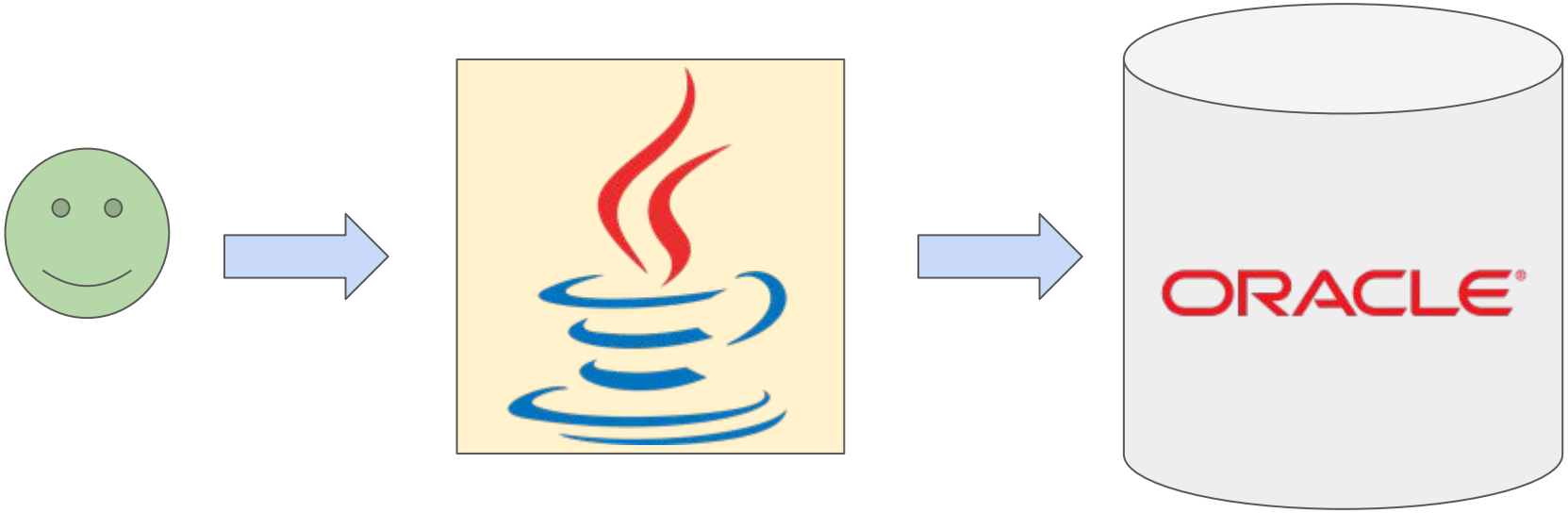
Riemann

monitors distributed systems

a long time ago in a galaxy far far away...



Cloud Native Landscape

v1.0

App Definition & Development

Database & Data Analytics

Streaming

SCM

Application Definition

CI/CD

Orchestration & Management

Scheduling & Orchestration

kubernetes (CNCF Project)

Coordination & Service Discovery

CoreDNS (CNCF Project)

Service Management

linkerd (CNCF Project)

GRPC (CNCF Project)

envoy (CNCF Project)

Runtime

Cloud-Native Storage

Container Runtime

containerd (CNCF Project)

rkt (CNCF Project)

Cloud-Native Network

CNI (CNCF Project)

Provisioning

Host Management / Tooling

Infrastructure Automation

Container Registries

Secure Images

Notary (CNCF Project)

TUF (CNCF Project)

Key Management

Cloud

Public

Private

Platforms

PaaS / Container Service

Serverless/Event based

Observability & Analysis

Monitoring

Prometheus (CNCF Project)

Logging

fluentd (CNCF Project)

Tracing

OpenTracing (CNCF Project)

github.com/cncf/landscape

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

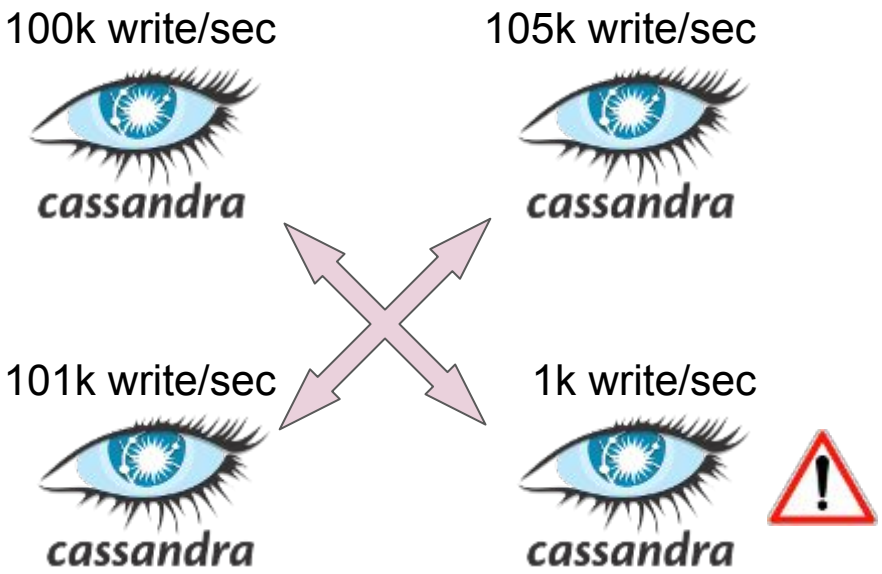
CLOUD NATIVE COMPUTING FOUNDATION

Redpoint

Amplify PARTNERS

Distributed architectures are hard

Monitoring distributed systems



- Time windows
- Rates
- Percentiles
- Cluster monitoring
- Correlation between metrics
- State transitions (OK => KO)
- Alerts (mail, slack, pagerduty…)
- Flexibility
- ...

RIEMANN

- Created by Kyle Kingsbury (Aphyr)
- Event processing
- Clojure
- Monitoring

An immutable event

:host	“foo.bar.com”
:service	“df_percent_bytes_used_root”
:state	“critical”
:time	1493243041
:metric	90
:description	“Disk is full”
:tags	[“disk”]
:ttl	60

Collectd
Telegraf
K8s/Heapster
Statsd
Graphite
...

Syslog-ng
Logstash
Fluentd
...

Java
Haskell
Go
Python
Perl
...

Kafka
Nagios check
Chef
...

Good

TCP
TLS

Drop packets

UDP

Slow

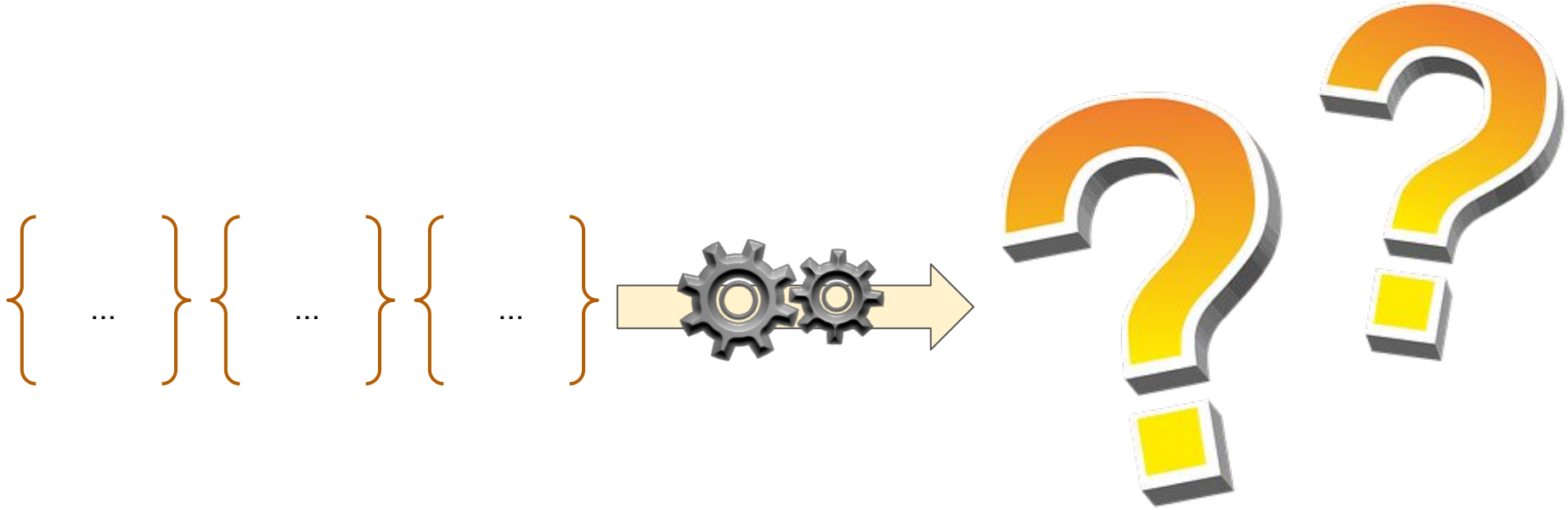
HTTP

Compat

Graphite
OpenTSBD

RIEMANN

Streams



```
{ :host     "foo1.com"
  :service  "api_rate"
  :time     1493243041
  :metric   90 }
```

```
{ :host     "foo1.com"
  :service  "foobar"
  :time     1493243041
  :metric   90 }
```

```
{ :host     "foo1.com"
  :service  "api_rate"
  :time     1493243044
  :metric   90 }
```

where = service "api_rate"

```
{ :host     "foo1.com"
  :service  "api_rate"
  :time     1493243041
  :metric   90 }
```

```
{ :host     "foo1.com"
  :service  "api_rate"
  :time     1493243044
  :metric   90 }
```

```
{ :host     "foo1.com"
  :service  "api_rate"
  :time     1493243041
  :metric   90 }
```

```
{ :host     "foo1.com"
  :service  "api_rate"
  :time     1493243044
  :metric   90 }
```

fixed-time-window 10

```
[ { :host     "foo1.com"
    :service  "api_rate"
    :time     1493243041
    :metric   90 }
  { :host     "foo1.com"
    :service  "api_rate"
    :time     1493243044
    :metric   90 } ]
```

```
{ :host    "foo1.com"
  :service "api_rate"
  :time    1493243041
  :metric  90 }
{ :host    "foo1.com"
  :service "api_rate"
  :time    1493243044
  :metric  90 }
```

smap sum

```
{ :host    "foo1.com"
  :service "api_rate"
  :time    1493243044
  :metric  180 }
```

```
:host      "foo1.com"
:service   "api_rate"
:time      1493243044
:metric    180
```

where < metric 200

```
:host      "foo1.com"
:service   "api_rate"
:time      1493243044
:metric    180
```

```
{ :host      "foo1.com"
  :service   "api_rate"
  :time      1493243044
  :metric    180 }
```

email "ops@riemann.io"

where = service “api_rate”

fixed-time-window 10

smap sum

where < metric 200

email “ops@riemann.io”

(**where** (= service “api_rate”)

(**fixed-time-window** 10

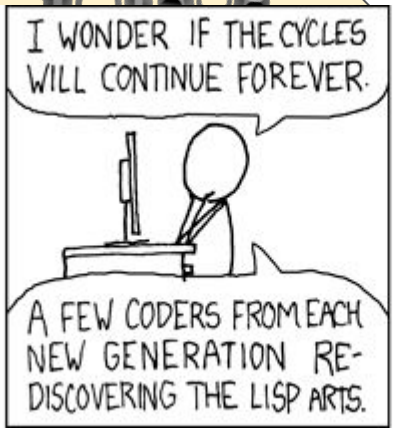
(**smap** sum

(**where** (< metric 200)

(**email** “ops@riemann.io”)))))

cljs


```
riemann.bin> "REPL driven development ftw"
"REPL driven development ftw"
```



Configuration as code

(Your config is 100 % Clojure)

```
(with {:description "Disk is full"
       :state       "critical"}
  child)
```

```
{:host      "foo.bar.com"
 :service   "df_home_mathieu"
 :state     "ok"
 :time      1493243041
 :metric    90}
```

```
{:host      "foo.bar.com"
 :service   "df_home_mathieu"
 :state     "ok"
 :time      1493243041
 :metric    90
 :description "Disk is full"}
```

```
(where (service "foo")

  (with {:description "cat"}
    (email "ops@riemann.io"))

  (with {:description "dog"}
    (email "dev@riemann.io")))
```

where has 2 children

First child

Second child

```
(where (service "foo")

  (with {:description "cat"}
    (email "ops@riemann.io"))

  (with {:description "dog"}
    (email "dev@riemann.io")))
```

where has 2 children

with has 1 child

First child

with has 1 child

Second child

Clojure datastructures

Immutability

No side effects between streams

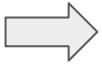
```
(where (service “df_percent_bytes_used_var_log”))
```

```
(where (service #“^df_percent_bytes_used_”))
```

```
(where (and (service #“^df_percent_bytes_used_”)
            (> (:metric event) 80)))
```

```
(default :ttl 60
  child)
```

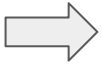
:host	“foo.bar.com”
:service	“df_home_mathieu”
:state	“ok”
:time	1493243041
:metric	90



:host	“foo.bar.com”
:service	“df_home_mathieu”
:state	“ok”
:time	1493243041
:metric	90
:ttl	60

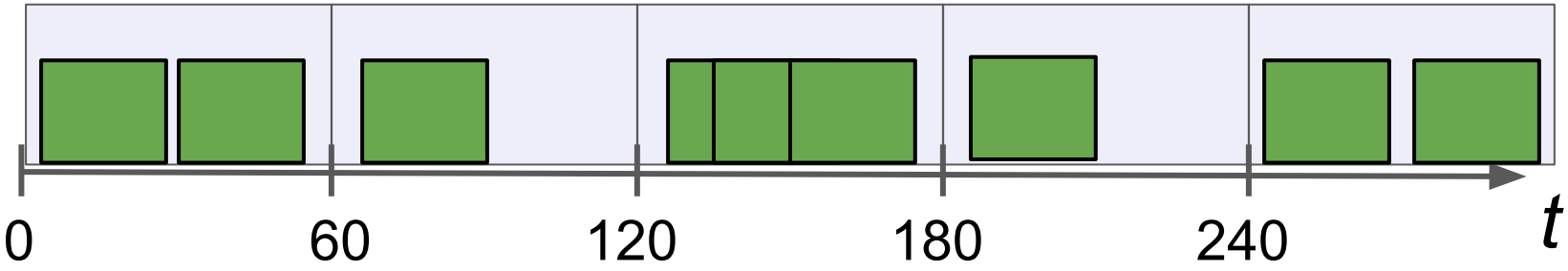
```
(smap (fn [event]
  (assoc event :ttl 60))
  child)
```

:host	“foo.bar.com”
:service	“df_home_mathieu”
:state	“ok”
:time	1493243041
:metric	90

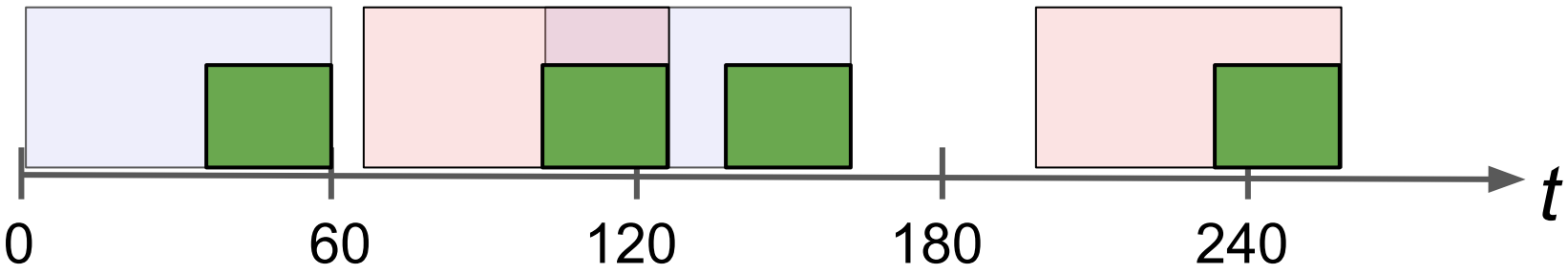


:host	“foo.bar.com”
:service	“df_home_mathieu”
:state	“ok”
:time	1493243041
:metric	90
:ttl	60

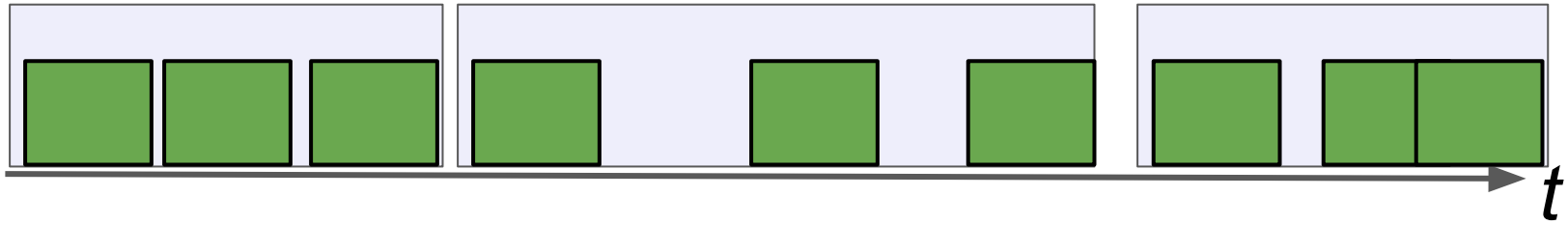
```
(fixed-time-window 60
  child1
  child2)
```



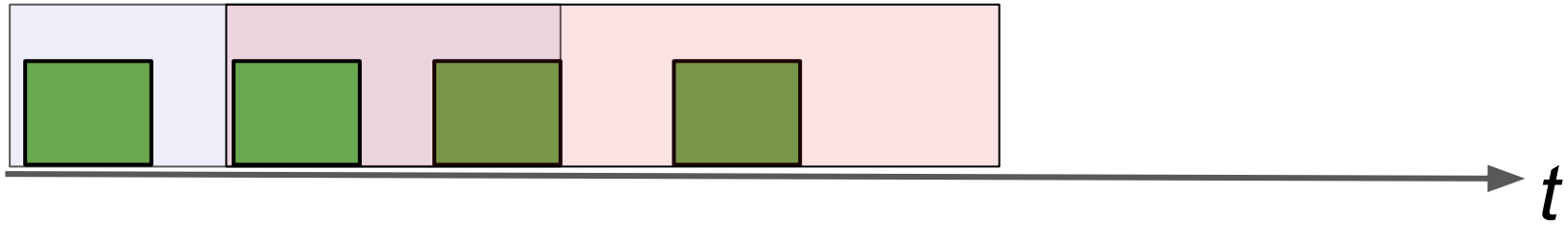
```
(moving-time-window 60
  child)
```



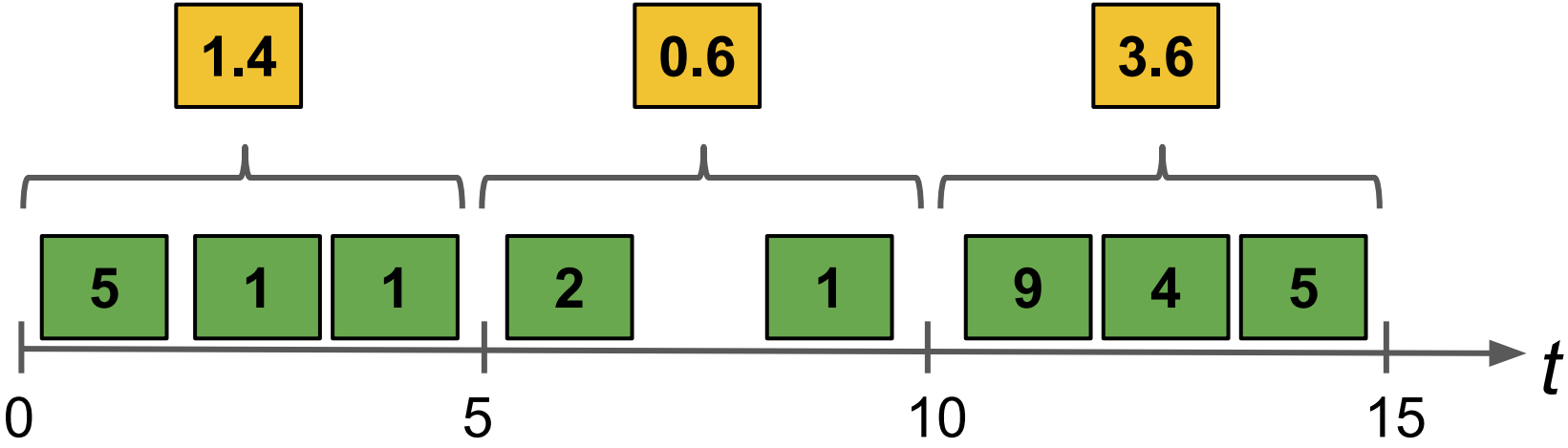
```
(fixed-event-window 3
  child)
```



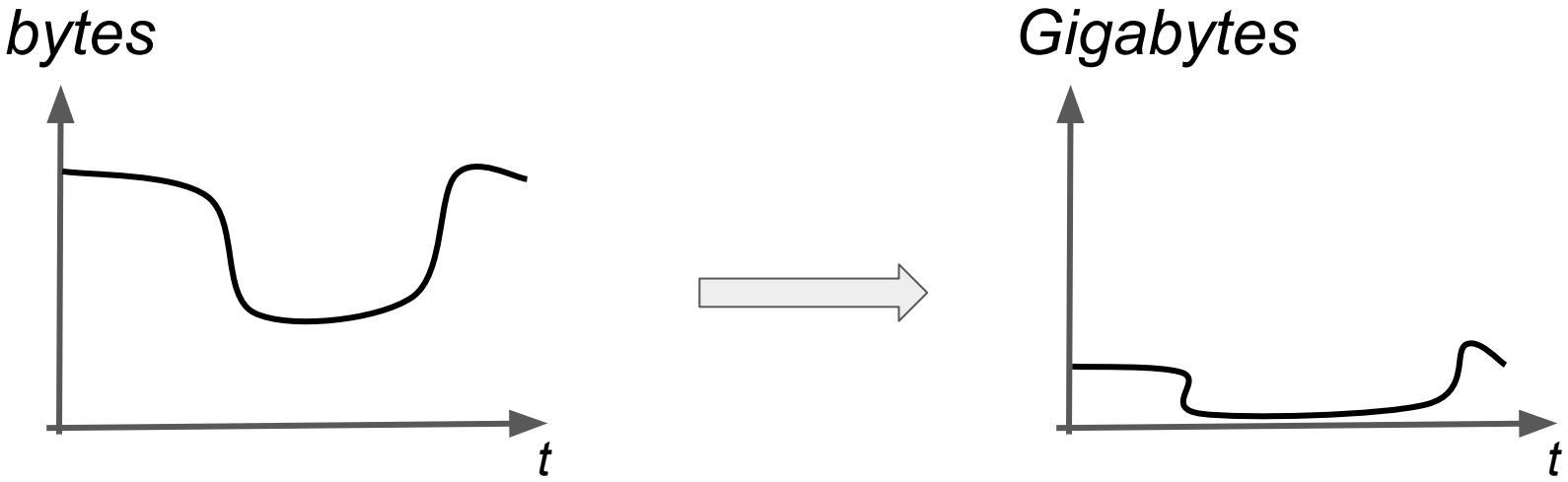
```
(moving-event-window 3
  child)
```



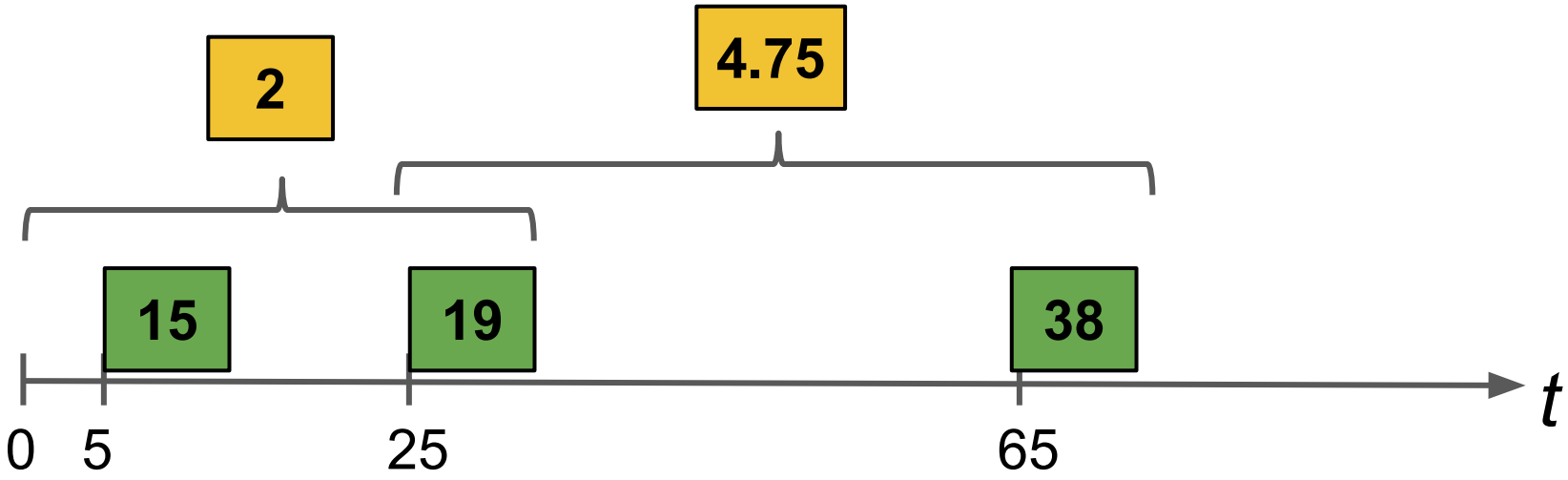
```
(rate 5
  child)
```



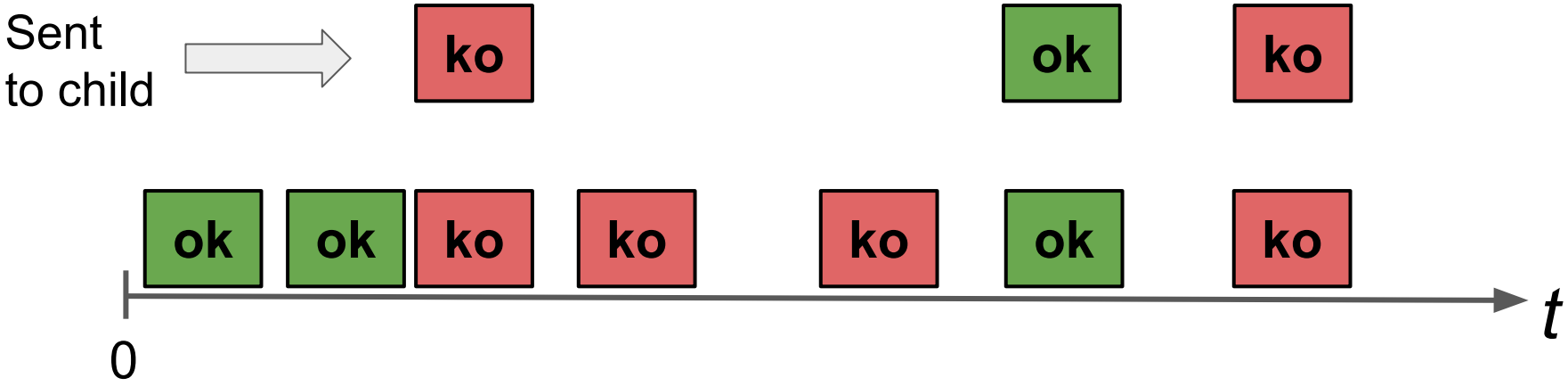
```
(scale (/ 1 1024 1024 1024)
  child)
```



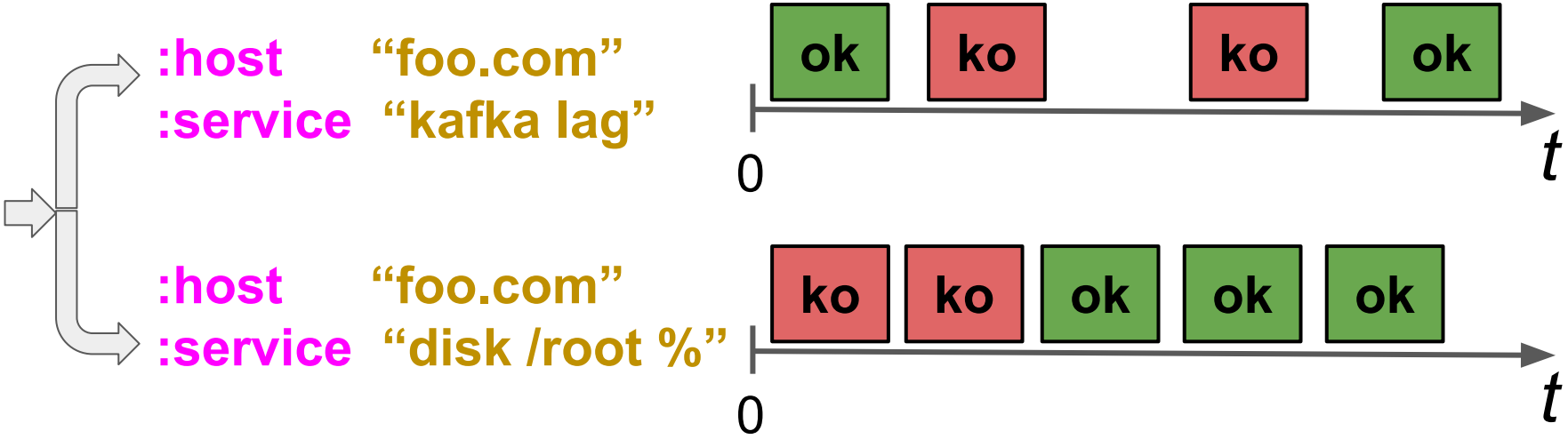
```
(ddt
 child)
```



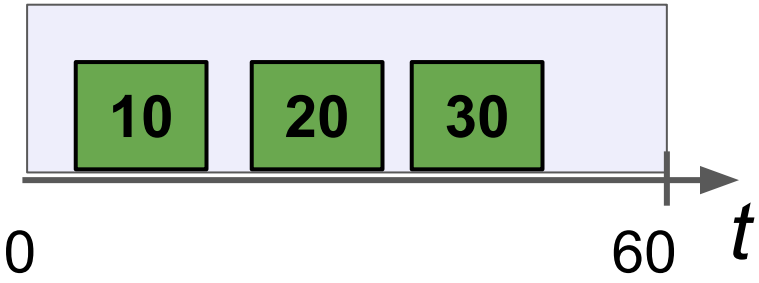
```
(changed :state {:init "ok"}
  child)
```



```
(by [:host :service]
  (changed :state {:init "ok"}
    child))
```

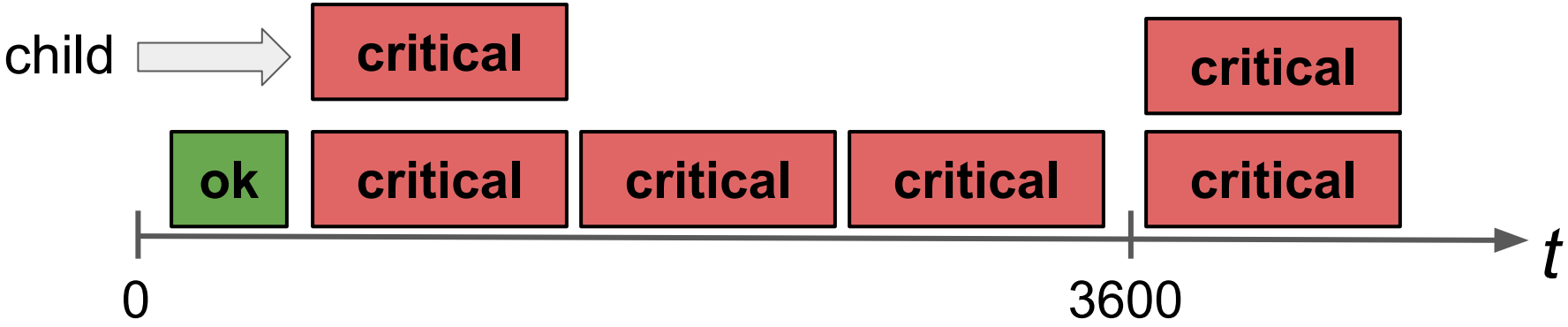


```
(where (service "api request")
  (percentiles 60 [0.5 0.99]
    child))
```

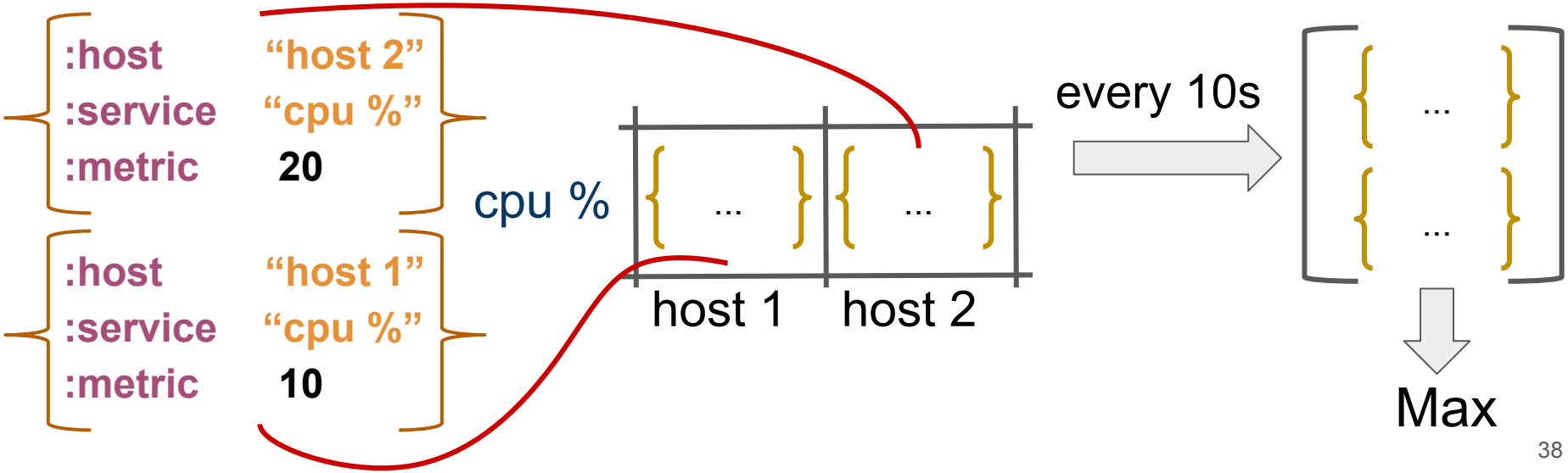


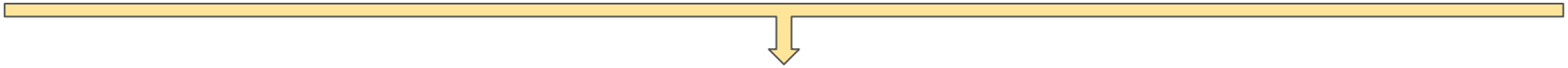

```
{ :host      "riemann.io"
  :service   "api request 0.5"
  :metric    20 }

{ :host      "riemann.io"
  :service   "api request 0.99"
  :metric    30 }
```

```
(where (state "critical")
  (throttle 2 3600
    (email "foo@riemann.io")))
```



```
(where (service "cpu %")
  (coalesce 10
    (smap max)))
```





InfluxDB
Elasticsearch
Graphite
Kafka
Logstash
Datadog
Cloudwatch
Riemann
...

Email
Hipchat
Slack
Mailgun
...

Pagerduty
VictorOps
Twilio
Alerta
...

Nagios
Shinken
...

```
(batch 100 1      ;; batch size = 100 every 1 sec
  (async-queue! :influxdb ;; create a threadpool
    {:queue-size 10000
     :core-pool-size 4}
    (influxdb {:host 127.0.0.1 ;; forward to influx
               :db "riemann"})))
```

```
(exception-stream
  (email “alert@riemann.io”)
  (influxdb {:host 127.0.0.1
             :db ”riemann”}))
```

Configuration as code

Split your configuration

```
(def check-critical-state        ;; a var containing a stream
  (where (state “critical”)
    (email “admin@riemann.io”)))

(defn check-state       ;; a function returning a stream
  [s email-addr]
  (where (state s)
    (email email-addr)))

(streams
   check-critical-state
   (check-state “critical” “admin@riemann.io”))
```

/etc/riemann/riemann.config

/mycorp/app/elasticsearch.clj

/mycorp/output/mail.clj

/mycorp/system/disk.clj

/mycorp/system/ram.clj

…

+ A plugin system

Configuration as code

Tests

```
(scale (/ 1 1024 1024 1024)
  (tap :scale-tap)
  child)


(tests
   (deftest foo-test
      (is (= (:scale-tap (inject! [{:metric 1000}]))
                [{:metric (/ 1000 1024 1024 1024)}]))))
```

The index

- In memory datastructure (hashmap)
  - Key : [host service]
  - Value: an event
  - The index stream adds event to the index

```
(where (service “ram_percent”)
  (index))
```

service

Time 3

ram_%	:host "foo.bar" :service "ram_percent" :metric 65 :ttl 120 :time 2	:host "fizz.buzz" :service "ram_percent" :metric 80 :ttl 120 :time 2
cpu_%	:host "foo.bar" :service "cpu_percent" :metric 40 :ttl 60 :time 1	:host "fizz.buzz" :service "cpu_percent" :metric 90 :ttl 60 :time 3
	foo.bar	**fizz.buzz**

host

service

Time 10

ram_%	:host "foo.bar" :service "ram_percent" :metric 65 :ttl 120 :time 2	:host "fizz.buzz" :service "ram_percent" :metric 80 :ttl 120 :time 2
cpu_%	:host "foo.bar" :service "cpu_percent" :metric 45 :ttl 60 :time 10	:host "fizz.buzz" :service "cpu_percent" :metric 90 :ttl 60 :time 3
	foo.bar	**fizz.buzz**

host

service

Time 64

ram_%	:host "foo.bar" :service "ram_percent" :metric 65 :ttl 120 :time 2	:host "fizz.buzz" :service "ram_percent" :metric 80 :ttl 120 :time 2
cpu_%	:host "foo.bar" :service "cpu_percent" :metric 45 :ttl 60 :time 10	:host "fizz.buzz" :service "cpu_percent" :metric 90 :ttl 60 :time 3
	foo.bar	**fizz.buzz**

host

service

Time 64

ram_%

```
:host    "foo.bar"
:service "ram_percent"
:metric  65
:ttl     120
:time    2
```

```
:host    "fizz.buzz"
:service "ram_percent"
:metric  80
:ttl     120
:time    2
```

cpu_%

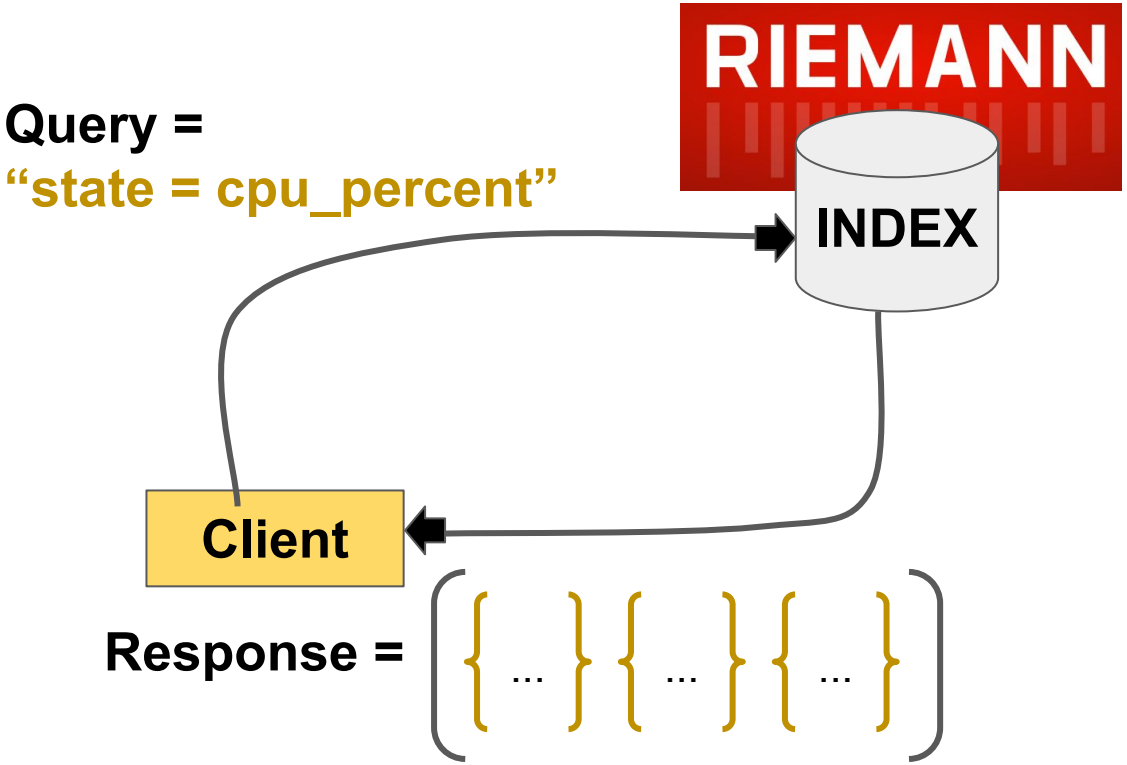
```
:host    "foo.bar"
:service "cpu_percent"
:metric  45
:ttl     60
:time    10
```

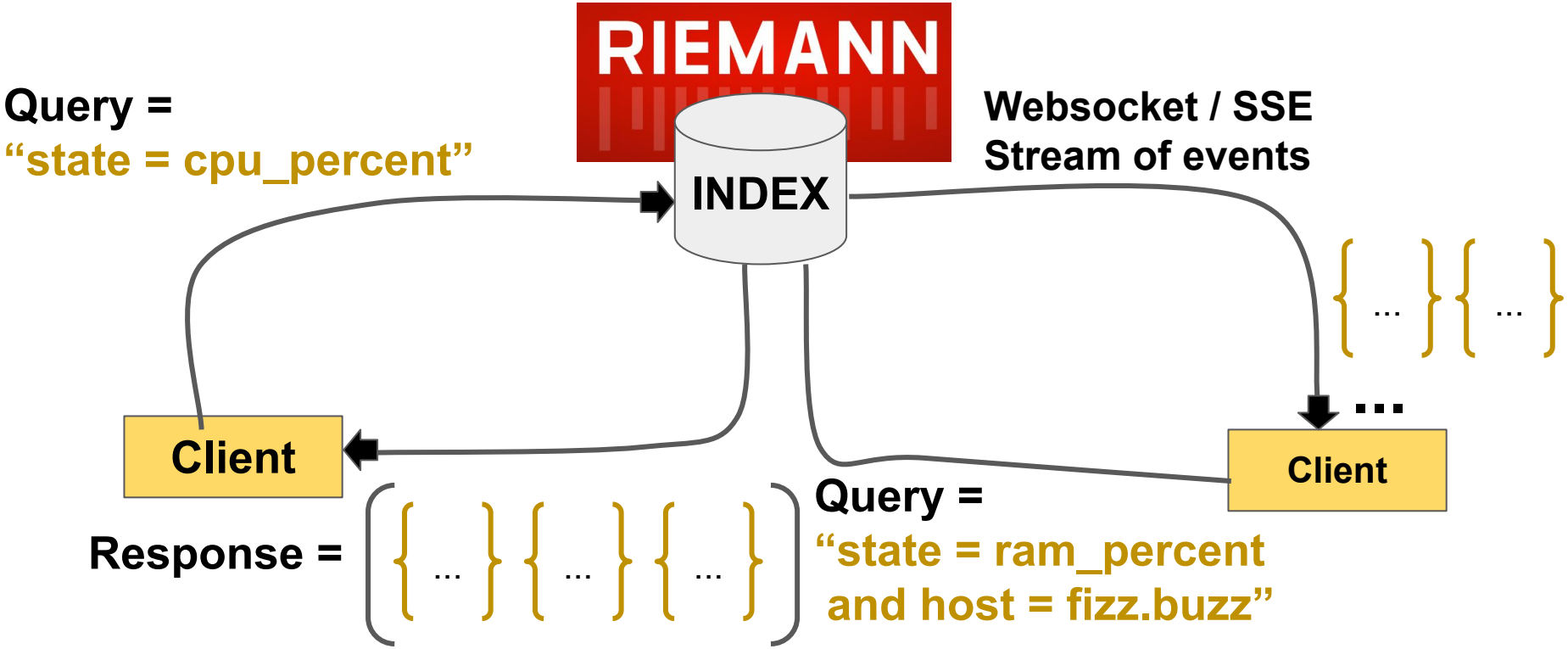
```
:host    "fizz.buzz"
:service "cpu_pe
:metric  90
:ttl     60
:time    3
```

Reinjected in Riemann with :state "expired"

foo.bar **fizz.buzz** *host*

```
(expired
  (email “expired@riemann.io”))
```


RIEMANN
Query =
“state = cpu_percent”
INDEX
Client
Response =
...
...
...


RIEMANN
INDEX
Query =
"state = cpu_percent"
Websocket / SSE
Stream of events
{ ... } { ... }
...
Client
Client
Response =
{ ... } { ... } { ... }
Query =
"state = ram_percent
and host = fizz.buzz"

Audience | render (670×200)

riemann.corp.factual.com/#Audience

Wiki | Calendar | Skuld | Riemann Issues | Riemann | Descartes | Dashboard | Other Bookmarks

Front | DSAPI | DSAPI JVM | Network | Clocks | Riemann | Hearsay | Omphalos | bm-mem | Solr | Front System | Front CPU | FrontFD | FrontReq | Audience × | Audience Test | Outpost | + | websockets sse

Audience

Health

	cpu	load	memory
ip-10-137-20-216	0.69	18.62	0.45
ip-10-137-20-237	0.52	12.14	0.45
ip-10-138-4-27	0.12	0.68	0.12
ip-10-140-9-123	0.08	4.75	0.06
ip-10-141-166-212	0.73	12.88	0.49
ip-10-151-16-87	0.81	19.48	0.48
ip-10-151-69-43	0.66	14.24	0.46
ip-10-190-155-167	0.03	0.32	0.13
ip-10-191-169-209	0.18	3.41	0.14
ip-10-191-185-108	0.04	0.16	0.14
ip-10-238-162-112	0.03	0.32	0.1
ip-10-239-130-244	0.81	16.41	0.46
ip-10-4-174-73	0.05	0.22	0.13
ip-10-4-223-79	0.09	4.67	0.06
ip-10-50-197-74	0.07	0.86	0.13

Disk

	disk /	mnt
ip-10-137-20-216	0.82	0.01
ip-10-137-20-237	0.8	0.01
ip-10-138-4-27	0.31	0.01
ip-10-140-9-123	0.56	0.01
ip-10-141-166-212	0.79	0.01
ip-10-151-16-87	0.78	0.01
ip-10-151-69-43	0.75	0.01
ip-10-190-155-167	0.1	0.01
ip-10-191-169-209	0.12	0.01
ip-10-191-185-108	0.14	0.01
ip-10-238-162-112	0.35	0.01
ip-10-239-130-244	0.72	0.01
ip-10-4-174-73	0.37	0.01
ip-10-4-223-79	0.55	0.01
ip-10-50-197-74	0.13	0.01

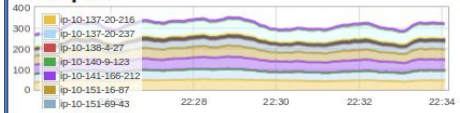
Riak

	disk	node_gets	node_puts	read_repairs	ring	vnode_gets	vnode_puts
ip-10-137-20-216	778.56	61.88	45.83	0.57		141.8	142.45
ip-10-137-20-237	759.44	45.55	43.77	0.35		123.27	126.83
ip-10-138-4-27	289.22	0	2.68	0		11.43	11.5
ip-10-140-9-123	2.13	0	0	0		0	52.23
ip-10-141-166-212	750.3	60.17	52.55	0.32		164.23	186.03
ip-10-151-16-87	738.22	46	48.9	0.33		127.57	138.35
ip-10-151-69-43	705.95	61.32	34.75	0.23		143.15	163.6
ip-10-190-155-167	86.63	0	4.43	0		17.97	18.07
ip-10-191-169-209	105.53	0	5.67	0		21.57	21.75
ip-10-191-185-108	119.22	0	6.32	0		25.1	25.32
ip-10-238-162-112	322.99	0	4.25	0		14.67	14.78
ip-10-239-130-244	678.85	45.27	56.18	0.22		166.83	184.4
ip-10-4-174-73	340.28	14.32	6.47	17.07		37.43	40.1
ip-10-4-223-79	2.48	0	0	0		0	11.08
ip-10-50-197-74	115.63	4.25	5.02	3.88		26.48	26.23

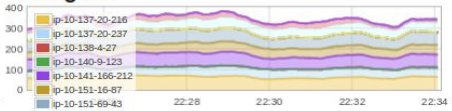
Riak Latencies

	get 50	get 95	get 99	put 50
ip-10-137-20-216	49.55	160.08	279.42	44.05
ip-10-137-20-237	47.64	138.93	244.05	39.52
ip-10-138-4-27	0	0	0	10.34
ip-10-140-9-123	0	0	0	0
ip-10-141-166-212	46.89	148.65	239.31	30.47
ip-10-151-16-87	51.01	161.76	304.2	97.86
ip-10-151-69-43	49.33	143.38	252.34	64
ip-10-190-155-167	0	0	0	3.1
ip-10-191-169-209	0	0	0	3.98
ip-10-191-185-108	0	0	0	2.29
ip-10-238-162-112	0	0	0	2.05
ip-10-239-130-244	47.69	149.85	259.22	32.45
ip-10-4-174-73	49.44	166.86	296.39	2.22
ip-10-4-223-79	0	0	0	0
ip-10-50-197-74	43.99	127.71	223.66	2.38

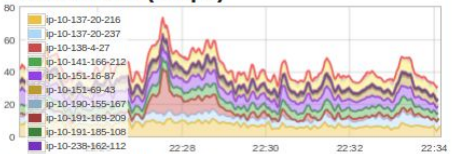
Riak puts



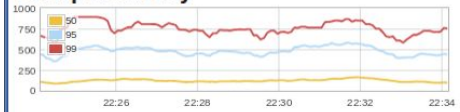
Riak gets



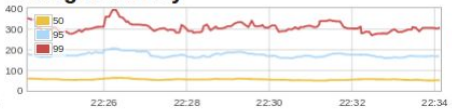
Network out (MBps)



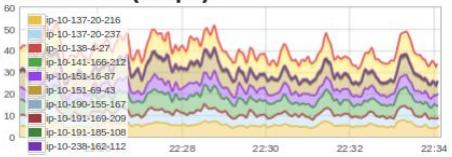
Riak put latency



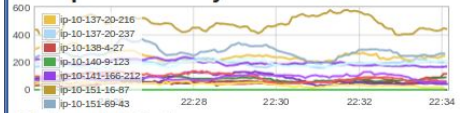
Riak get latency



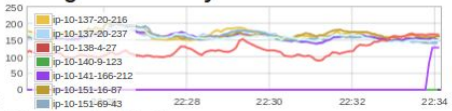
Network in (MBps)



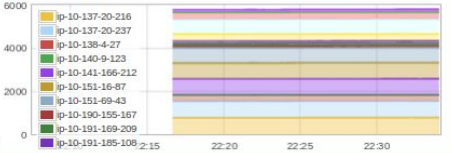
Riak put 95 latency



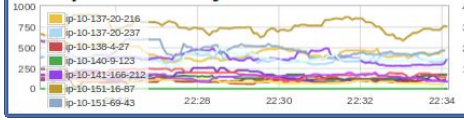
Riak get 95 latency



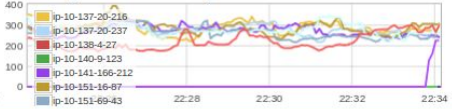
Riak disk



Riak put 99 latency



Riak get 99 latency



JVM

Multithreaded (<3 Clojure)

Netty

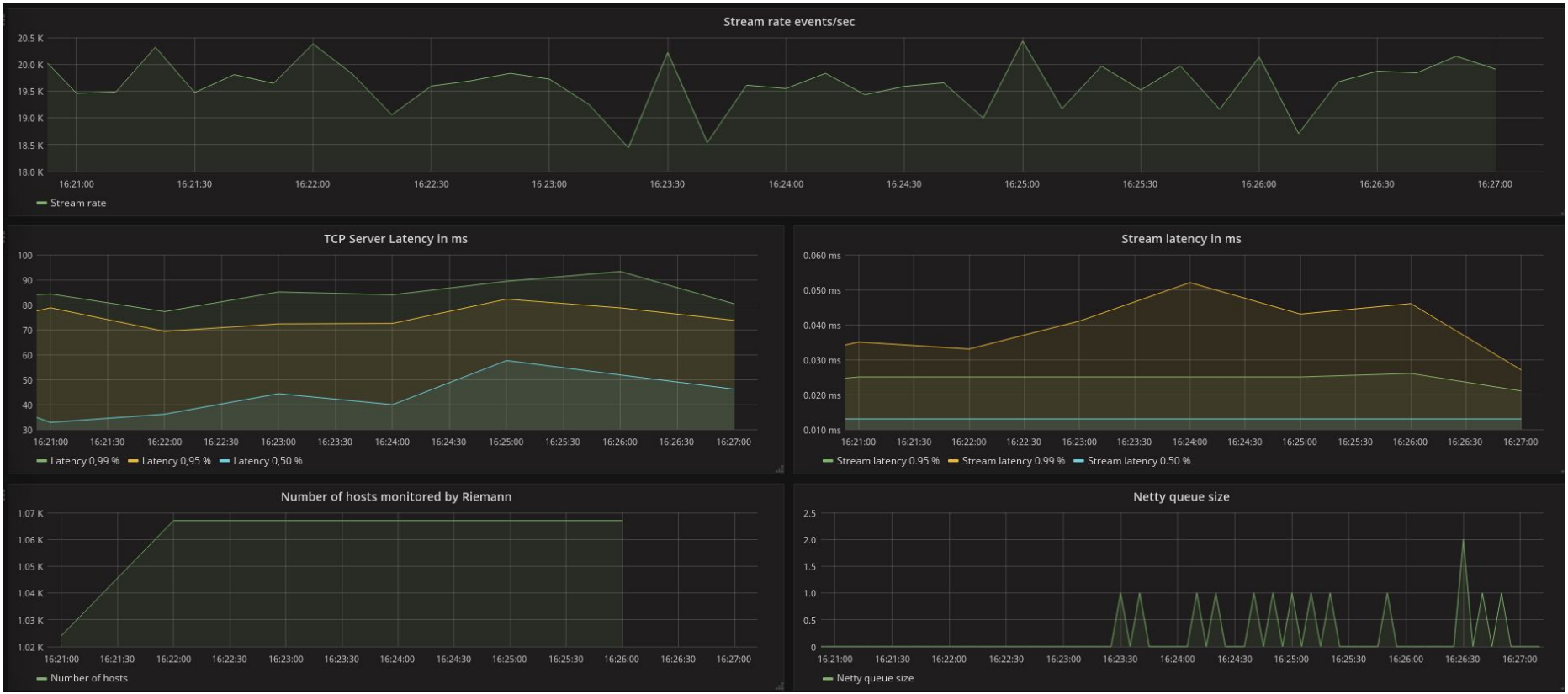
Protobuf

In Memory

Back pressure

...

FAST


Stream rate events/sec
20.5 K
20.0 K
19.5 K
19.0 K
18.5 K
18.0 K
16:21:00
16:21:30
16:22:00
16:22:30
16:23:00
16:23:30
16:24:00
16:24:30
16:25:00
16:25:30
16:26:00
16:26:30
16:27:00
Stream rate
TCP Server Latency in ms
100
90
80
70
60
50
40
30
Latency 0,99 %
Latency 0,95 %
Latency 0,50 %
Stream latency in ms
0.060 ms
0.050 ms
0.040 ms
0.030 ms
0.020 ms
0.010 ms
Stream latency 0.95 %
Stream latency 0.99 %
Stream latency 0.50 %
Number of hosts monitored by Riemann
1.07 K
1.06 K
1.05 K
1.04 K
1.03 K
1.02 K
Number of hosts
Netty queue size
2.5
2.0
1.5
1.0
0.5
0
Netty queue size

No HA

- Sharding
- Send events to 2 instances
- Keepalived

But Clojure is hard ! (((((lisp)))))

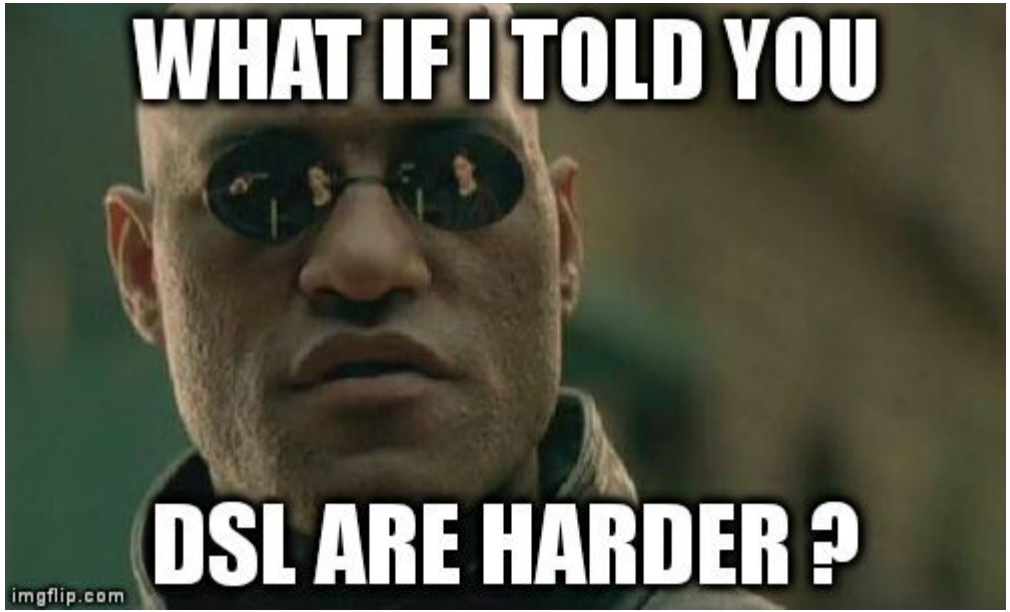
Prometheus :

```
ALERT IncreasedErrorRate
IF (sum(backend_code:haproxy_server_http_responses_total:irate1m{tier="lb", environment
="prd", code="5xx"}) by (code,backend)) > 5
FOR 15s
LABELS {severity="critical"}
ANNOTATIONS {
             title="Increased Error Rate Across Fleet",
             description="We are having a high rate of 5xx accross the fleet. It's like
ly that customers are impacted.",
             runbook="troubleshooting/high-error-rate.md"
             }
```

Zabbix :

```
{Template PfSense:hrStorageFree[{#SNMPVALUE}].last()}<{Template PfSense:hrStorageSize[{
#SNMPVALUE}].last()}*0.1
```

But Clojure is hard ! (((((lisp)))))



Thanks !

riemann.io

Questions ?