Fluent Bit Open Source Project & AWS
Logging with Data Insights

Eduardo Silva
eduardo@treasure-data.com
@edsiper / Principal Engineer at Arm

Wesley Pettit
wppttt@amazon.com
@TheWesleyPettit / SW Engineer at Amazon
Data is Everywhere
Data Analysis

Gather Insights
Data Analysis

Gather Insights

Extract Value

Analysis
Challenges
Data Challenges

Multiple Sources of Information

- TCP / UDP
- File system, common log files
- Systemd / Journald
- Sensors
- Remote Hardware
End to End

Communication Workflow

HW / SW → Data / Event / Log → DB → Analysis
Data Ingestion

Performance Penalties

HW / SW -> Data / Event / Log -> Analysis

HW / SW -> Data / Event / Log

HW / SW -> Data / Event / Log
Logging
Logging Challenges

Multiple Sources of Information

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

- **JSON Maps**
  
  `{"log": "Hey GEC!", "stream": "stdout", "time": "2019-05-07T10:03:11.33507113Z"}`

- **Many others…!**
Hey KubeCon!

```json
{
  "log": "Hey KubeCon!",
  "stream": "stdout",
  "time": "2019-11-18T14:03:11.33507113Z"
}
```
Structured Messages

Metadata

```json
{
    "log": "Hey KubeCon!",
    "stream": "stdout",
    "time": "2019-11-18T14:03:11.33507113Z"
}
```
Structured Messages

Metadata

```json
{
  "log": "Hey KubeCon!",
  "stream": "stdout",
  "time": "2019-11-18T14:03:11.33507113Z",
  "kubernetes": {
    "host": "minikube",
    "pod_name": "kubecon",
    "pod_id": "c76927af-f563-11e4-b32d-54ee1227188d",
    "container_name": "kubecon",
    "namespace_name": "default",
    "namespace_id": "53437884-8e08-4d95-850b-e94378c9b2fd"
  }
}
```
About

Fluent Bit

- 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`

```
[ timestamp, message (map) ]
```
Logging: basics

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

Application | Record
[timestamp, message (map)]

Data serialized with MessagePack
Logging Handling

Workflow

Application → Log → Record

[ timestamp, message (map) ]

Data serialized with MessagePack
Logging & Routing

Tag `apache.vhost1`
- Record 1
- Record 2

Tag `syslog`
- Record 3
- Record 4

Routing:
- Output Plugin 1
  - Match `apache.*`
  - Record 1
  - Record 2

- Output Plugin 2
  - Match `syslog`
  - Record 3
  - Record 4

Storage
Fluent Bit Adoption

General info

- >250,000 deployments EVERY SINGLE DAY

- Wide Adoption
  - AWS
  - Google Cloud Platform
  - DataDog
Logging Processing in Kubernetes

Read Logs from the Filesystem or Journald

Node

Master

Pod 1

Pod N

/\var/log/containers/*

Read Logs

Fluent Bit Pod
Logging Processing in Kubernetes

Read Logs from the Filesystem or Journald

Node

Master

Pod 1

Pod N

/var/log/containers/*

Read Logs

Fluent Bit Pod
Logging Processing in Kubernetes

Read Logs from the Filesystem or Journald

/var/log/containers/*

Pod 1
Pod N

Pod 1
Pod N

Read Logs

Master

/aws

Database
Fluent Bit & AWS
Amazon/aws-for-fluent-bit

Output Logs to:
• CloudWatch Logs
• Kinesis Data Firehose
• Kinesis Data Streams (coming soon)
• All destinations supported natively in Fluent Bit
Regional Amazon ECR Repositories

Pull the image from within your region, highly available, save on data transfer

<table>
<thead>
<tr>
<th>Region</th>
<th>Registry ID</th>
<th>Full Image URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>us-east-1</td>
<td>906394416424</td>
<td>906394416424.dkr.ecr.us-east-1.amazonaws.com/aws-for-fluent-bit:latest</td>
</tr>
<tr>
<td>eu-west-1</td>
<td>906394416424</td>
<td>906394416424.dkr.ecr.eu-west-1.amazonaws.com/aws-for-fluent-bit:latest</td>
</tr>
<tr>
<td>us-west-1</td>
<td>906394416424</td>
<td>906394416424.dkr.ecr.us-west-1.amazonaws.com/aws-for-fluent-bit:latest</td>
</tr>
</tbody>
</table>
FireLens for Amazon ECS

- An AWS “managed” experience that makes it easy to use Fluentd and Fluent Bit
- (Don’t boo me) Not currently available for Kubernetes.
What should we work on?

- Github.com/aws/aws-for-fluent-bit
- Github.com/aws/containers-roadmap
- Tweet at me - @TheWesleyPettit
Tutorials and more

Check out the AWS blogs for interesting posts on Fluentd and Fluent Bit:
• AWS Containers Blog
• AWS Open Source Blog
• AWS Compute Blog
Centralized Container Logging with Fluent Bit

by Michael Hausenblas | on 09 JUL 2019 | in Amazon Athena, Amazon CloudWatch, Amazon EC2 Container Registry, Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, Amazon Kinesis, Amazon Simple Storage Services (S3), Open Source | Permalink | Comments | Share

by Wesley Pettit and Michael Hausenblas

AWS is built for builders. Builders are always looking for ways to optimize, and this applies to application logging. Not all logs are of equal importance. Some require real-time analytics, others
Logging on Steroids

Stream Processing on the Edge
Fluent Bit

Logging is challenging, data processing even more!
HW / SW

Log Events Collector
(aggregator)

Stream Processor

Analysis

DB

Edge

Cloud
Stream Processor

Stream Processing

- Input
- Parser
- Filter
- Storage
- Router

Output 1
Output 2
Output N

Stream Processor
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
Fluent Bit + Stream Processing

SQL: SELECT statement

- All keys selection
  ```
  SELECT * FROM STREAM:apache;
  ```

- Keys selection
  ```
  SELECT host, status, size FROM STREAM:apache;
  ```
Fluent Bit + Stream Processing

SQL: Aggregation functions

- Supported aggregation functions
  - COUNT()
  - SUM()
  - MAX()
  - MIN()
  - AVG()
SELECT
   device_id,
   AVG(temp)
FROM
   STREAM:devices WINDOW TUMBLING (5 second)
GROUP BY
   device_id
Fluent Bit + Stream Processing

SQL: Streams creations

- Create a stream using results of a query
  ```sql
  CREATE STREAM events AS SELECT a, b, c FROM STREAM:apache;
  ```

- Tag stream for Fluent Bit data pipeline
  ```sql
  CREATE STREAM events WITH (tag='myevents') AS SELECT a, b, c FROM STREAM:apache;
  ```
Roadmap

Where are we going: v1.4 / January 2020

- Core
  - Config Maps
  - Old charset encoding to UTF-8
- Stream Processor
  - Snapshots
  - Machine Learning Algorithms
- Performance improvements and more connectors!
Questions?
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

Router

Output Plugin

- 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)
static void cb_es_flush(...) {
    int ret;
    char *pack;
    size_t b_sent;

    /* Get upstream connection */
    u_conn = flb_upstream_conn_get(ctx->u);
    if (!u_conn) {
        FLB_OUTPUT_RETURN(FLB_RETRY);
    }

    /* Convert format */
    pack = Elasticsearch_format(data, bytes, ...);

    /* Compose HTTP Client request */
    c = flb_http_client(u_conn, FLB_HTTP_POST, ctx->uri,
                          pack, bytes_out, NULL, 0, NULL, 0);

    /* Issue HTTP request */
    ret = flb_http_do(c, &b_sent);

    /* Cleanup */
    flb_free(pack);
    FLB_OUTPUT_RETURN(FLB_OK);
}
Example

```c
static void cb_es_flush(...)
{
    int ret;
    char *pack;
    size_t b_sent;

    /* Get upstream connection */
    u_conn = flb_upstream_conn_get(ctx->u);
    if (!u_conn) {
        FLB_OUTPUT_RETURN(FLB_RETRY);
    }

    /* Convert format */
    pack = elasticsearch_format(data, bytes, ...);

    /* Compose HTTP Client request */
    c = flb_http_client(u_conn, FLB_HTTP_POST, ctx->uri,
                        pack, bytes_out, NULL, 0, NULL, 0);

    /* Issue HTTP request */
    ret = flb_http_do(c, &b_sent);

    /* Cleanup */
    flb_free(pack);
    FLB_OUTPUT_RETURN(FLB_OK);
}
```
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

<table>
<thead>
<tr>
<th>Input Plugins</th>
<th>Filter Plugins</th>
<th>Output Plugins</th>
</tr>
</thead>
<tbody>
<tr>
<td>tail</td>
<td>grep</td>
<td>treasure data</td>
</tr>
<tr>
<td>kmsg</td>
<td>throttle</td>
<td>http</td>
</tr>
<tr>
<td>serial</td>
<td>parser</td>
<td>elasticsearch</td>
</tr>
<tr>
<td>systemd</td>
<td>kubernetes</td>
<td>splunk</td>
</tr>
<tr>
<td>syslog (tcp/udp)</td>
<td>lua</td>
<td>azure</td>
</tr>
<tr>
<td>cpu, mem, disk</td>
<td>nest</td>
<td>kafka</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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