From logs to metrics.

on the desperate attempt to combat entropy

Who I am.

Leonardo Di Donato

Senior Software Engineer at influxdata®

Creator of *go-syslog*

github.com/leodido

twitter.com/leodido

www.linkedin.com/in/leodidonato

How many logs do we generate every day?

The quantity is not the only factor ...

How many standards - if any - we use to log?

How strictly we follow those standards formats?

How to transform kubernetes logs into metrics with the TICK stack.

Almost everyone needs to govern their logs.
Deriving metrics, synthesizing and visualizing them helps in decision making.

git.io/k8s-logs-to-metrics-tick (PoC)

First of all we needed a log parser.

But to parse which format ... ?

BSD-syslog - *RFC 3164* - resembled a de-facto standard.
Wide usage, lot of tools, long lifetime span (2001).
But ...
messy/informal RFC ...
no strict well-defined grammar
no single stable framing technique
too many customisations around.

Nope!

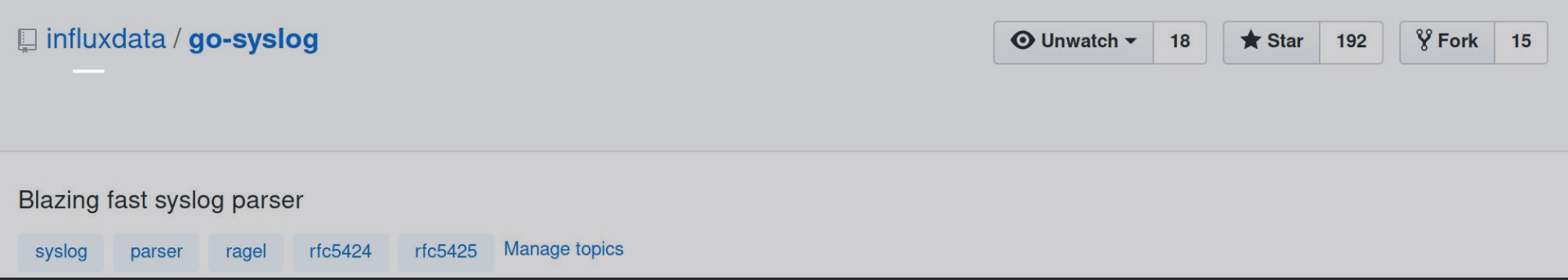
Thus we chose

RFC 5424 deprecates RFC 3164

- Well-defined grammar
- Octet counting framing
  - finally the stack trace for a panic in a single syslog …
- TLS transport mapping
  - secure logs
- Only 9 years old - ie., 2009

```
SYSLOG-MSG      = HEADER SP STRUCTURED-DATA [SP MSG]

HEADER          = PRI VERSION SP TIMESTAMP SP HOSTNAME SP APP-NAME SP PROCID SP MSGID
PRI             = "<" PRIVAL ">"
PRIVAL          = 1*3DIGIT ; range 0 .. 191
VERSION         = NONZERO-DIGIT 0*2DIGIT
HOSTNAME        = NILVALUE / 1*255PRINTUSASCII

APP-NAME        = NILVALUE / 1*48PRINTUSASCII
PROCID          = NILVALUE / 1*128PRINTUSASCII
MSGID           = NILVALUE / 1*32PRINTUSASCII

TIMESTAMP       = NILVALUE / FULL-DATE "T" FULL-TIME
FULL-DATE       = DATE-FULLYEAR "-" DATE-MONTH "-" DATE-MDAY
DATE-FULLYEAR   = 4DIGIT
DATE-MONTH      = 2DIGIT ; 01-12
DATE-MDAY       = 2DIGIT ; 01-28, 01-29, 01-30, 01-31 based on month/year
FULL-TIME       = PARTIAL-TIME TIME-OFFSET
PARTIAL-TIME    = TIME-HOUR ":" TIME-MINUTE ":" TIME-SECOND [TIME-SECFRAC]
TIME-HOUR       = 2DIGIT ; 00-23
TIME-MINUTE     = 2DIGIT ; 00-59
TIME-SECOND     = 2DIGIT ; 00-59
TIME-SECFRAC    = "." 1*6DIGIT
TIME-OFFSET     = "Z" / TIME-NUMOFFSET
TIME-NUMOFFSET  = ("+" / "-") TIME-HOUR ":" TIME-MINUTE

STRUCTURED-DATA = NILVALUE / 1*SD-ELEMENT
SD-ELEMENT      = "[" SD-ID *(SP SD-PARAM) "]"
SD-PARAM        = PARAM-NAME "=" %d34 PARAM-VALUE %d34
SD-ID           = SD-NAME
PARAM-NAME      = SD-NAME
PARAM-VALUE     = UTF-8-STRING ; characters '"', '\' and ']' MUST be escaped.
SD-NAME         = 1*32PRINTUSASCII ; except '=', SP, ']', %d34 (")

MSG             = MSG-ANY / MSG-UTF8
MSG-ANY         = *OCTET ; not starting with BOM
MSG-UTF8        = BOM UTF-8-STRING
BOM             = %xEF.BB.BF
UTF-8-STRING    = *OCTET ; UTF-8 string as specified in RFC 3629

OCTET           = %d00-255
SP              = %d32
PRINTUSASCII    = %d33-126
NONZERO-DIGIT   = %d49-57
DIGIT           = %d48 / NONZERO-DIGIT
NILVALUE        = "-"
```

real syslog.



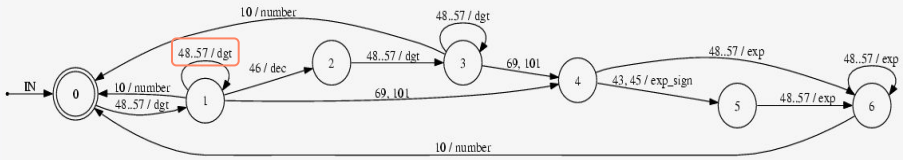
We chose Ragel to create the (Go) syslog parser

github.com/influxdata/go-syslog

A state machine compiler

- regular languages -> FSM
- can execute code (actions) at arbitrary points
- non-determinism operators
- table or control flow driven state machines
- various host languages - c, c++, obj-c, asm, d, go, java, ruby, c#, ocaml

```
action dgt      { printf("DGT: %c\n", fc); }
action dec      { printf("DEC: .\n"); }
action exp      { printf("EXP: %c\n", fc); }
action exp_sign { printf("SGN: %c\n", fc); }
action number   { /*NUMBER*/ }

number = (
    [0-9]+ $dgt ( '.' @dec [0-9]+ $dgt )?
    ( [eE] ( [+\-] $exp_sign )? [0-9]+ $exp )?
) %number;

main := ( number '\n' )*;
```




```
st0:
    if ( ++p == pe )
        goto out0;
    if ( 48 <= (*p) && (*p) <= 57 )
        goto tr0;
    goto st_err;
tr0:
    { printf("DGT: %c\n", (*p)); }
st1:
    if ( ++p == pe )
        goto out1;
    switch ( (*p) ) {
        case 10: goto tr5;
        case 46: goto tr7;
        case 69: goto st4;
        case 101: goto st4;
    }
    if ( 48 <= (*p) && (*p) <= 57 )
        goto tr0;
    goto st_err;
// ...
```

The gotos are your best friends. Only when you do not write them!

go-syslog provides parsers for RFC 5424 and RFC 5425.

`<85>4 2018-10-11T22:14:15.003Z leodido - 31932 - [ex@32473 iut="3"] An auth token...`

Numerical Code	Facility
0	kernel messages
1	user-level messages
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon (note 2)
16	local use 0 (local0)
17	local use 1 (local1)
18	local use 2 (local2)
19	local use 3 (local3)
20	local use 4 (local4)
21	local use 5 (local5)
22	local use 6 (local6)
23	local use 7 (local7)

Table 1. Syslog Message Facilities

Numerical Code	Severity
0	Emergency: system is unusable
1	Alert: action must be taken immediately
2	Critical: critical conditions
3	Error: error conditions
4	Warning: warning conditions
5	Notice: normal but significant condition
6	Informational: informational messages
7	Debug: debug-level messages

Table 2. Syslog Message Severities

```
prival = facility * 8 + severity
```

- *tools.ietf.org/html/rfc5424.html* (Syslog grammar)
- *tools.ietf.org/html/rfc5425.html* (TLS + octet counting)
- *tools.ietf.org/html/rfc5426.html* (UDP)
- *tools.ietf.org/html/rfc6587.html* (TCP + octet counting)
- *man7.org/linux/man-pages/man3/syslog.3.html*
- *man7.org/linux/man-pages/man0/syslog.h.0p.html*
- *man7.org/linux/man-pages/man1/logger.1.html*

```
bestEffortOn := true
i := []byte(`<165>4 2018-10-11T22:14:15.003Z mymach.it e - 1 [ex@32473 iut="3"] An app event...`)
p := rfc5424.NewParser()
m, e := p.Parse(i, &bestEffortOn) // best effort mode on means both m and e can have value ...
```

This results in `m` being equal to the following `SyslogMessage` instance. While error `e` is `nil` in this case.

```
// (*rfc5424.SyslogMessage)({
//  priority: (*uint8)(165),
//  facility: (*uint8)(20),
//  severity: (*uint8)(5),
//  version: (uint16) 4,
//  timestamp: (*time.Time)(2018-10-11 22:14:15.003 +0000 UTC),
//  hostname: (*string)((len=9) "mymach.it"),
//  appname: (*string)((len=1) "e"),
//  procID: (*string)(<nil>),
//  msgID: (*string)((len=1) "1"),
//  structuredData: (*map[string]map[string]string)((len=1) {
//   (string) (len=8) "ex@32473": (map[string]string) (len=1) {
//    (string) (len=3) "iut": (string) (len=1) "3"
//   }
//  }),
//  message: (*string)((len=33) "An app event...")
// })
```

It provides also a builder.

Incrementally build valid syslog messages

```
msg := &SyslogMessage{}
msg.SetTimestamp("not a RFC3339MICRO timestamp")
// Not yet a valid message (try msg.Valid())
msg.SetPriority(191)
msg.SetVersion(1)
msg.Valid() // Now it is minimally valid
str, _ := msg.String()
// str is “<191>1 - - - - - -”
```

Notice that its API ignores input values that does not follow the grammar.

Performances.

- ~250ns to parse the smallest legal message
- ~2µs to parse an average legal message
- ~4µs to parse a very long legal message

```
[no]_empty_input_____________________________________-4  30000000       253 ns/op     224 B/op       3 allocs/op
[no]_multiple_syslog_messages_on_multiple_lines___-4  20000000       433 ns/op     304 B/op      12 allocs/op
[no]_impossible_timestamp____________________________-4  10000000      1080 ns/op     528 B/op      11 allocs/op
[no]_malformed_structured_data_______________________-4  20000000       552 ns/op     400 B/op      12 allocs/op
[no]_with_duplicated_structured_data_id______________-4   5000000      1246 ns/op     688 B/op      17 allocs/op
[ok]_minimal_________________________________________-4  30000000       264 ns/op     247 B/op       9 allocs/op
[ok]_average_message_________________________________-4   5000000      1984 ns/op    1536 B/op      26 allocs/op
[ok]_complicated_message_____________________________-4   5000000      1644 ns/op    1280 B/op      25 allocs/op
[ok]_very_long_message_______________________________-4   2000000      3826 ns/op    2464 B/op      28 allocs/op
[ok]_all_max_length_and_complete_____________________-4   3000000      2792 ns/op    1888 B/op      28 allocs/op
[ok]_all_max_length_except_structured_data_and_mes-4   5000000      1830 ns/op     883 B/op      13 allocs/op
[ok]_minimal_with_message_containing_newline_____-4  20000000       294 ns/op     250 B/op      10 allocs/op
[ok]_w/o_procid,_w/o_structured_data,_with_message-4  10000000       956 ns/op     364 B/op      11 allocs/op
[ok]_minimal_with_UTF-8_message______________________-4  20000000       586 ns/op     359 B/op      10 allocs/op
[ok]_with_structured_data_id,_w/o_structured_data_-4  10000000       998 ns/op     592 B/op      14 allocs/op
[ok]_with_multiple_structured_data___________________-4   5000000      1538 ns/op    1232 B/op      22 allocs/op
[ok]_with_escaped_backslash_within_structured_data-4   5000000      1316 ns/op     920 B/op      20 allocs/op
[ok]_with_UTF-8_structured_data_param_value,_with_-4   5000000      1580 ns/op    1050 B/op      21 allocs/op
```

—

Telegraf is the plugin-driven server agent for collecting & reporting metrics.

github.com/influxdata/telegraf

Thus we created the *syslog input plugin* for it, using *go-syslog*

- Listens for syslog messages transmitted over UDP - RFC 5426 - or TCP.
- Supports (atm) only messages formatted according to RFC 5424.
- Supports TLS, octet framing (both over TCP - RFC 6587 - and TLS - RFC 5425).
- BSD format - RFC 3164 - in progress.

Metrics

Measurement: syslog

- tags
  - severity (string)
  - facility (string)
  - hostname (string)
  - appname (string)
- fields
  - version (integer)
  - severity_code (integer)
  - facility_code (integer)
  - timestamp (integer) - the time recorded in the syslog message
  - procid (string)
  - msgid (string)
  - sdid (bool)
  - structured data elements (string)
- timestamp - the time the messages was received

```
[[inputs.syslog]]
  ## Specify an ip or hostname with port - eg., tcp://localhost:6514, tcp://10.0.0.1:6514
  ## Protocol, address and port to host the syslog receiver.
  ## If no host is specified, then localhost is used.
  ## If no port is specified, 6514 is used (RFC5425#section-4.1).
  server = "tcp://:6514"

  ## TLS Config
  # tls_allowed_cacerts = ["/etc/telegraf/ca.pem"]
  # tls_cert = "/etc/telegraf/cert.pem"
  # tls_key = "/etc/telegraf/key.pem"

  ## Period between keep alive probes.
  ## 0 disables keep alive probes.
  ## Defaults to the OS configuration.
  ## Only applies to stream sockets (e.g. TCP).
  # keep_alive_period = "5m"

  ## Maximum number of concurrent connections (default = 0).
  ## 0 means unlimited.
  ## Only applies to stream sockets (e.g. TCP).
  # max_connections = 1024

  ## Read timeout is the maximum time allowed for reading a single message (default = 5s).
  ## 0 means unlimited.
  # read_timeout = "5s"

  ## Whether to parse in best effort mode or not (default = false).
  ## By default best effort parsing is off.
  # best_effort = false

  ## Character to prepend to SD-PARAMs (default = "_").
  ## A syslog message can contain multiple parameters and multiple identifiers within structured data
  ## Eg., [id1 name1="val1" name2="val2"][id2 name1="val1" nameA="valA"]
  ## For each combination a field is created.
  ## Its name is created concatenating identifier, sdparam_separator, and parameter name.
  # sdparam_separator = "_"
```

Input (with octet counting)

```
169 <165>1 2018-10-01:14:15.000Z mymachine.example.com evntslog - ID47 [exampleSDID@32473
iut="3" eventSource="Application" eventID="1011"] An application event log entry...
```

Output

```
syslog,appname=evntslog,facility=local4,hostname=mymachine.example.com,severity=notice
exampleSDID@32473_eventID="1011",exampleSDID@32473_eventSource="Application",exampleSDID@32
473_iut="3",facility_code=20i,message="An application event log
entry...",msgid="ID47",severity_code=5i,timestamp=1065910455003000000i,version=1i
1538421339749472344
```

Source code at *git.io/k8s-logs-to-metrics-tick*

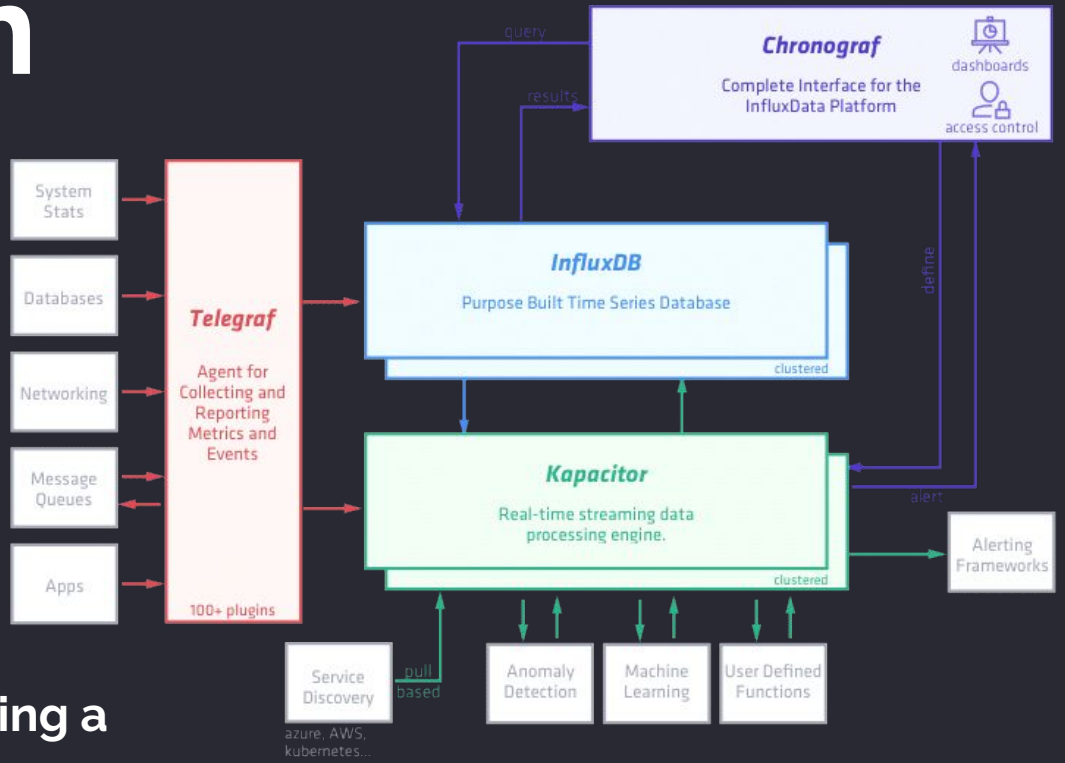
Our solution

Grab k8s and kernel logs from journald.

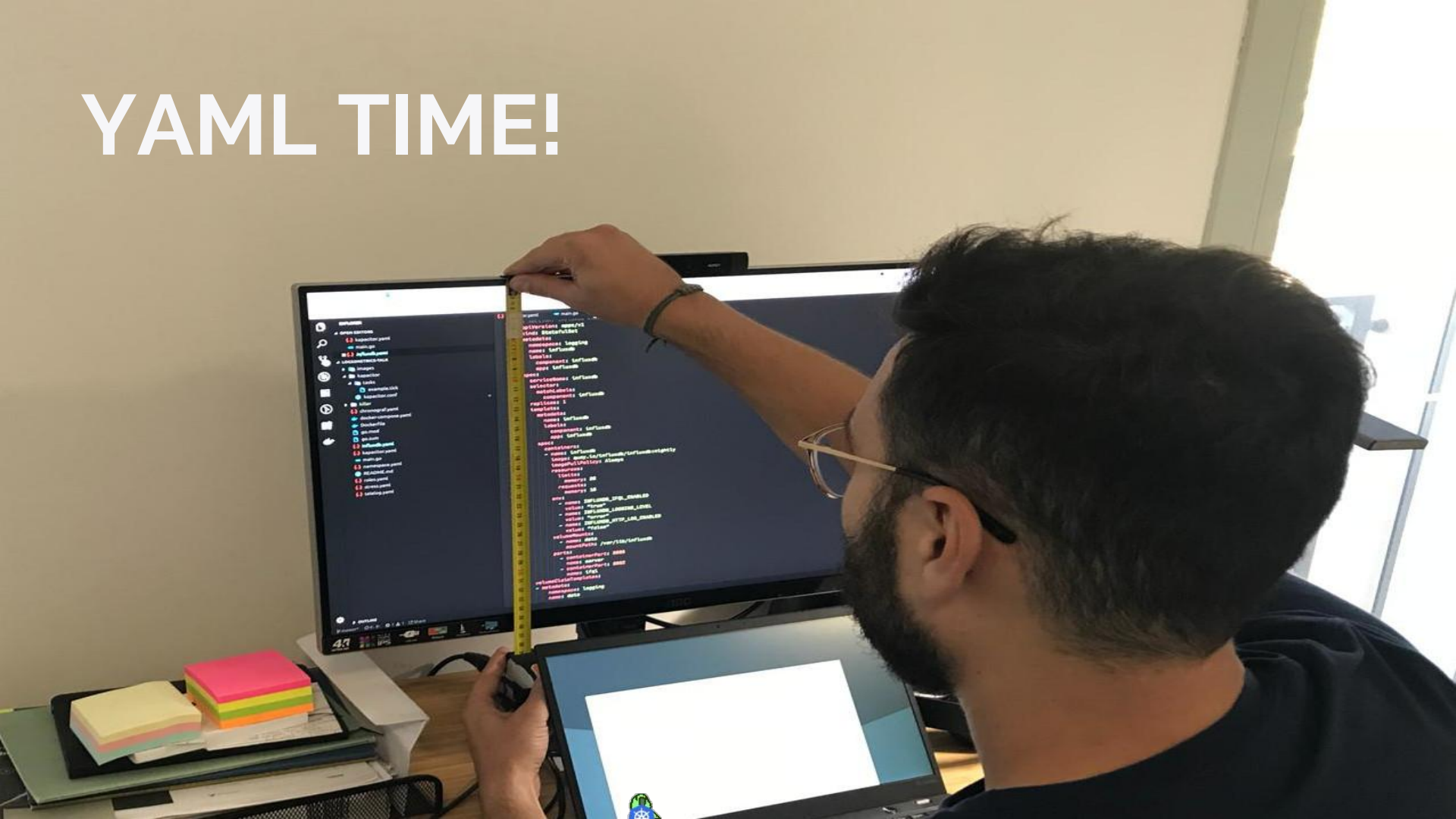
Parse them via telegraf syslog input plugin.

Visualize logs with chronograf log viewer.

Elicit new metrics to plot applying a kapacitor UDF.



YAML TIME!



Using rsyslog to grab RFC 5424 syslog messages from journald.

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: rsyslog
 namespace: logging
 labels:
   component: rsyslog
   app: rsyslog
data:
 rsyslog.conf: |+
   # …
   module(load="imjournal" ...)
   # This module only works with the journald and json-file docker log drivers
   module(load="mmkubernetes" tls.cacert="..." tokenfile="..." annotation_match=["."])
   # Extracts k8s metadata
   action(type="mmkubernetes")
   # …
   # Compose RFC5424 message
   template(name="rfc5424" type="list") { … }
   action(type="omfwd" target="127.0.0.1" port="6514" protocol="tcp" tcp_framing="octet-counted"
     template="rfc5424" ...)
```

Setup telegraf syslog plugin to receive log messages over TCP.

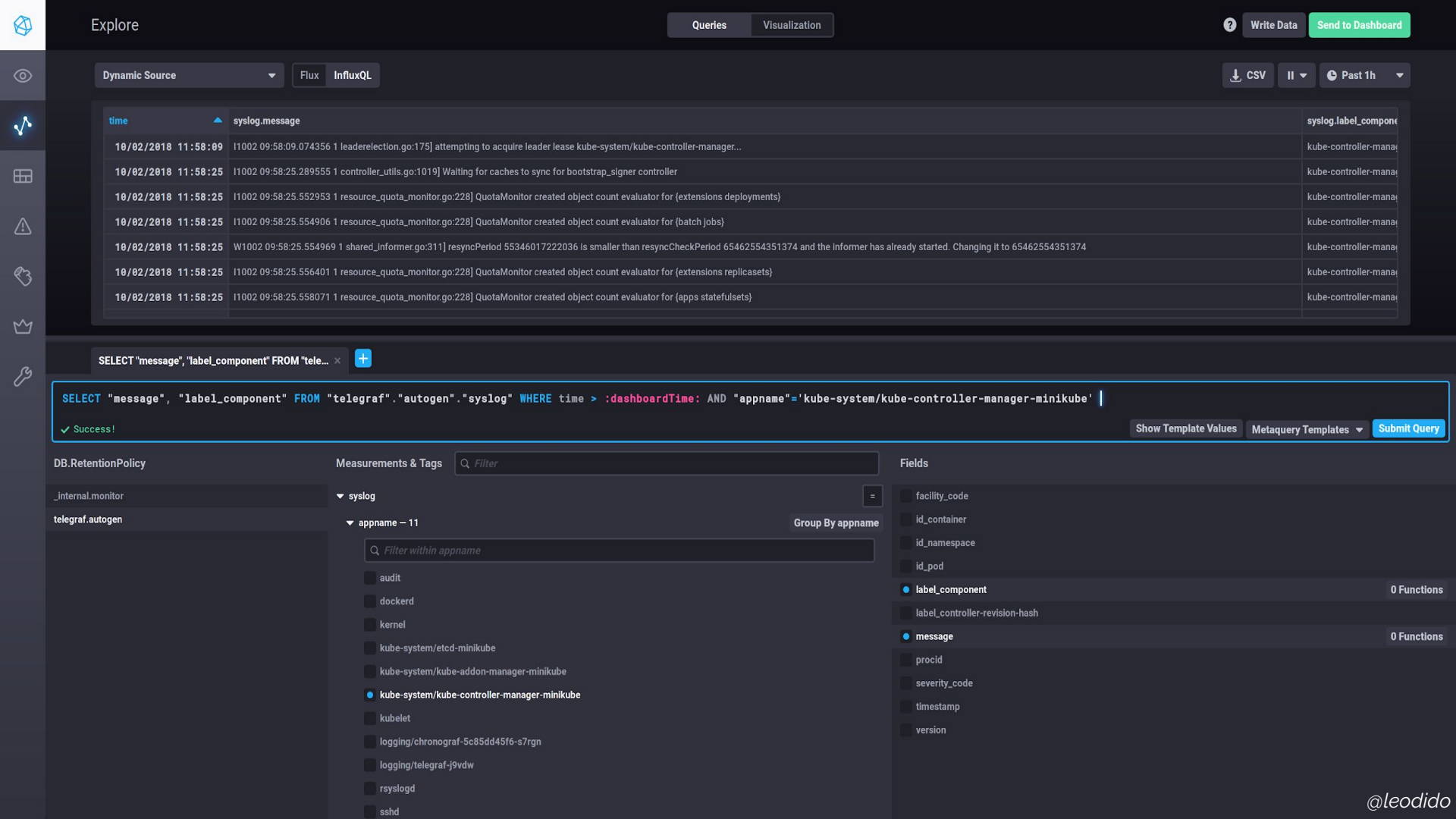
```
apiVersion: v1
kind: ConfigMap
metadata:
 name: telegraf
 namespace: logging
 labels:
   component: telegraf
   app: telegraf
data:
 telegraf.conf: |+
   # ...
   [agent]
     interval = "10s"
     round_interval = true
     metric_batch_size = 1000
     # ...
   [[outputs.influxdb]]
     urls = ["http://influxdb:8086"] # required
     database = "telegraf" # required
     retention_policy = "autogen"
     write_consistency = "any"
     timeout = "1m"
   [[inputs.syslog]]
     server = "tcp://:6514"
     best_effort = true
```

Let’s deploy chronograf and influxDB

```yaml
apiVersion: v1
kind: Service
metadata:
 name: chronograf
 namespace: logging
 labels:
   component: chronograf
   app: chronograf
spec:
 ports:
 - port: 80
   targetPort: 8888
   name: server
 selector:
   component: chronograf
---
apiVersion: apps/v1
kind: Deployment
# ...
```

```yaml
apiVersion: v1
kind: Service
metadata:
 name: influxdb
 namespace: logging
 labels:
   component: influxdb
   app: influxdb
 annotations:
   service.alpha.kubernetes.io/tolerate-unready-endpoints: “true”
spec:
 clusterIP: None
 ports:
 - port: 8086
   name: server
 selector:
   component: influxdb
---
apiVersion: apps/v1
kind: StatefulSet
# ...
```


Log Viewer
InfluxDB @ http://influxdb.logging.svc:8086
telegraf.autogen
Displaying 2582 events in histogram
1h Window
100
80
60
40
20
0
09:10
09:15
09:20
09:25
09:30
09:35
09:40
09:45
09:50
09:55
10 PM
10:05
20 err
7 warning
48 notice
19 info
Search logs using keywords or regular expressions...
1h ago
Search
Truncate
Wrap
Severity
Timestamp
Message
Facility
Proc ID
Application
Host
notice 2018-10-02 21:53:52 audit: type=1325 audit(1538510032.727:2438): table=filter family=2 entries=33 kern kernel
notice 2018-10-02 21:53:52 PROCTITLE proctitle=69707461626C65732D726573746F7265002D2D6E6F666C757368002D2D636F756E74657273 auth audit
notice 2018-10-02 21:53:52 SYSCALL arch=c000003e syscall=54 success=yes exit=0 a0=3 a1=0 a2=40 a3=557d9402b7c0 items=0 ppid=3823 pid=26670 auid=42949672… auth 26670 audit
notice 2018-10-02 21:53:52 NETFILTER_CFG table=filter family=2 entries=33 auth audit
info 2018-10-02 21:53:37 INFO: == Kubernetes addon reconcile completed at 2018-10-02T19:53:37+0000 == user 2398 kube-system/kube-...
info 2018-10-02 21:53:37 serviceaccount "storage-provisioner" unchanged user 2398 kube-system/kube-...
info 2018-10-02 21:53:37 INFO: == Reconciling with addon-manager label == user 2398 kube-system/kube-...
err 2018-10-02 21:53:37 error: no objects passed to apply user 2398 kube-system/kube-...
err 2018-10-02 21:53:36 rsyslogd: omfwd: TCPSendBuf error -2027, destruct TCP Connection to 127.0.0.1:6514 [v8.36.0 try http://www.rsyslog.com/e/2027… user 2398 logging/telegraf-...
info 2018-10-02 21:53:36 INFO: == Reconciling with deprecated label == user 2398 kube-system/kube-...
info 2018-10-02 21:53:36 action 'action 2' resumed (module 'builtin:omfwd') [v8.36.0 try http://www.rsyslog.com/e/2359] syslog rsyslogd
warning 2018-10-02 21:53:36 action 'action 2' suspended (module 'builtin:omfwd'), retry 0. There should be messages before this one giving the reason for… syslog rsyslogd
err 2018-10-02 21:53:36 omfwd: TCPSendBuf error -2027, destruct TCP Connection to 127.0.0.1:6514 [v8.36.0 try http://www.rsyslog.com/e/2027] syslog rsyslogd
info 2018-10-02 21:53:36 INFO: == Kubernetes addon ensure completed at 2018-10-02T19:53:36+0000 == user 2398 kube-system/kube-...
info 2018-10-02 21:53:36 INFO: Leader is minikube user 2398 kube-system/kube-...
err 2018-10-02 21:53:23 rsyslogd: omfwd: TCPSendBuf error -2027, destruct TCP Connection to 127.0.0.1:6514 [v8.36.0 try http://www.rsyslog.com/e/2027… user 2398 logging/telegraf-...
info 2018-10-02 21:53:23 action 'action 2' resumed (module 'builtin:omfwd') [v8.36.0 try http://www.rsyslog.com/e/2359] syslog rsyslogd
warning 2018-10-02 21:53:23 action 'action 2' suspended (module 'builtin:omfwd'), retry 0. There should be messages before this one giving the reason for… syslog rsyslogd
err 2018-10-02 21:53:23 omfwd: TCPSendBuf error -2027, destruct TCP Connection to 127.0.0.1:6514 [v8.36.0 try http://www.rsyslog.com/e/2027] syslog rsyslogd
notice 2018-10-02 21:53:22 audit: type=1327 audit(1538510002.687:2437): proctitle=69707461626C65732D726573746F7265002D2D6E6F666C757368002D2D636F756E7465… kern kernel
notice 2018-10-02 21:53:22 audit: type=1300 audit(1538510002.687:2437): arch=c000003e syscall=54 success=yes exit=0 a0=3 a1=0 a2=40 a3=5652315d95e0 item… kern kernel
notice 2018-10-02 21:53:22 audit: type=1325 audit(1538510002.687:2437): table=nat family=2 entries=61 kern kernel
notice 2018-10-02 21:53:22 auth audit
SYSCALL arch=c000003e syscall=54 success=yes exit=0 a0=3 a1=0 a2=40 a3=5652315d95e0 items=0 ppid=3823 pid=26447
auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="iptables-restor"
exe="/sbin/xtables-multi" subj=kernel key=(null)
notice 2018-10-02 21:53:22 auth 26447 audit
notice 2018-10-02 21:53:22 auth audit
notice 2018-10-02 21:53:22 kern kernel
notice 2018-10-02 21:53:22 audit: type=1300 audit(1538510002.685:2436): arch=c000003e syscall=54 success=yes exit=0 a0=3 a1=0 a2=40 a3=5652315d07c0 item… kern kernel
notice 2018-10-02 21:53:22 audit: type=1325 audit(1538510002.685:2436): table=filter family=2 entries=33 kern kernel
notice 2018-10-02 21:53:22 PROCTITLE proctitle=69707461626C65732D726573746F7265002D2D6E6F666C757368002D2D636F756E74657273 auth audit
notice 2018-10-02 21:53:22 SYSCALL arch=c000003e syscall=54 success=yes exit=0 a0=3 a1=0 a2=40 a3=5652315d07c0 items=0 ppid=3823 pid=26447 auid=42949672… auth 26447 audit
notice 2018-10-02 21:53:22 NETFILTER_CFG table=filter family=2 entries=33 auth audit
err 2018-10-02 21:52:53 rsyslogd: omfwd: TCPSendBuf error -2027, destruct TCP Connection to 127.0.0.1:6514 [v8.36.0 try http://www.rsyslog.com/e/2027… user 2398 logging/telegraf-...
@leodido


Explore
Queries
Visualization
Write Data
Send to Dashboard
Dynamic Source
Flux
InfluxQL
CSV
Past 1h
time
syslog.message
syslog.label_compone
10/02/2018 11:58:09
I1002 09:58:09.074356 1 leaderelection.go:175] attempting to acquire leader lease kube-system/kube-controller-manager...
kube-controller-manag
10/02/2018 11:58:25
I1002 09:58:25.289555 1 controller_utils.go:1019] Waiting for caches to sync for bootstrap_signer controller
kube-controller-manag
10/02/2018 11:58:25
I1002 09:58:25.552953 1 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for {extensions deployments}
kube-controller-manag
10/02/2018 11:58:25
I1002 09:58:25.554906 1 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for {batch jobs}
kube-controller-manag
10/02/2018 11:58:25
W1002 09:58:25.554969 1 shared_informer.go:311] resyncPeriod 55346017222036 is smaller than resyncCheckPeriod 65462554351374 and the informer has already started. Changing it to 65462554351374
kube-controller-manag
10/02/2018 11:58:25
I1002 09:58:25.556401 1 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for {extensions replicasets}
kube-controller-manag
10/02/2018 11:58:25
I1002 09:58:25.558071 1 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for {apps statefulsets}
kube-controller-manag
SELECT "message", "label_component" FROM "tele...
SELECT "message", "label_component" FROM "telegraf"."autogen"."syslog" WHERE time > :dashboardTime: AND "appname"='kube-system/kube-controller-manager-minikube'
Success!
Show Template Values
Metaquery Templates
Submit Query
DB.RetentionPolicy
_internal.monitor
telegraf.autogen
Measurements & Tags
Filter
syslog
=
appname — 11
Group By appname
Filter within appname
audit
dockerd
kernel
kube-system/etcd-minikube
kube-system/kube-addon-manager-minikube
kube-system/kube-controller-manager-minikube
kubelet
logging/chronograf-5c85dd45f6-s7rgn
logging/telegraf-j9vdw
rsyslogd
sshd
Fields
facility_code
id_container
id_namespace
id_pod
label_component
0 Functions
label_controller-revision-hash
message
0 Functions
procid
severity_code
timestamp
version

Logs are just a mess!

Inspecting logs coming from a single server is easy.
Inspecting logs coming from a distributed system is hard.

We need metrics!

—

Now we want to detect and count the OOMs.

Logs are streams.
We need a streaming processor!

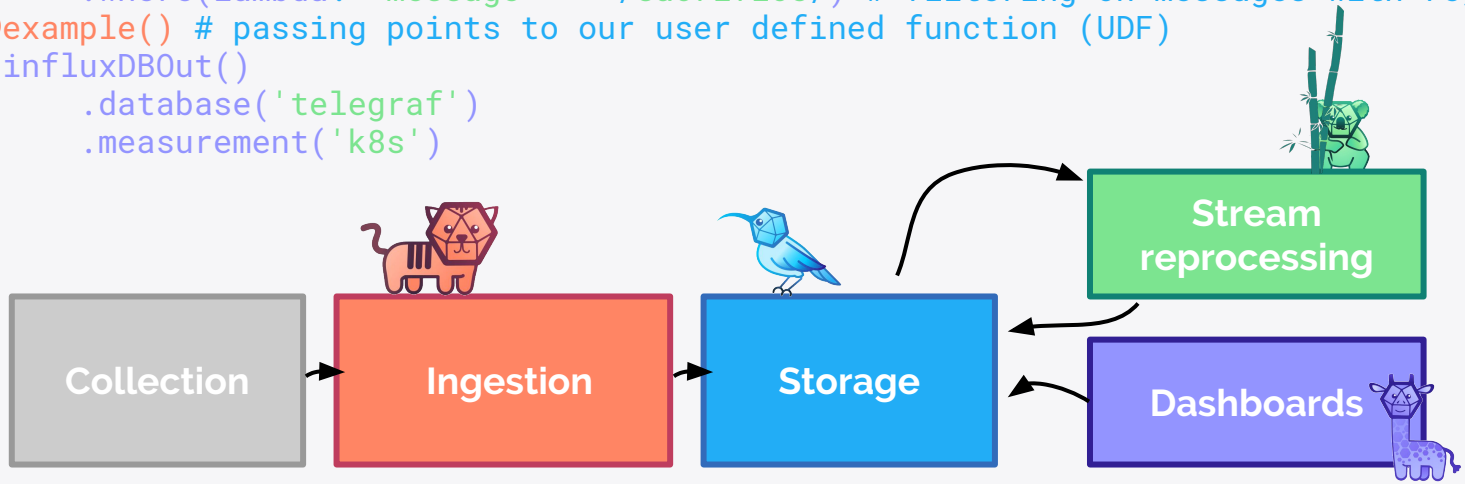
github.com/influxdata/kapacitor

A streaming processor can be programmed to identify the patterns we want and act on them, e.g: OOM Kills.

```
Memory cgroup out of memory: Kill process 13012 (stress) score 1958 or sacrifice child
```

Let's write a tick script to grab log points

```
dbrp "telegraf"."autogen"
stream
     |from()
          .measurement('syslog')
          .truncate(1ms)
          .where(lambda: "appname" == 'kernel') # filter by points tag
          .where(lambda: "message" =~ /sacrifice/) # filtering on messages with regex
     @example() # passing points to our user defined function (UDF)
     |influxDBOut()
          .database('telegraf')
          .measurement('k8s')
```



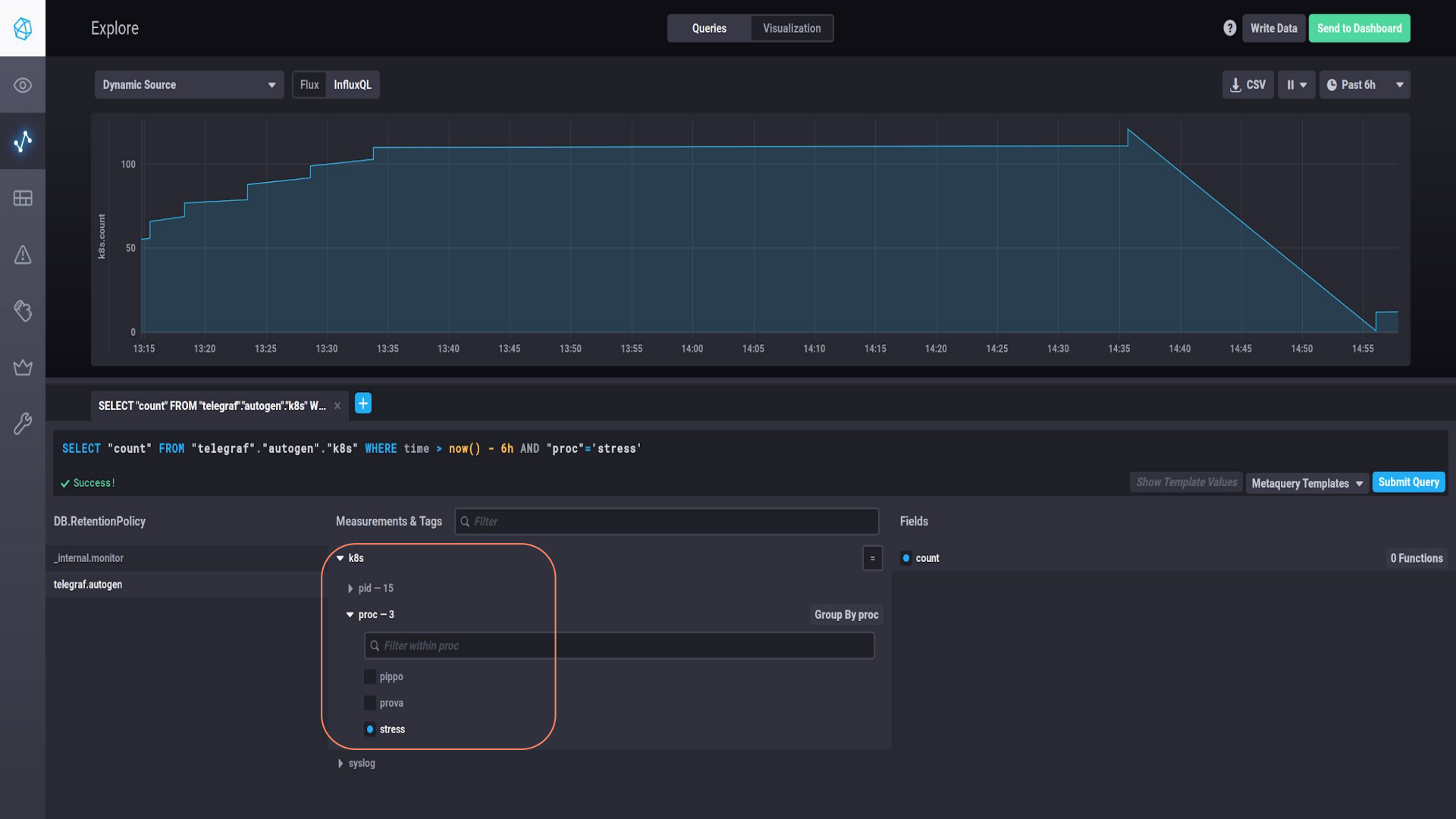
Let’s configure kapacitor

```
# …
[udf]
[udf.functions]
   [udf.functions.example]
     socket = "/tmp/example.sock"
     timeout = "10s"

[[influxdb]]
 enabled = true
 default = true
 name = "logging"
 urls = ["http://localhost:8086"]
 timeout = 0
 startup-timeout = "5m"

 [influxdb.subscriptions]
   telegraf = ["autogen"]
```

Let's write the UDF

```
func (h *handler) Point(p *agent.Point) error {
    var r = regexp.MustCompile(`(?m).*Kill process (?P<pid>\d+) (?P<proc>\(.*\)) score (?P<score>\d+)`)
    message, ok := p.FieldsString["message"]
    if ok {
        m := r.FindStringSubmatch(message)
        data := mapSubexpNames(m, r.SubexpNames())
        proc := strings.Trim(data["proc"], "()")
        state := h.state[proc]
        if state == nil {
            state := &myState{Counter: 0}
            h.state[proc] = state
        }
        h.state[proc].update()
        newpoint := &agent.Point{
            Time: time.Now().UnixNano(),
            Tags: map[string]string{
                "proc": proc,
                "pid":  string(data["pid"]),
            },
            FieldsInt: map[string]int64{
                "count": h.state[proc].Counter,
            },
        }
        // Send point
        h.agent.Responses <- &agent.Response{
            Message: &agent.Response_Point{
                Point: newpoint,
            },
        }
    }
    return nil
}
```


OOM monitor
Variables
Annotations
Past 1h
stacked
now() - 1h
count(oom)
120
100
80
60
40
20
14:10
14:20
14:25
14:30
14:35
14:40
14:45
14:50
14:55
15:00
15:05
2018/10/04 14:56:14.319
k8s.max_count[proc=pippo] 132
k8s.max_count[proc=prova] 72
k8s.max_count[proc=stress] 12
Untitled Graph
100
150
50
200
0
250
24
Untitled Graph
72.00
Untitled Graph
132.00
@leodido


Explore
Queries
Visualization
Write Data
Send to Dashboard
Dynamic Source
Flux
InfluxQL
CSV
Past 6h
k8s.count
100
50
0
13:15
13:20
13:25
13:30
13:35
13:40
13:45
13:50
13:55
14:00
14:05
14:10
14:15
14:20
14:25
14:30
14:35
14:40
14:45
14:50
14:55
SELECT "count" FROM "telegraf"."autogen"."k8s" W...
SELECT "count" FROM "telegraf"."autogen"."k8s" WHERE time > now() - 6h AND "proc"='stress'
Success!
Show Template Values
Metaquery Templates
Submit Query
DB.RetentionPolicy
Measurements & Tags
Filter
Fields
_internal.monitor
telegraf.autogen
k8s
=
count
0 Functions
pid — 15
proc — 3
Group By proc
Filter within proc
pippo
prova
stress
syslog

Thanks.

@leodido
git.io/k8s-logs-to-metrics-tick
git.io/go-syslog
github.com/influxdata