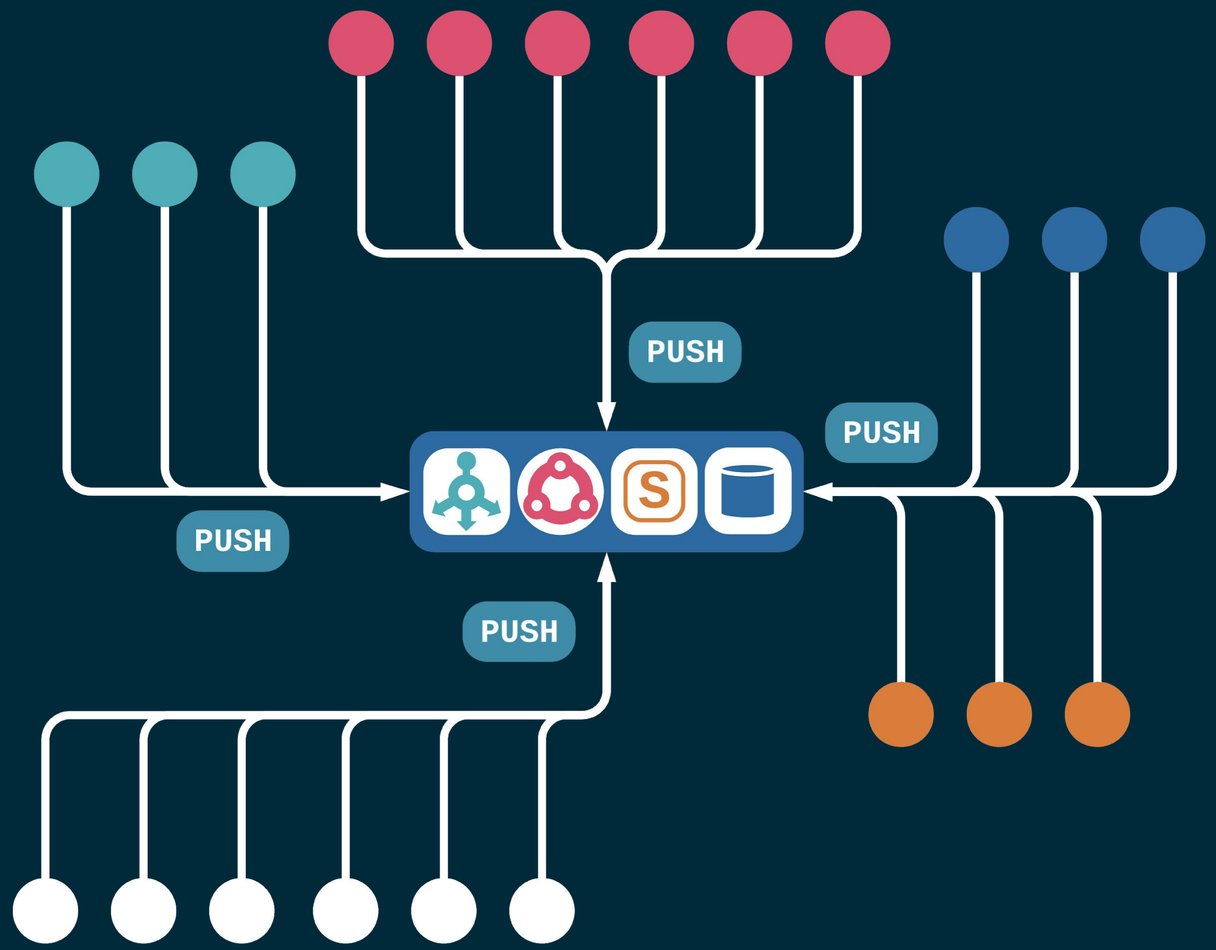
STATSD buffered aggregations

DATA long-term storage



SCALING RESILIENCE

RELAY	stateless
HASH	stateless
STATSD	stateless
DATA	stateful



DOWNSIDES?

KEEP EVERYTHING FOREVER

doesn’t scale well

SOA?

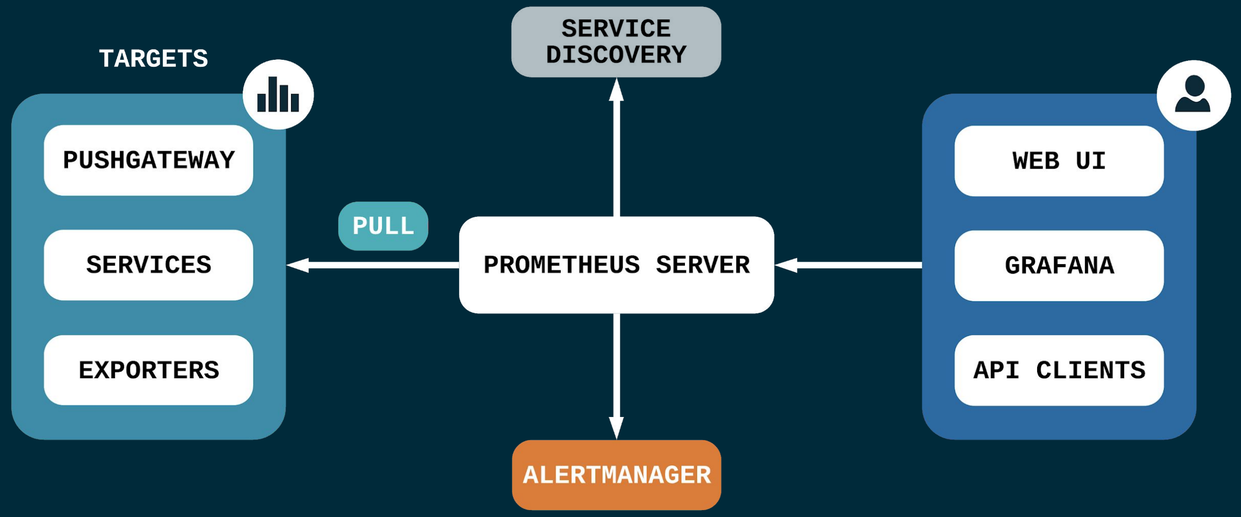
not suitable for dynamic environments

LACK OF AD-HOC TUNING

pre-defined configuration

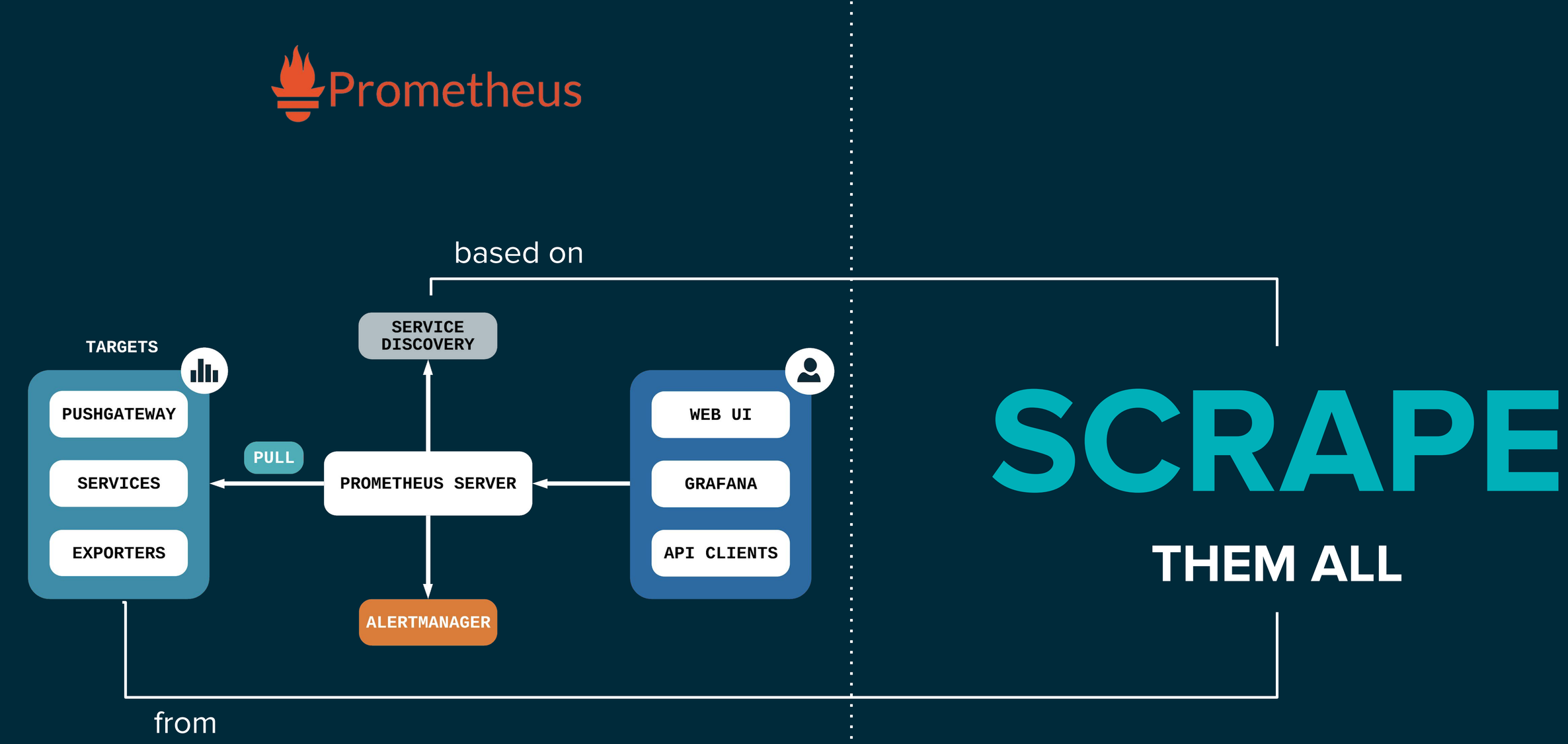
PROMETHEUS

Prometheus

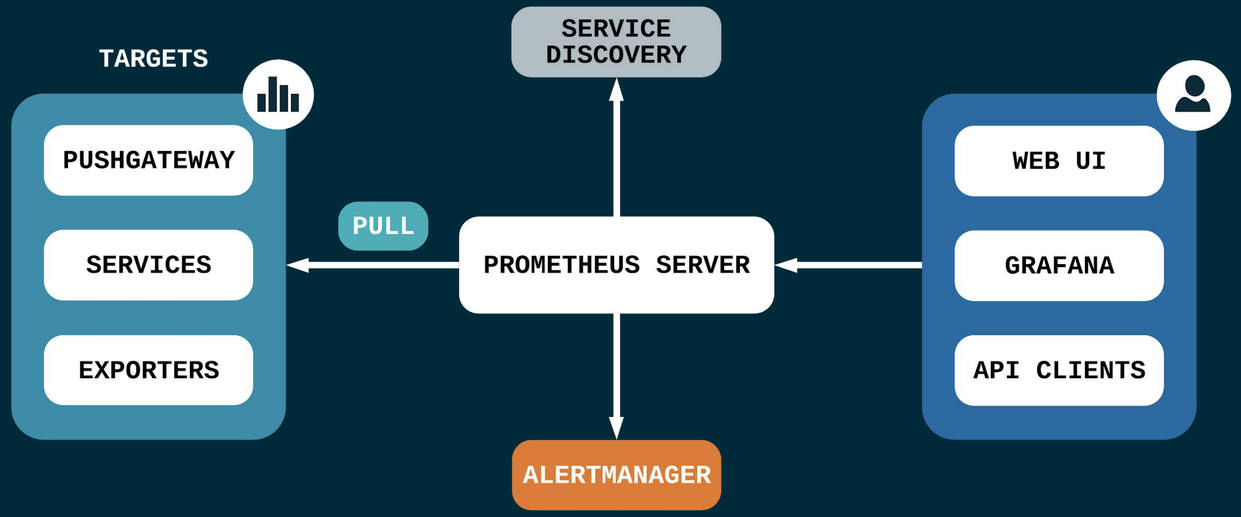


ABOUT

- open-source monitoring toolkit and TSDB
- written in Go
- made for dynamic cloud environments
- instrument / collect / store / query / alert / graph



Prometheus



COMPONENTS

PROMETHEUS SERVER

scrape and store metrics

PUSHGATEWAY

short-lived jobs support

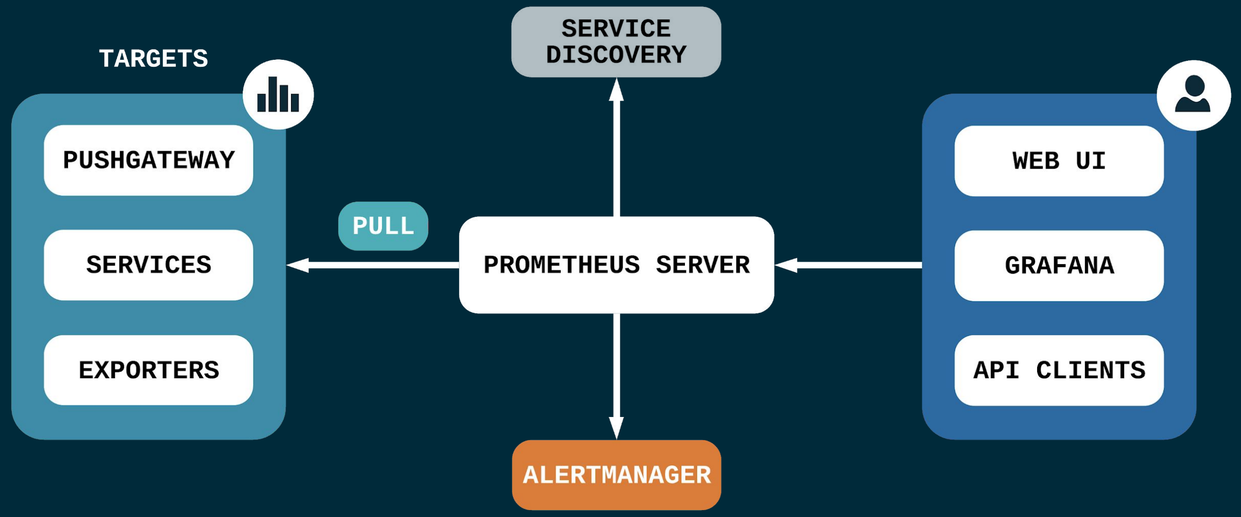
ALERTMANAGER

handle alerts

CLIENT LIBS & EXPORTERS

instrument code & 3P-services support

Prometheus



STORAGE

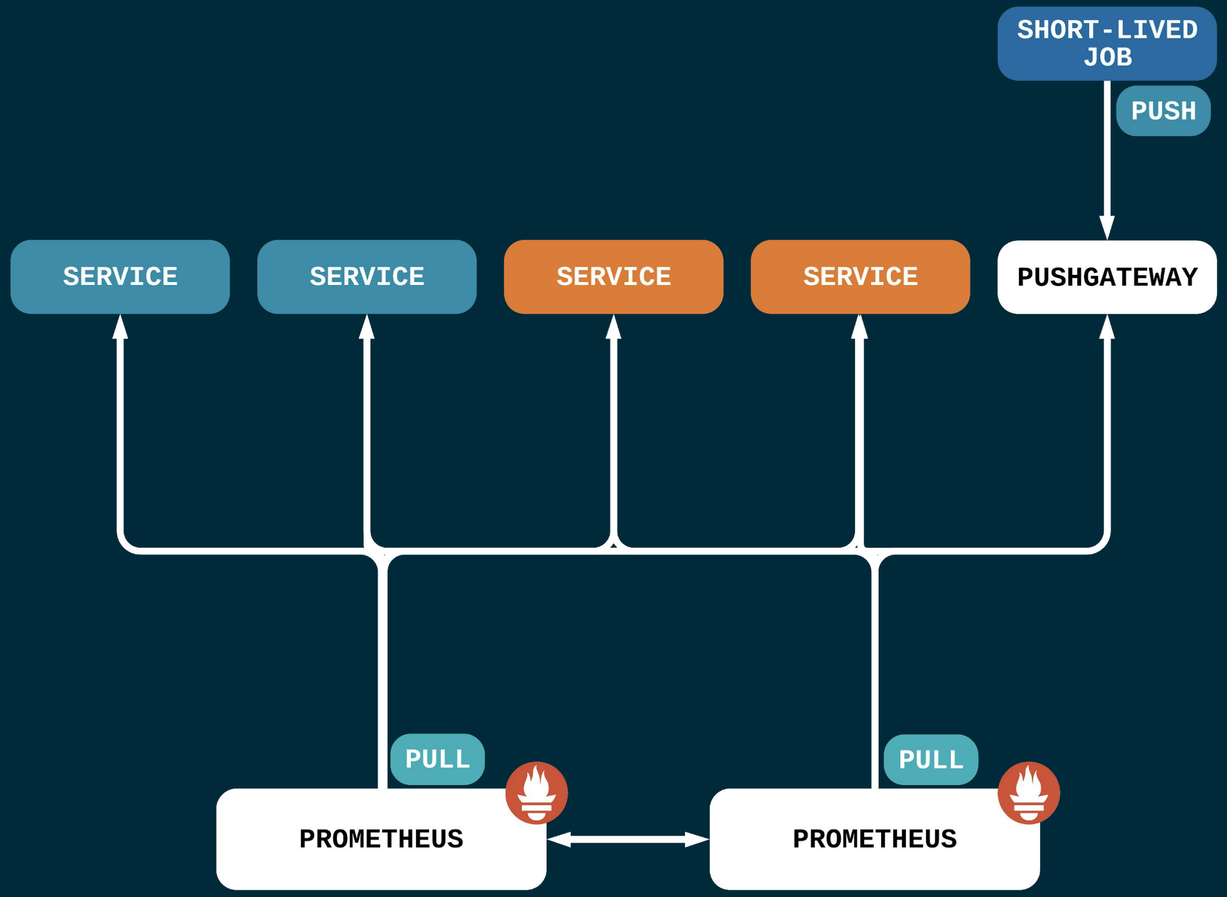
NOT SCALABLE OR DURABLE

by design

REMOTE STORAGE INTEGRATIONS

read & write

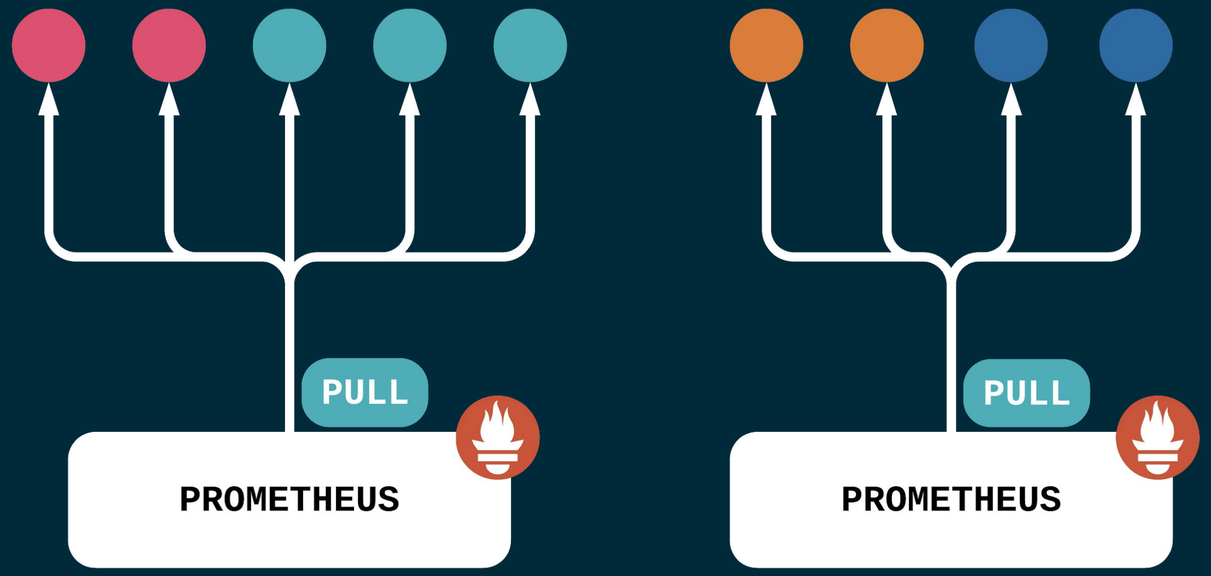
Prometheus



RESILIENCE

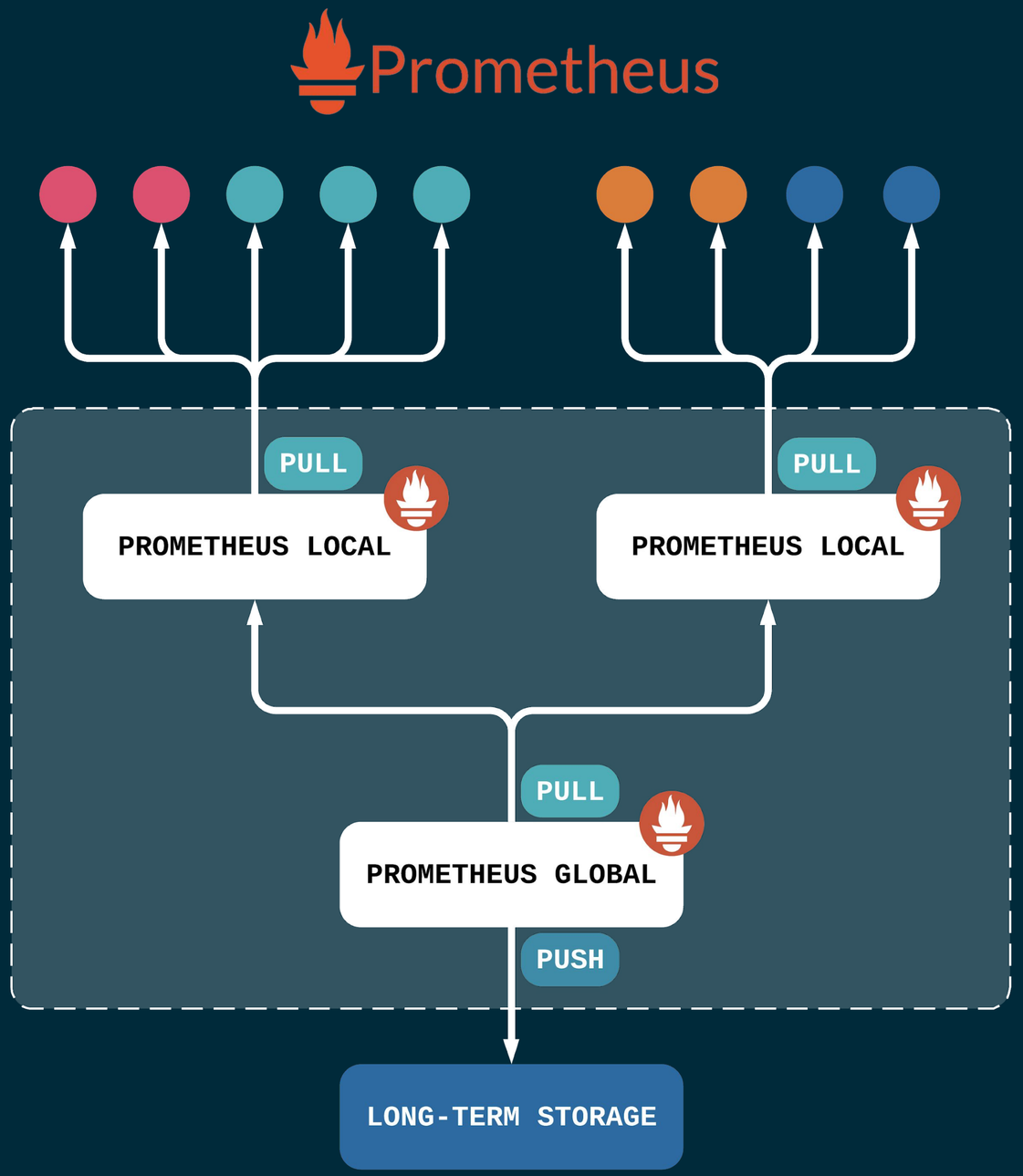
multiple identical instances
data redundancy

Prometheus



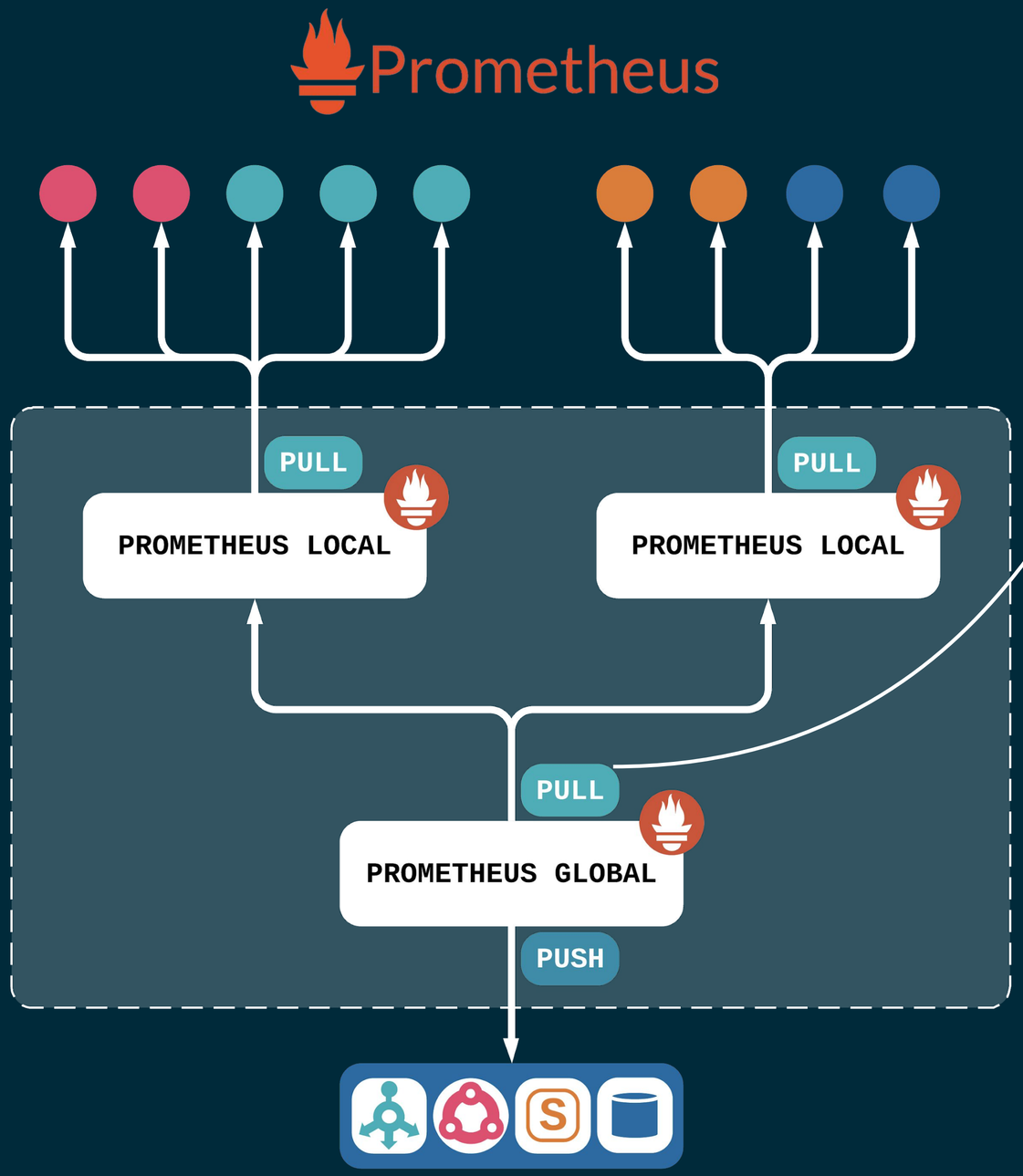
SCALING

services slice sharding



FEDERATION

scrape time-series data from another server
hierarchical / cross-server

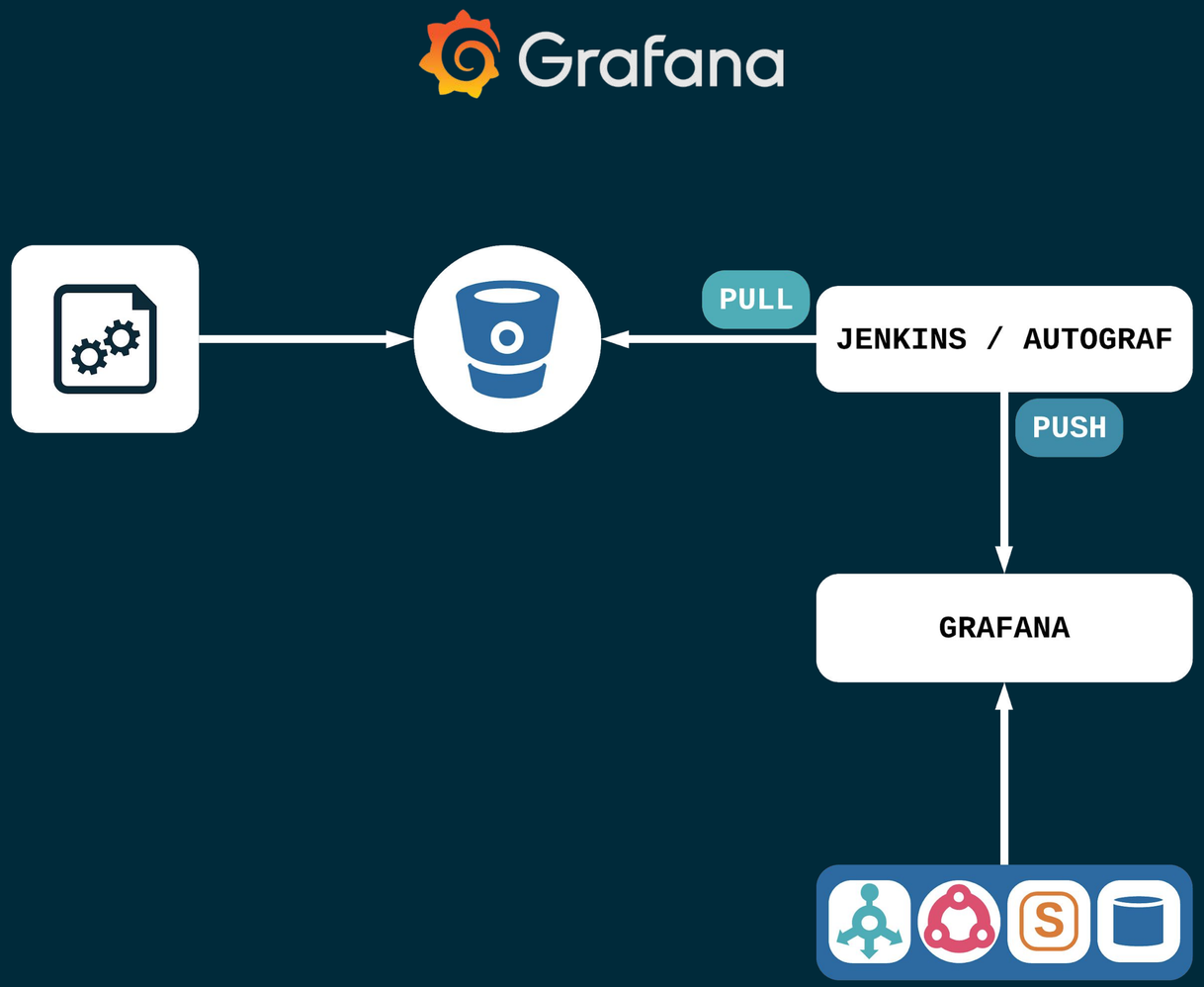


RECORDING RULES

precompute expressions
(frequently used / computationally expensive)
+
long-term storage ingestion

VISUALIZATION

VISUALIZATION



GRAFANA

open-source dashboard & graph composer
built-in visualization options
pluggable data sources

AUTOGRAF

Python
codify grafana dashboards creation
dynamic dashboards (canary)

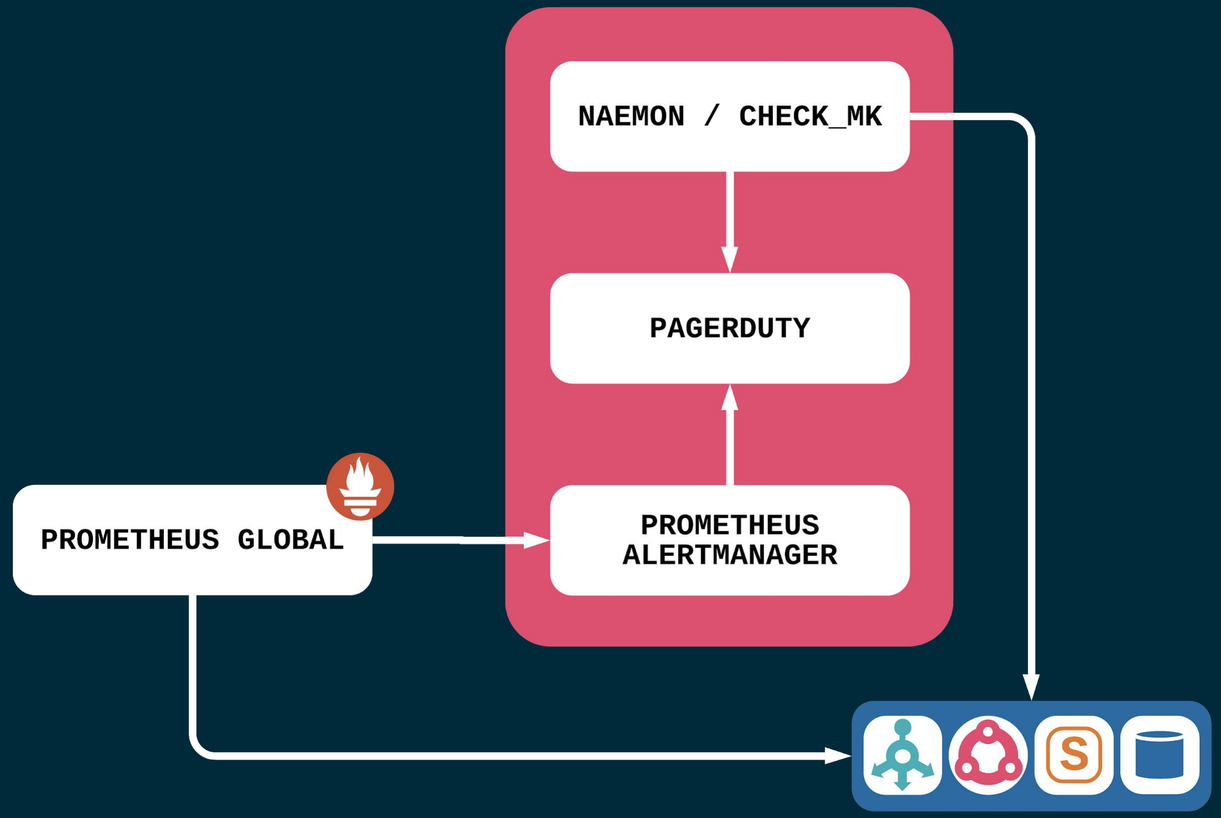
ALERTS

NOTIFY ME!

if SLO violated - someone should be woken up

pagerduty

Prometheus



ALERTS

PAGERDUTY

on-call management & notification
service & team
scheduling & escalation rules

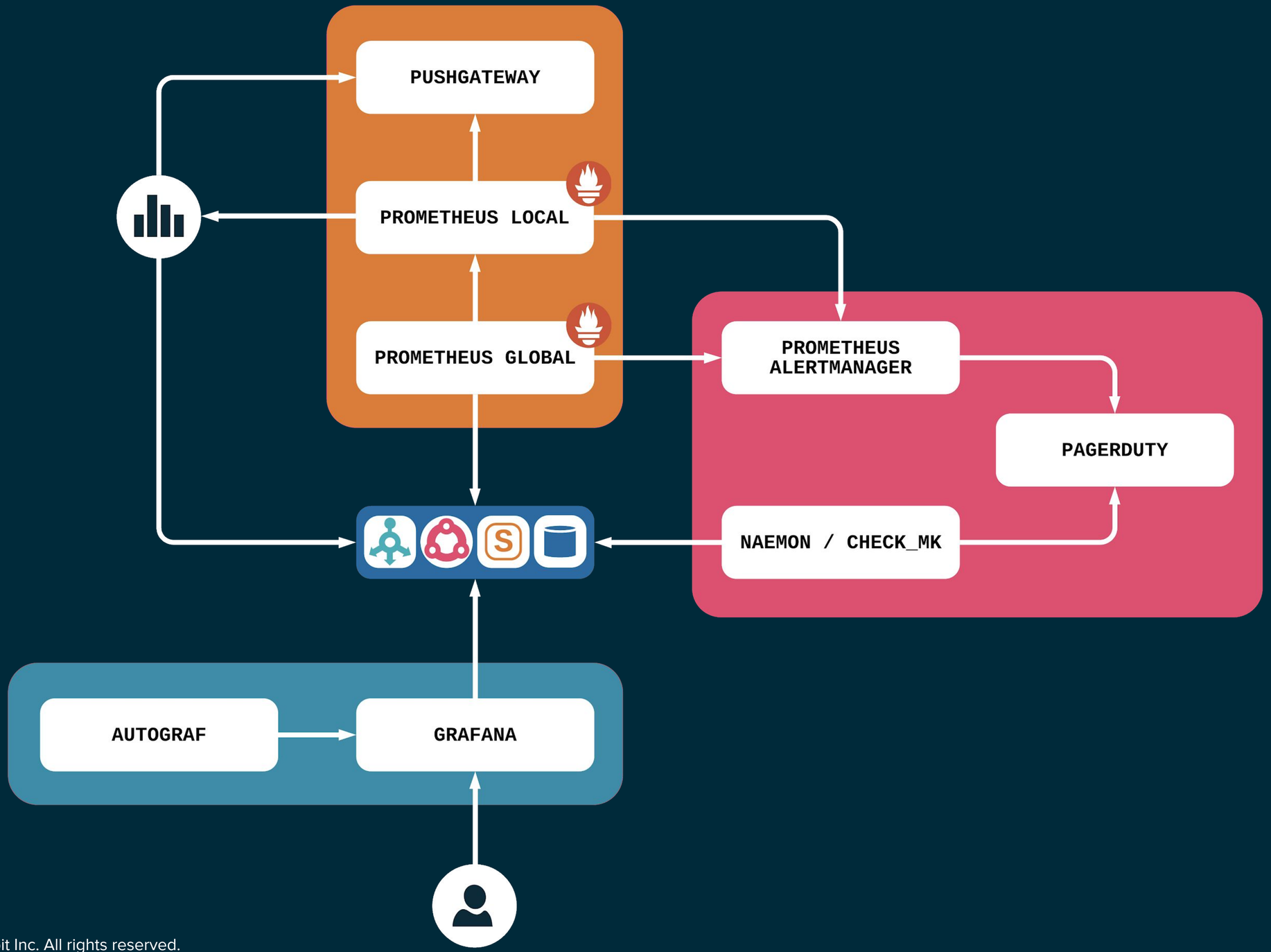
NAEMON + CHECK_MK

system / network / infrastructure monitoring suite
active metrics checks

DEPRECATED

PROMETHEUS ALERTMANAGER

alerting rules in prometheus server
grouping / inhibition / silencing



PART III

monitoring culture

OWNERSHIP

you wrote it - you own it

SERVICE REGISTRY

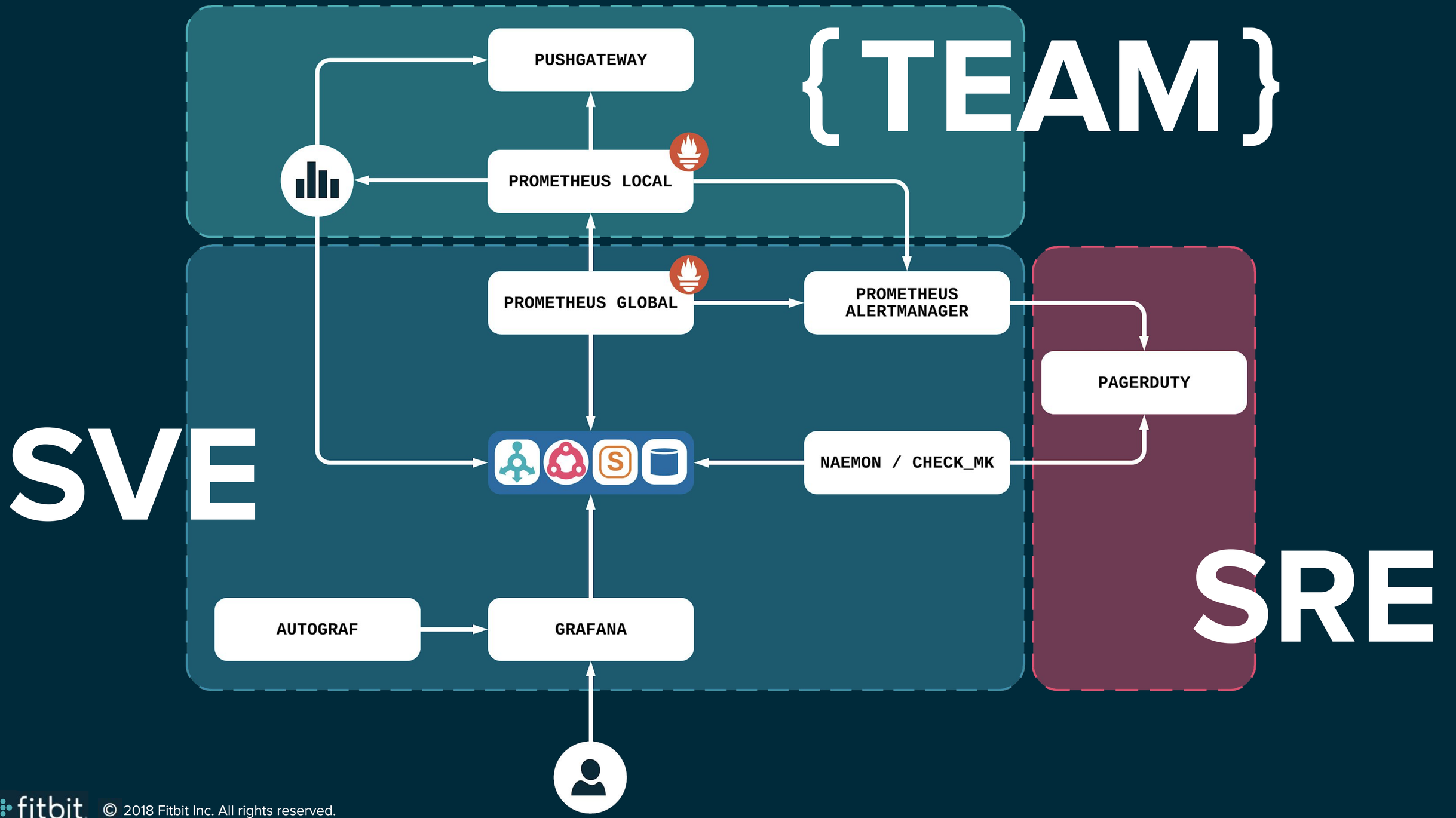
service owners
runbooks
disaster recovery / backups
SLO / SLA

BEING ON-CALL

per team duties
scheduled rotation

TEAMS

SRE (Site Reliability Engineering)
SVE (Site Visibility Engineering)
others



RECAP

monitor? **monitor!**

trade-offs it is all about it

use tools **wisely**

build your monitoring **culture**

THANK YOU

questions?