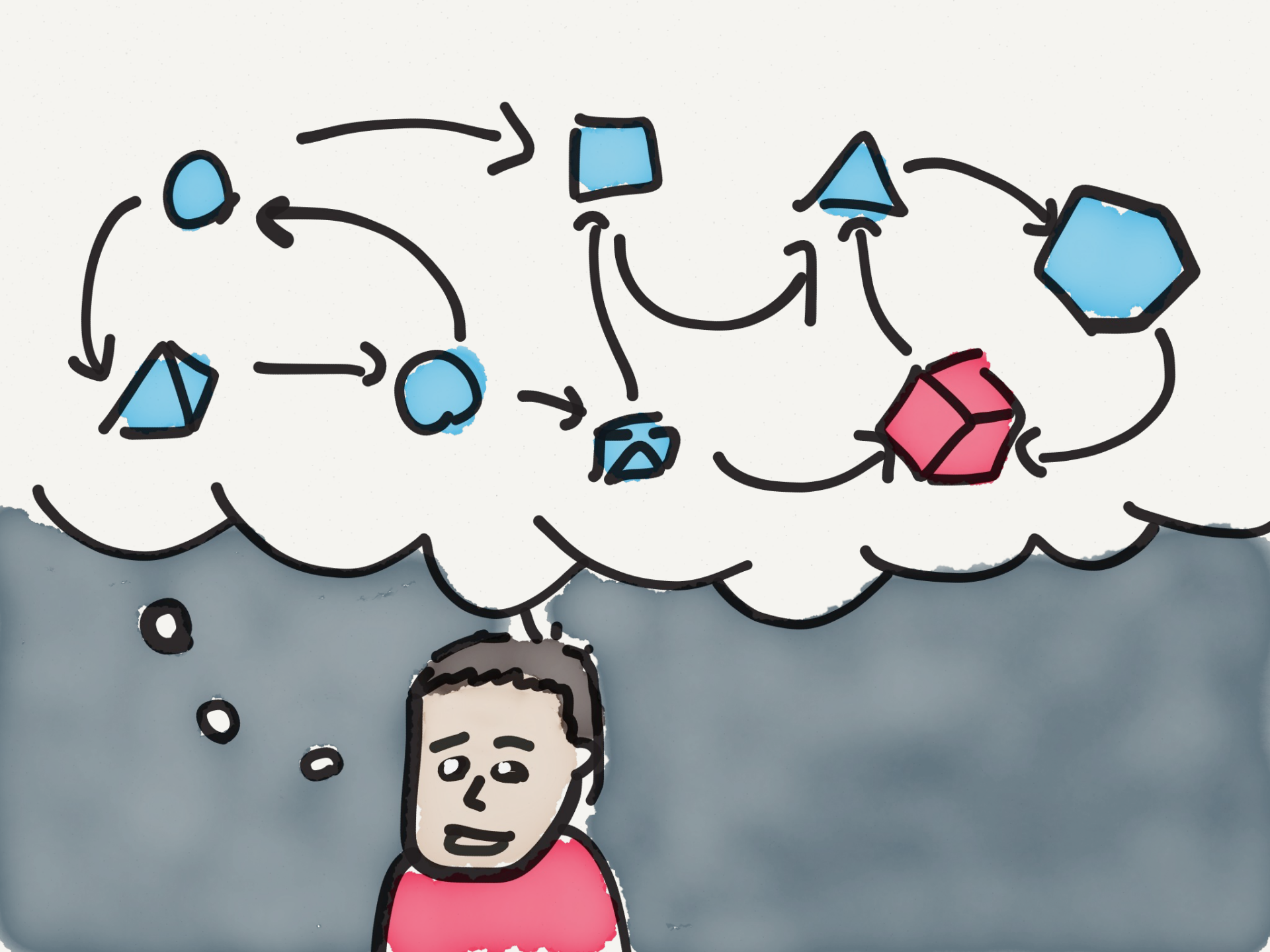
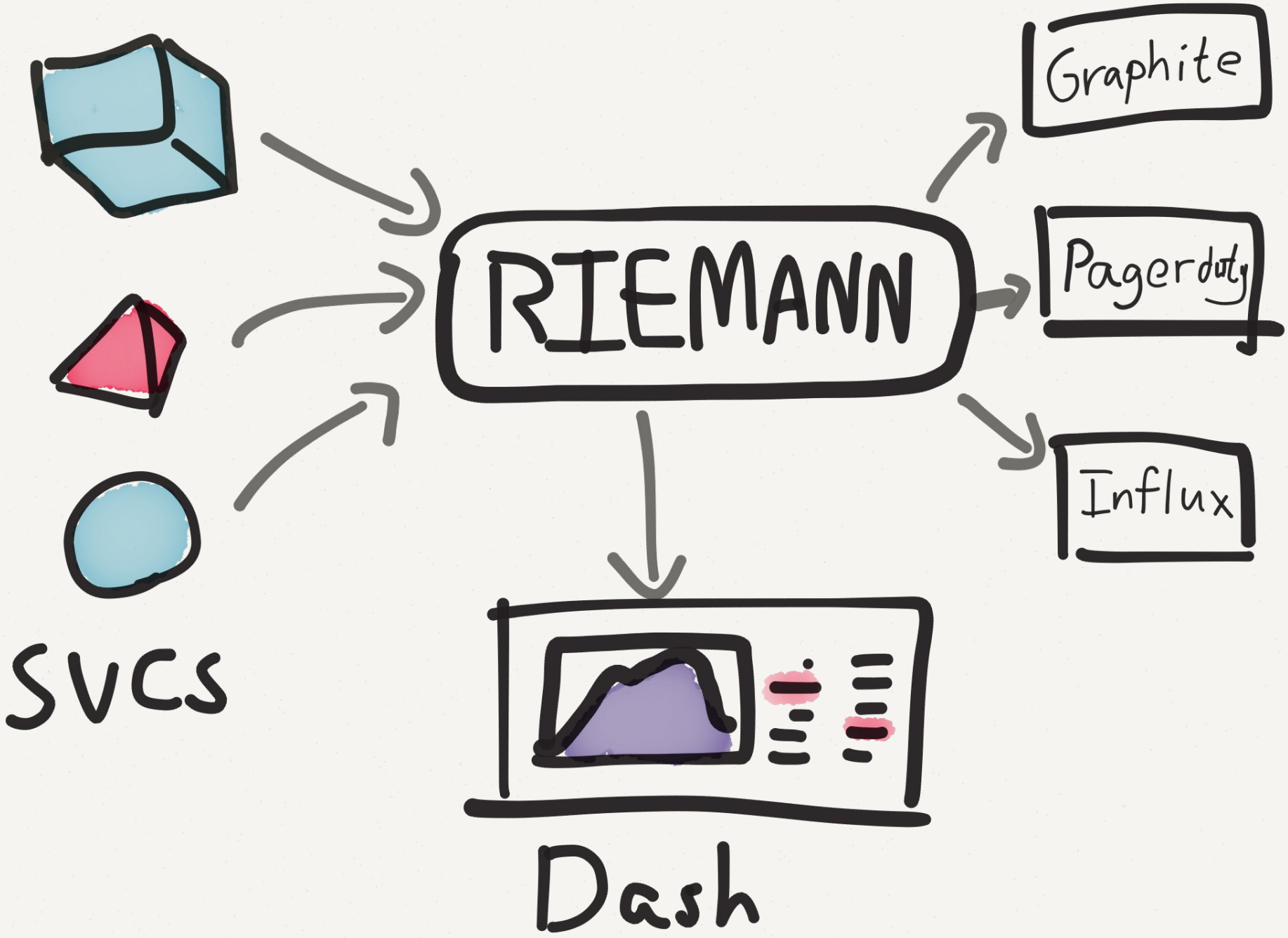

Working with
RIEMANN
Monitorama!

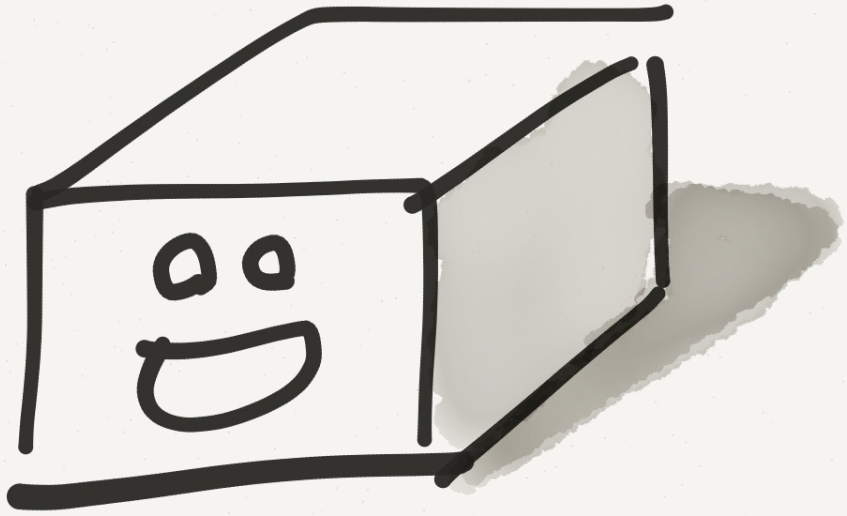


VISIBILITY

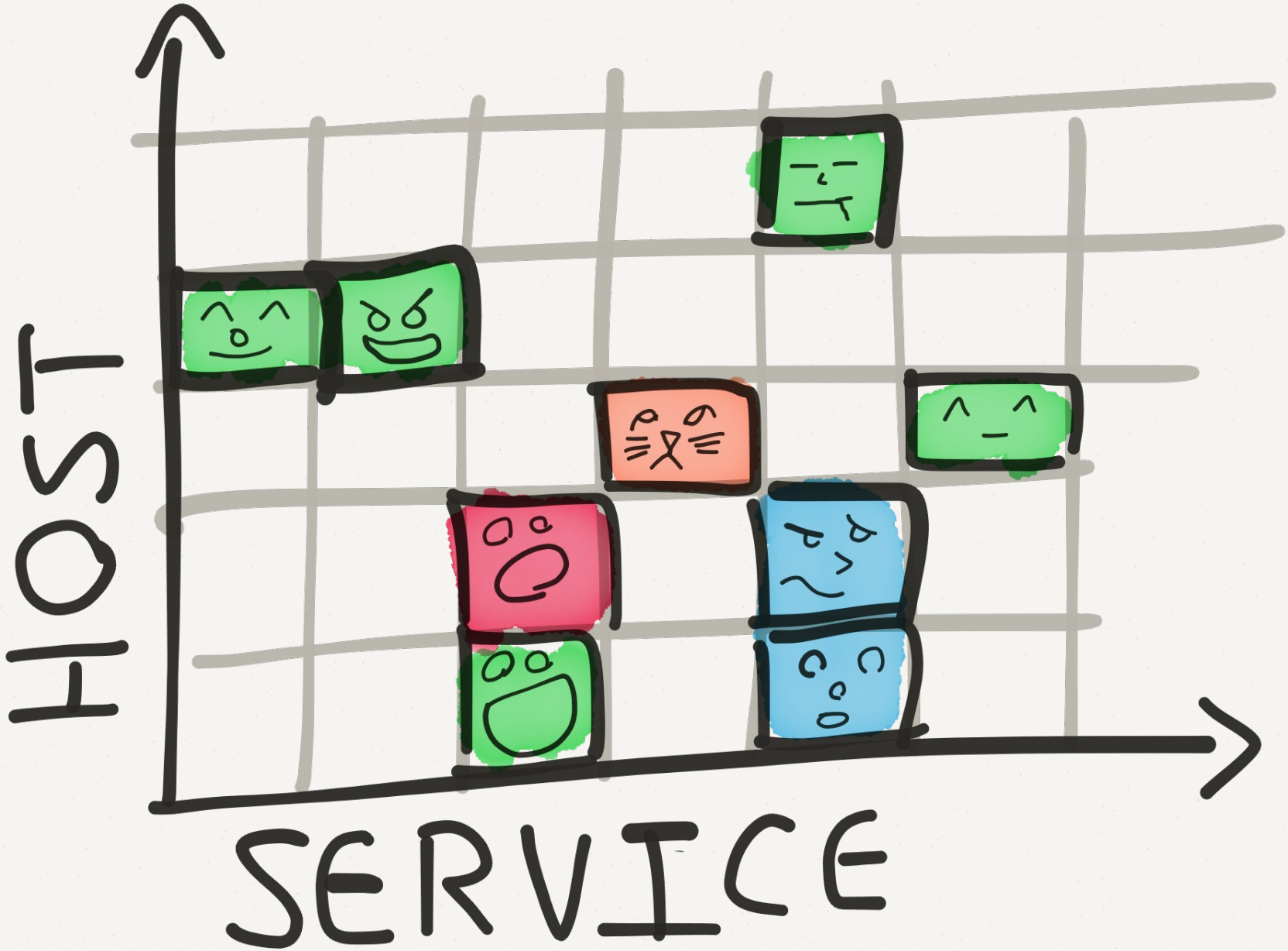

Graphite
RIEMANN
Pagerduty
Influx
SVCs
Dash

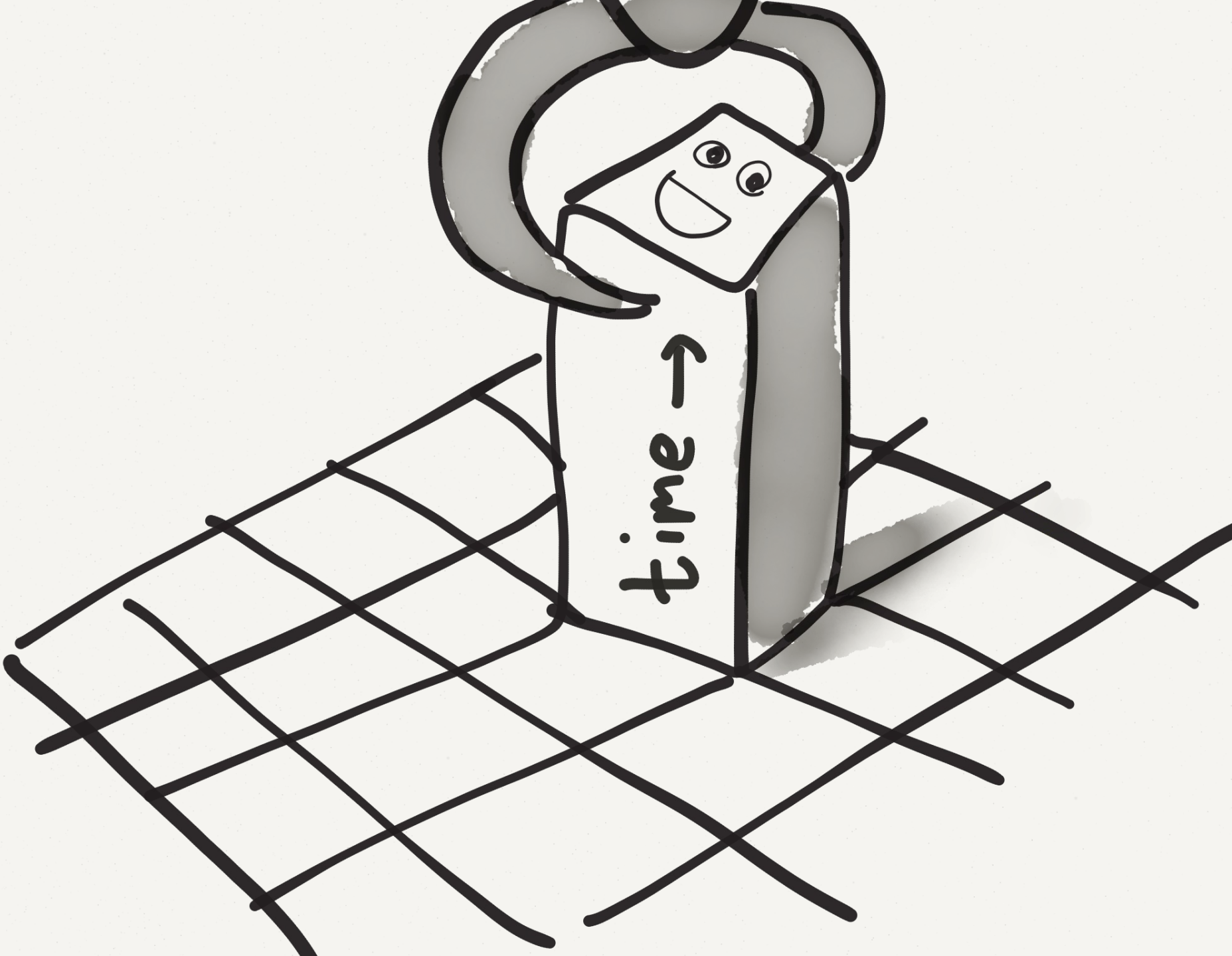
What KIND of INFORMATION?

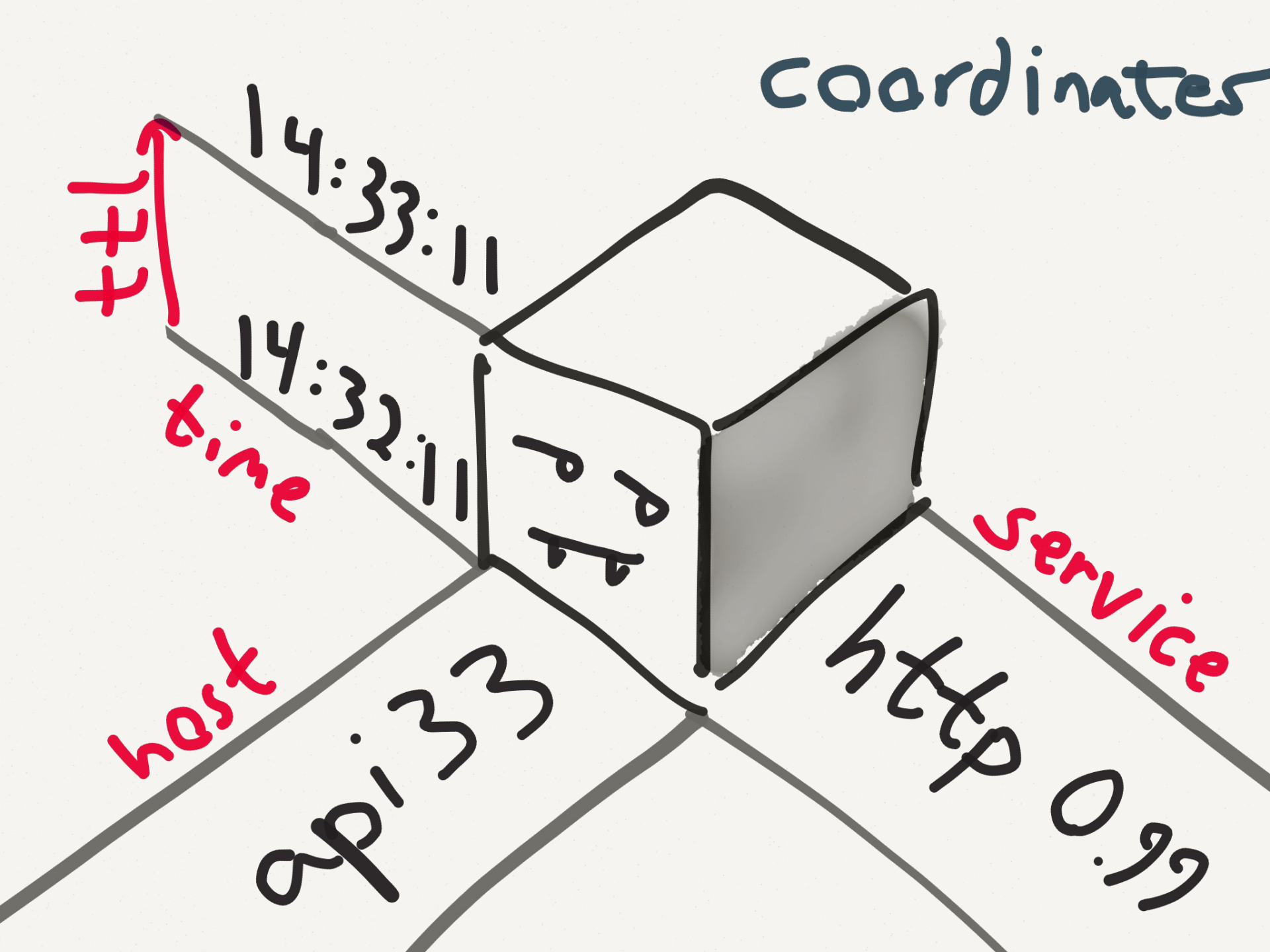
fig. 1



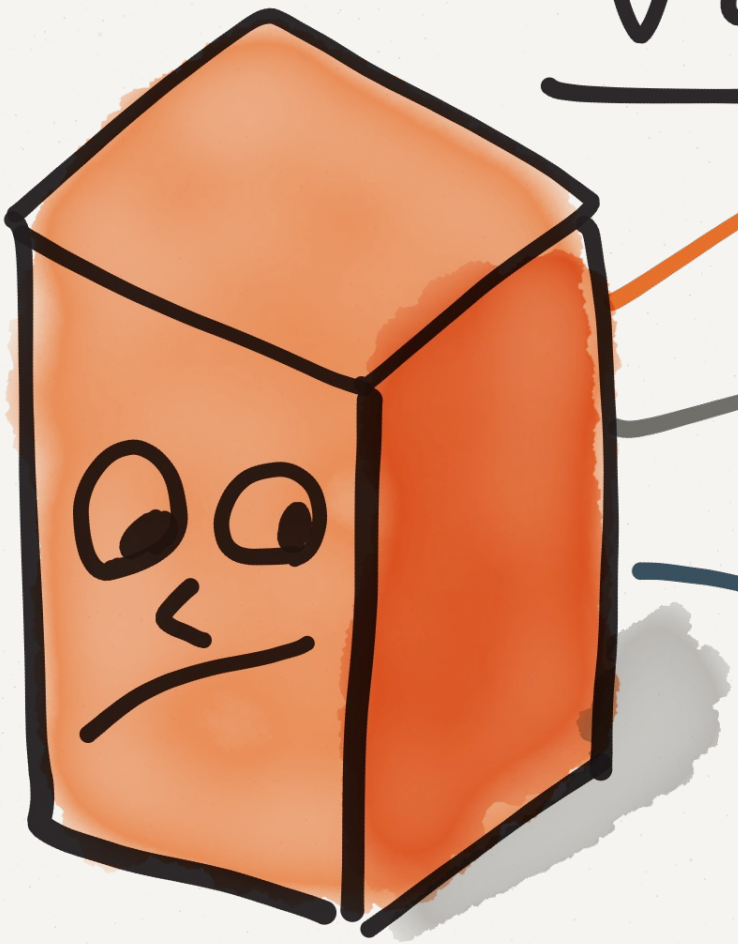
An Event.


HOST
SERVICE


time →


coordinates
14:33:11
14:32:11
time
host
api:33
service
http 0.99

Values

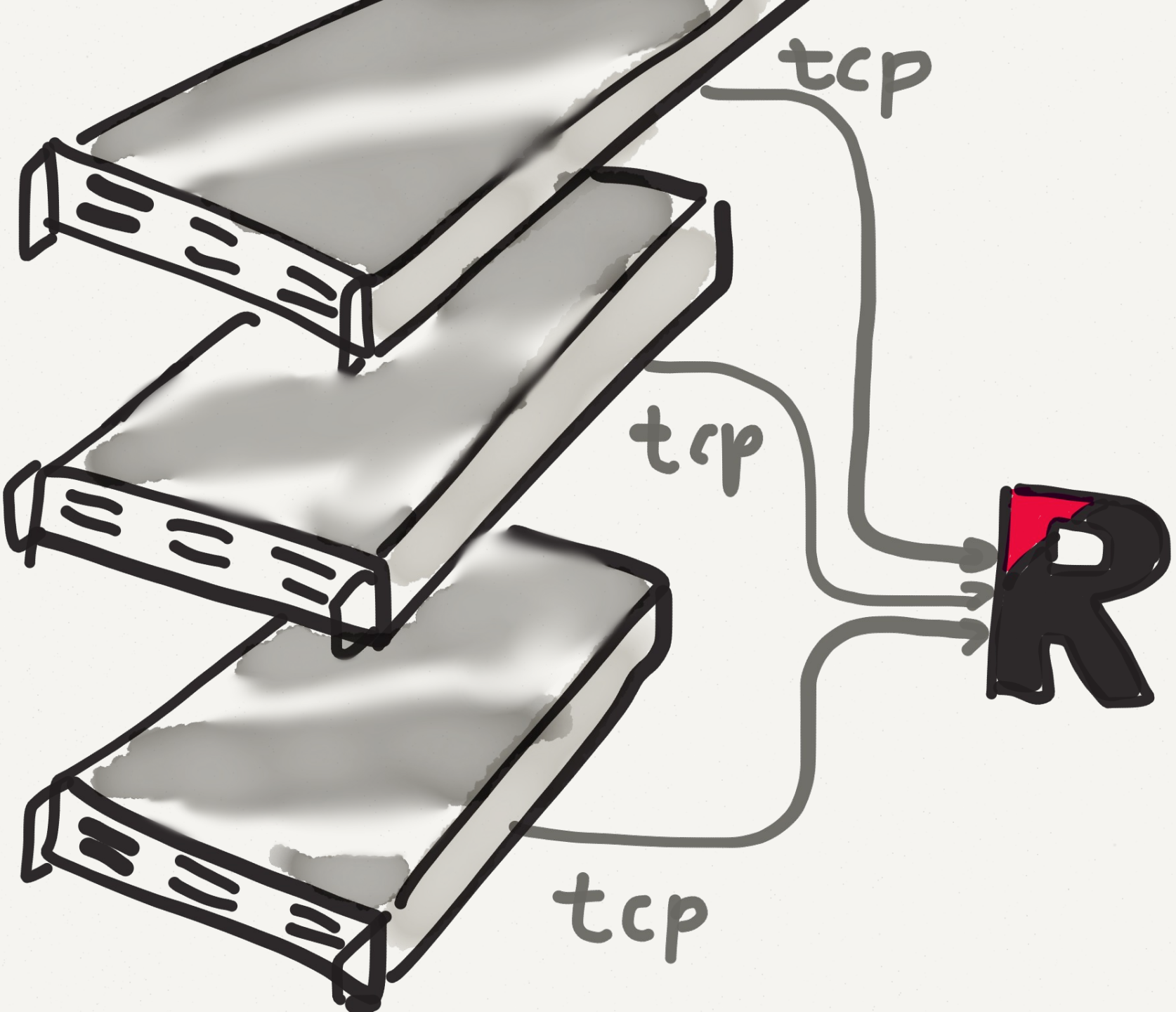


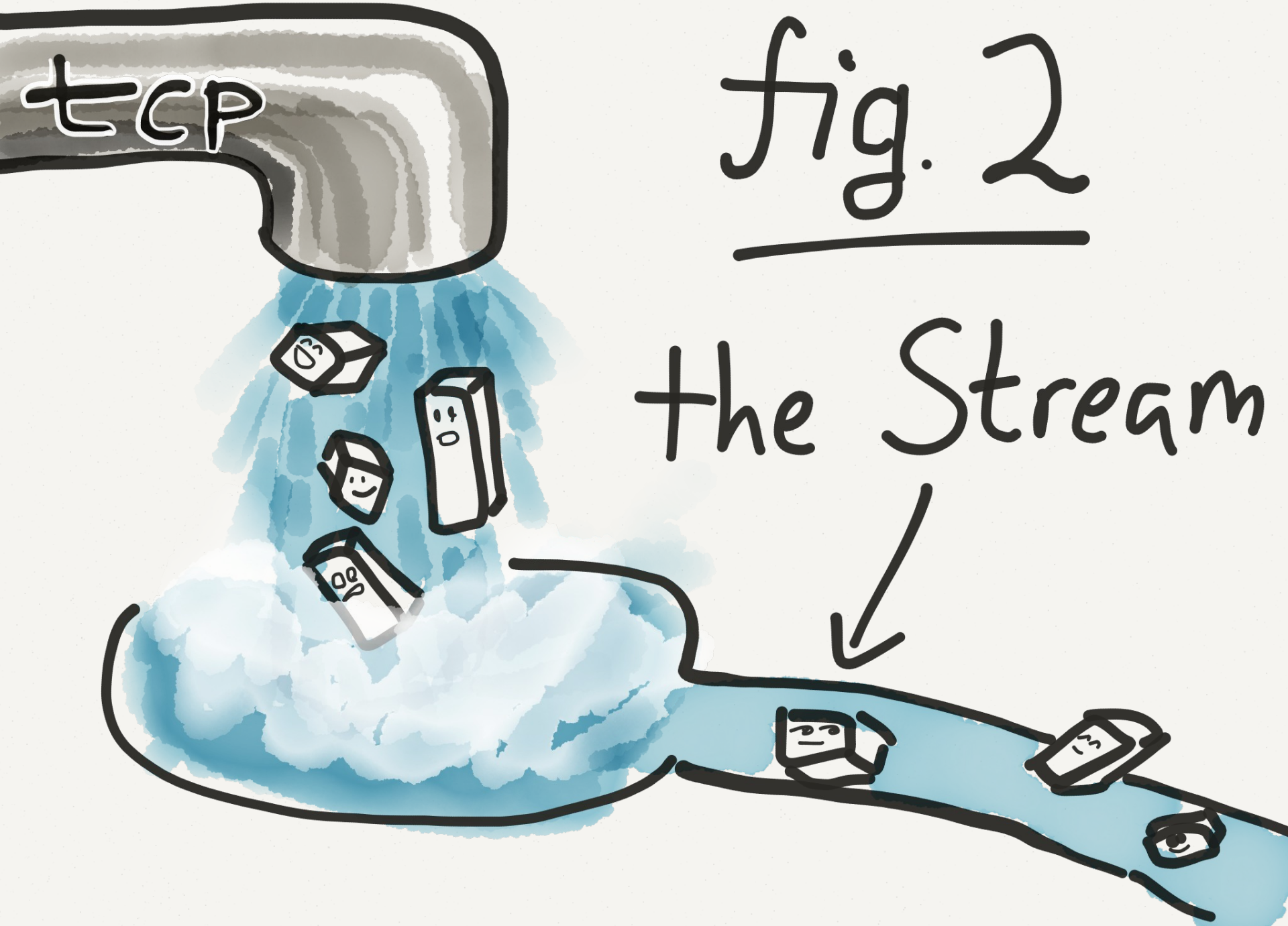
Warning

0.337

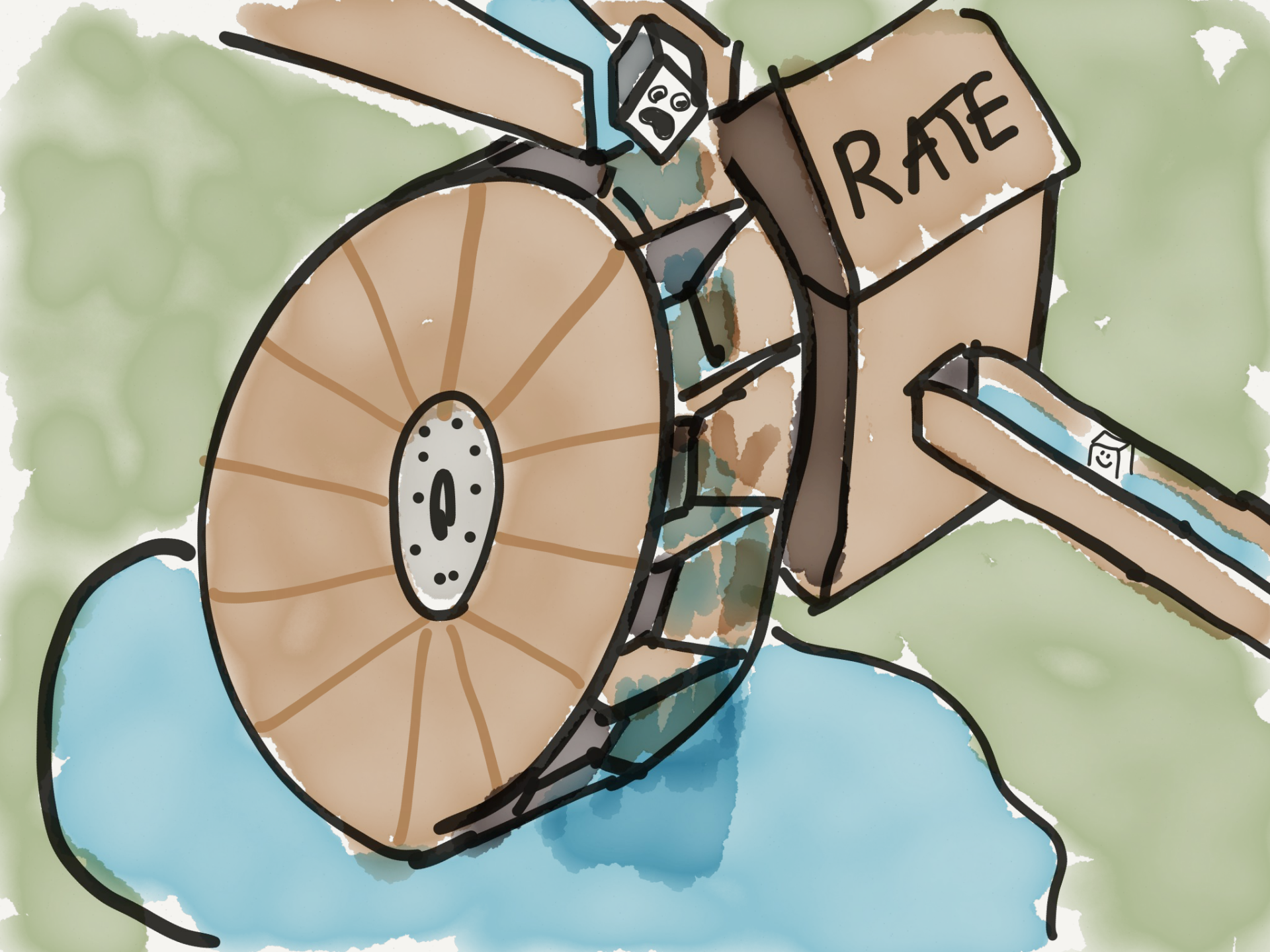
unexpected token \n in request body...


:host api3.tx
:service disk /var free
:time 1230774 9
:ttl 60
:state ok
:metric 0.37
:description 502 GB free
:tags api, disk

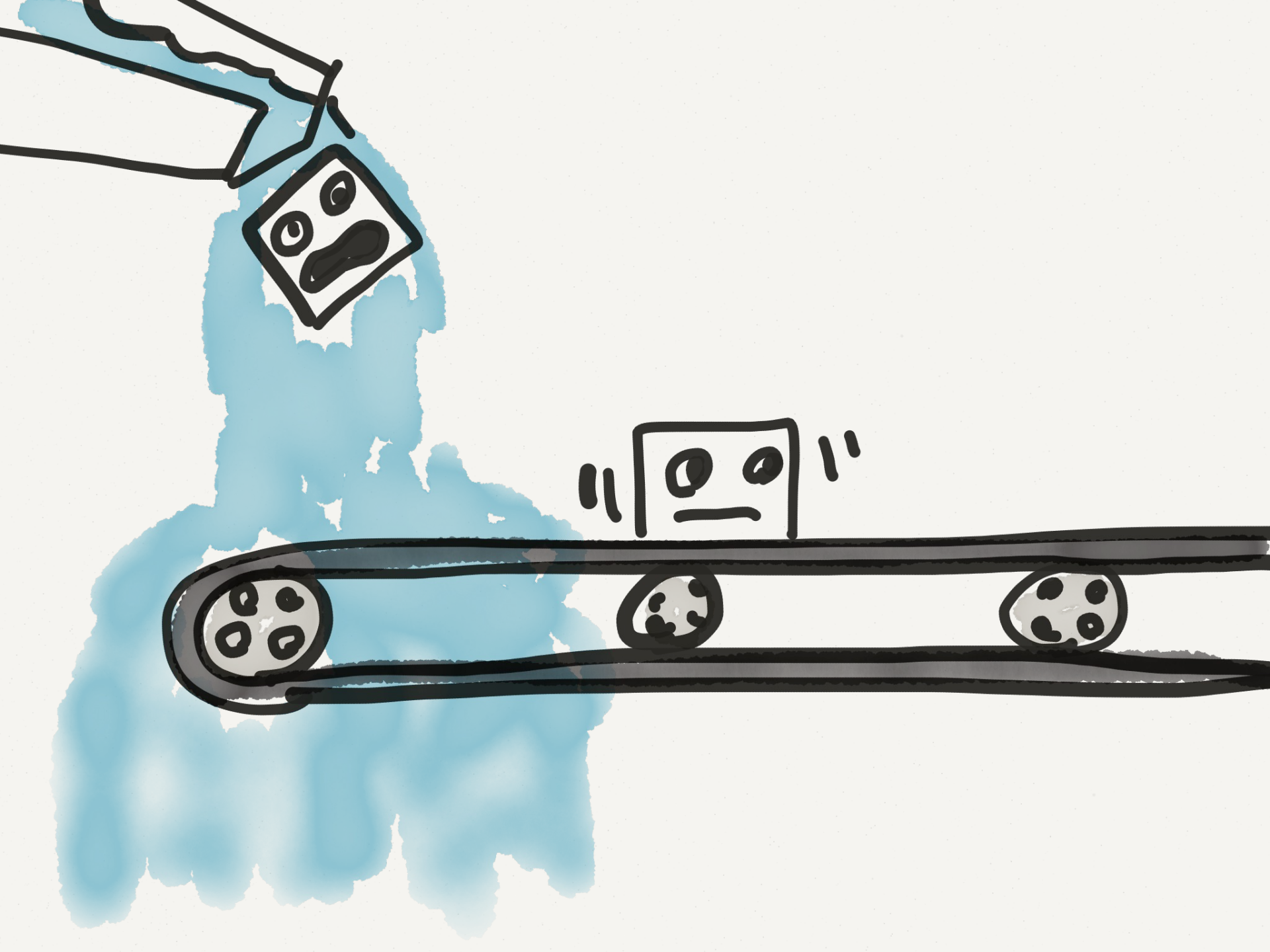

tcp
tcp
tcp
R

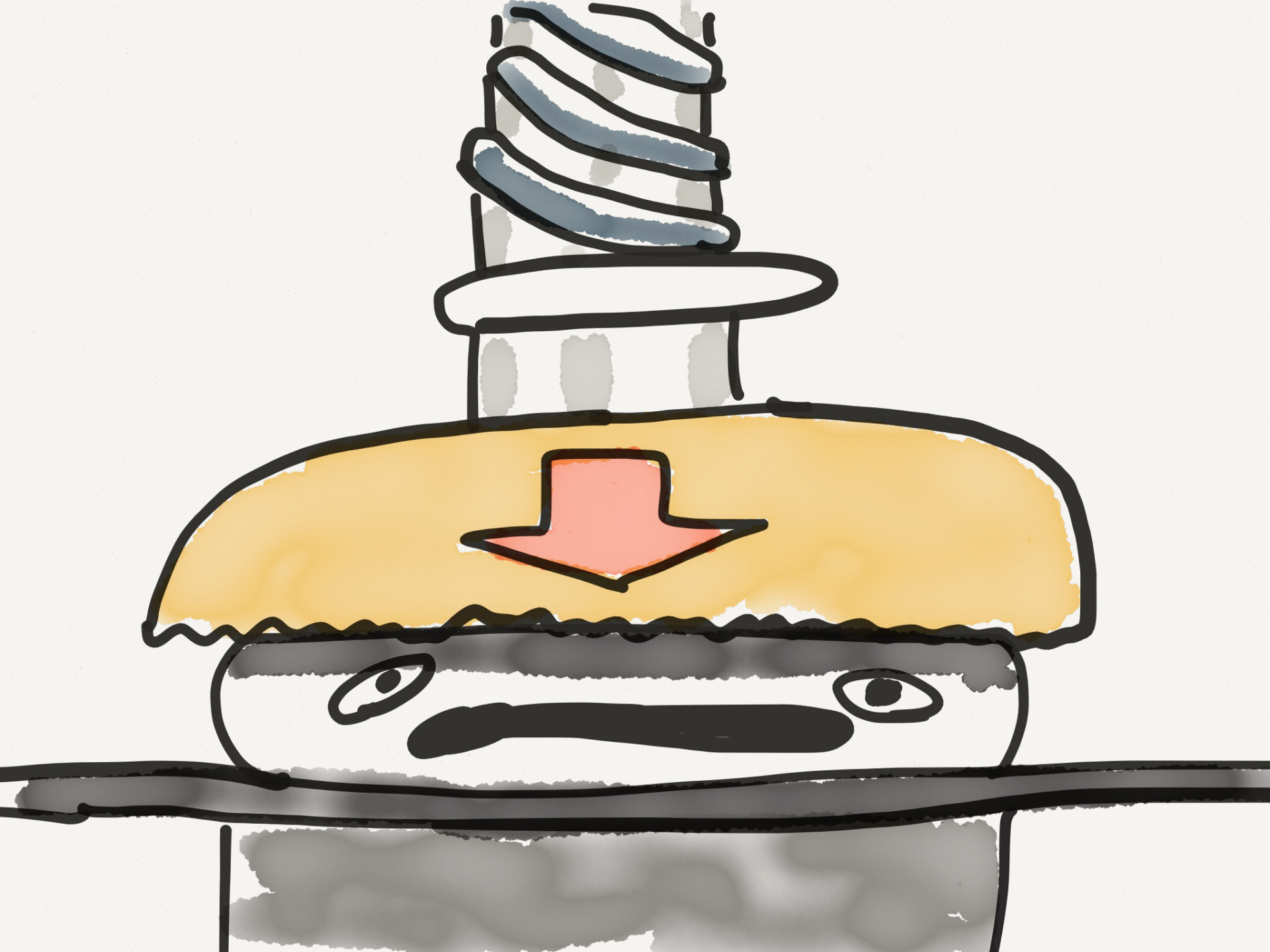

tcp
fig. 2
the Stream

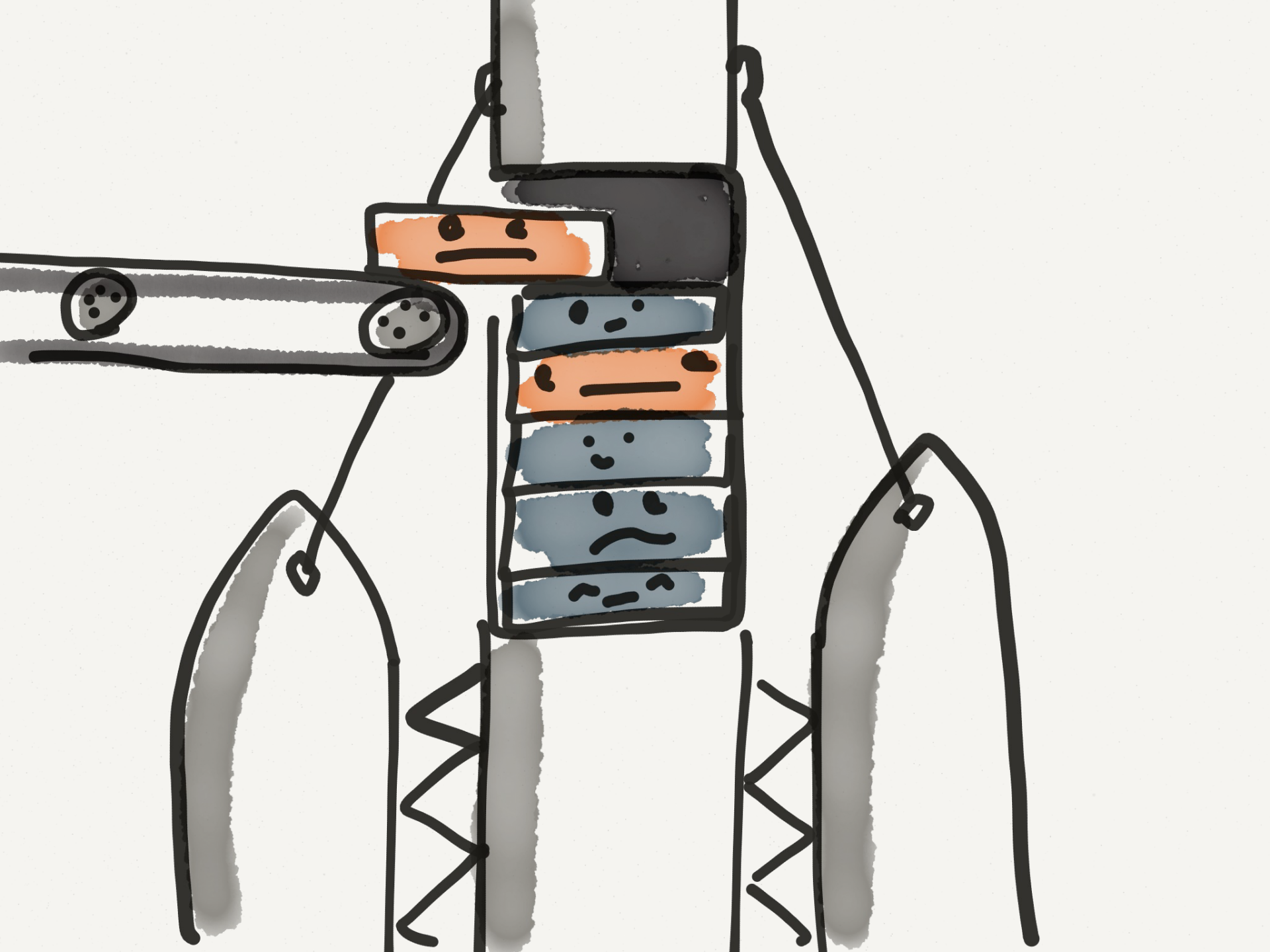

where

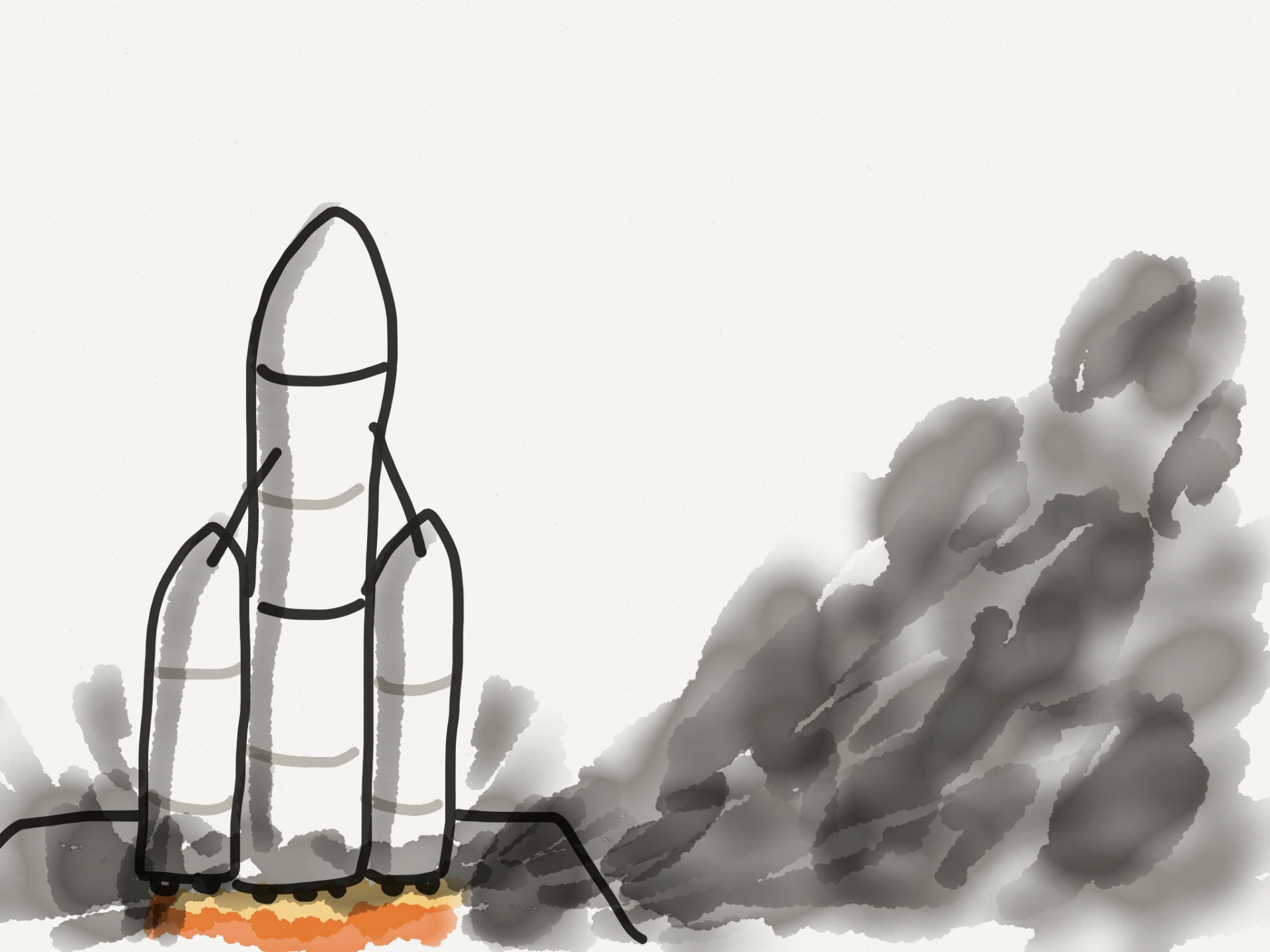

RATE

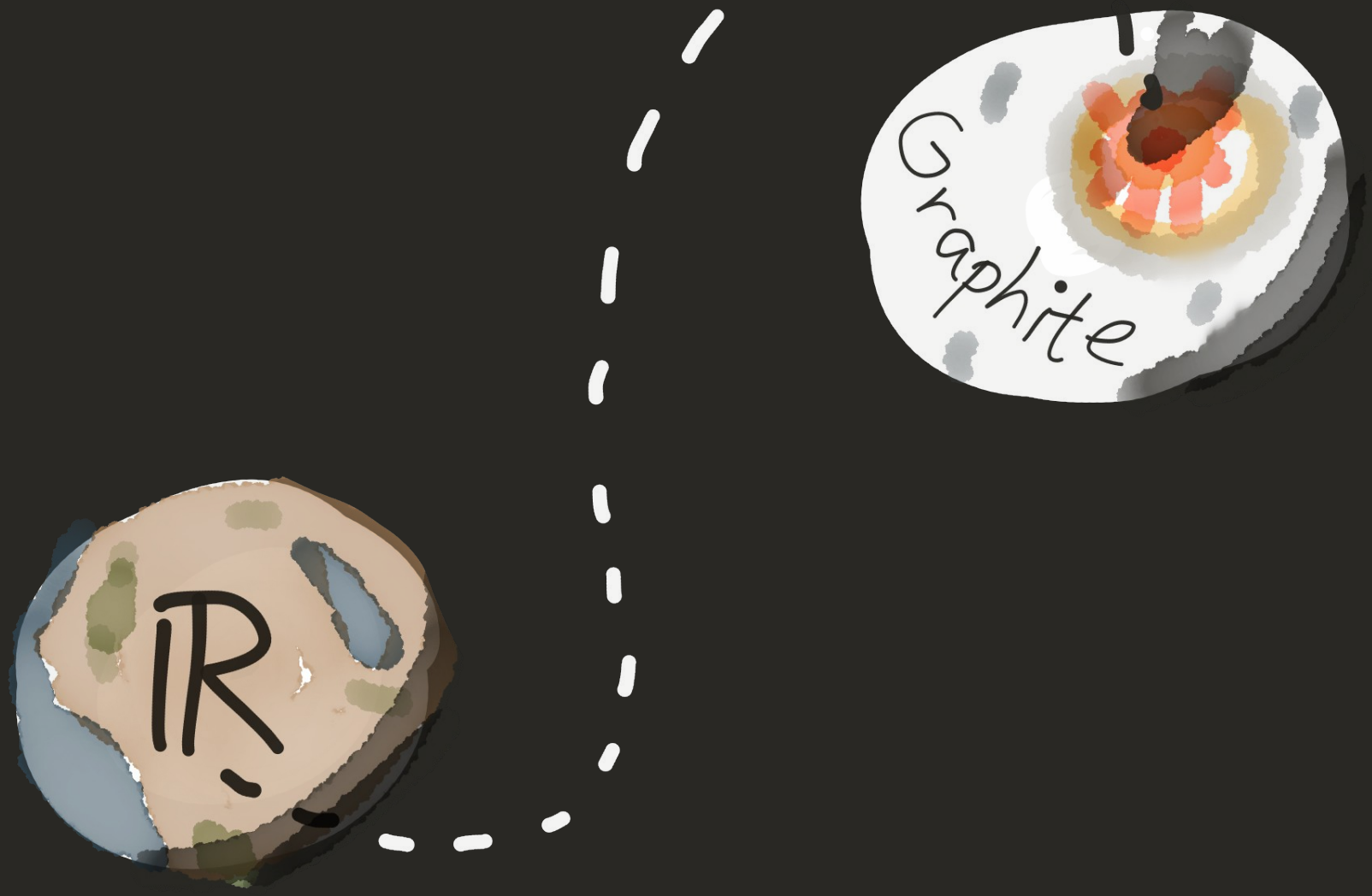


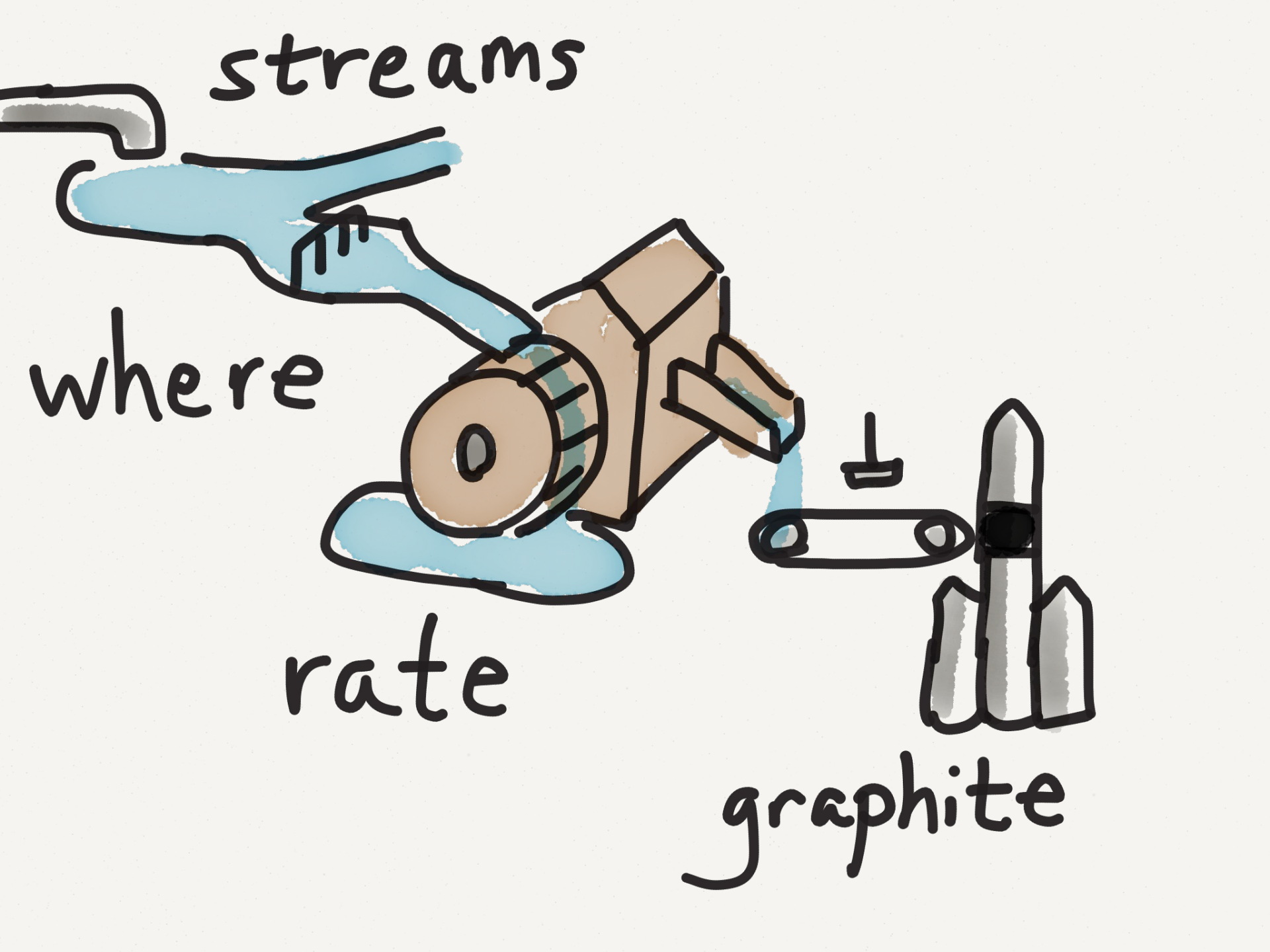









Graphite
R


streams
where
rate
graphite

(streams



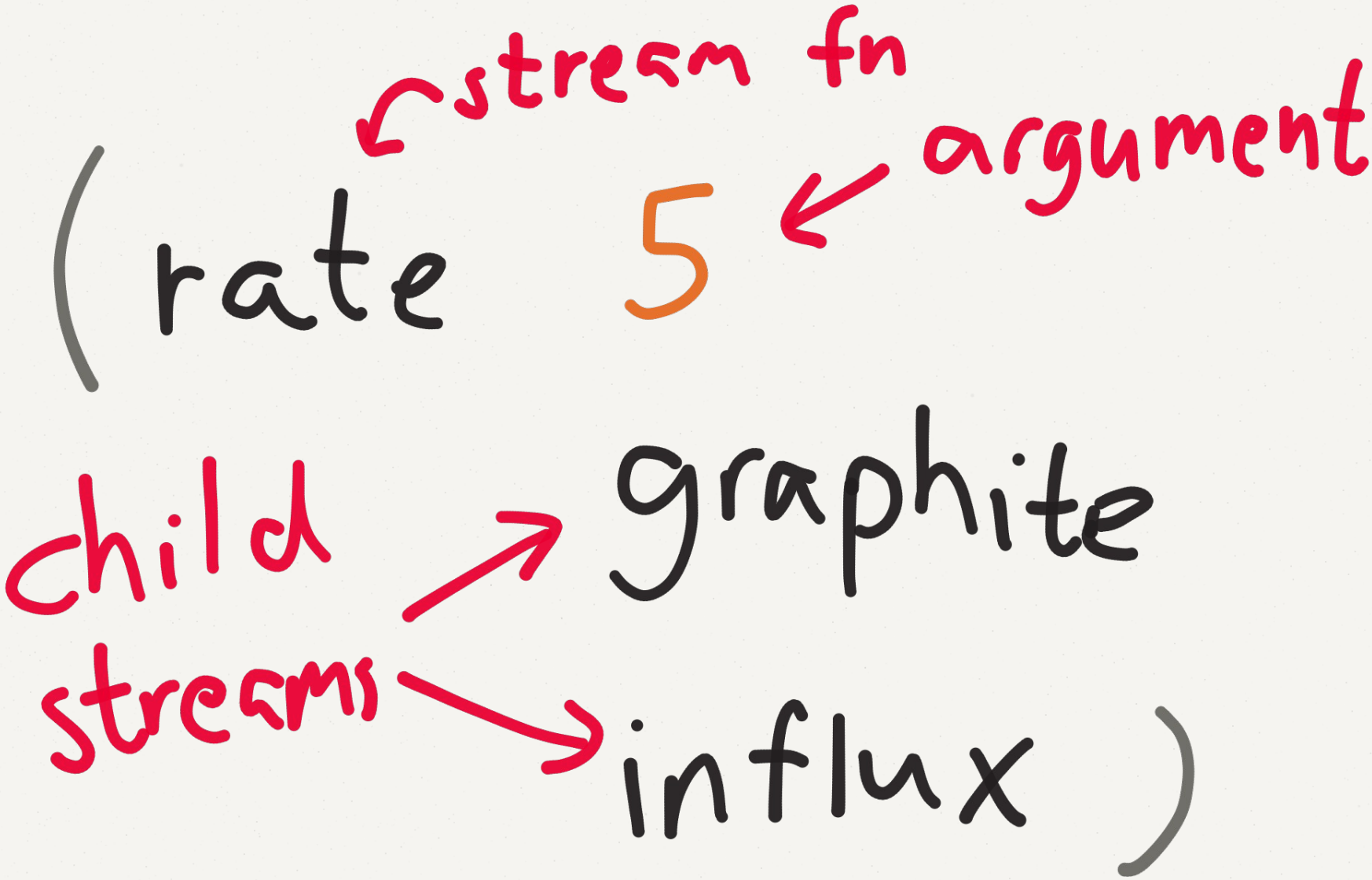
(where

(rate

graphite)))

(streams
(where
(rate
graphite)))

```
(streams
  (where (service "reqs")
    (rate 5
      graphite)))
```


stream fn
argument
(rate 5
child streams
graphite
influx)

Filtering

```
(where (host #"api\d+"))
 (where (< 2 metric 5))
(where (tagged "cat"))
```

```
(where
  (and (not (state "ok"))
       (service "foo")
       (host #".+\.tx"))
  ...)
```

Changing Events

```
(with :host nil
      graph)
```

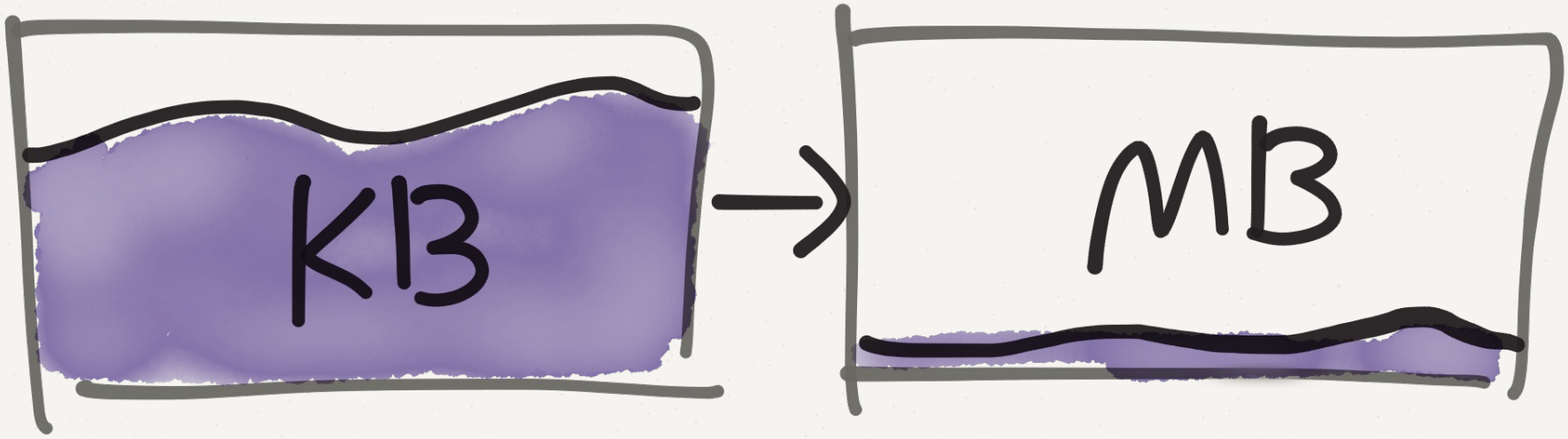
```
(default :ttl 60
  graph)
```

```
(adjust [:service
         str
         "  rate"]
  (rate 5 graph))
```

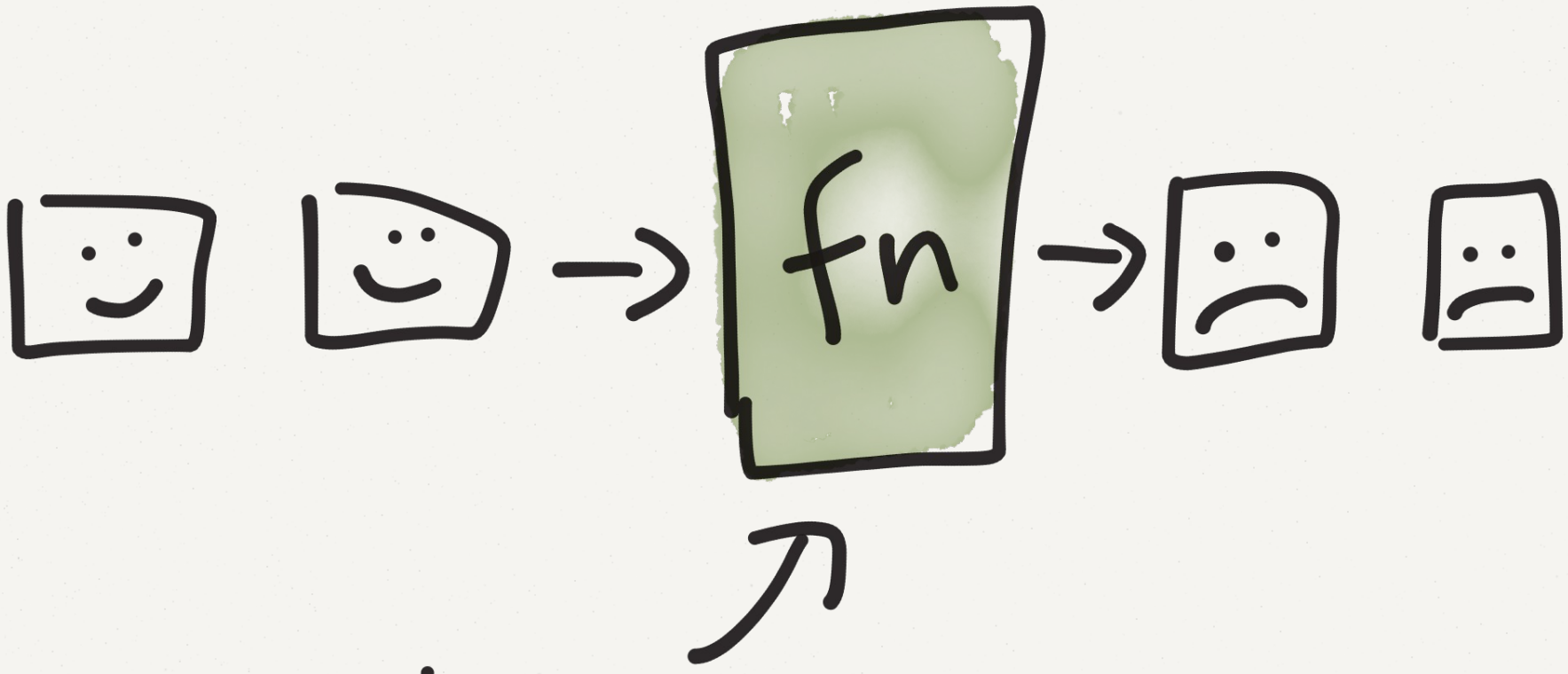
```
(str (:service event)
     " rate")
```

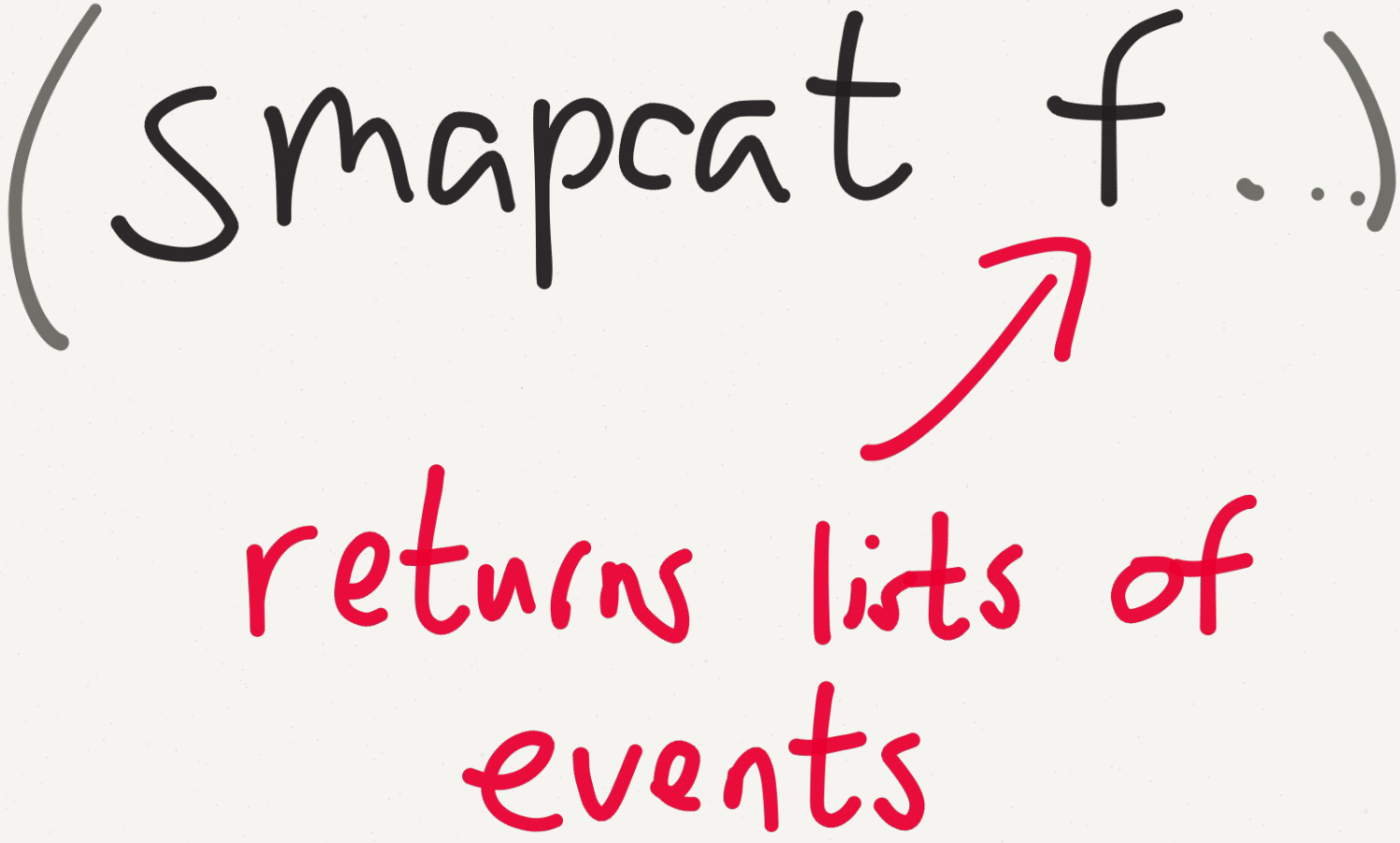
"foa" → "foo rate"

(scale 1/1024 graph)



```
(smap (fn [e]
  (assoc e :datacenter
           "texas"))
  ...)
```


fn
sadtron 2000

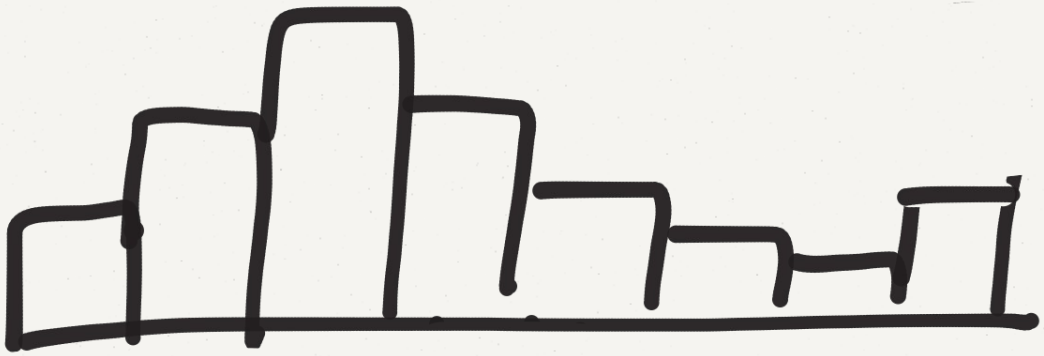

(smapcat f ...)
returns lists of events

```
(sreduce
  (fn [prev-event
       cur-event]
    ...)
  graph))
```

Differentials & Integrals

(ddt graph)

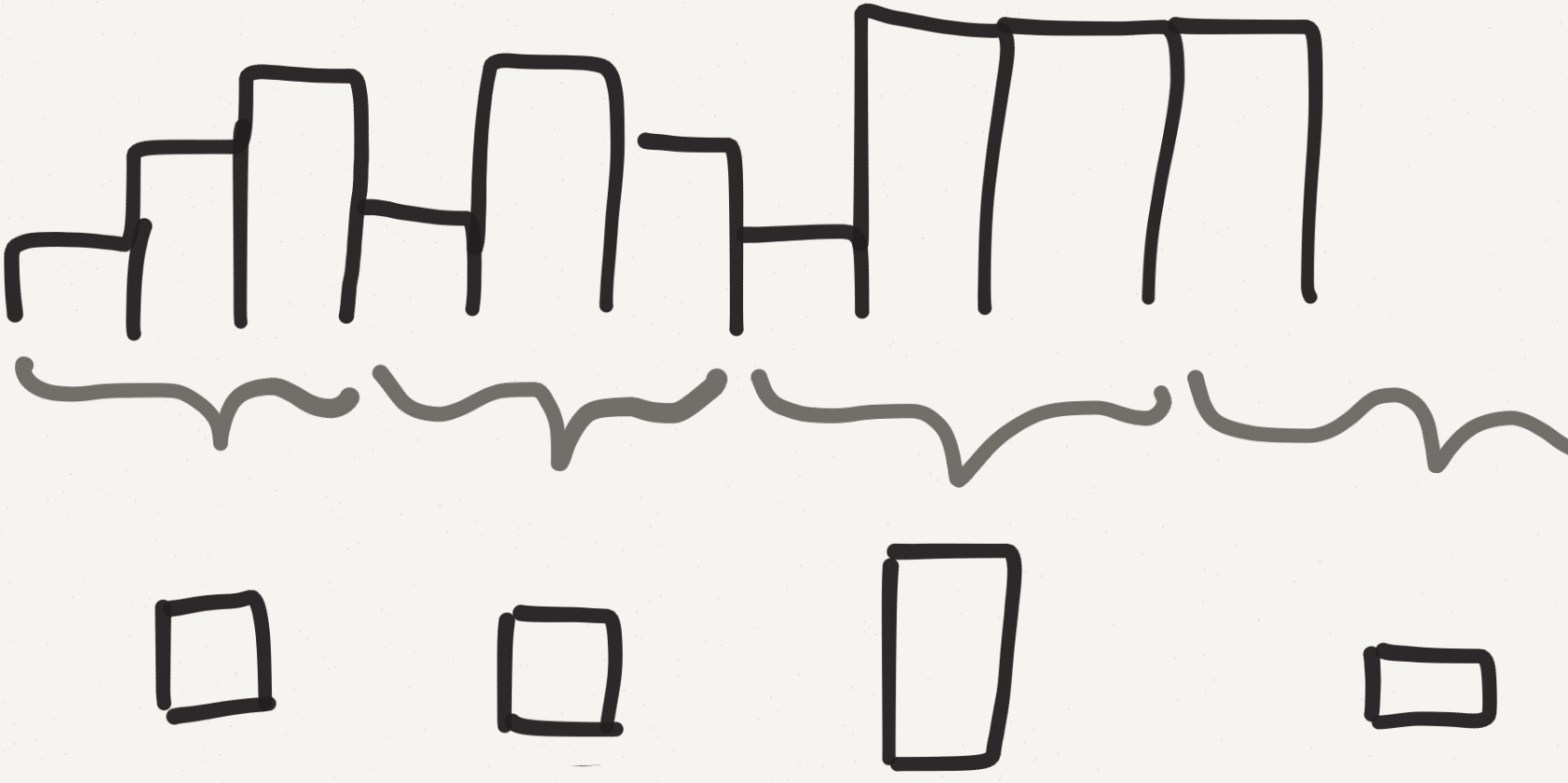
in:



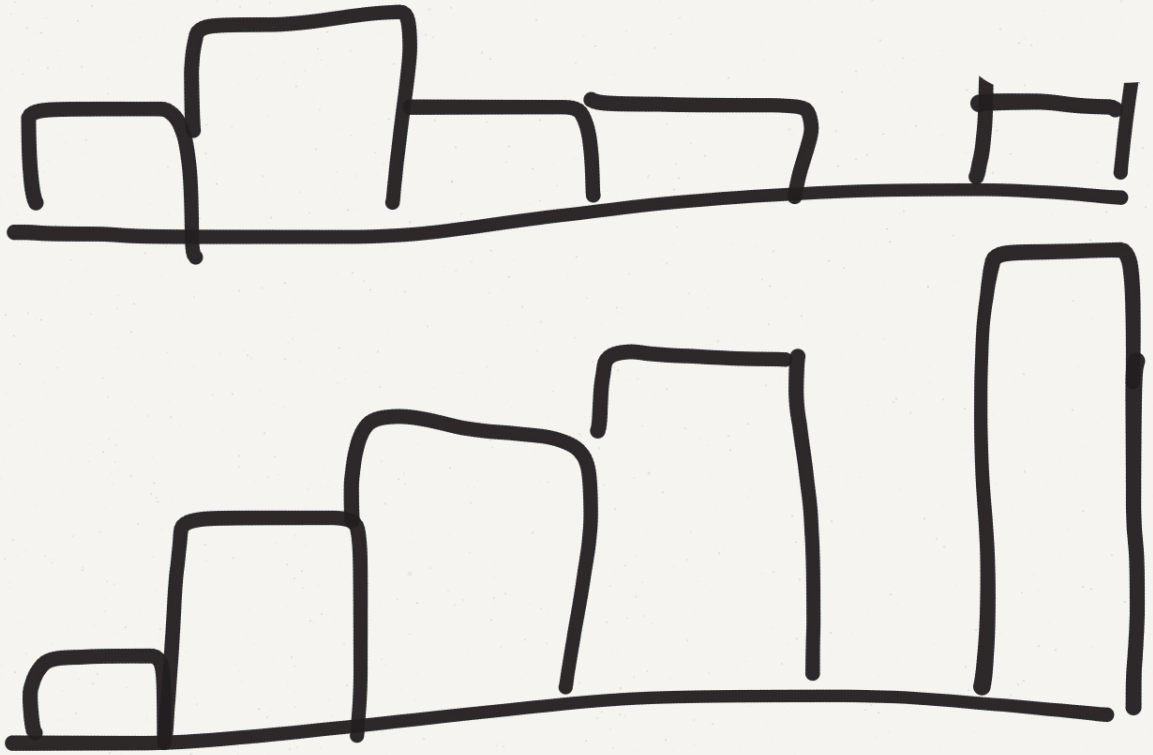
out:



(rate 5 graph)



(counter ∘ graph)



Smoothing

≠

Filtering

```
(changed
  (fn [e] ...))

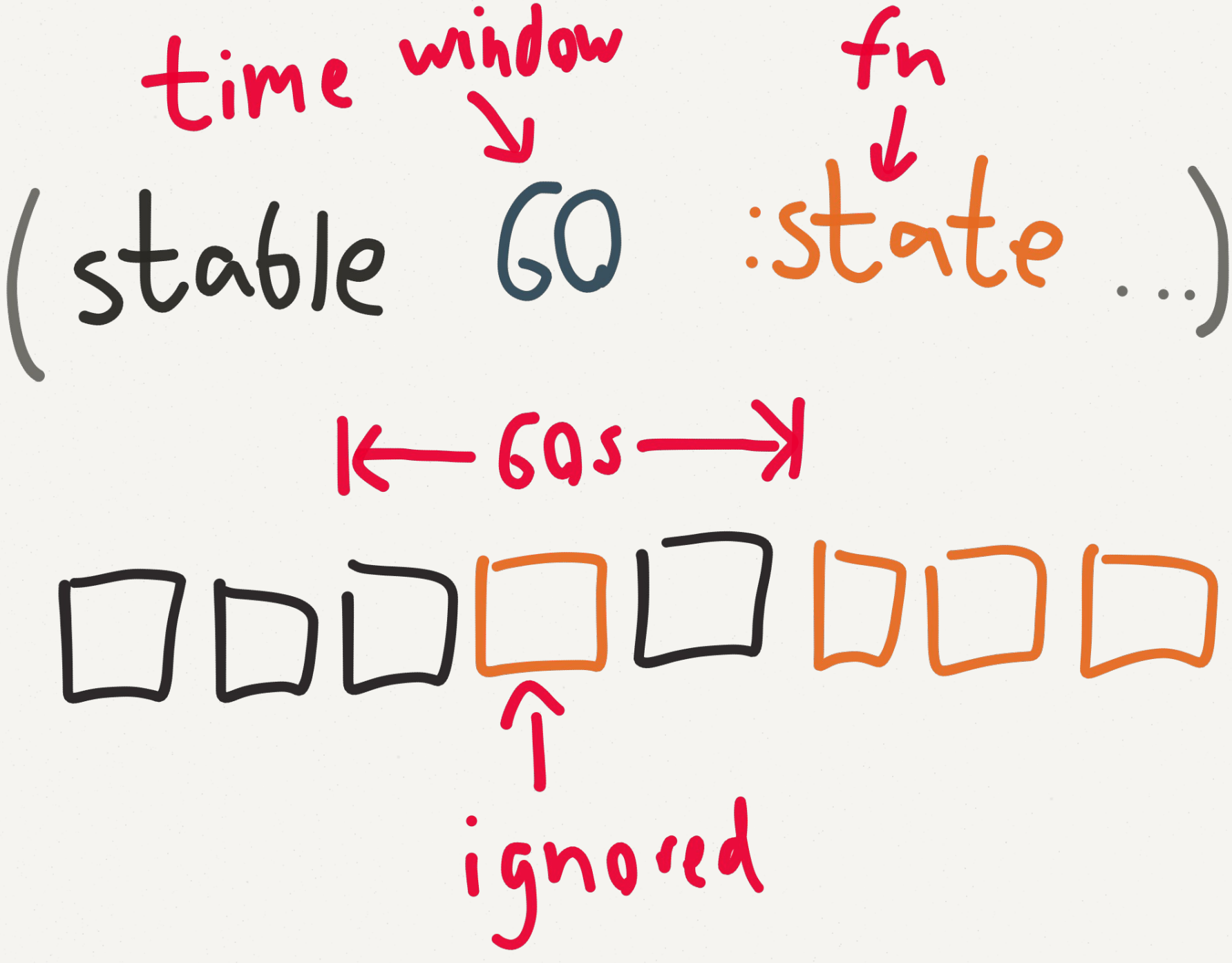
(changed :state)
```

half-life

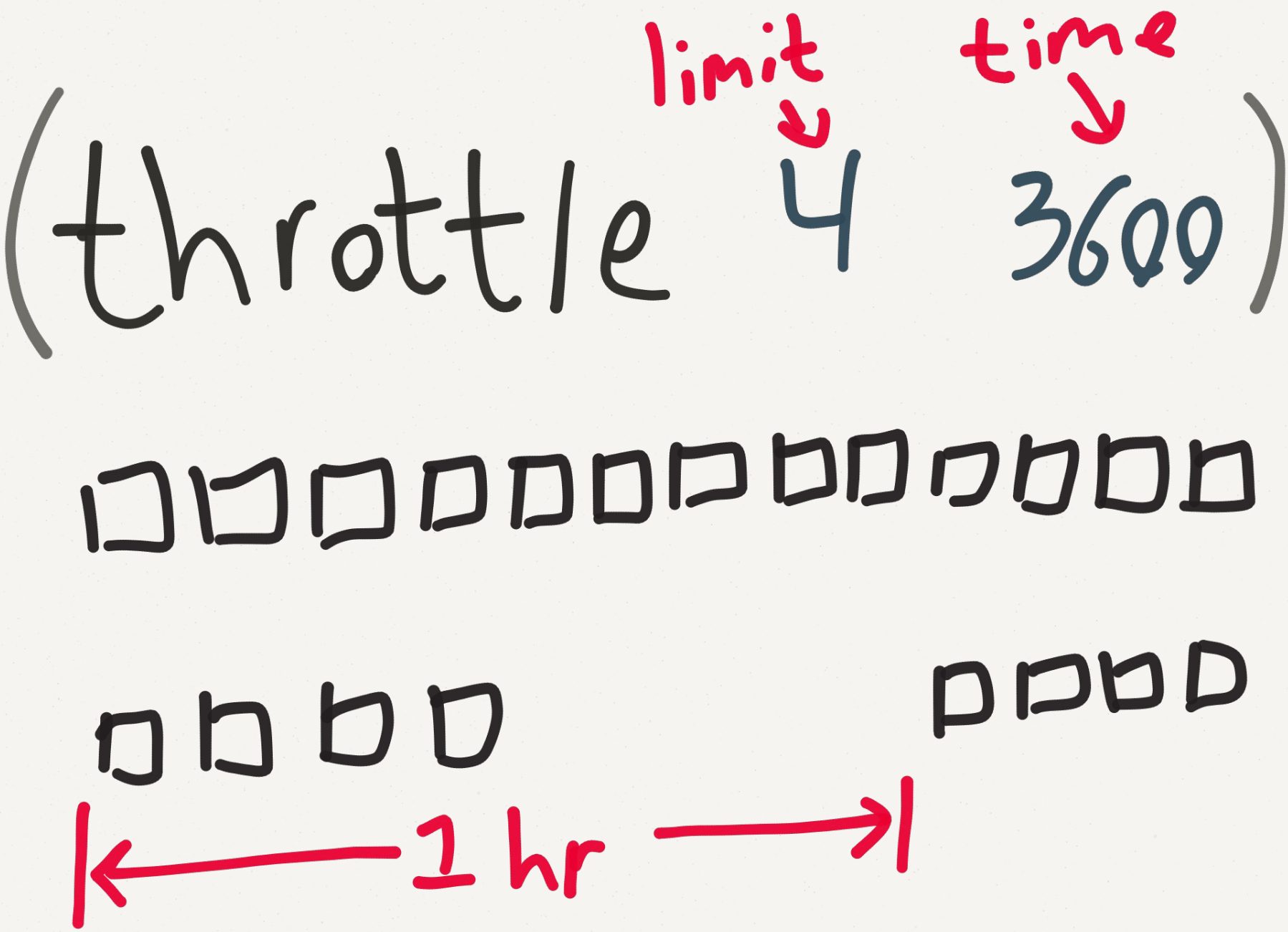
(ewma 10

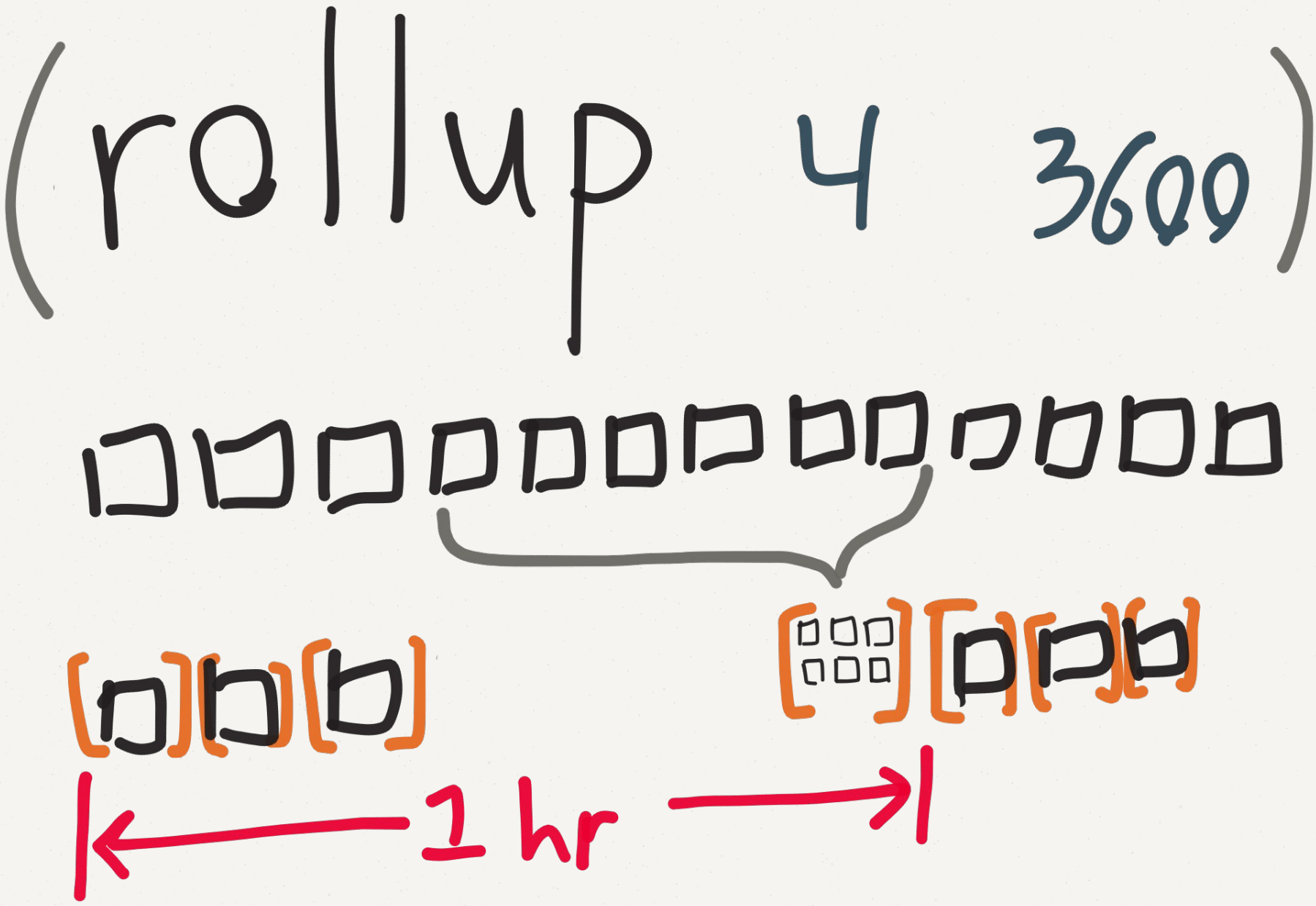
graph)

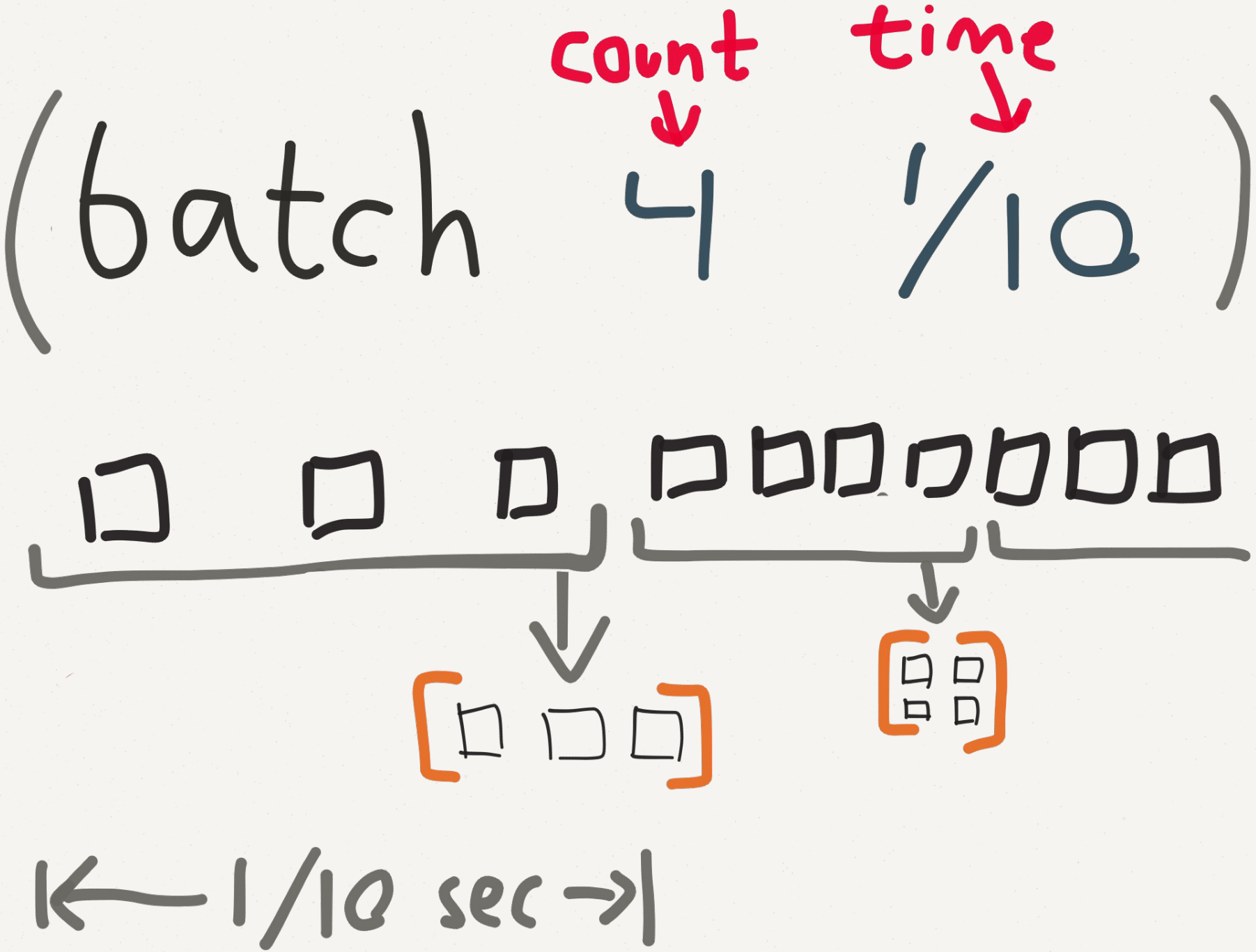


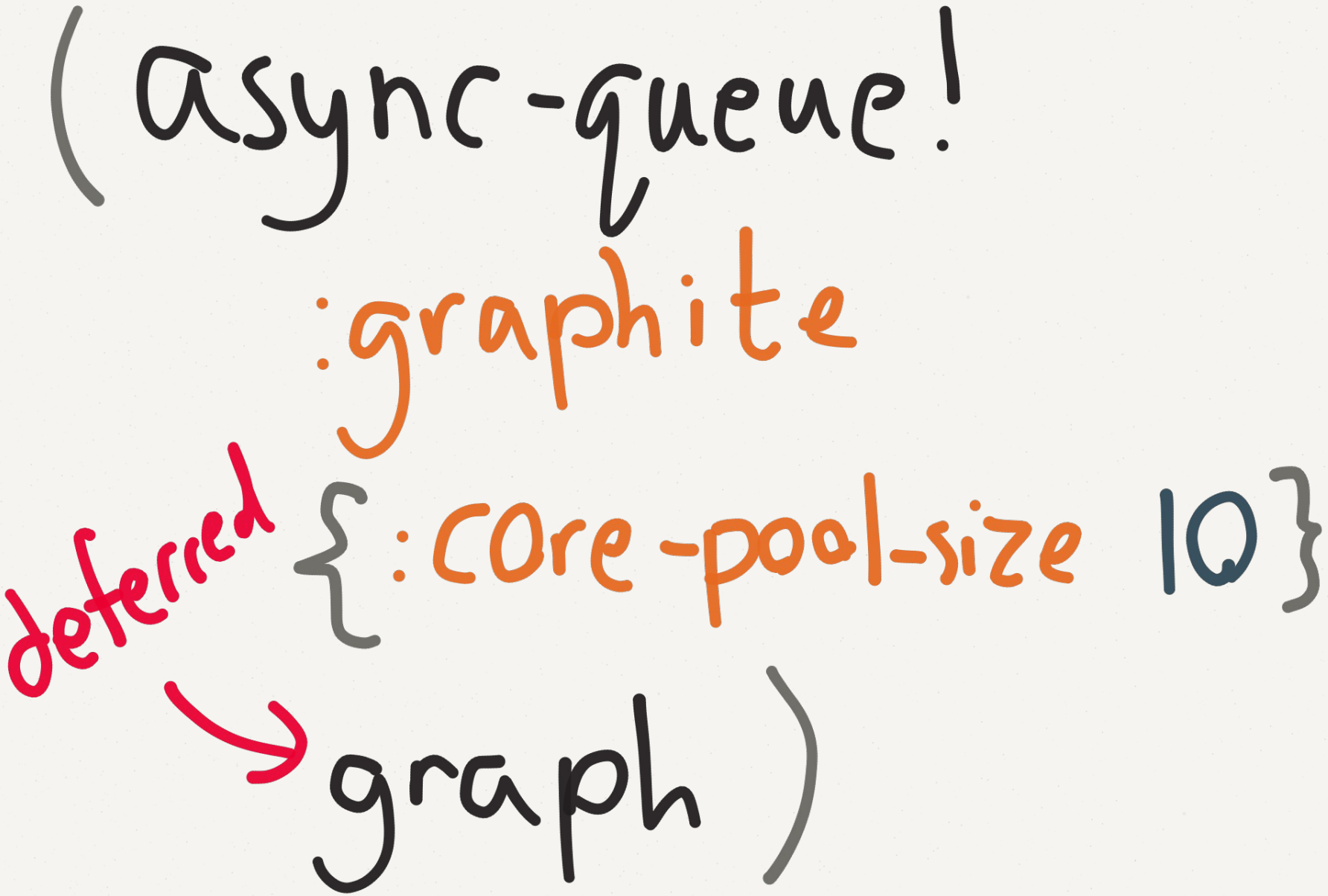

time window
fn
(stable 60 :state ...)
60s
ignored

Flow Control


(throttle
limit
4
time
3600
1 hr


(rollup 4 3600)
1 hr


count
time
batch 4 1/10
1/10 sec

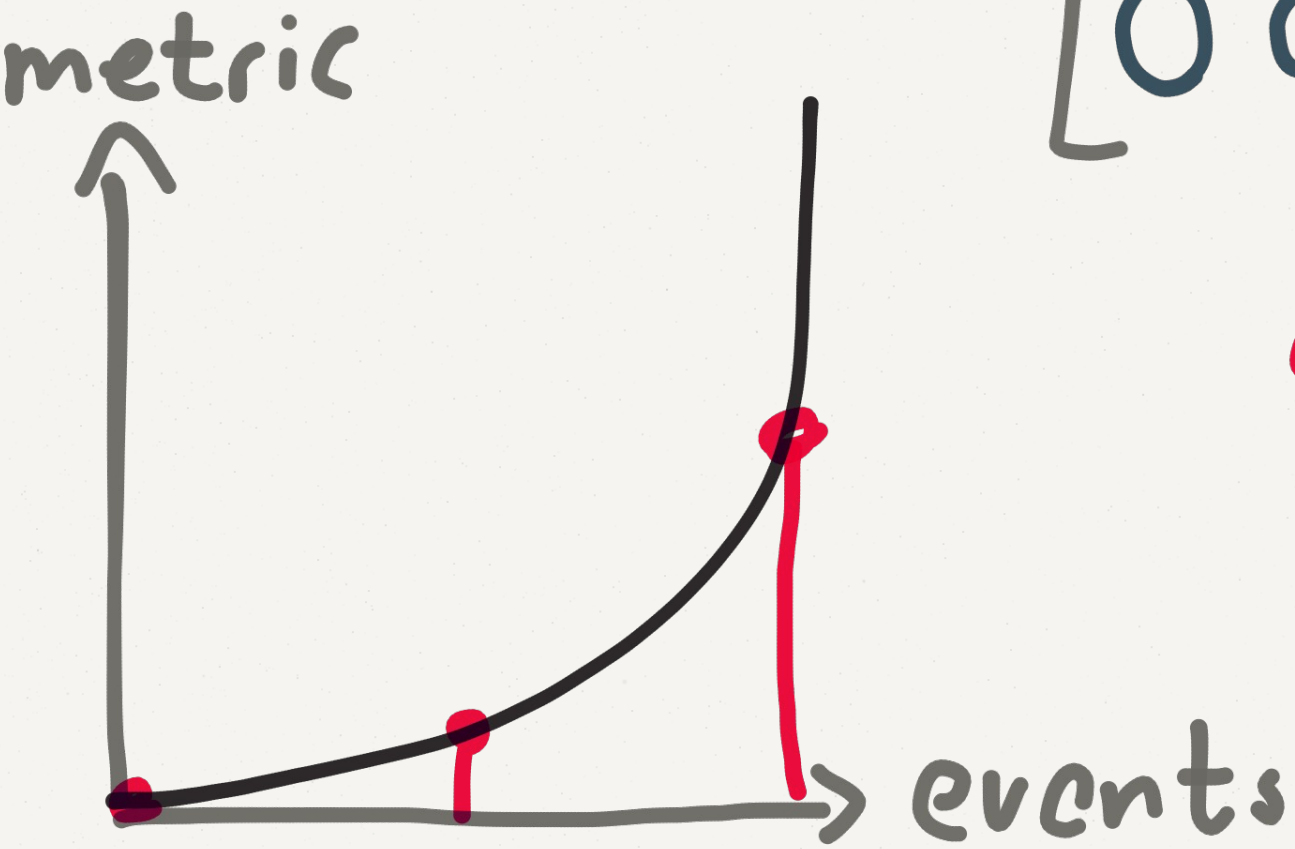

(async-queue!
:graphite
{:core-pool-size 10}
graph)
deferred

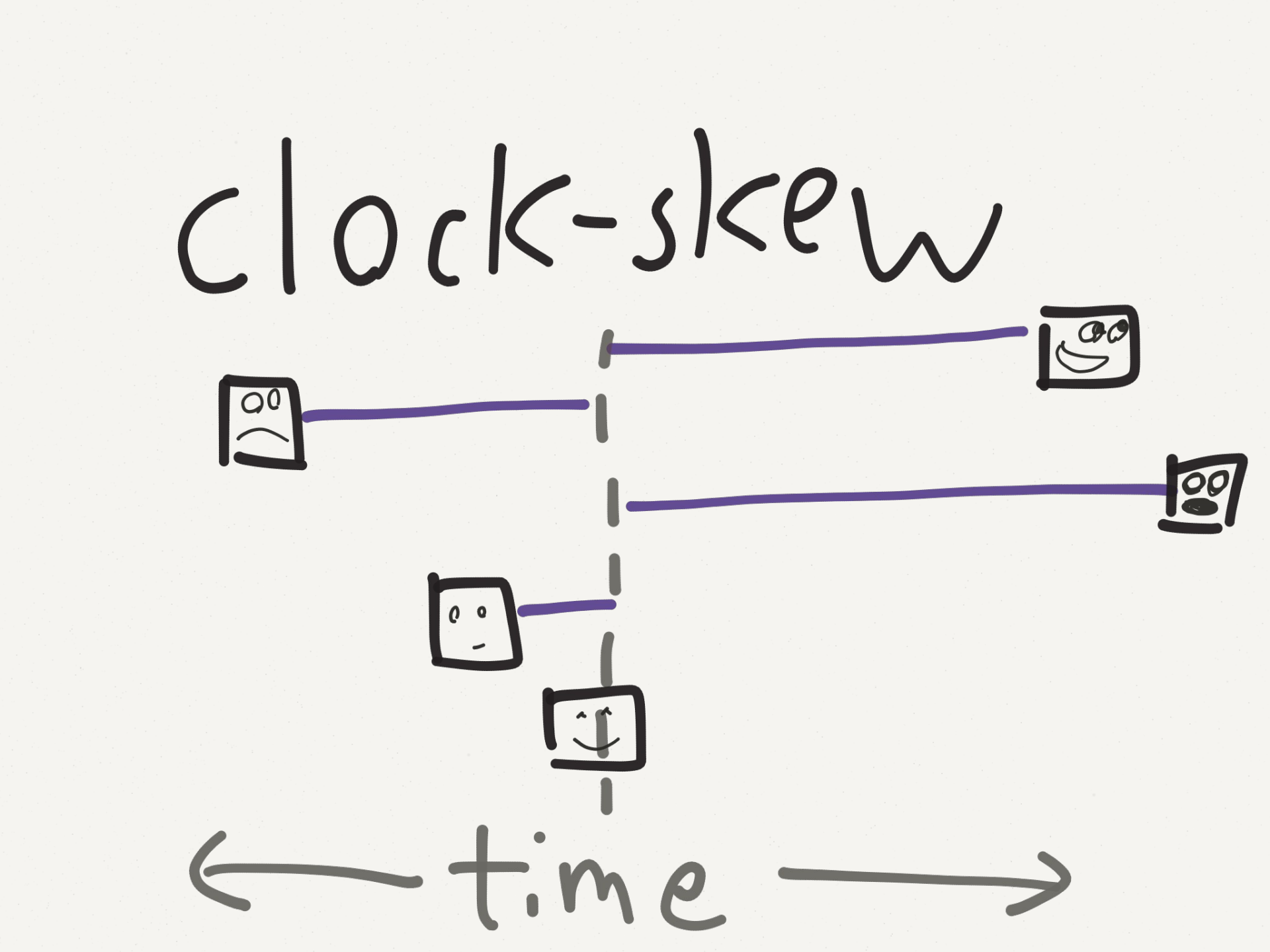
Analysis

(percentiles 10 [0 0.5 0.99])

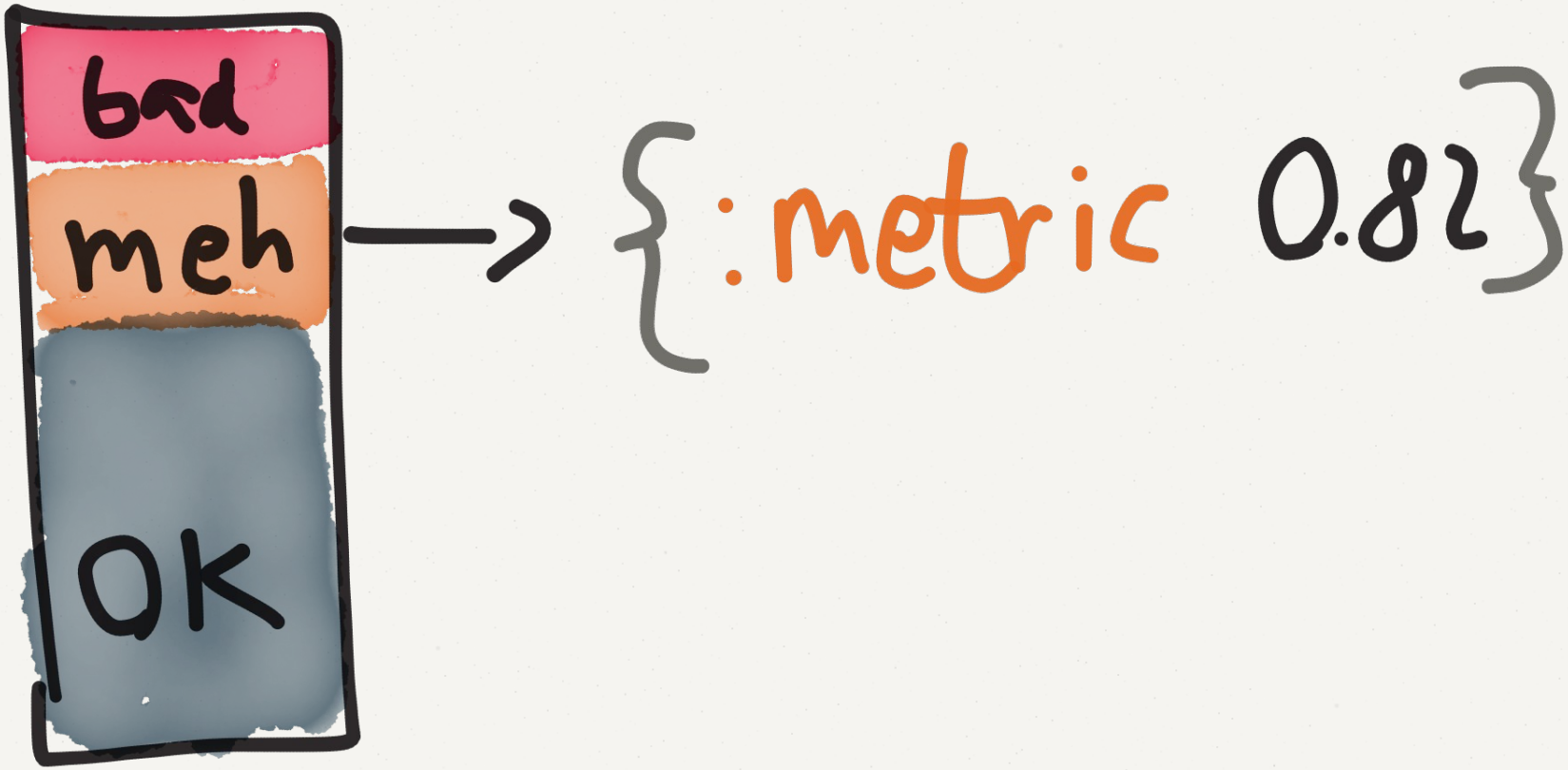
time

Quantiles



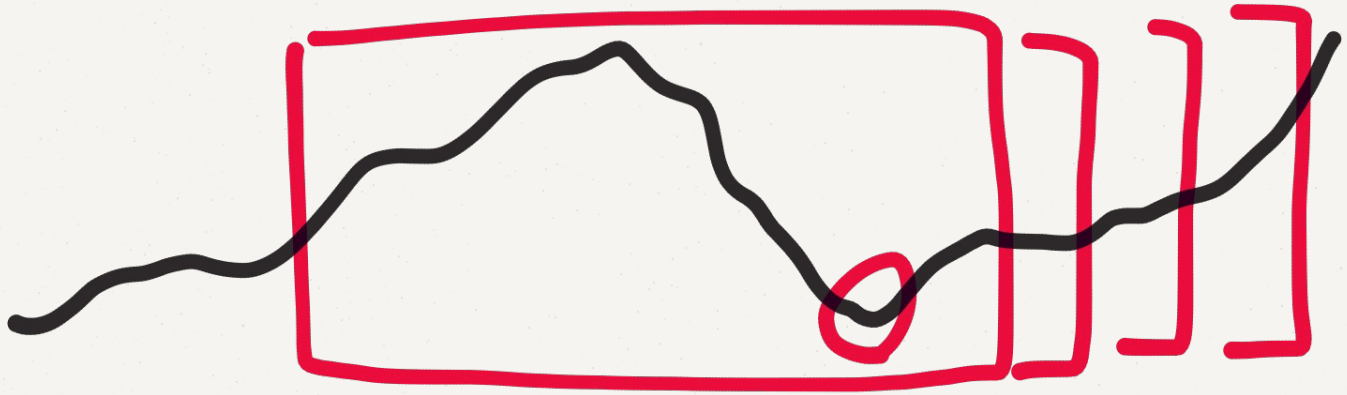

clock-skew
time

apdex

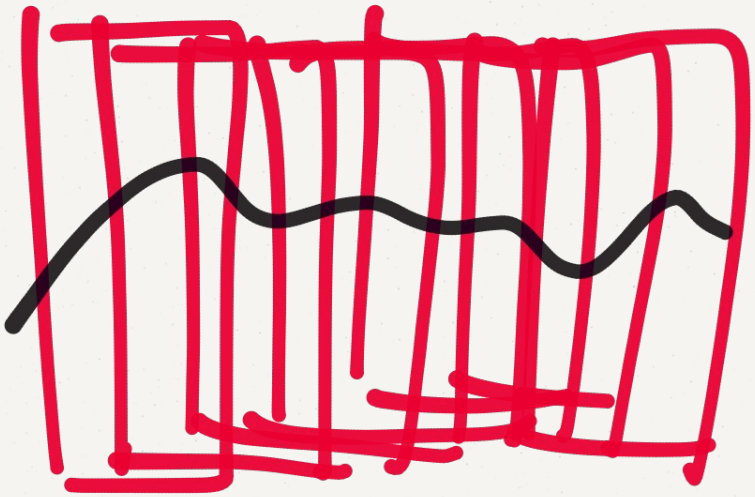


Combining across Time

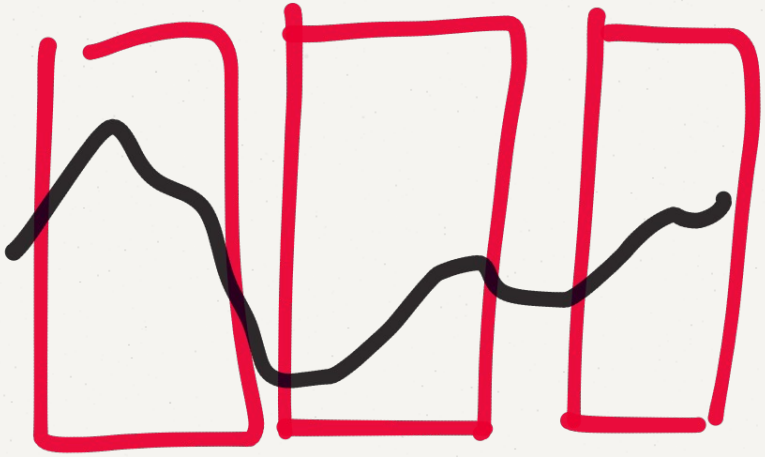
```
(moving-time-window
  100 ← seconds
  (smap folds/min ...))
```



moving

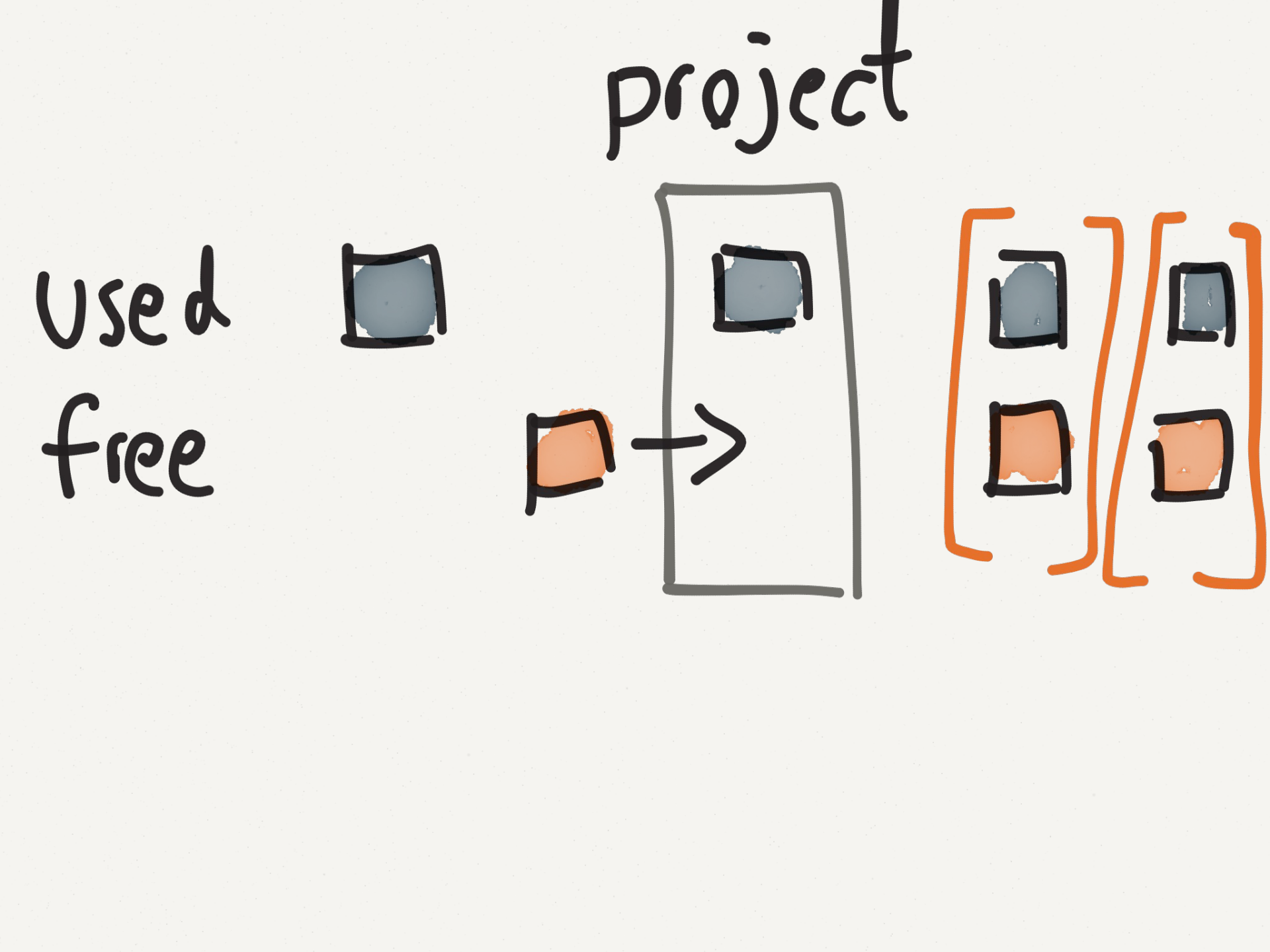


fixed

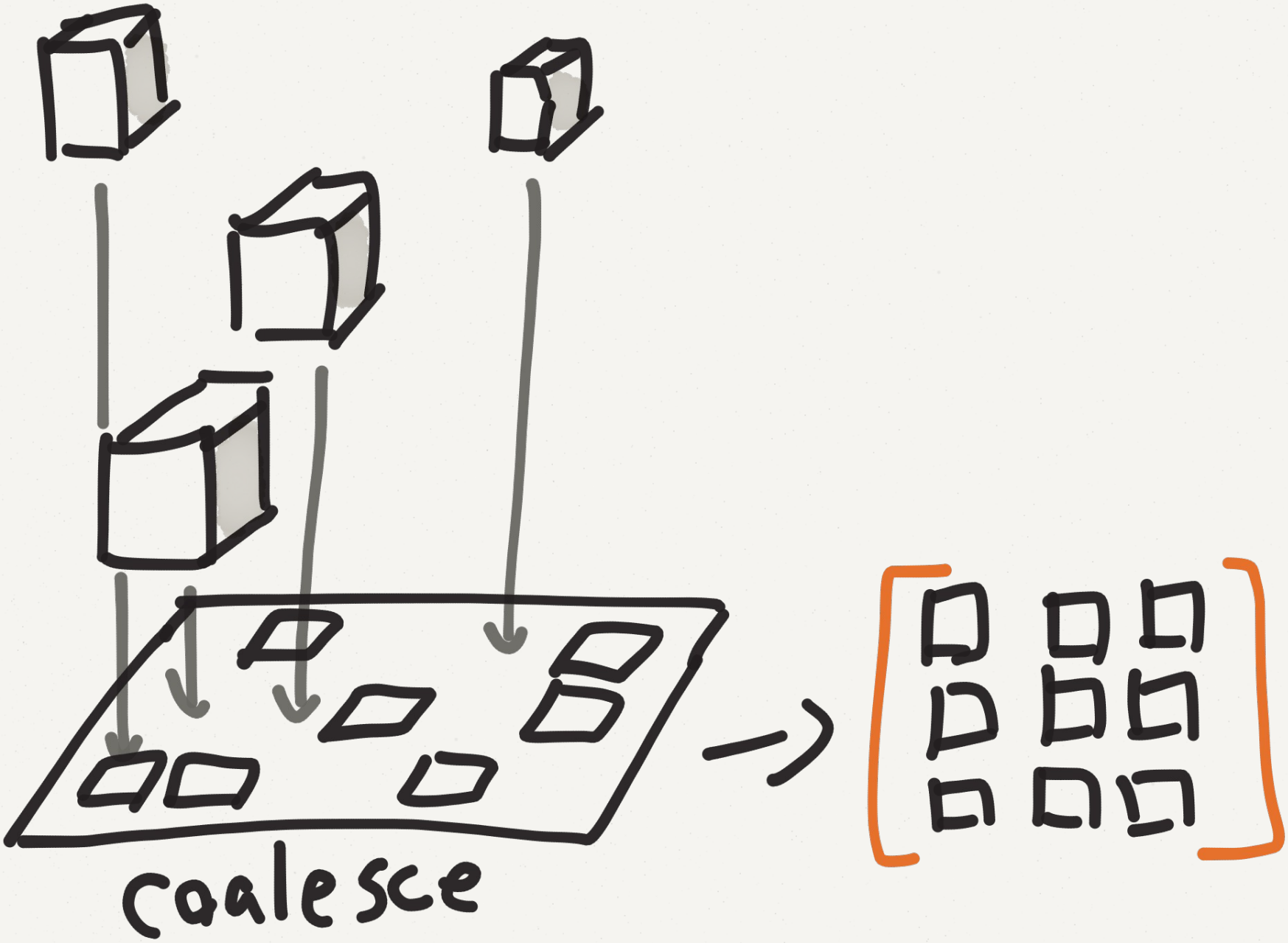


Cambining acrass space

```
(project [(service "free")
          (service "used")]
  (smap folds/sum
    (with :service "total"
      graph)))
```

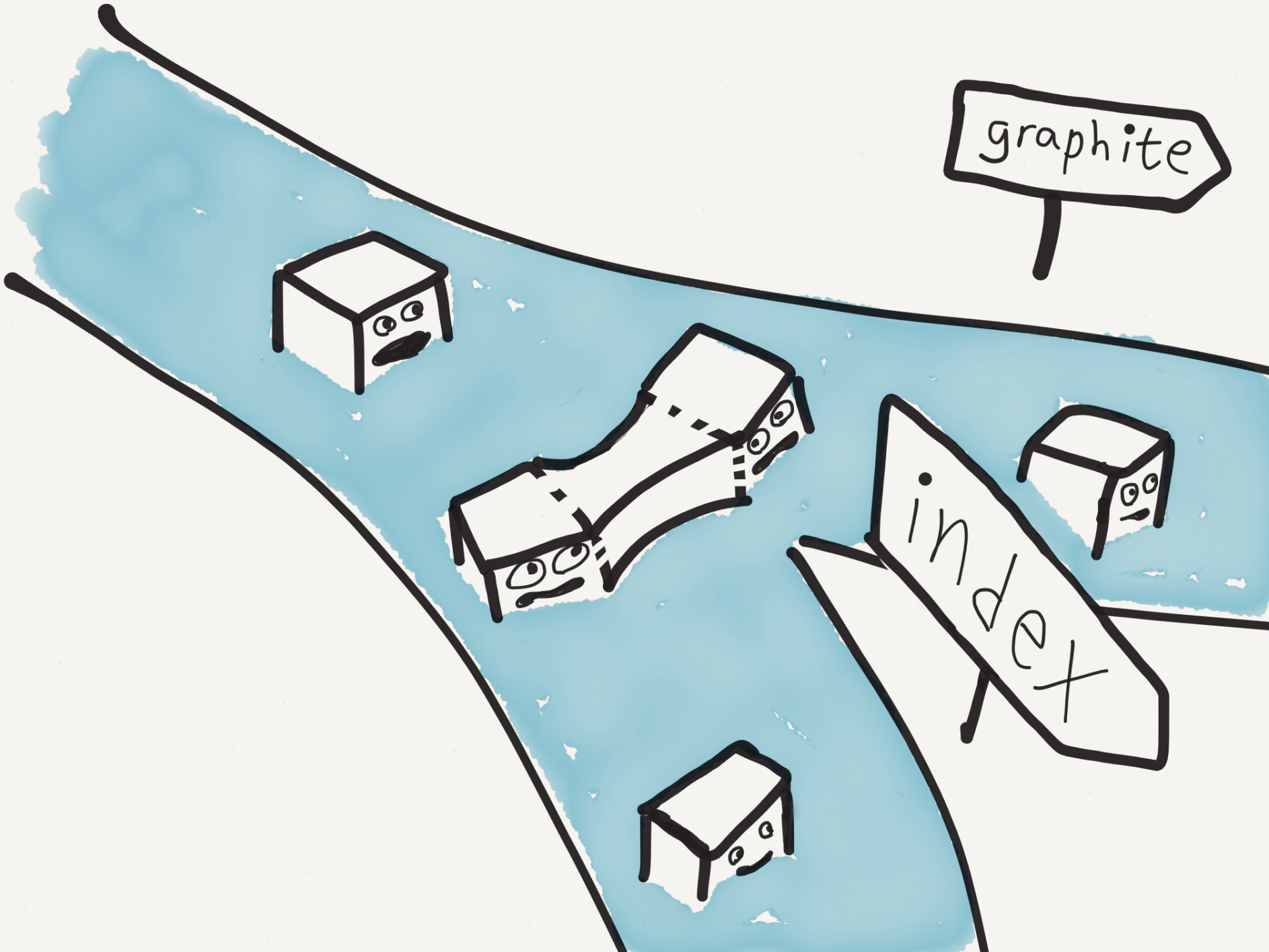

project
used
free

```
(coalesce
  (smap folds/mean
    ...))
```


coalesce

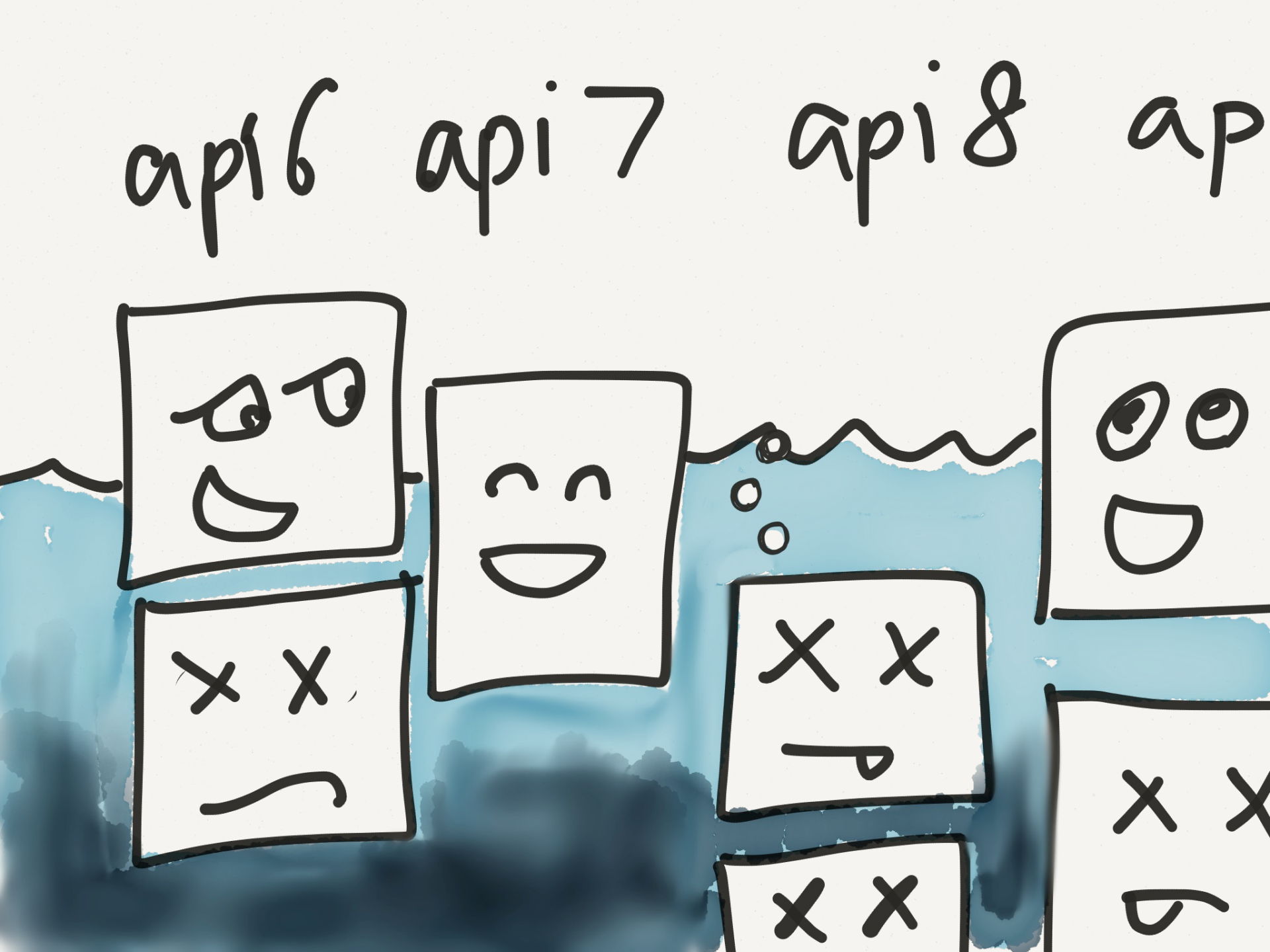
Composition

```
(rate 5
  graphite
  index)
```

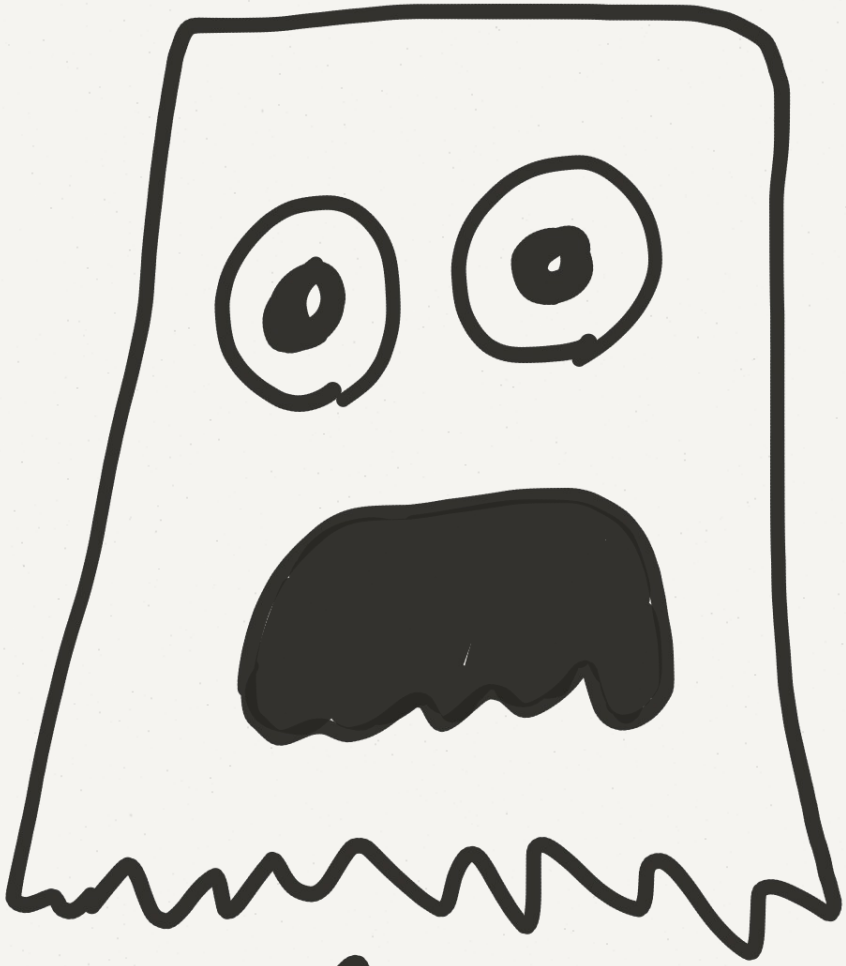

graphite
index

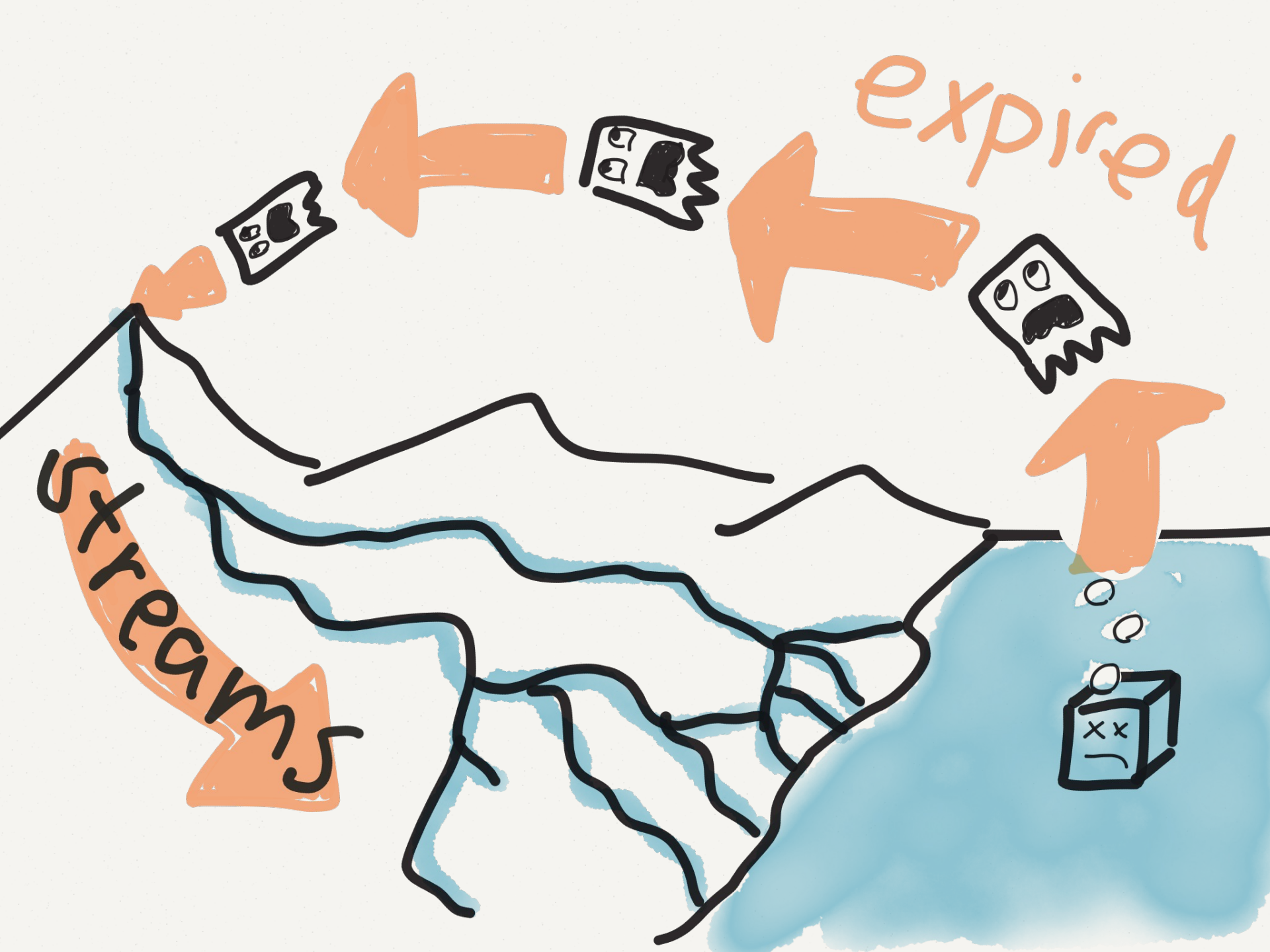
The Index?


streams


api6
api7
api8
ap

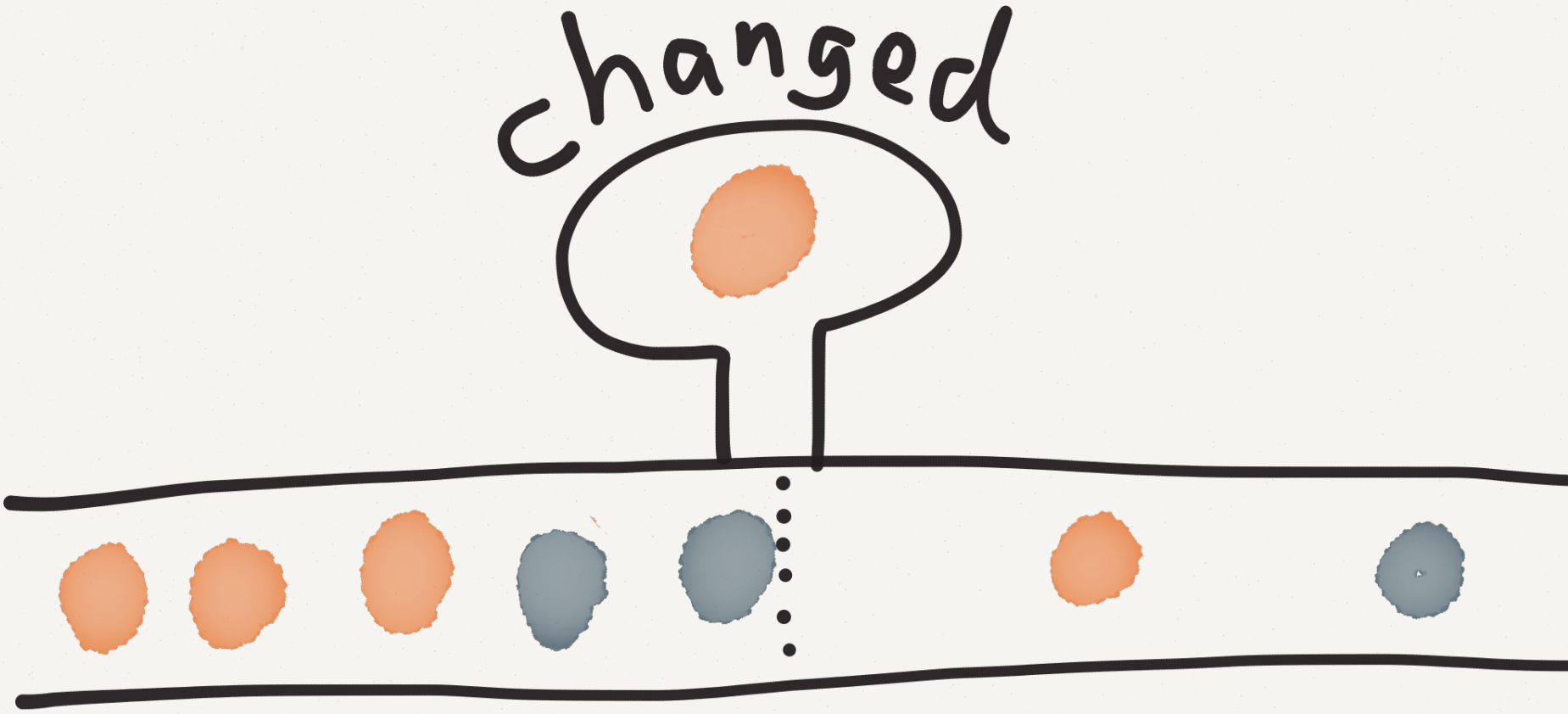
"I am the ghost of nginx on api8!"

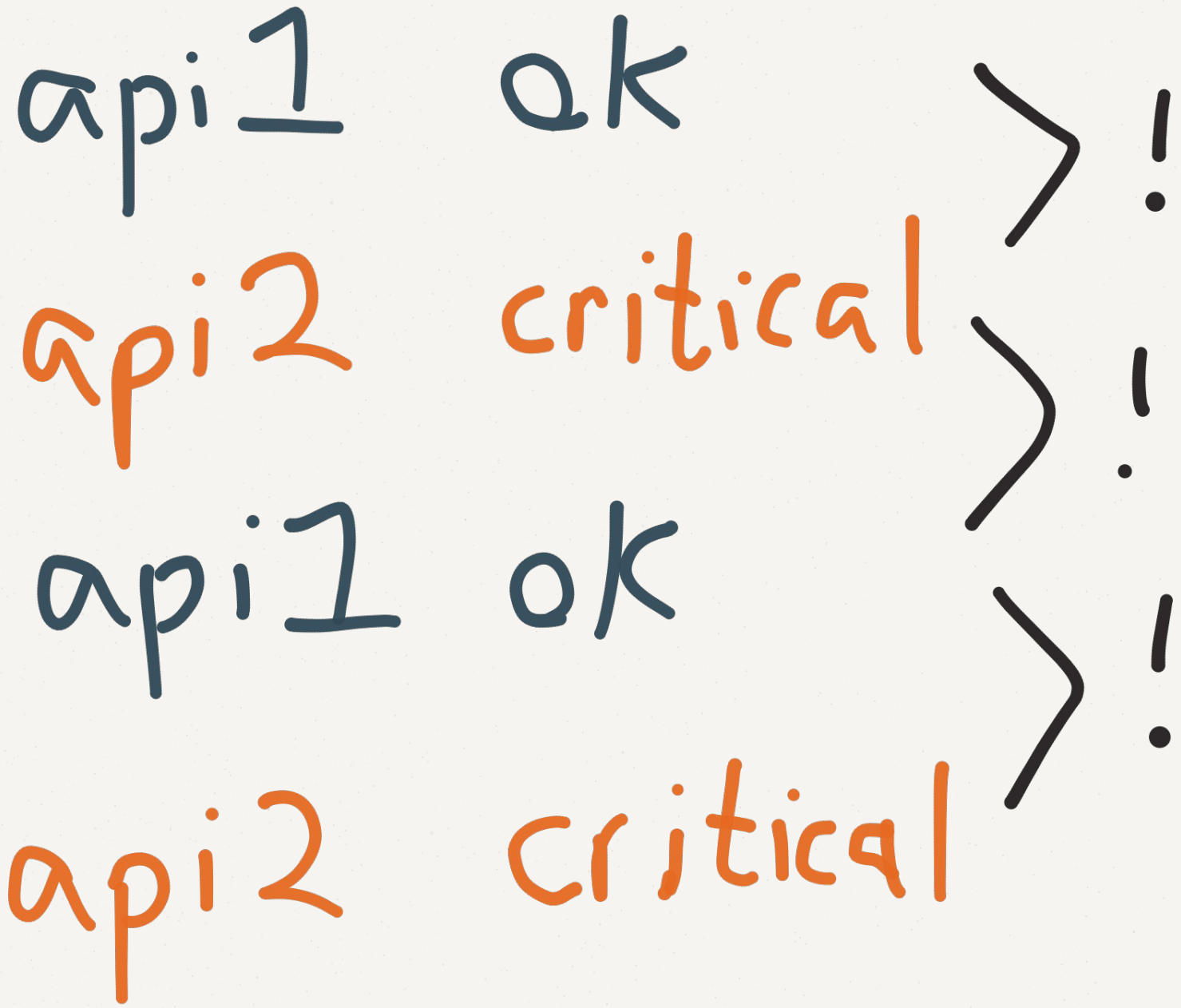



expired
streams

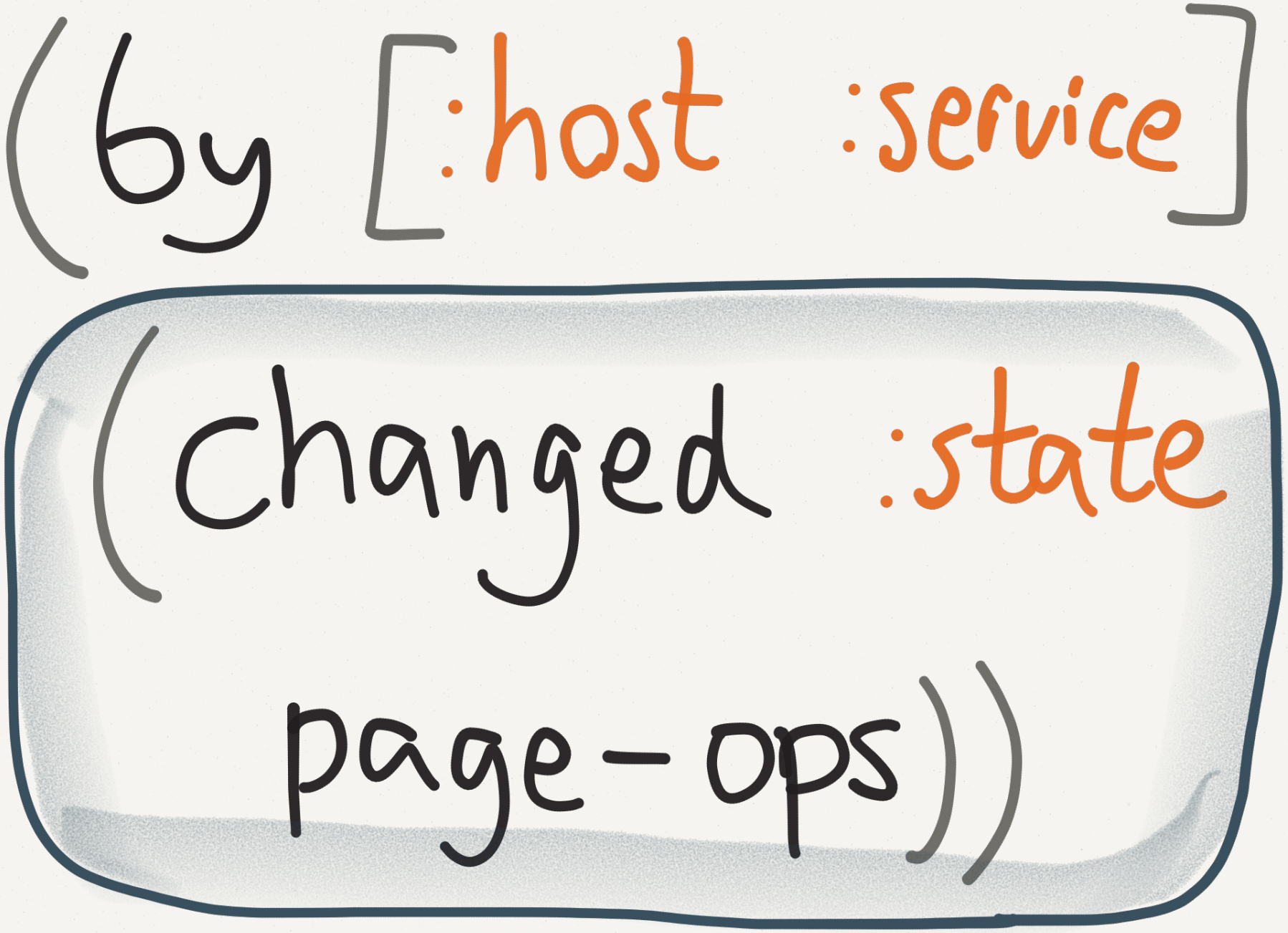
```
(expired
 #(warn "Lost" %))
```

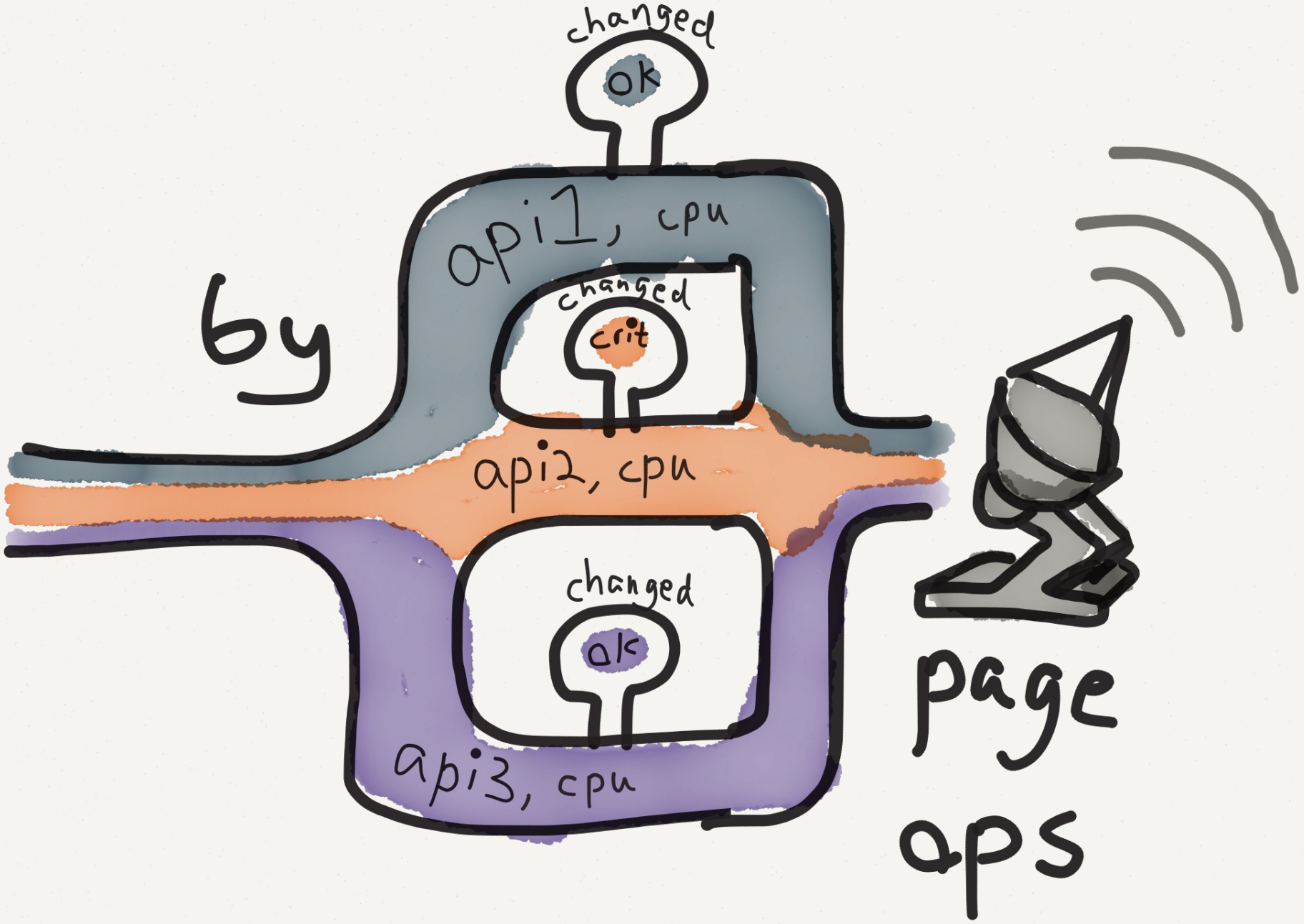
```
(changed :state
  page-ops)
```


changed


api1 ok
api2 critical
api1 ok
api2 critical

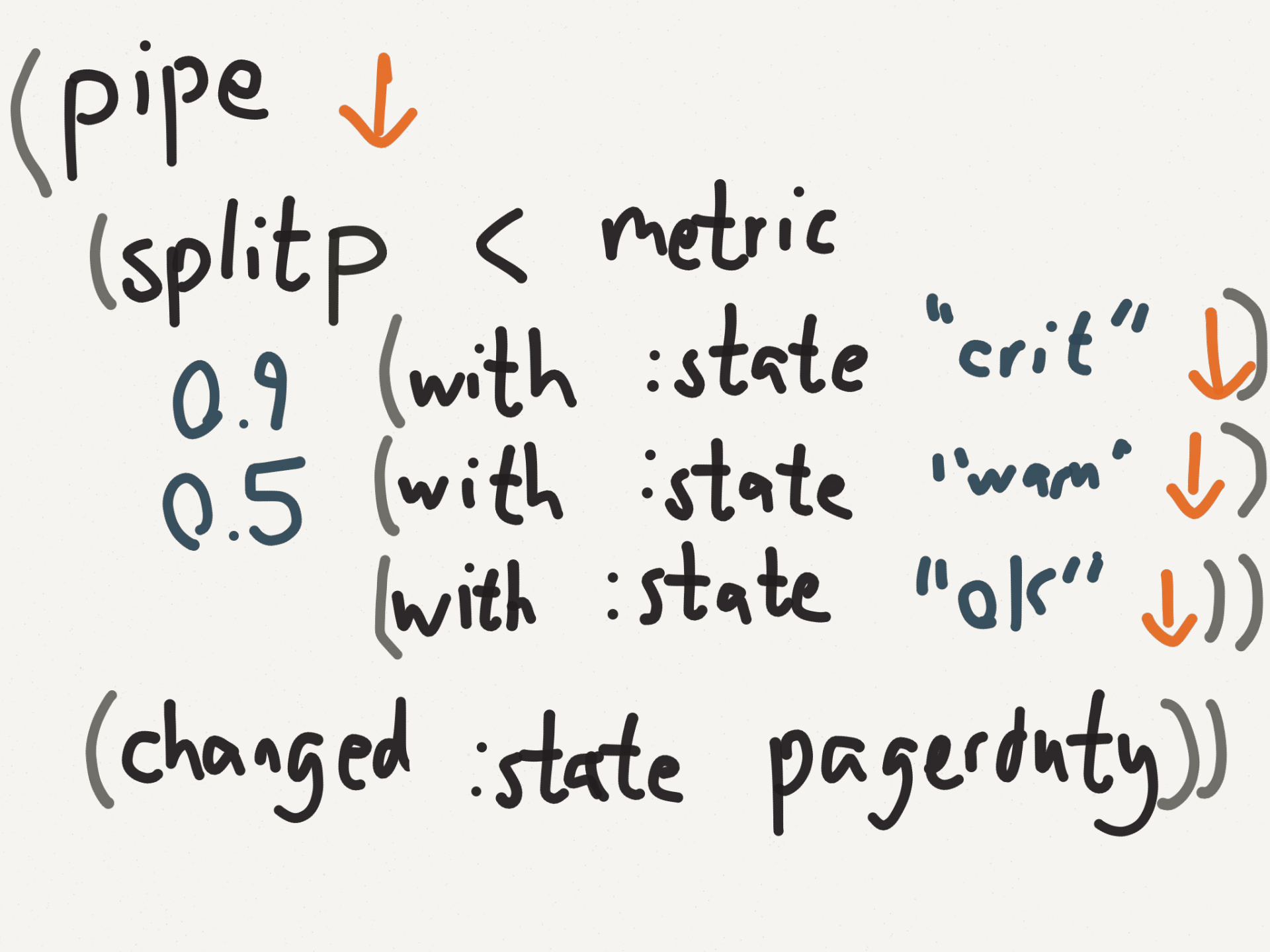
```
(by [:host :service]
  (changed :state
    page-ops))
```

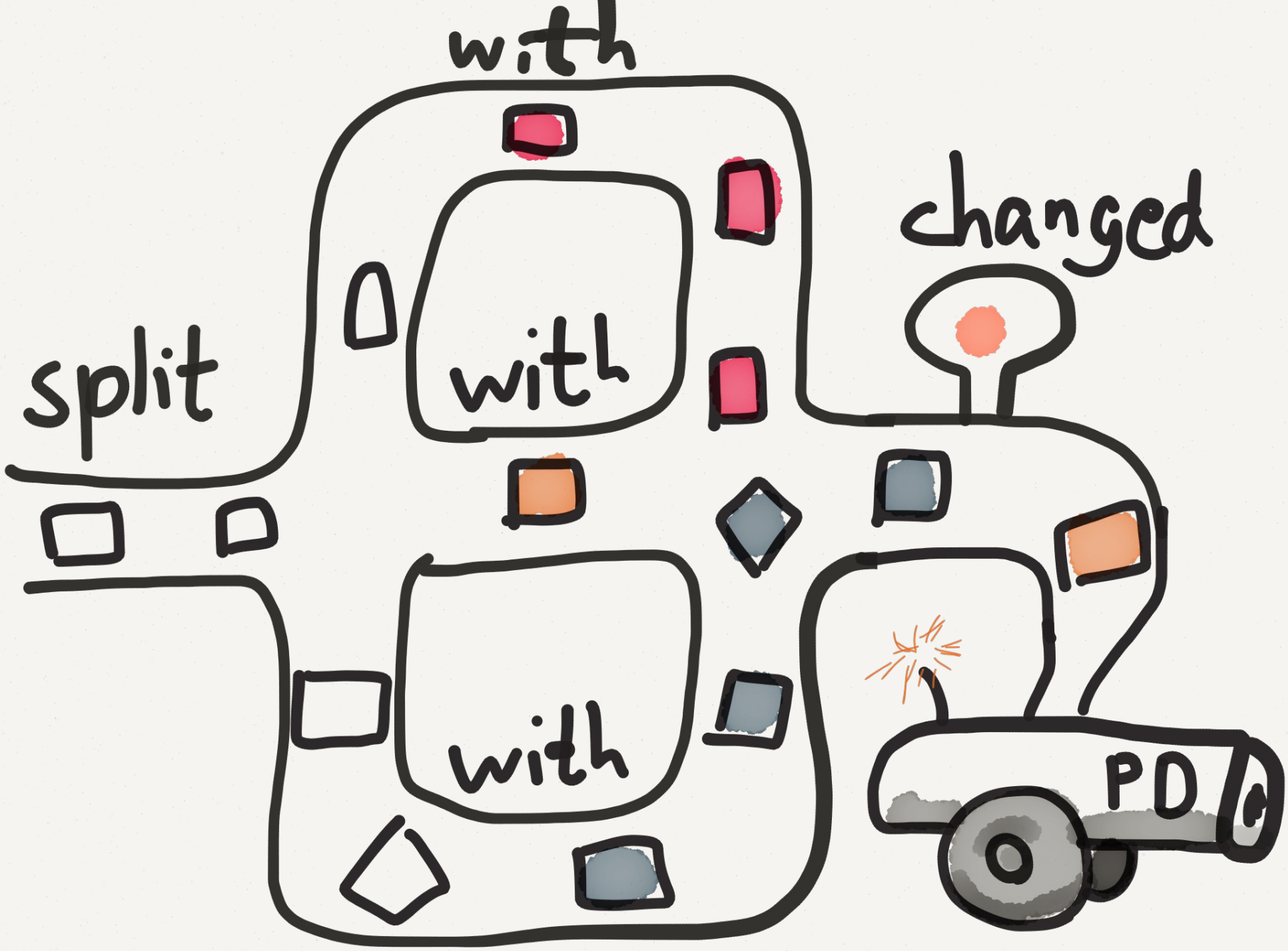

(by [:host :service]
(changed :state
page-ops))


changed
ok
api1, cpu
changed
crit
by
api2, cpu
changed
ok
api3, cpu
page
ops

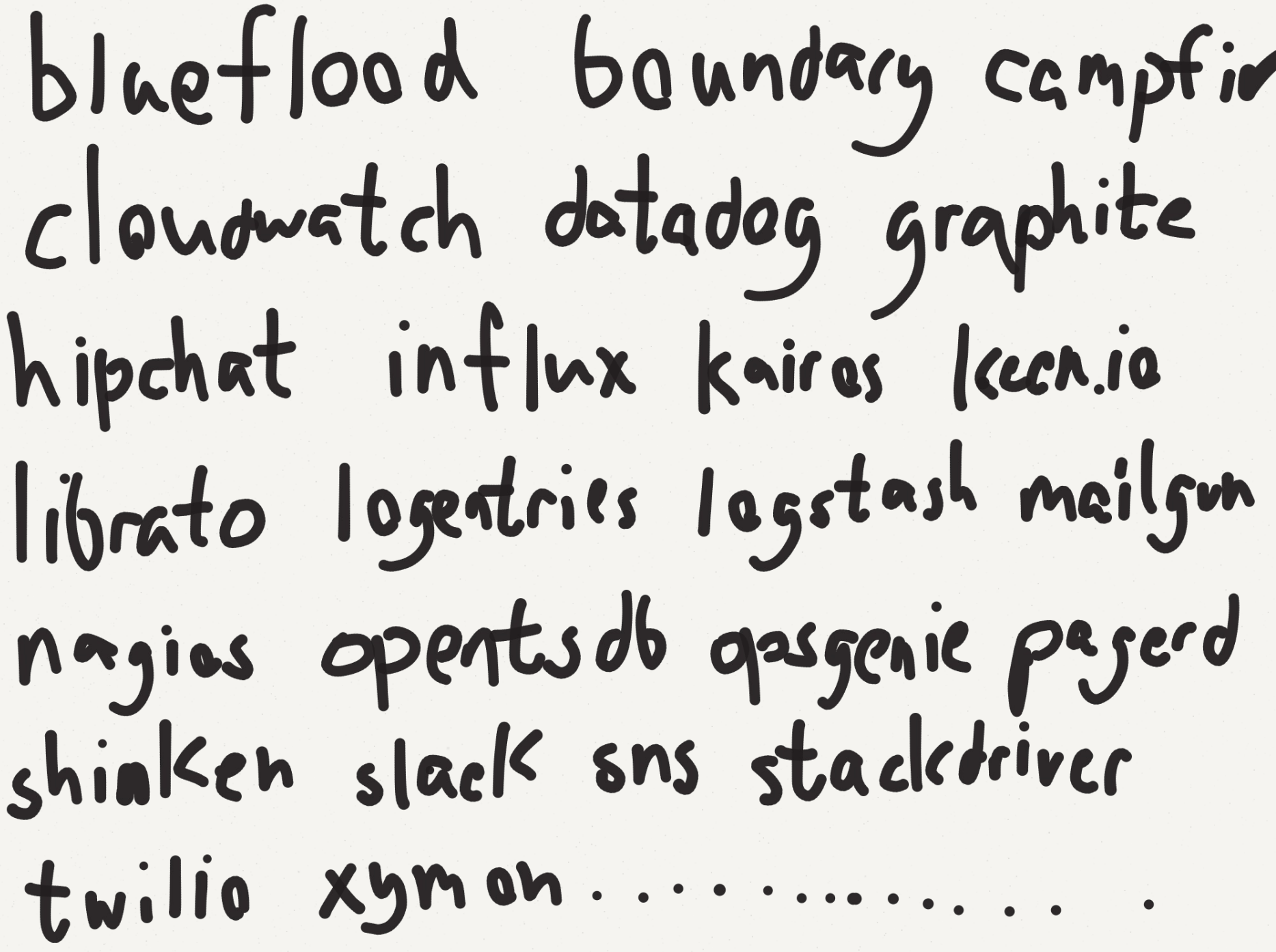
How do you recombine streams in a tree?

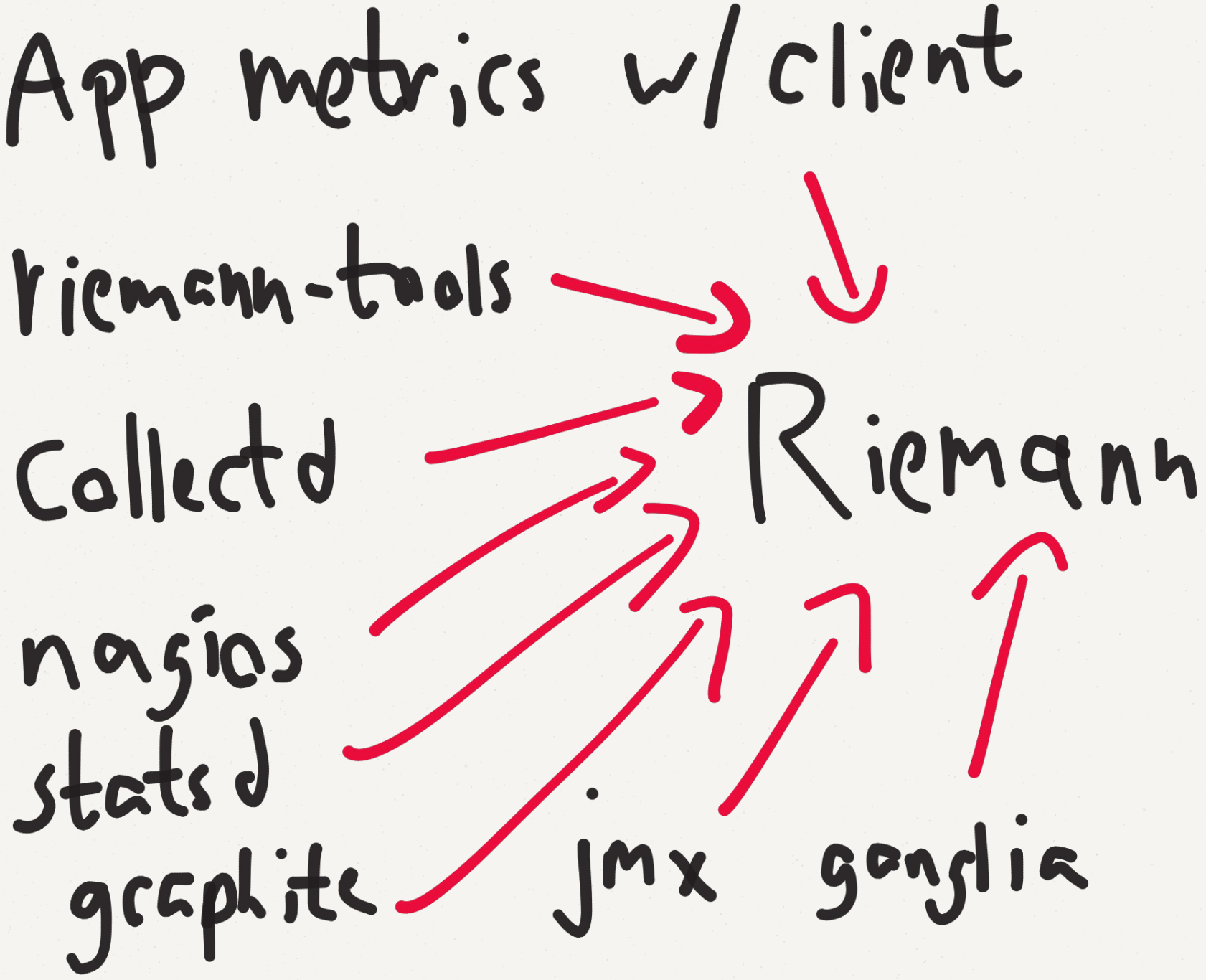
```
(let [index   (index)
      graph (graphite...)]
  (streams
    (where (...) index)
    (where (...) index
           graph)))
```

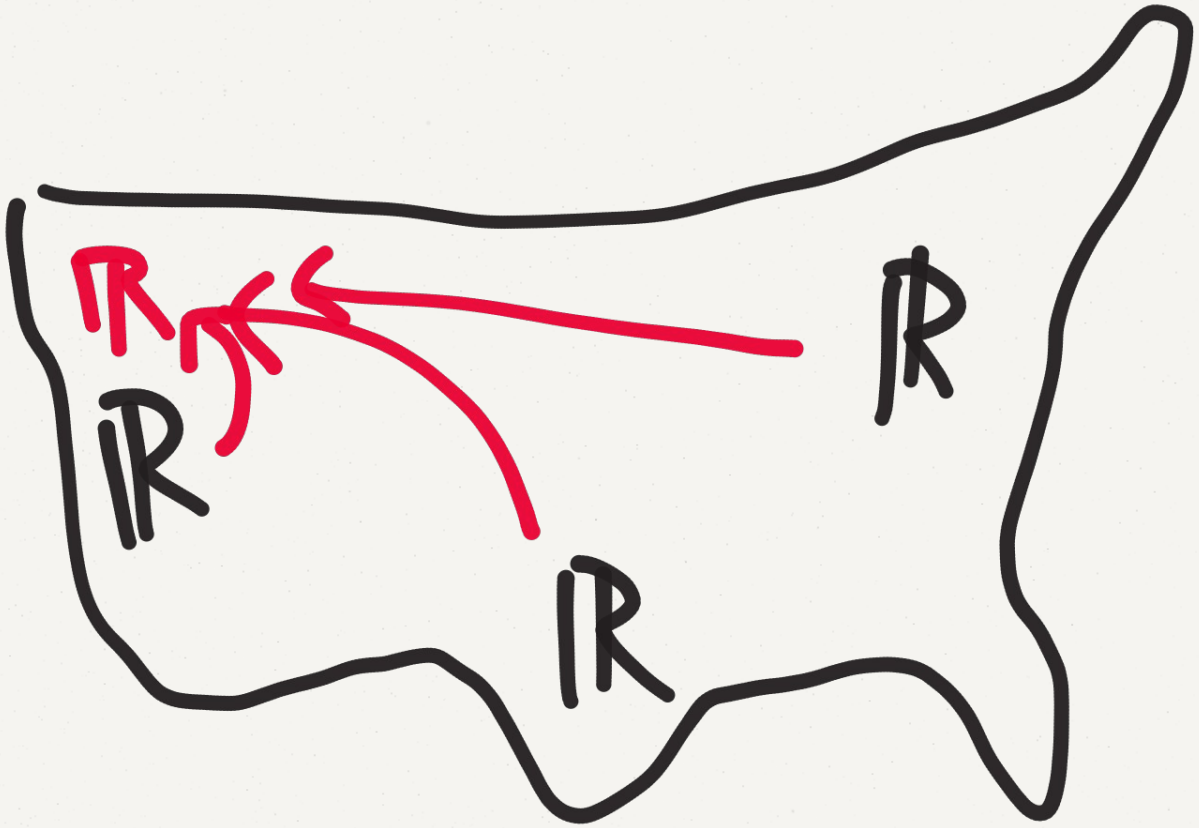

(pipe
(splitp < metric
0.9 (with :state "crit"))
0.5 (with :state "warn")
(with :state "ok")))
(changed :state pagerduty))


with
changed
split
with
with
PD

Playing well with OTHERS


blueflood boundary campfi
cloudwatch datadog graphite
hipchat influx kairos keen.io
librato logentries logstash mailgun
nagios opentsdb opsgenie pagerd
shinken slack sns stackdriver
twilio xymon


App metrics w/ client
riemann-tools
Collectd
nagios
statsd
graphite
Riemann
jmx
ganglia



(forward (tcp-client ...))

Organizing Configs

```
(defn tell-ops [index]
  (sdo
    index
    (throttle 5 3600
      (email "ops@foo"))))
```

```
(ns meowcorp.queues
  (:require
    [meowcorp.misc :as m]
    [riemann.config :refer :all]

    ...)
```

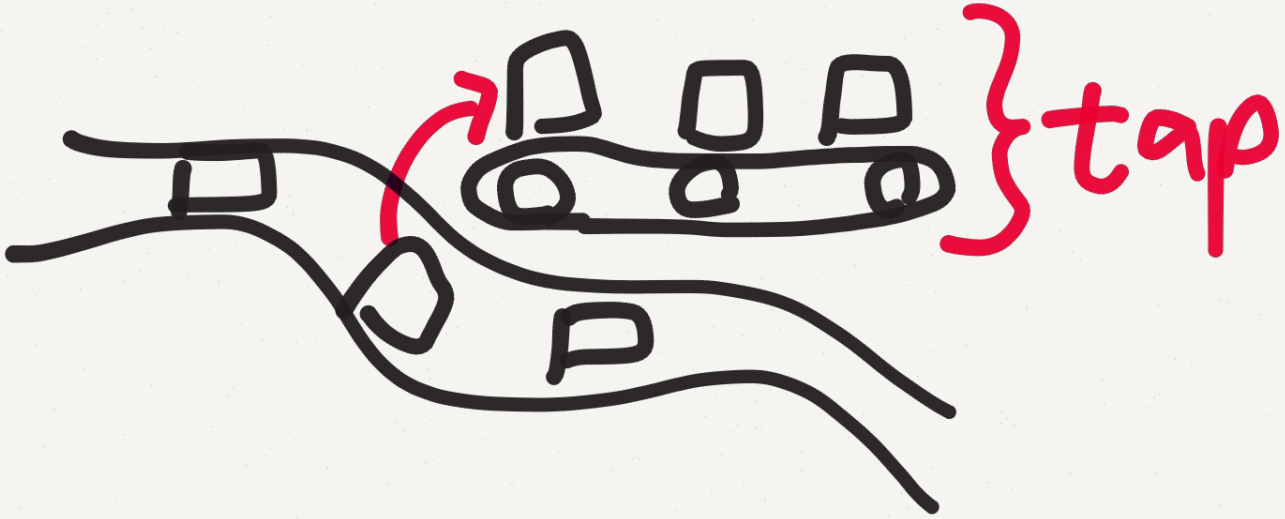
Testing

(io graph)

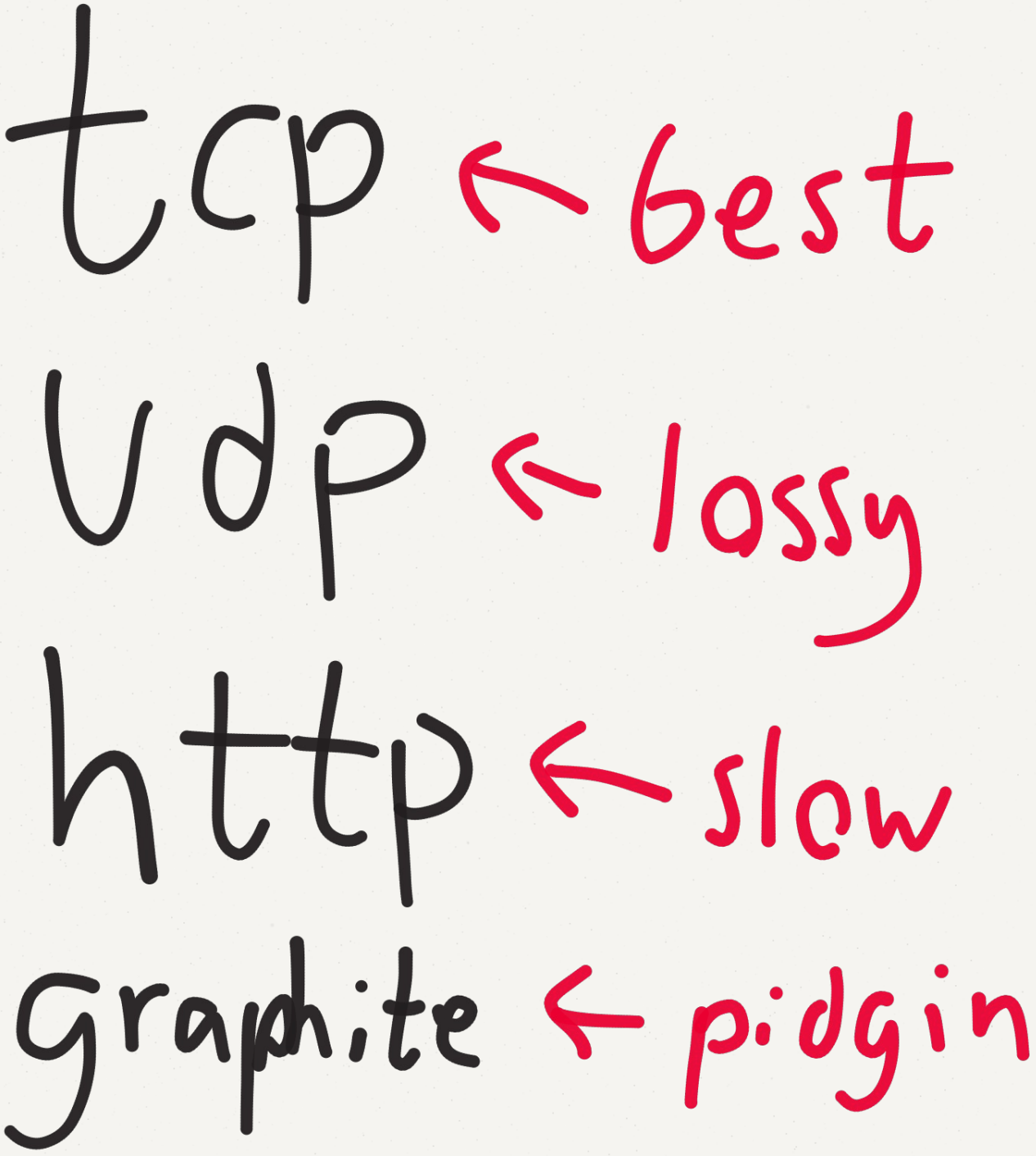


wont run in a test

```
(tap  :graphite
  (io  graph))
```



Recommendation


tcp ← best
udp ← lossy
http ← slow
graphite ← pidgin

Use tests!

Use hot config reloading

Clock skew will ruin you

Riemann is

Transient

(no persistence!)

Riemann is not distributed

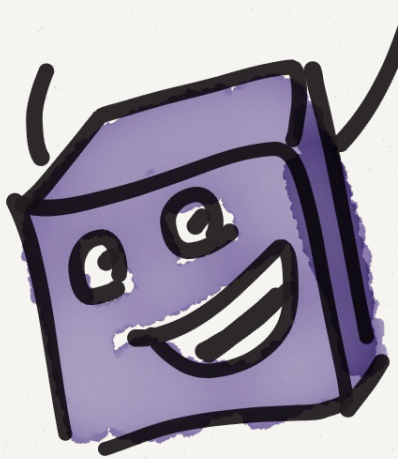
(use linux-ha for naïve failover)

Riemann may be faster than you're used to

~10 MHz tops?

...or way slower

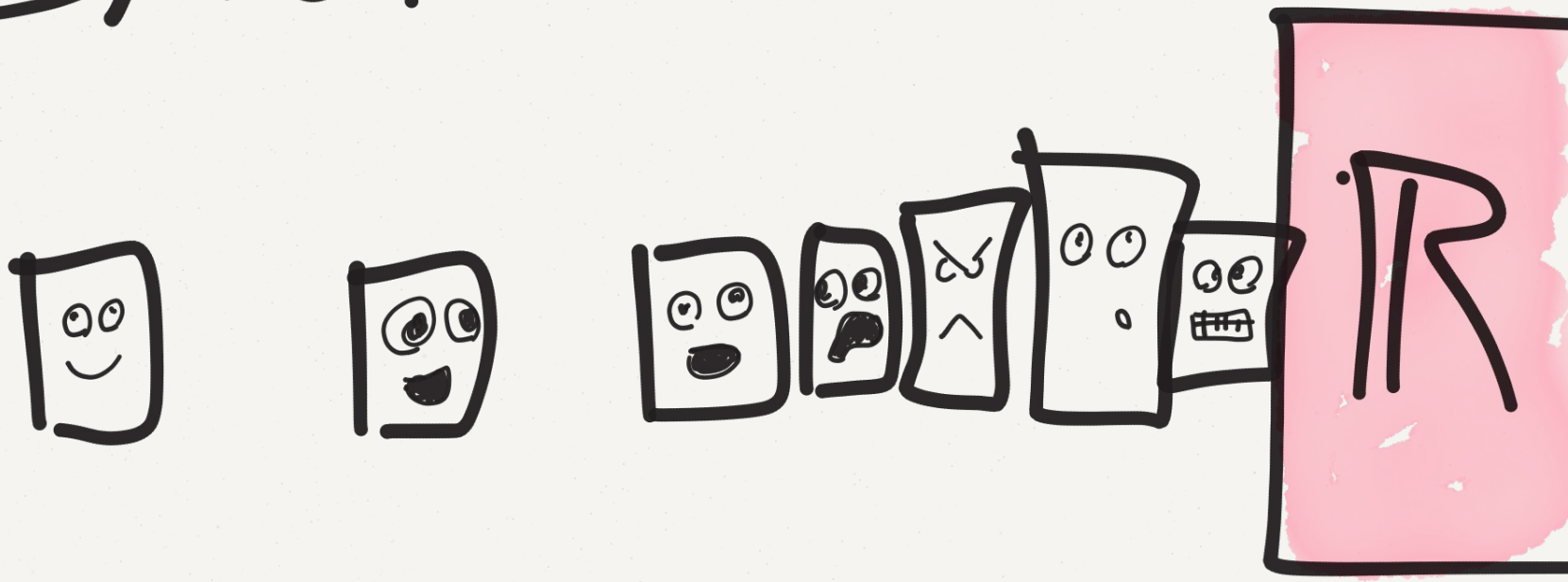
Arbitrary Computation! Wooo!



Use TLS 6idr certificates if you want crypto + auth.

Riemann provides

BACKPRESSURE



Your clients should

- Use bounded queues
- Load-shed if needed
- Pipeline & batch


Review

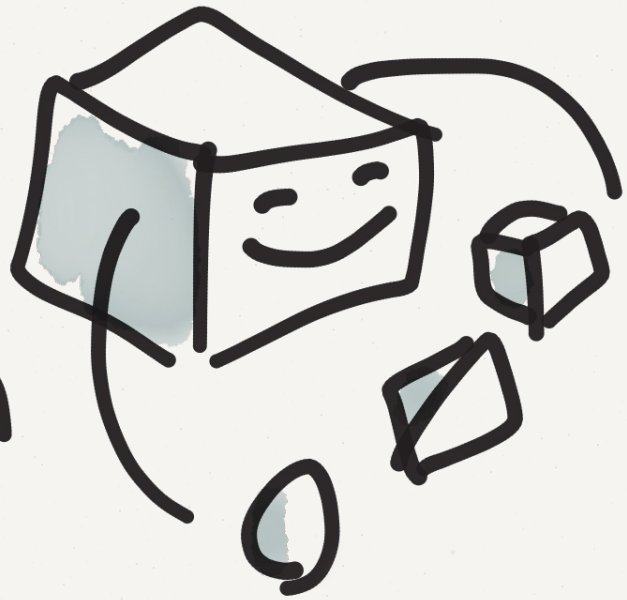
A fast, transient

Clearinghouse for

distsys visibility

- App latencies
- Signups
- Node health
- DB stats

Events are a unifying representation



Public transit for instrumentation & analytics

Clients $ app
code only need
to speak Riemann

Frees you to route to $ combine other monitoring systems

Powerful stream language for complex transforms!

Horrible UI

(I'm so sorry)

Built for

WEIRD

PEOPLE



But you might like it too!

https://riemann.io

