logs as

data

*

as data

code as data

HTTP requests as data

HTML markup as data

exceptions as data

SQL queries as data

CLI options as data

decision trees as data

build specs as data

cloud deploys as data

*

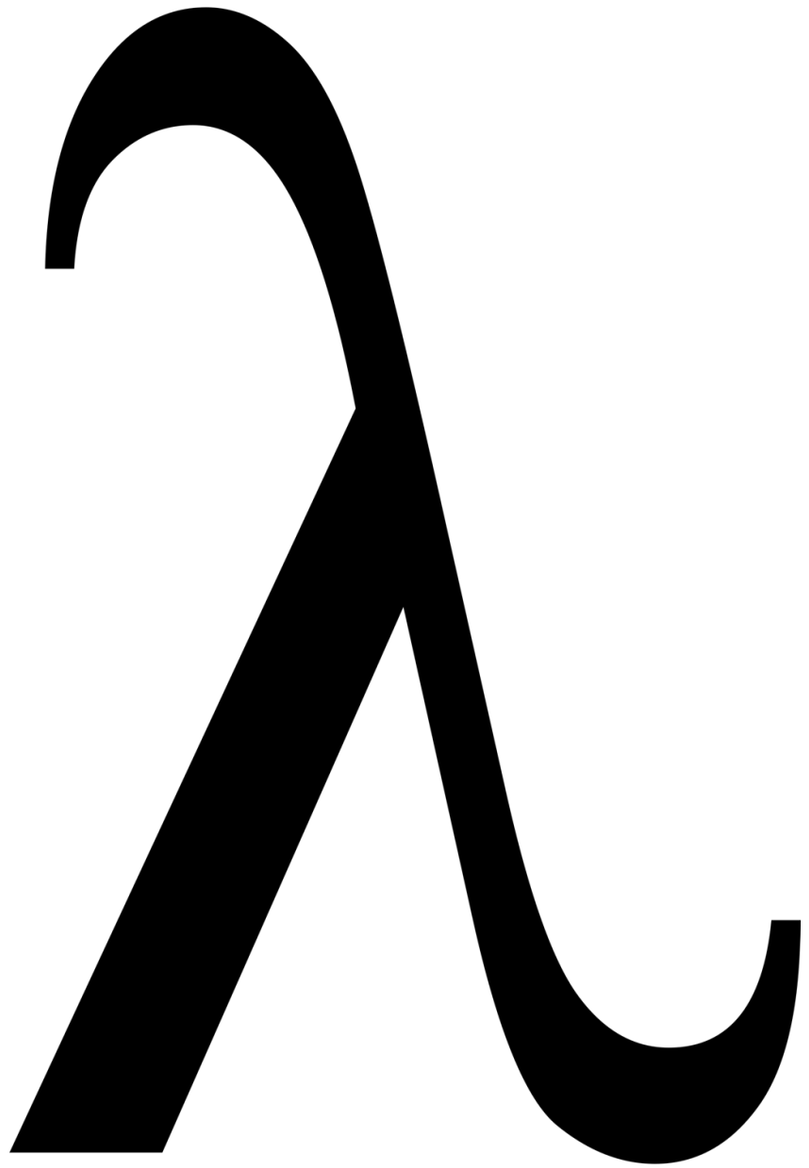
as data

logs as data

logs?

possible power

current power



data

via simple abstractions

with a general library

data

```
{:name  "Conj"
 :state "NC"
 :days  3}
```

```
public class Conference {
    private String name;
    private String state;
    private Integer days;
    ...
}
```

via simple abstractions

map

seq

fn

with a general library

```
(->> foods
     (filter :hot?)
     (sort-by :cost)
     (take 3))
```

```
(useful/find-first
 tasty?
 foods)
```

data

via simple abstractions

with a general library

logs

data?

"info: user 24 viewed /home"

via simple abstractions?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration>
<log4j:configuration>
  <appender name="stdout"
    class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern"
        value="%d{ABSOLUTE} %5p %c{1}:%L - %m%n" />
    </layout>
  </appender>
  ...
  <root>
    <level value="debug" />
    <appender-ref ref="stdout" />
  </root>
</log4j:configuration>
```

with a general library?

```
org.apache.log4j.*
```

org.apache.log4j.Heirarchy

org.apache.log4j.LogRecordFilter

unix

grep / awk / sort / head

data

via simple abstractions

with a general library

logs?

"info: user 24 viewed /home"

/var/log/app/access.log

events

stream of everything

that is happening

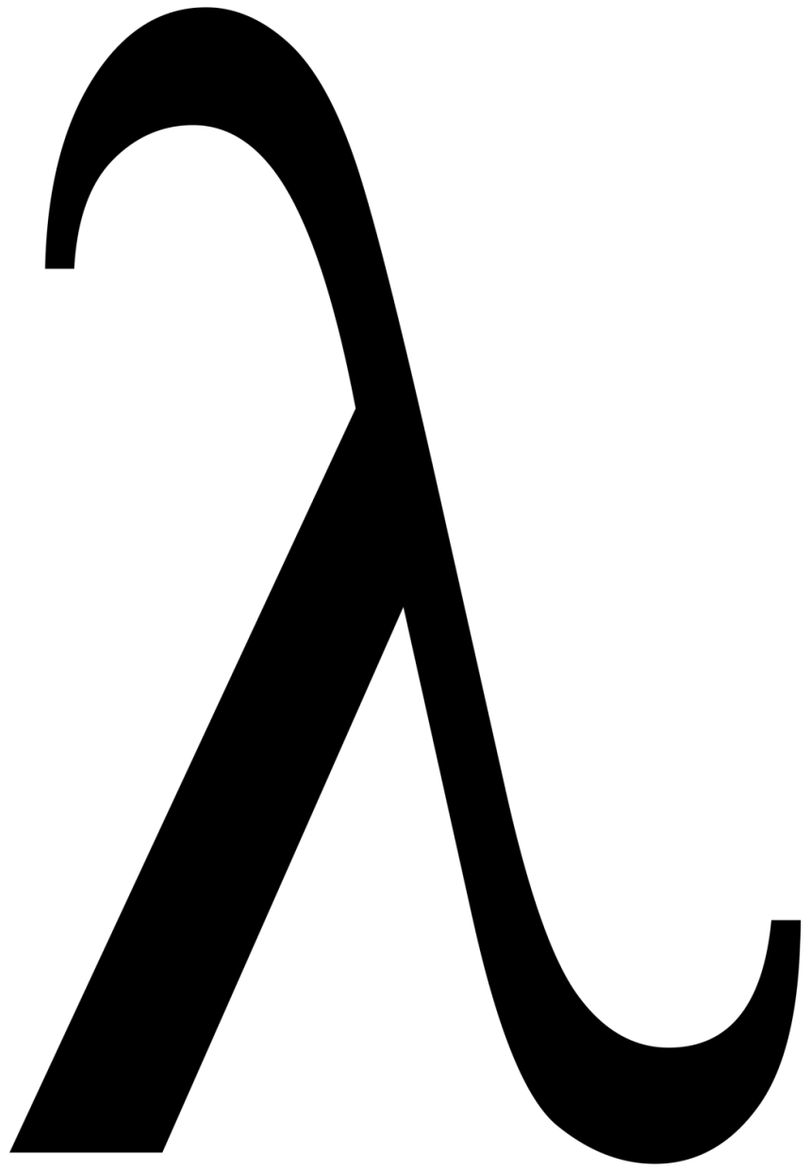
record of everything

that ever happened

extremely general

possible power

current power



data

via simple abstractions

with a general library

data

"info: user 24 viewed /home"

```
{:level   "info"
 :viewing true
 :user_id 24
 :path    "/home"}
```

via simple abstractions

```
(log
  {:level    :info
   :viewing true
   :user_id 24
   :path     "/home"})
```

```
(emit
  {:level   :info
   :viewing true
   :user_id 24
   :path    "/home"})
```

(emit event)

IFn, IPersistentMap

with a general library

map / fn / seq*

```
(->> events
  (filter :viewing)
  (count-by :path)
  (timechart))
```

```
(->> events
     (fn1 arg1)
     (fn2 arg2)
     (fn3))
```

data

via simple abstractions

with a general library

log lines → events

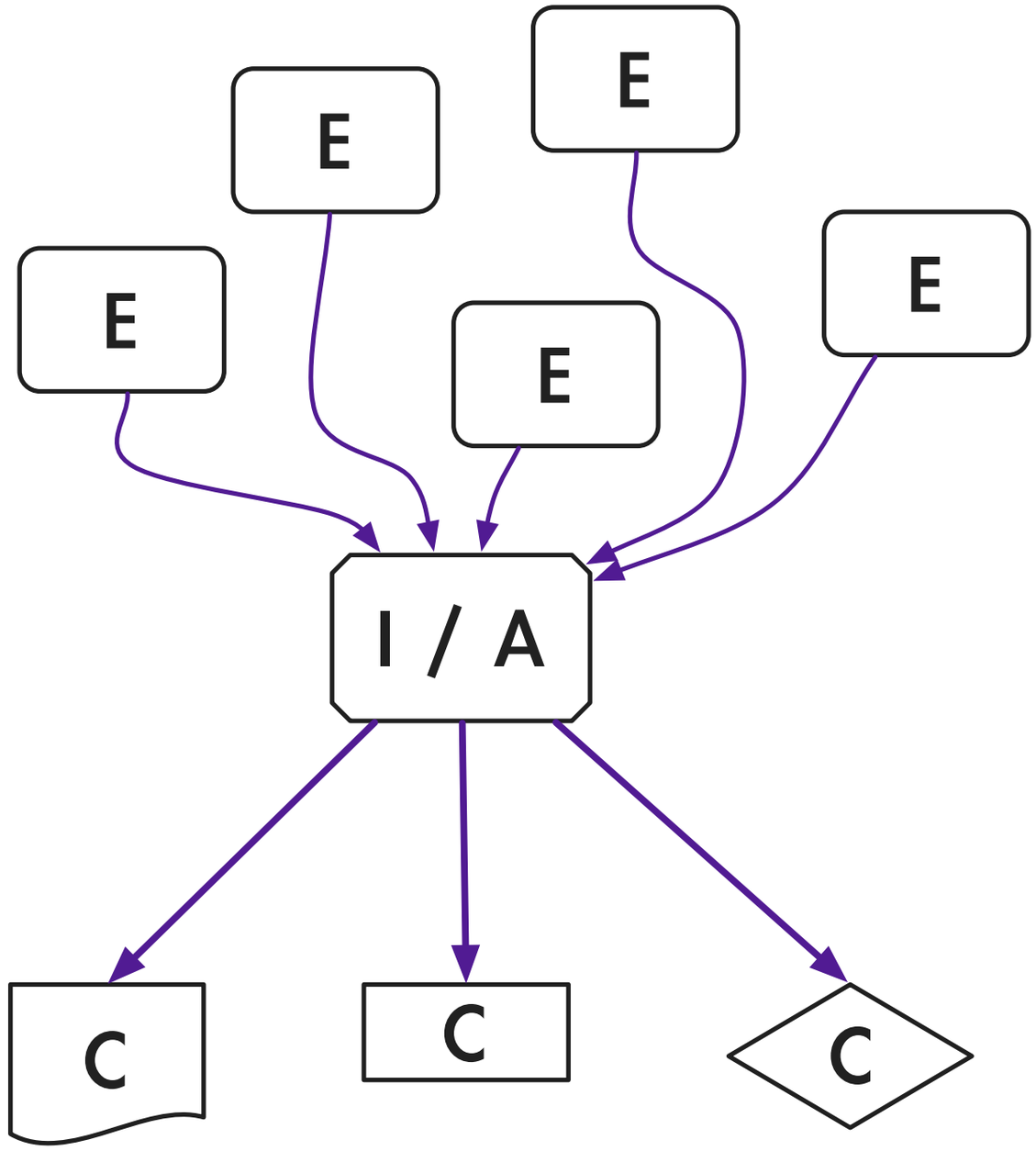
strings → maps

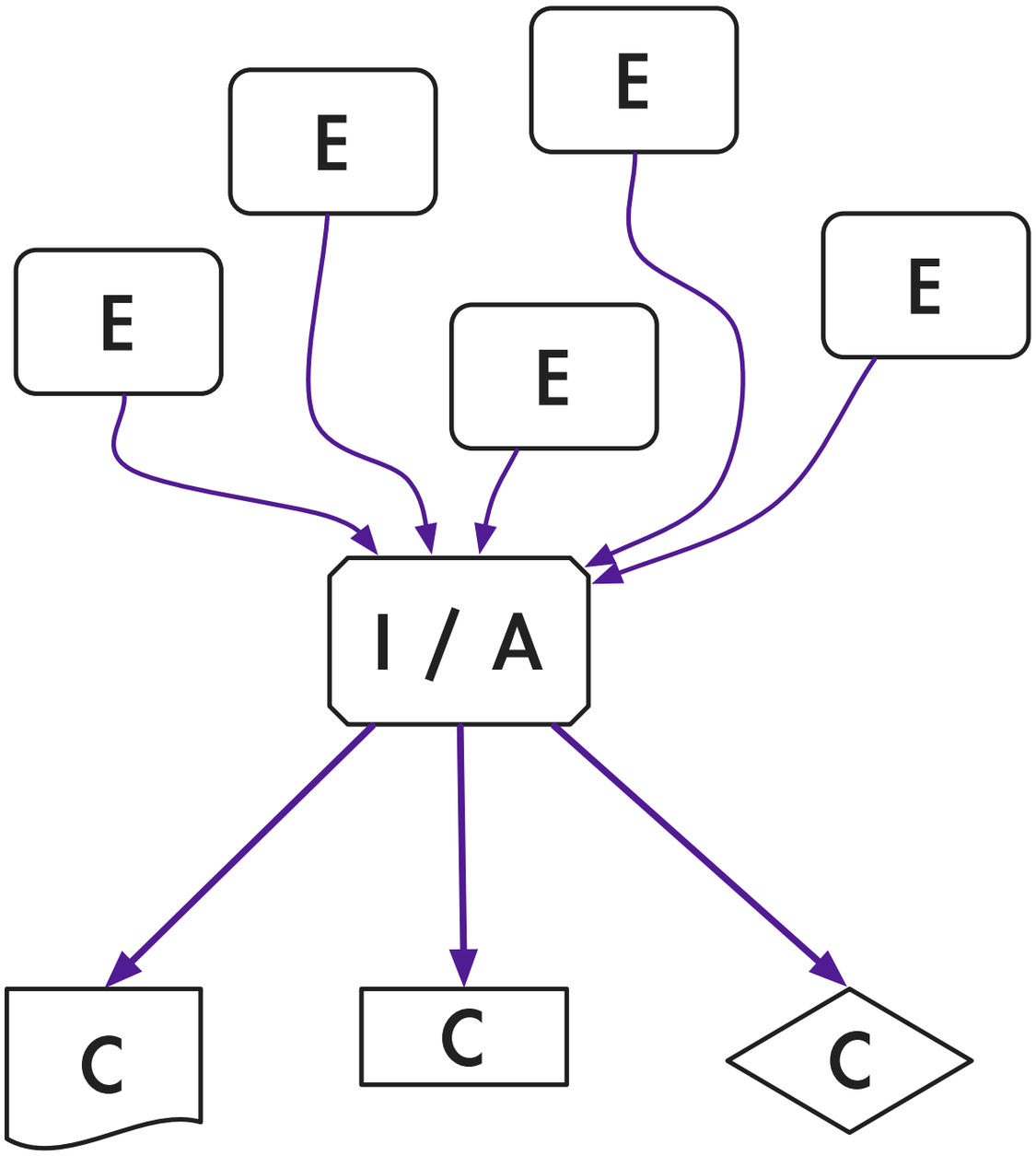
files → streams

possible power

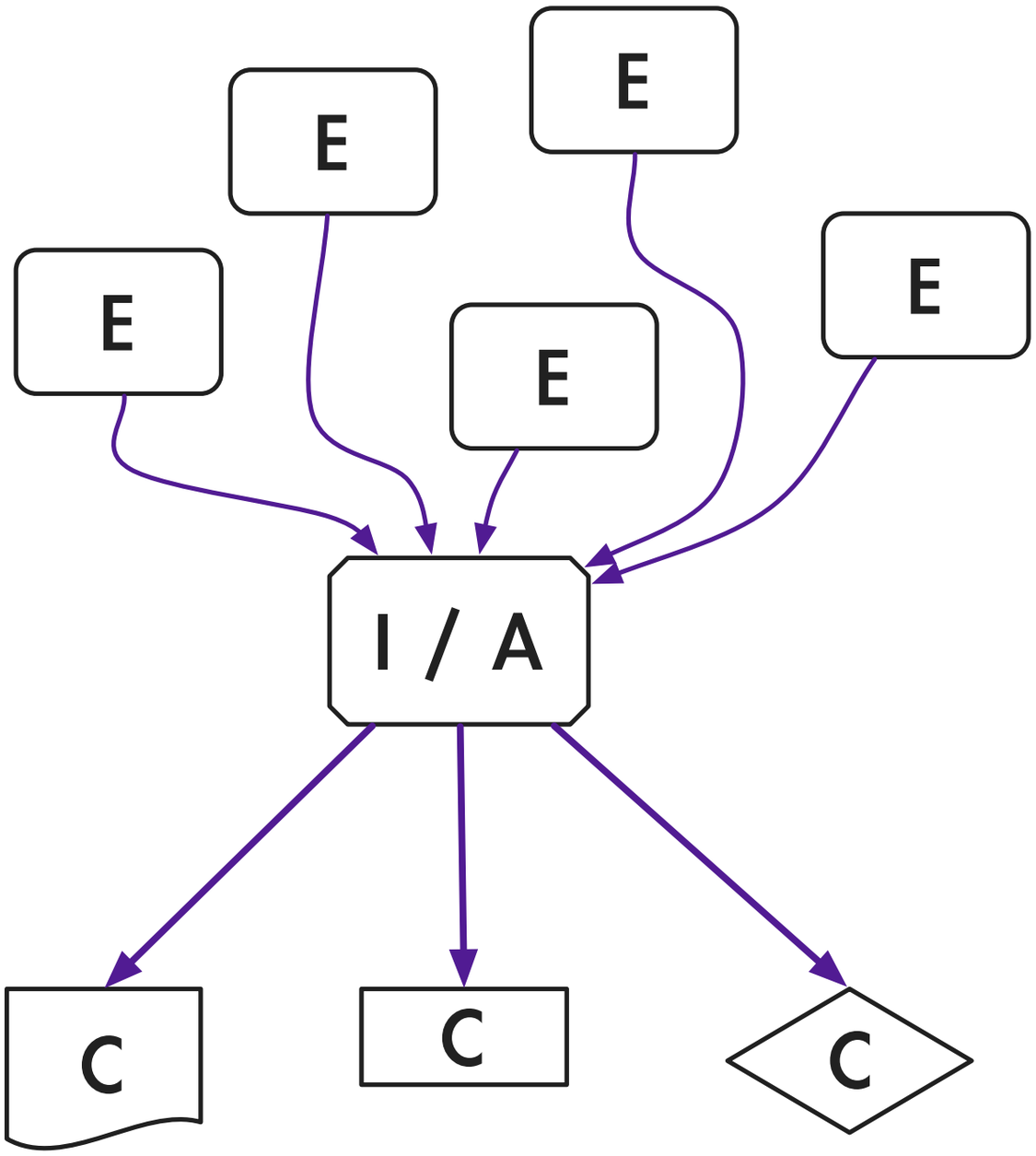
current power

log crunching → event processing


E
E
E
E
E
E
I / A
C
C
C



(f event-stream)



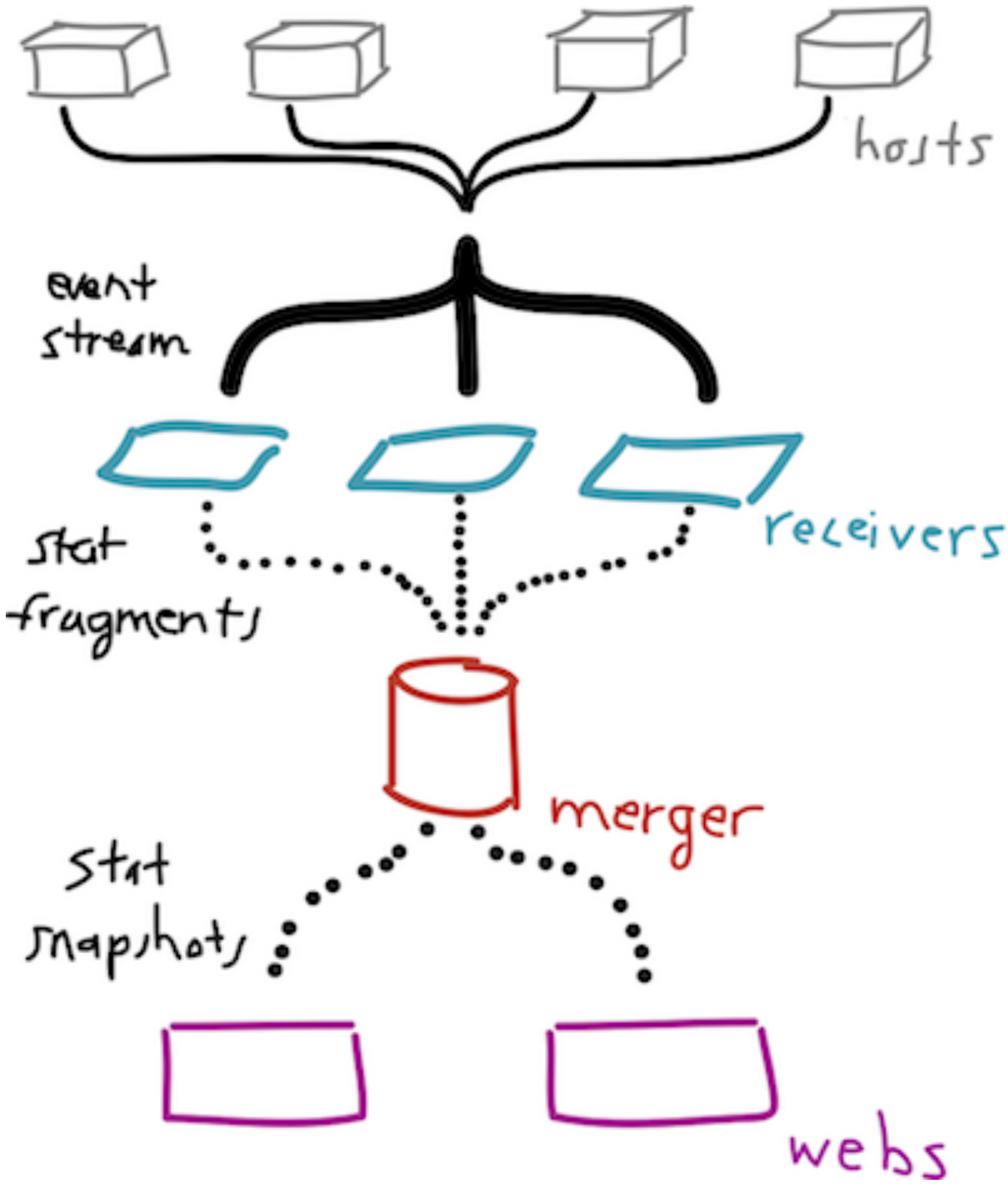
(g event-stream)

debugging
auditing
metrics
analytics
alerting

heroku/pulse

real-time metrics

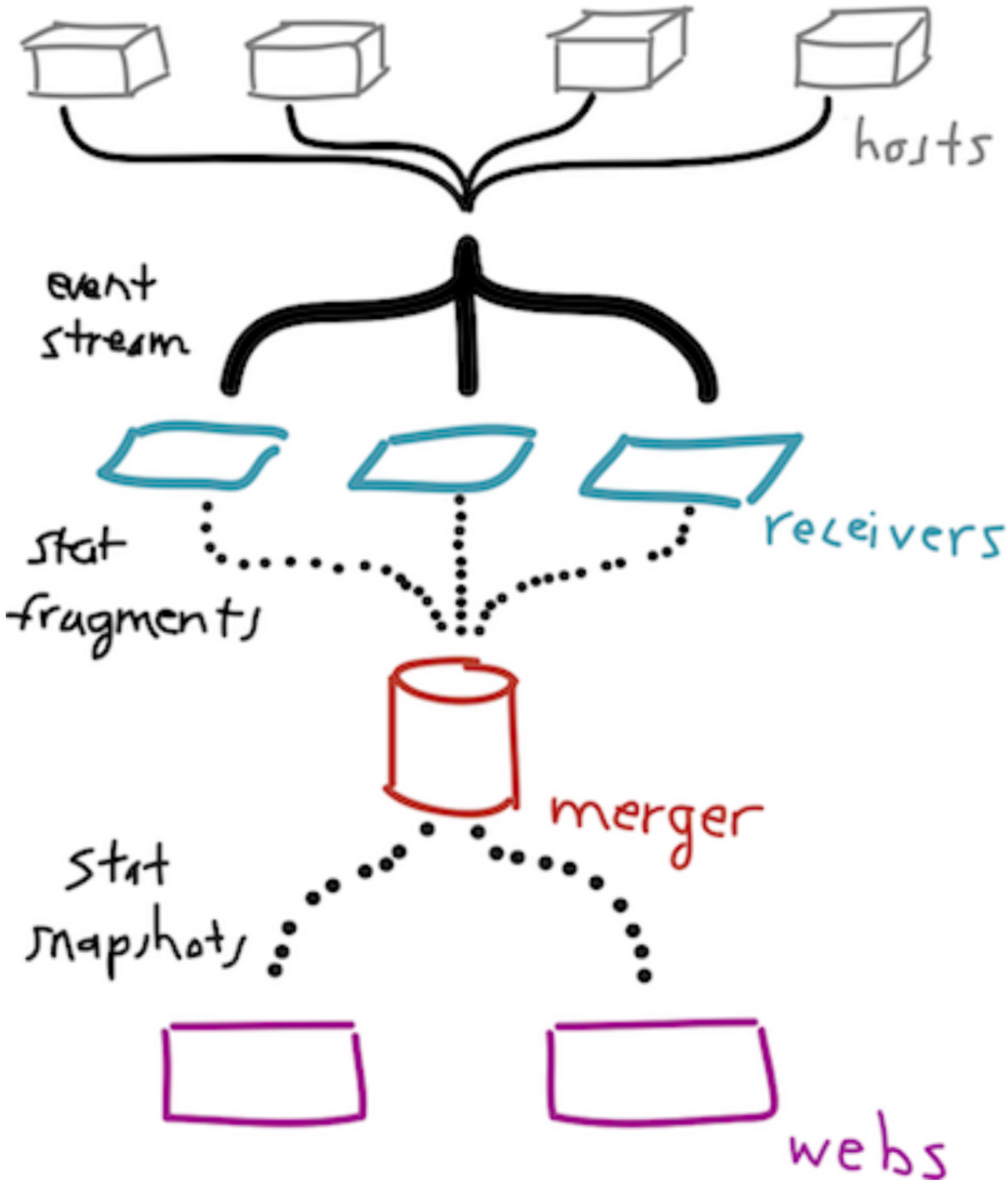
```
$ heroku open
```


hosts
event
stream
receivers
stat
fragments
merger
stat
snapshots
webs

```
(defstat requests-per-second
  (per-second
    (fn [e] (= (:source e) "nginx"))))
```

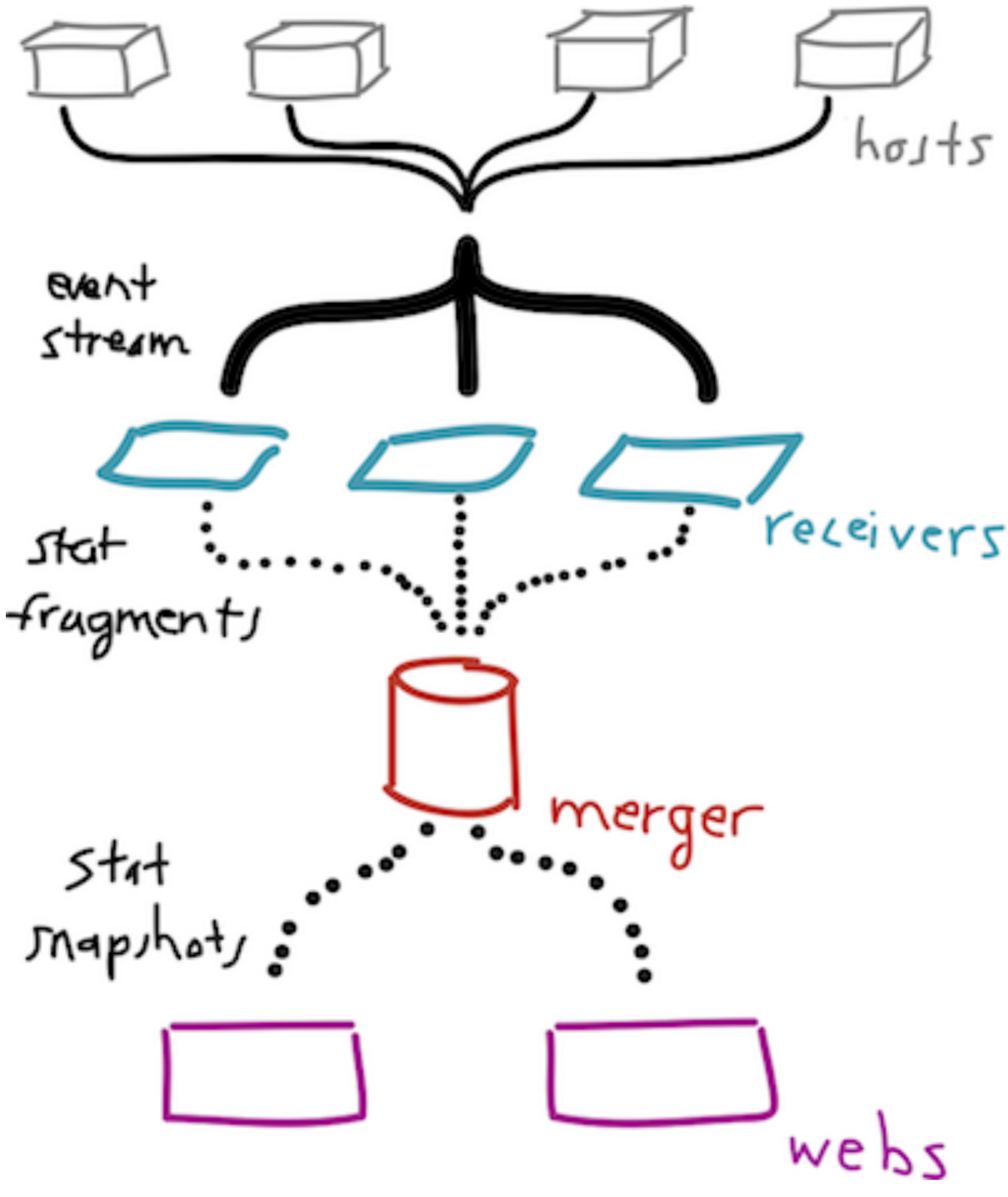
```
(defstat ps-launch-time-mean
  (mean 60
    (fn [e] (and (:monitor_boot e)
                 (= (:at e) "up")))
    (fn [e] (:age e))))
```

```
(defstat ps-running-total-last
  (last-sum
    (fn [e] (and (:runtime e)
                 (:counts e)))
    (fn [e] (:instance_id e))
    (fn [e] (:num e))))
```


hosts
event
stream
receivers
stat
fragments
merger
stat
snapshots
webs

```
(defstat requests-per-second
  (per-second
    (fn [e] (= (:source e) "nginx"))))
```

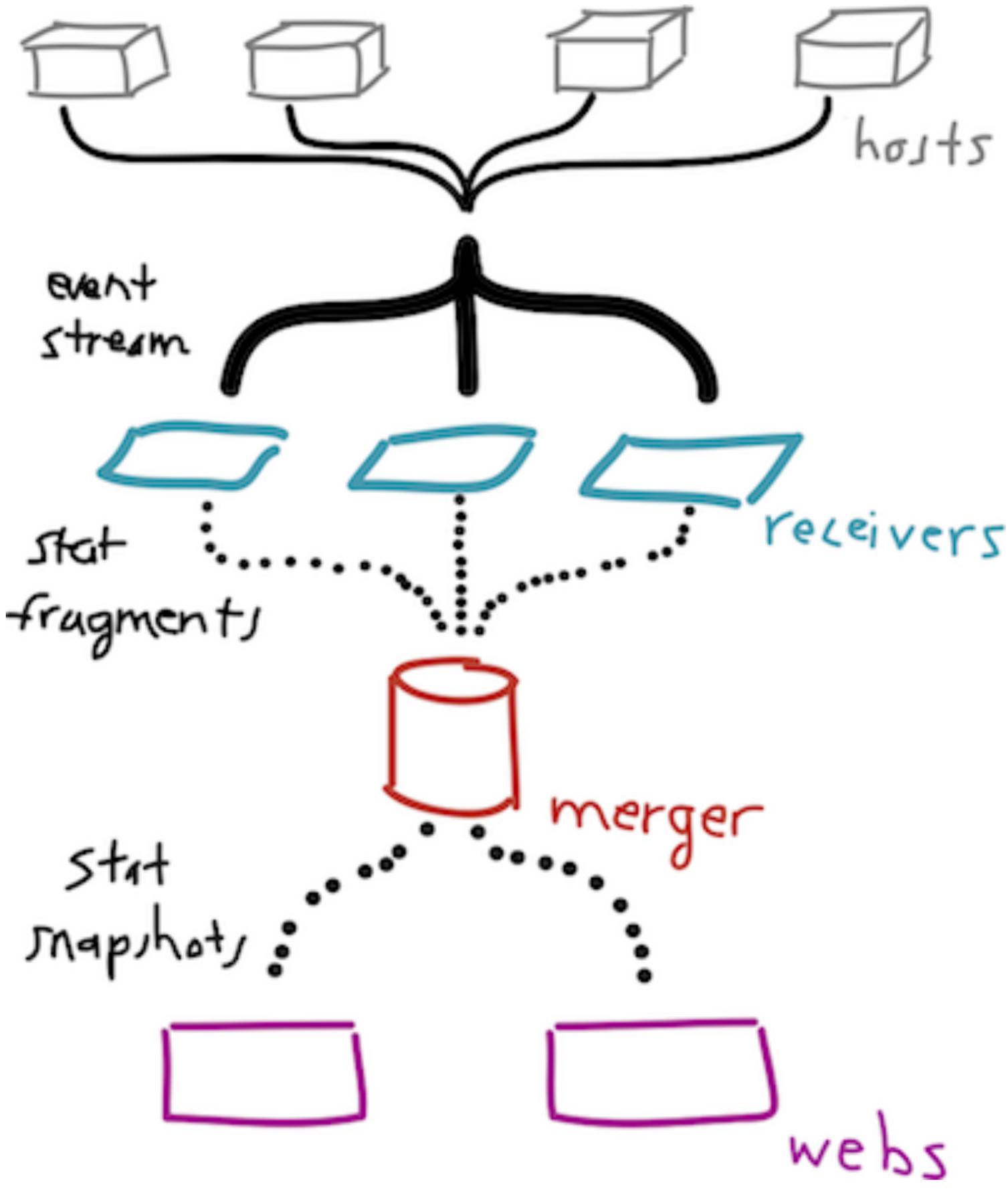
```
{:receive-init
   (fn [] ...)
 :receive-apply
   (fn [last-val event] ...)
 :receive-emit
   (fn [last-val] ...)

 :merge-init
   (fn [] ...)
 :merge-apply
   (fn [last-val received] ...)
 :merge-emit
   (fn [last-val] ...)}
```


hosts
event
stream
receivers
stat
fragments
merger
stat
snapshots
webs

```
(let [line  (q/take apply-queue)
      event (p/parse line)]
  (s/receive-apply stats event))
```

```
"runtime counts instance_id=1523 count=17"
```

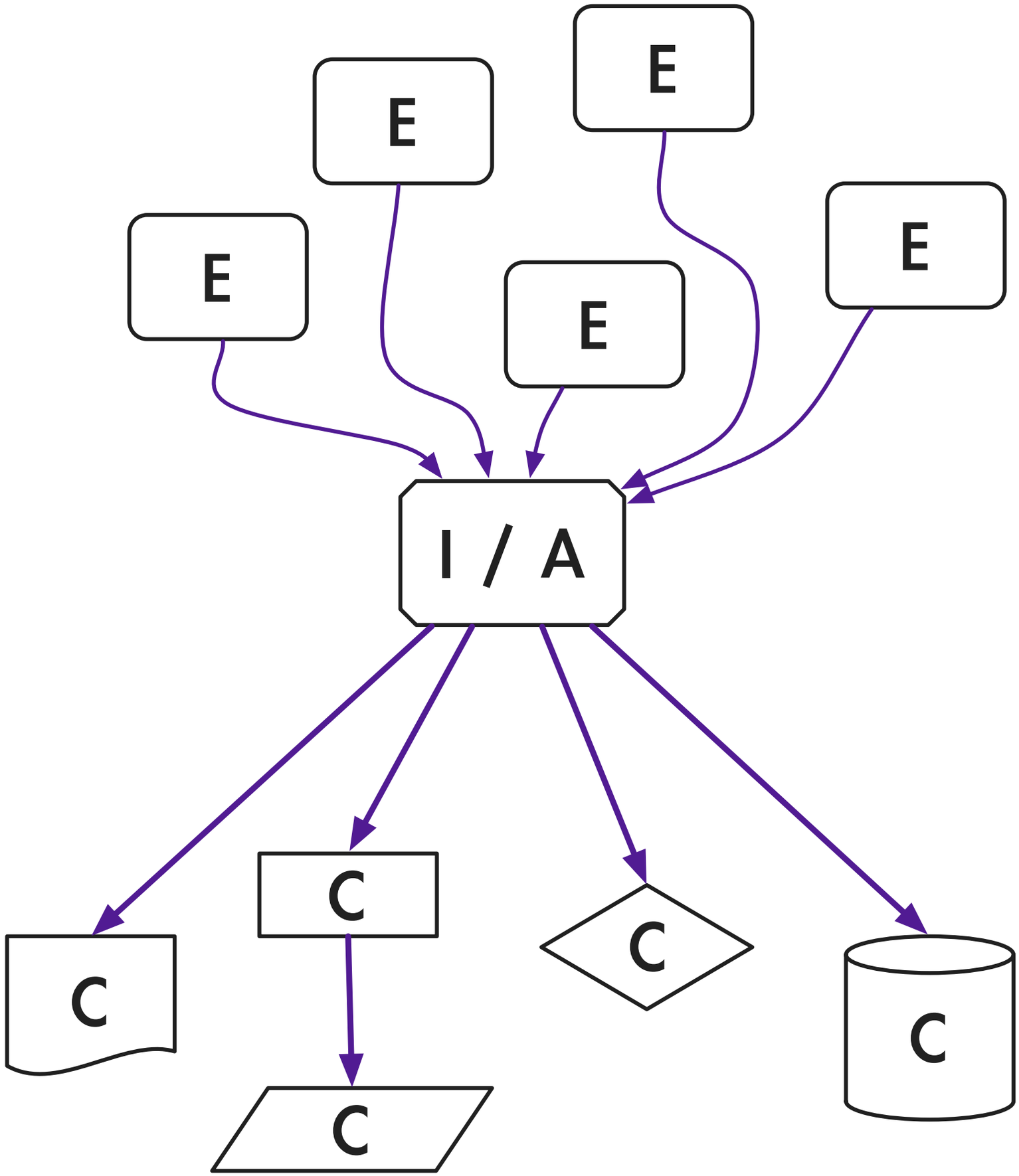
```
{:runtime     true
 :counts      true
 :instance_id 1523
 :count       17}
```

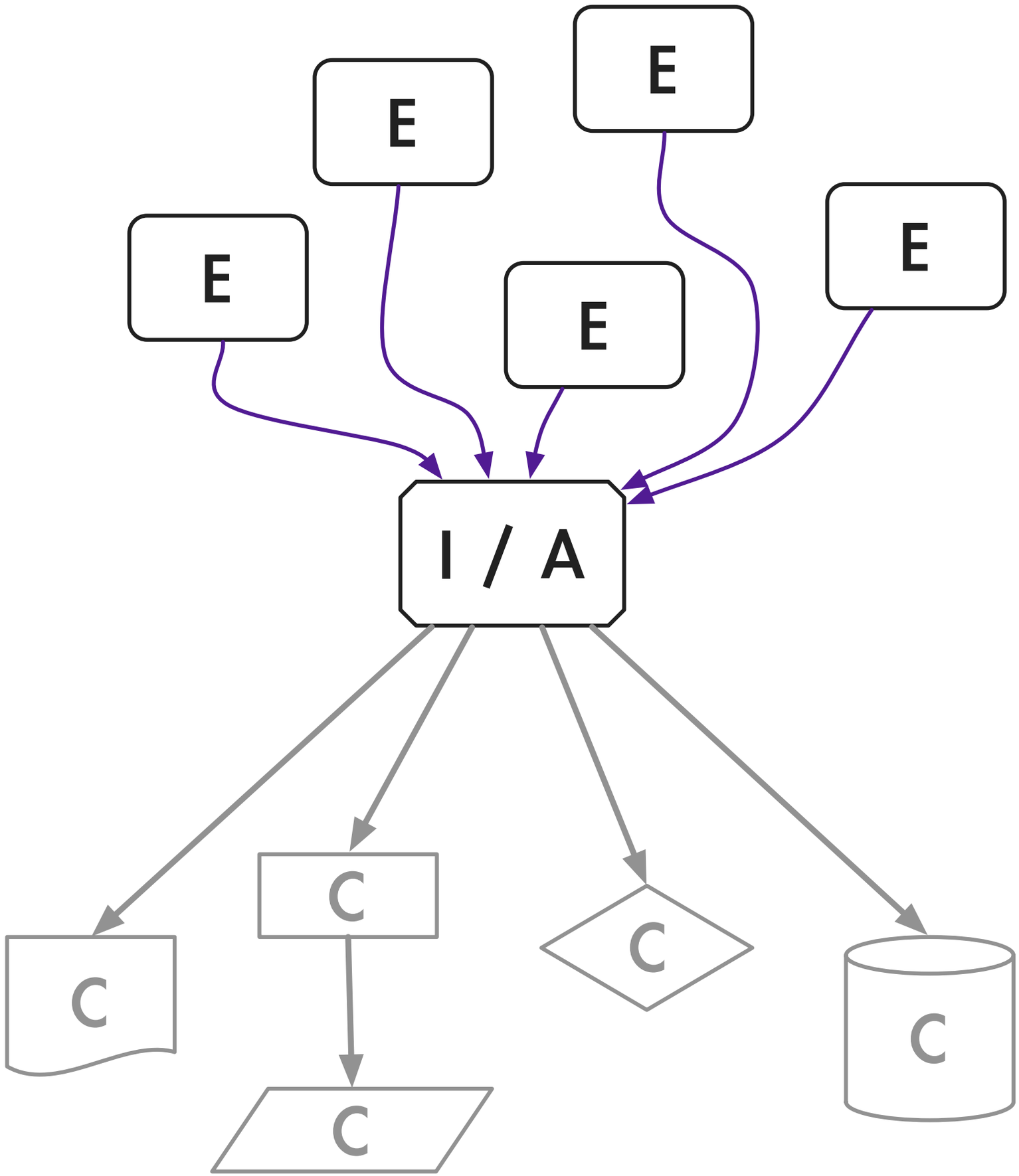

hosts
event
stream
receivers
stat
fragments
merger
stat
snapshots
webs

```
(f event-stream)
```

maps+seqs : Clojure

events : ?


E
E
E
E
E
E
I / A
C
C
C
C
C

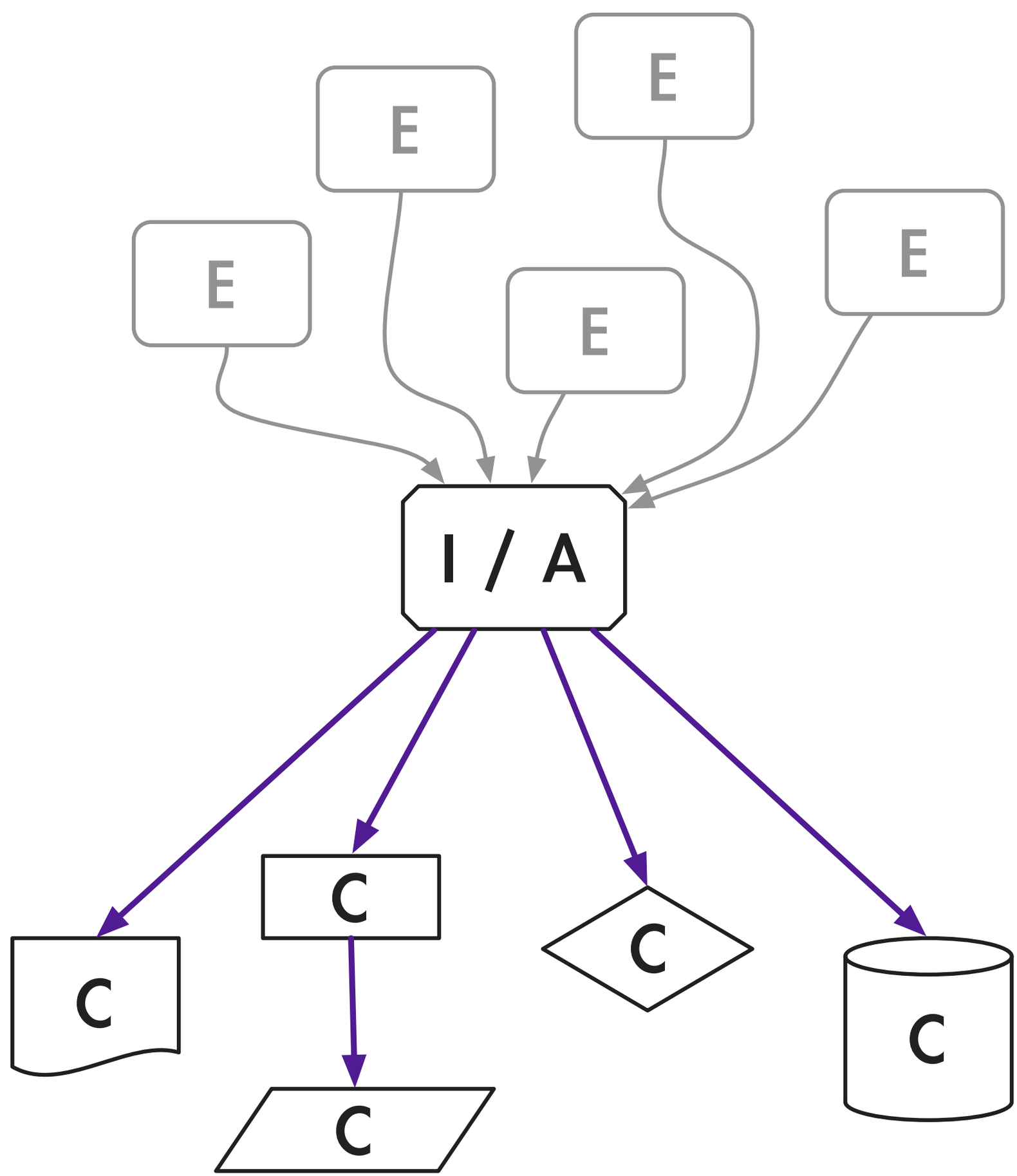

E
E
E
E
E
E
I / A
C
C
C
C
C

pervasive

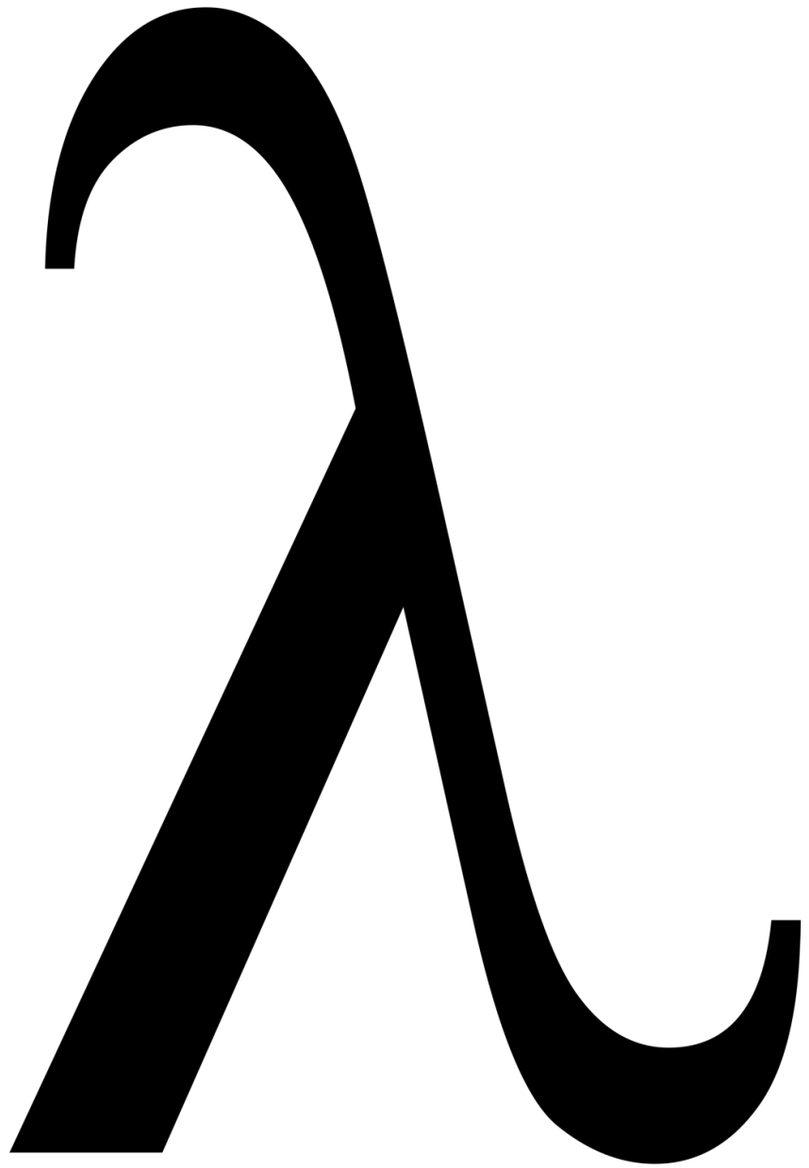
unified

event

collection


E
E
E
E
E
E
I / A
C
C
C
C
C

event processing services



Esper

Hadoop

Cascading

Cascalog

Storm

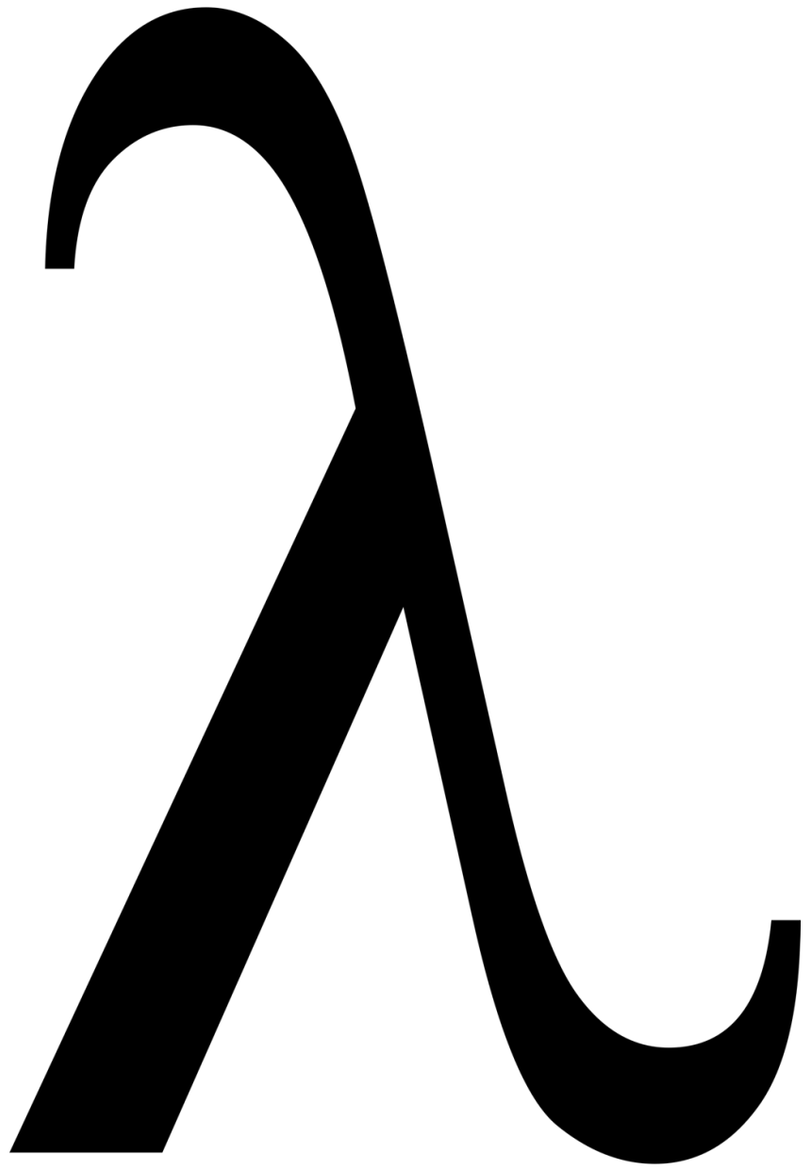
Conduit


heroku

logs as data

*

as data



thanks!

questions?