● Open Source Engineer at Treasure Data

● Repositories / Projects
  ○ github.com/edsiper
  ○ fluentbit.io
  ○ duda.io
  ○ monkey-project.com
“Logging is Simple”
“Logging is Simple”
“Logging is Simple”

Logging exists because of Analysis needs
Someone have to do some work
In a galaxy not so far away...
Analysis
Internally, Logging is not Simple
Scale Logging Requires Understanding
Logging Pipeline
Logging Pipeline

- Log files
- Journald
- TCP

Routing
- Elasticsearch
- InfluxDB
- Others
Logging Pipeline

Format / Structure:

- JSON
- Regex / Named Capture
Logging Pipeline

- Log Messages
- Parse
- Filter
- Buffer
- Routing

Alter content: Grep, Exclude, Metadata

- Elasticsearch
- InfluxDB
- Others
Logging Pipeline

- Log Messages
- Parse
- Filter
- Buffer
- Elasticsearch
- InfluxDB
- Others
- Memory / Filesystem
- Routing
Logging Pipeline

- Log Messages
- Parse
- Filter
- Buffer

Routing

- Elasticsearch
- InfluxDB
- Others

Deliver buffers to N destinations
Logging Pipeline

Log Messages -> Parse -> Filter -> Buffer

Routing:
- Elasticsearch
- InfluxDB
- Others
● How to deal with the Logging Pipeline?

● Is there any solution around?
About Fluentd

- Created by
  
  TREASURE DATA

- Now hosted at
  
  CLOUD NATIVE COMPUTING FOUNDATION
About Fluentd

- More than 600 plugins available
- Pluggable Architecture
- Built-in Reliability
- Full integration with Docker and Kubernetes
- Written in Ruby + C
Fluentd Modes

- Log Forwarder
- Log Aggregator
Log Aggregator = (Forwarder + Buffering Capabilities)
Edge Nodes / Forward to Aggregators

Node 1

- App
- DB

- App
- DB

- Forward to Aggregators
Edge Nodes & Costs
Edge Nodes & Costs

- Fluentd requires ~40MB as minimum
- Deploying a few hundred could be expensive
- Can we make Forward cheaper?
About Fluentd

- Written in C
- Pluggable Architecture
- Built-in Reliability
- Event Driven - Async I/O
Why Fluent Bit as a Forwarder

● Features
  ○ Input, Filter and Output Plugins
  ○ Built-in parsing support
  ○ Minimum memory required 450KB
Edge Nodes / Forward to Aggregators
Cheap Forwarding
Cloud Native Features

- Docker & Kubernetes Support
- Buffering fully controlled
  - pause() / resume() for input plugins
- Easy to containerize
  - Small memory footprint
  - No dependencies (all are built-in)
Hands on!

DEMO #1

Unstructured vs Structured data
Unstructured v/s Structured

Why

● Structured data have a schema
● Easy to convert to different representations
● It can be filtered
DEMO #2

Process Docker Logs
Kubernetes use case

- Applications runs in Containers
- Containers runs in a POD
- Multiple PODs can exists in a Node
- How to solve logging?
DEMO #3
Kubernetes: parse logs and append Metadata
Metadata Support Status

The new `kubernetes` filter takes care of the following metadata handling:

- Local data: POD Name, Namespace, Container Name and Container ID.
- Remote (API Server): Labels and Annotations
Networking and Co-routines

Easier implementation of output plugins that interact with networking operations like socket(), connect(), read(), write(), etc.

Fluent Bit provides non-blocking networking API that uses the event-loop with co-routines to implement:

- Network I/O
- TLS/SSL usage
- HTTP Client
Github Repository

- https://github.com/fluent/fluent-bit-kubernetes-daemonset

Docker Image (ubuntu-slim)

- quay.io/fluent/fluent-bit-kubernetes-daemonset
Roadmap

Next Release v0.11 (March 2017)

- Kubernetes support (filter_kubernetes)
- Parsers & Filters
- Memory optimizations

Release v0.12 (May 2017)

- in_tail + Multiline support
- Monitoring - re-enable HTTP service end-point: memory, records flow, others.
Thanks!

Project information

- Web site: Fluentbit.io
- Documentation: http://fluentbit.io/documentation/
- Github: http://github.com/fluent/fluent-bit

Contact

- Slack: http://slack.fluentd.org (fluent-bit channel)
- Twitter: @fluentbit