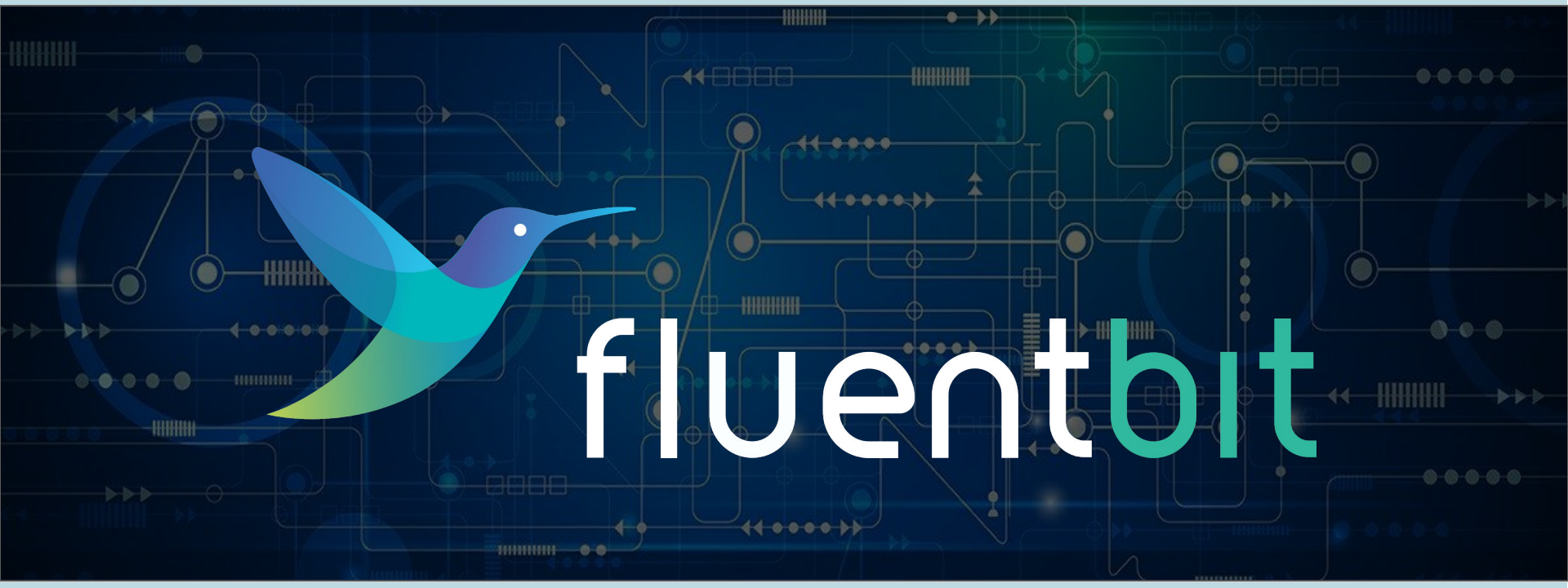


Eduardo Silva <eduardo@treasure-data.com>

@edsiper / Principal Engineer at **Arm**

arm
TREASURE DATA


Logging

Logging Challenges

Multiple Sources of Information

- TCP / UDP
- File system, common log files
- Systemd / Journald
- Sensors !

arm TREASURE DATA

Logging Challenges

.. and each one with different data formats

- Apache Logs

  [14/Mar/2019:23:43:52 +0000] GET /Frasera HTTP/1.0 500 2216

- MySQL

  2019-04-30T21:32:39.095880Z 0 [Note] InnoDB: Mutexes use GCC atomic builtins

- JSON Maps

  {“log”: “Hey GEC!”, “stream”: “stdout”, “time”: “2019-05-07T10:03:11.33507113Z”}

- Many others...!

arm
TREASURE DATA

Structured Messages

Unstructured to Structured

Hey LinuxDev Brazil!

```
{
    "log": "Hey LinuxDev Brazil!",
    "stream": "stdout",
    "time": "2019-08-03T17:03:11.33507113Z"
}
```

Structured Messages

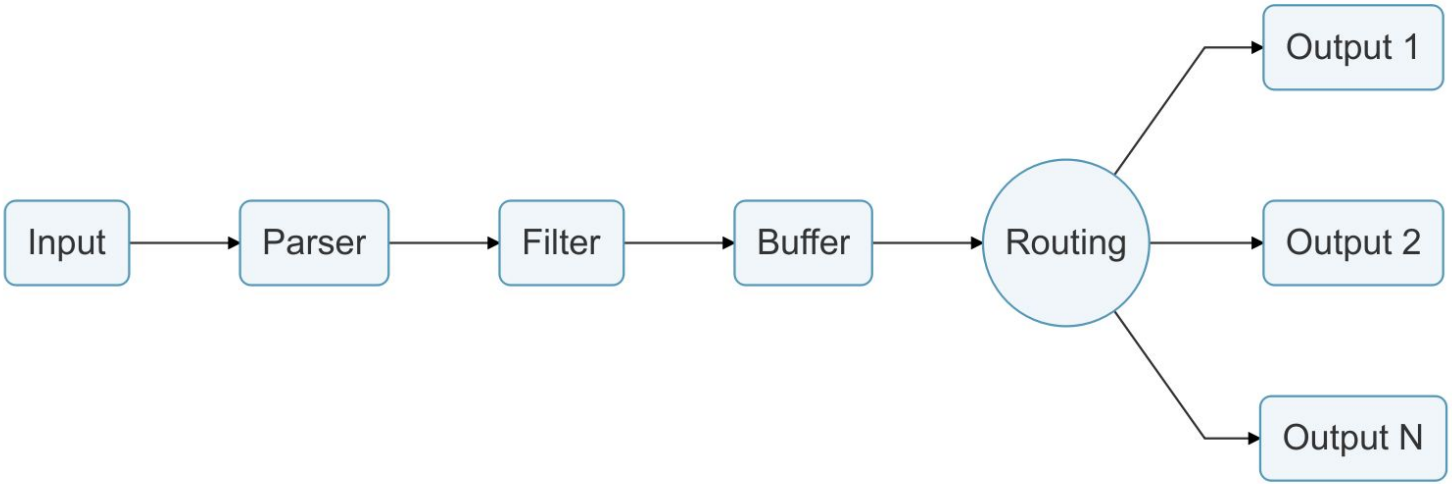
Metadata

```
{
    "log": "Hey LinuxDev Brazil!",
    "stream": "stdout",
    "time": "2019-08-03T17:03:11.33507113Z"
}
```

Structured Messages

Metadata

```
{
    "log": "Hey LinuxDev Brazil!",
    "stream": "stdout",
    "time": "2019-08-03T17:03:11.33507113Z",
    "kubernetes": {
        "host": "minikube",
        "pod_name": "linuxdevbr",
        "pod_id": "c76927af-f563-11e4-b32d-54ee1227188d",
        "container_name": "linuxdevbr",
        "namespace_name": "default",
        "namespace_id": "23437884-8e08-4d95-850b-e94378c9b2fd"
    }
}
```




fluentbit

About

Fluent Bit

- **Fluentd** sub-project
- Started in 2015
- Origins: Lightweight log processor for Embedded Linux
- Quickly evolved as a solution for the Cloud space
- Apache License v2.0

Fluent Bit

Design & Internals

- Written in **C** language
- **Low** memory and CPU footprint (memory around **500KB**)
- Pluggable Architecture (**~50** plugins available)
- Built-in security: TLS on Network I/O

Logging: basics

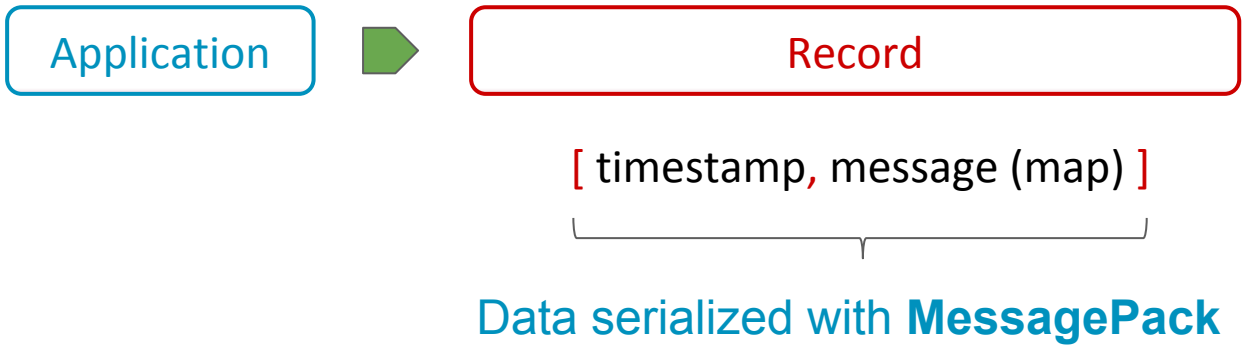
- Application generates a message: **record**

Logging: basics

- Application generates a message: **record**
- Record is appended with metadata: **timestamp**

Logging: basics

- Application generates a message: **record**
- Record is appended with metadata: **timestamp**
- Record is **serialized** and ready for processing



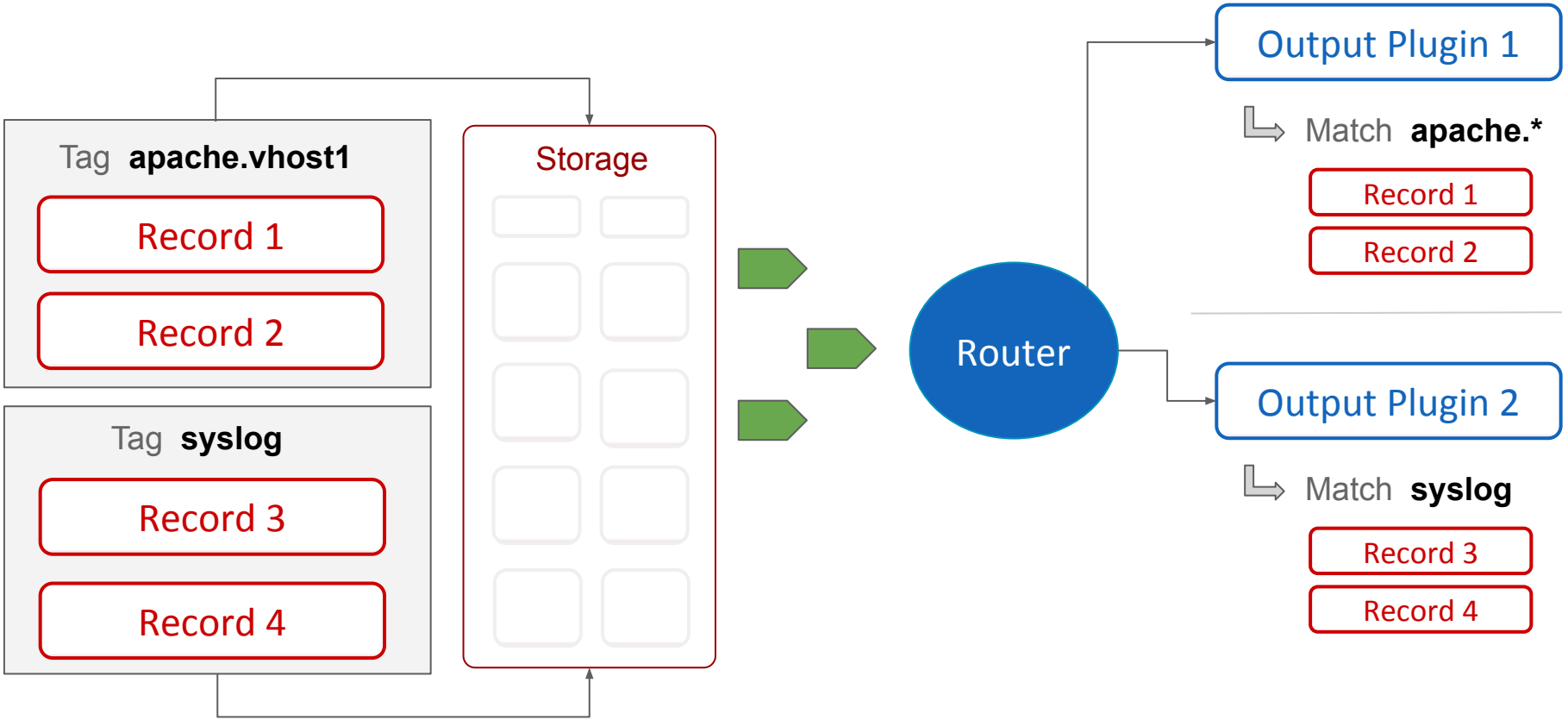
Logging Handling

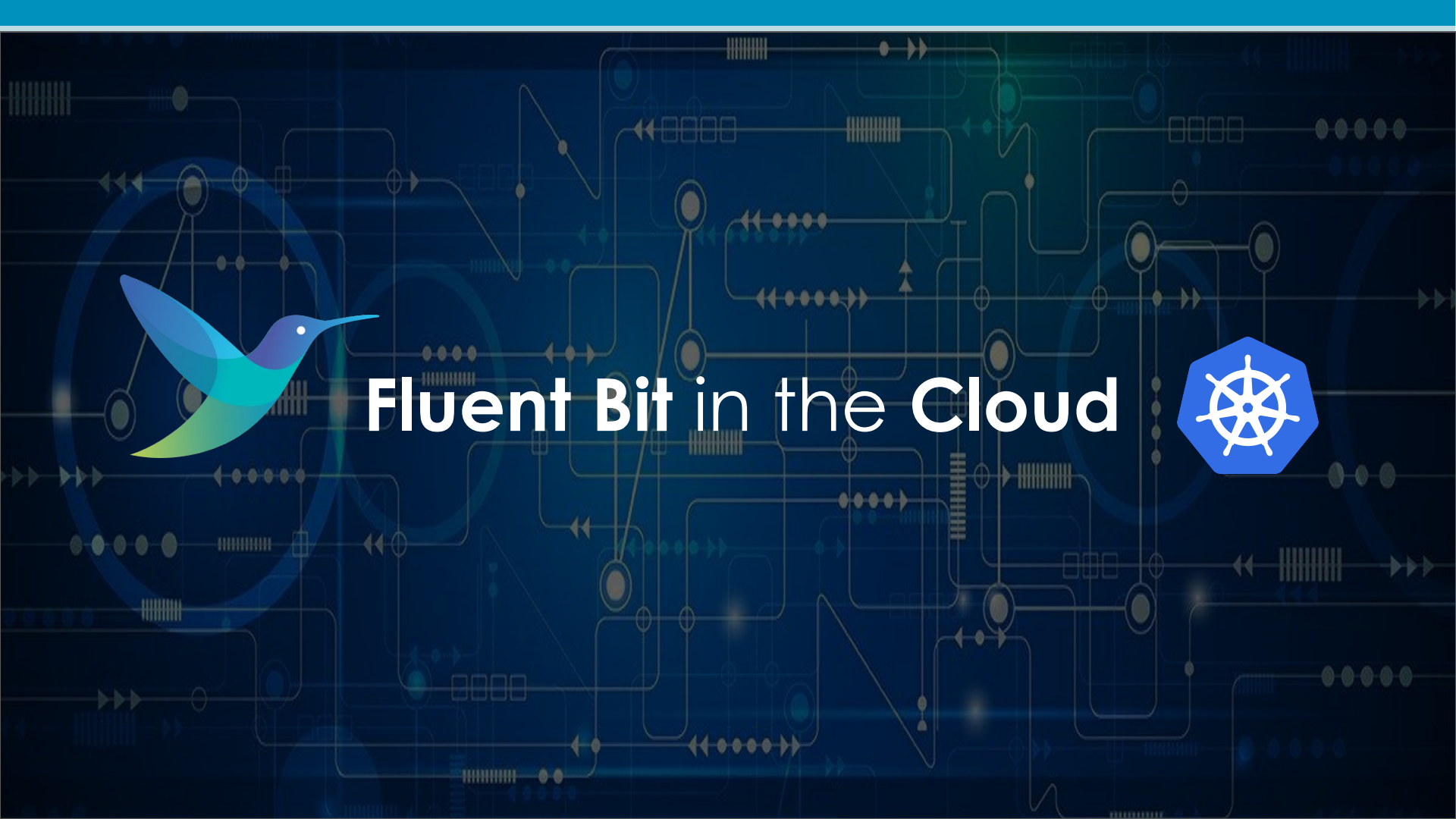
Workflow

[timestamp, message (map)]

Data serialized with **MessagePack**

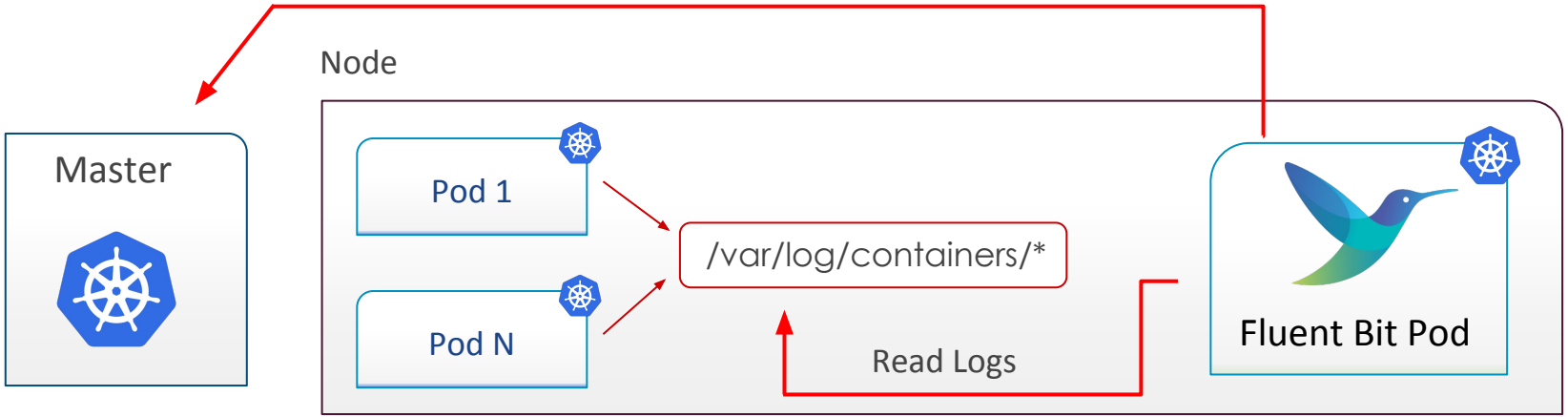
Logging & Routing




Fluent Bit in the Cloud

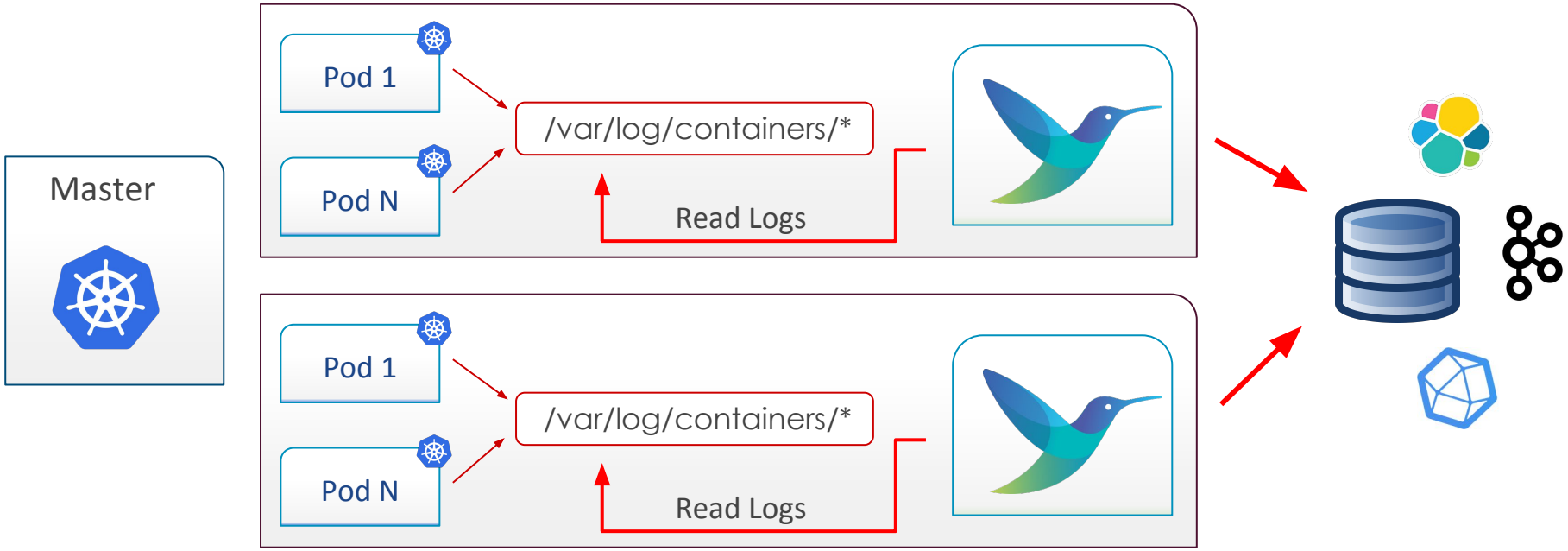
Logging Processing in Kubernetes

Read Logs from the Filesystem or Journald



Logging Processing in Kubernetes

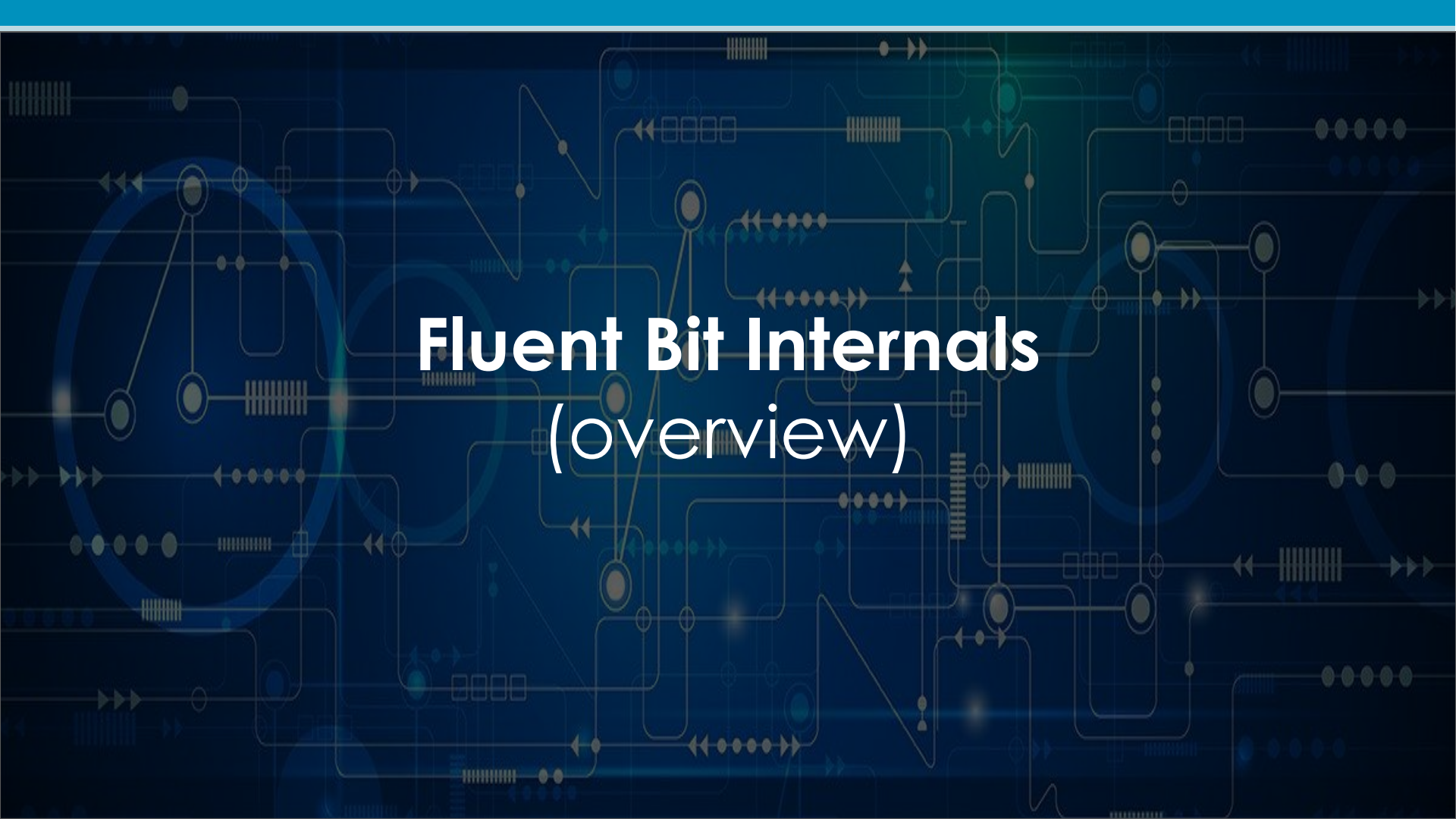
Read Logs from the Filesystem or Journald



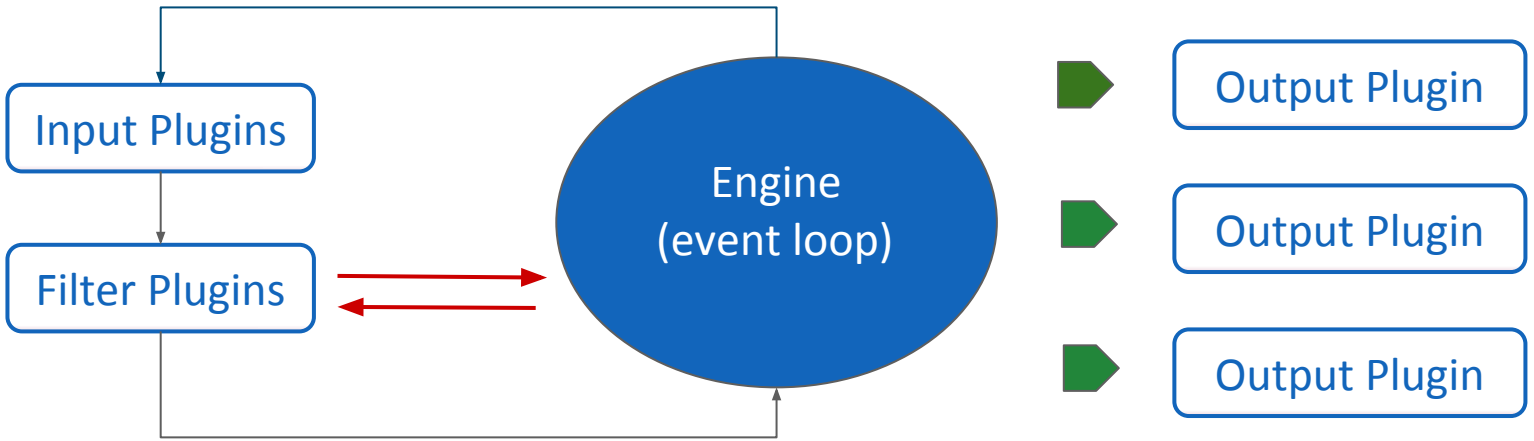
Fluent Bit Adoption

General info

- > **200.000** deployments EVERY SINGLE DAY
- Wide Adoption
  - AWS
  - Google Cloud Platform
  - DataDog (coming soon!)


Fluent Bit Internals
(overview)

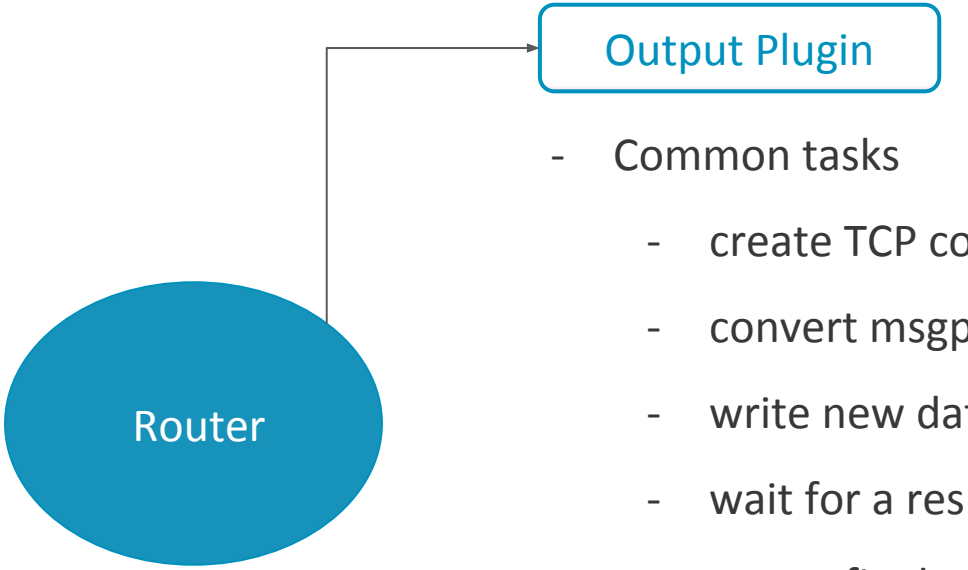
Internals


Input Plugins
Filter Plugins
Engine
(event loop)
Output Plugin
Output Plugin
Output Plugin

Internals / Output Plugins

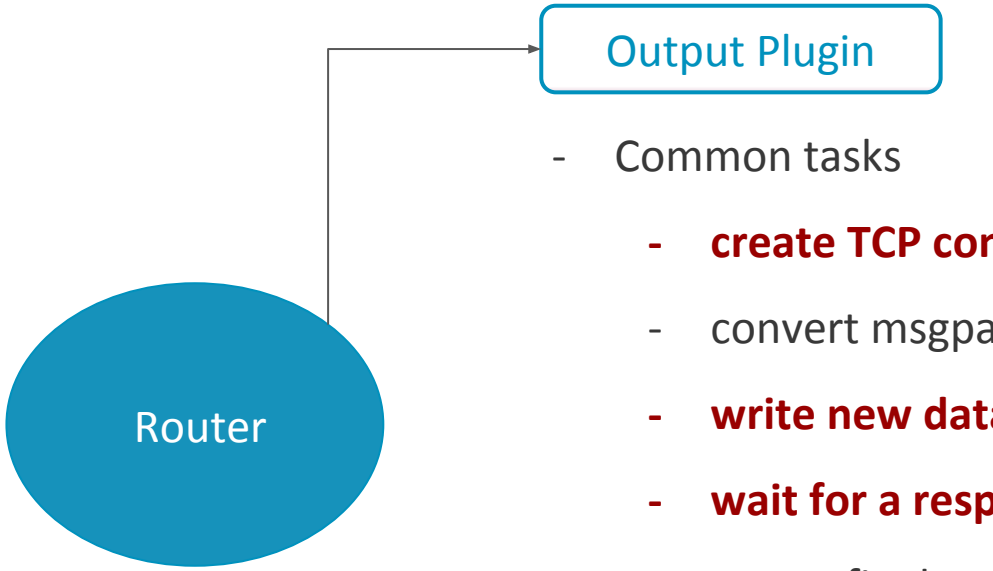
- Most of output plugins relies on Network I/O
- Simple design to avoid callbacks hell
- Reduce blocking time when possible: suspend and resume

Internals / Output Plugins



- Common tasks
  - create TCP connection
  - convert msgpack data to outgoing format
  - write new data over the network
  - wait for a response (most of cases)
  - report final status (OK, RETRY, ERROR)

Internals / Output Plugins



- Common tasks
  - **create TCP connection**
  - convert msgpack data to outgoing format
  - **write new data over the network**
  - **wait for a response (most of cases)**
  - report final status (OK, RETRY, ERROR)

Example

```
1   static void cb_es_flush(...)
2   {
3       int ret;
4       char *pack;
5       size_t b_sent;
6
7       /* Get upstream connection */
8       u_conn = flb_upstream_conn_get(ctx->u);
9       if (!u_conn) {
10          FLB_OUTPUT_RETURN(FLB_RETRY);
11      }
12
13      /* Convert format */
14      pack = elasticsearch_format(data, bytes, ...);
15
16      /* Compose HTTP Client request */
17      c = flb_http_client(u_conn, FLB_HTTP_POST, ctx->uri,
18                          pack, bytes_out, NULL, 0, NULL, 0);
19
20      /* Issue HTTP request */
21      ret = flb_http_do(c, &b_sent);
22
23      /* Cleanup */
24      flb_free(pack);
25      FLB_OUTPUT_RETURN(FLB_OK);
26  }
```

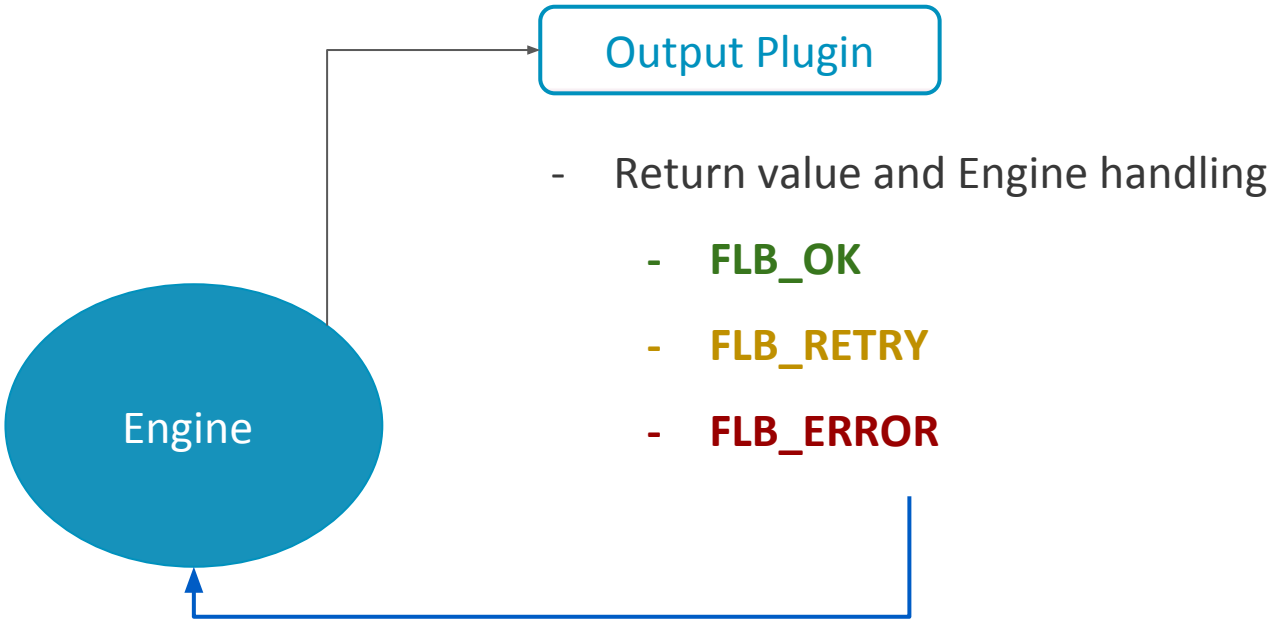
Example

```
1   static void cb_es_flush(...)
2   {
3       int ret;
4       char *pack;
5       size_t b_sent;
6
7       /* Get upstream connection */
8       u_conn = flb_upstream_conn_get(ctx->u);
9       if (!u_conn) {
10          FLB_OUTPUT_RETURN(FLB_RETRY);
11      }
12
13      /* Convert format */
14      pack = elasticsearch_format(data, bytes, ...);
15
16      /* Compose HTTP Client request */
17      c = flb_http_client(u_conn, FLB_HTTP_POST, ctx->uri,
18                          pack, bytes_out, NULL, 0, NULL, 0);
19
20      /* Issue HTTP request */
21      ret = flb_http_do(c, &b_sent);
22
23      /* Cleanup */
24      flb_free(pack);
25      FLB_OUTPUT_RETURN(FLB_OK);
26  }
```

suspend / resume

suspend / resume

Output Plugins: return values & retry logic



- Return value and Engine handling
  - **FLB_OK**
  - **FLB_RETRY**
  - **FLB_ERROR**

Fluent Bit & Plugin Helpers

- Upstream (TCP/TLS connection handling)
- HTTP Client
- OAuth2
- Timers
- Crypto (mbedTLS)
- Lua (LuaJIT)
- ...

Fluent Bit: Plugins

Input Plugins

- **tail**
- kmsg
- serial
- **systemd**
- syslog (tcp/udp)
- cpu, mem, disk
- ...

Filter Plugins

- grep
- throttle
- parser
- **kubernetes**
- **lua**
- nest
- ...

Output Plugins

- treasure data
- http
- elasticsearch
- splunk
- azure
- kafka
- ...

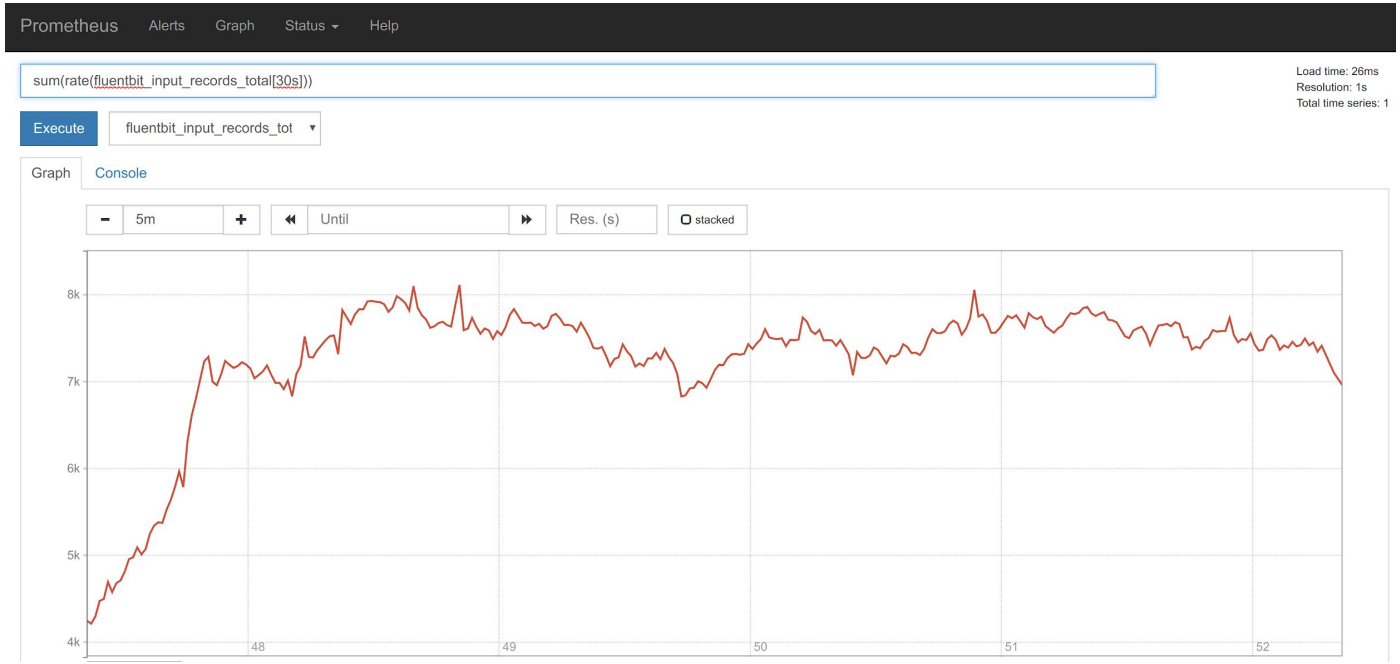
Fluent Bit & Filtering

Optional filtering with Lua !

```
function cb_replace(tag, timestamp, record)
      new_record = {}
      new_record["new"] = 12345
      new_record["old"] = record
      return 1, timestamp, new_record
end
```

Fluent Bit & Monitoring

Metrics and Prometheus Support



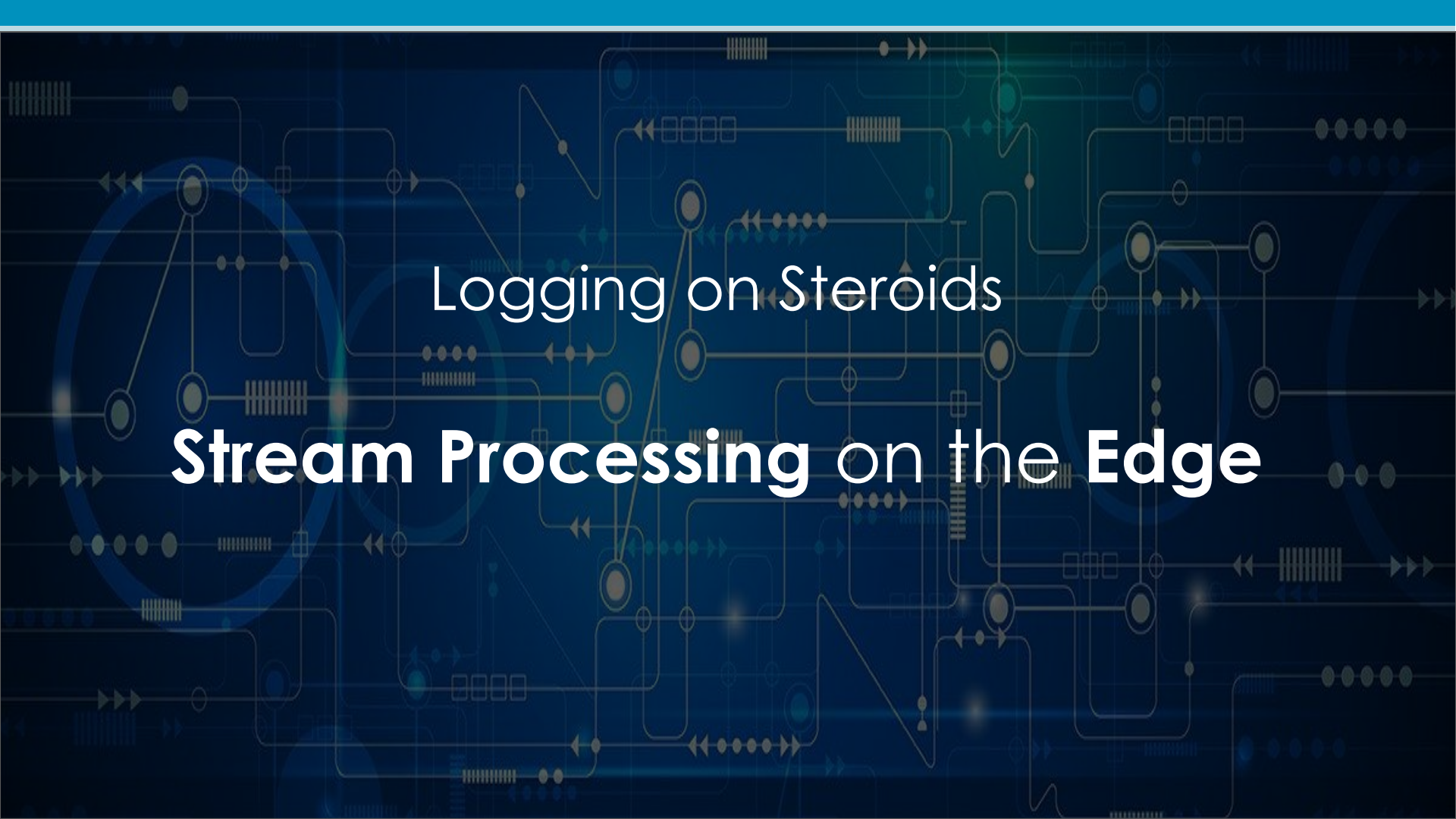
Thanks!

Eduardo Silva <eduardo@treasure-data.com>
@edsiper

fluentbit.io

fluent/**fluent-bit**

arm
TREASURE DATA


Logging on Steroids
Stream Processing on the Edge

What is Stream Processing ?

Stream Processing

“ It’s the ability to perform

data processing while **it** still in motion ”

Stream Processing

“ It’s the ability to perform

data processing while **it** still in motion ”

but on the **Edge**

Stream Processing on the Edge

Give me a stronger reason!

PERFORMANCE

arm
TREASURE DATA

Logging pains

Performance penalties are in many places

Access to File System Data

Data Parsing

Database Indexing

arm TREASURE DATA

Logging & Performance

Performance penalties are in many places

Access to File System Data

Data Parsing

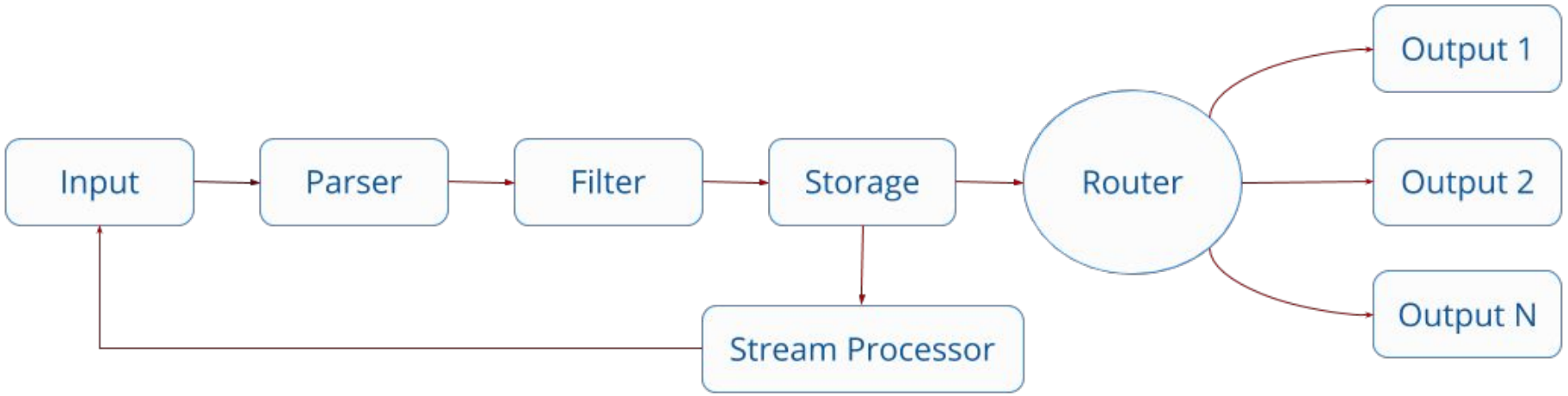
Database Indexing

Fluent Bit v1.2

Logging is challenging, data processing even more!

Stream Processor

Stream Processing



arm

Fluent Bit + Stream Processing

Vision

- Input plugins generate a stream of data
- Right after storage phase, we do optional Stream Processing
- Perform **SQL** queries
- Create **new streams** using results from previous queries

arm

Fluent Bit + Stream Processing

SQL: SELECT statement

- **All keys selection**

```
SELECT * FROM STREAM:apache;
```

- **Keys selection**

```
SELECT host, status, size FROM STREAM:apache;
```

arm

Fluent Bit + Stream Processing

SQL: Aggregation functions

- Supported **aggregation** functions
  - COUNT()
  - SUM()
  - MAX()
  - MIN()
  - AVG()

arm

Fluent Bit + Stream Processing

WINDOWING

```
SELECT
    device_id,
    AVG(temp)
FROM
    STREAM:devices WINDOW TUMBLING (5 second)
GROUP BY
    device_id
```

arm

Fluent Bit + Stream Processing

SQL: Streams creations

- Create a stream using results of a query

```
CREATE STREAM events AS SELECT a, b, c FROM STREAM:apache;
```

- Tag stream for Fluent Bit data pipeline

```
CREATE STREAM events WITH (tag='myevents') AS SELECT a, b, c FROM STREAM:apache;
```

arm

Thanks #2 !

Eduardo Silva <eduardo@treasure-data.com>
@edsiper

fluentbit.io

fluent/**fluent-bit**

arm
TREASURE DATA