Fluentd forward protocol modes in v0

... and how to implement minimal Fluentd logger.

Hiroshi Hatake

Internal technical information sharing workshop

Introduction

What is Fluentd?

Fluentd is a fully free open-source log collector for unified logging layer.

Introduction

What is Fluentd?

Fluentd is a fully free open-source log collector for unified logging layer.

- Unified logging with JSON and msgpack

Introduction

What is Fluentd?

Fluentd is a fully free open-source log collector for unified logging layer.

- Unified logging with JSON and msgpack
- Pluggable architecture

Introduction

What is Fluentd?

Fluentd is a fully free open-source log collector for unified logging layer.

- Unified logging with JSON and msgpack
- Pluggable architecture
- Minimum resources required

Introduction

What is Fluentd?

Fluentd is a fully free open-source log collector for unified logging layer.

- Unified logging with JSON and msgpack
- Pluggable architecture
- Minimum resources required
- Built-in reliability

Forward protocol - Introduction

Forward protocol[1]

Fluentd has three modes in forward protocol.

- Message Mode

[1] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0

Forward protocol - Introduction

Forward protocol[1]

Fluentd has three modes in forward protocol.

- Message Mode
- Forward Mode

[1] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0

Forward protocol - Introduction

Forward protocol[1]

Fluentd has three modes in forward protocol.

- Message Mode
- Forward Mode
- PackedForward Mode

[1] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0

Message Mode protocol

Message Mode[2]

It carries just an event.

- **tag** is a string separated with '.' (e.g. myapp.access) to categorize events.
- **time** is a number of seconds since Unix epoch.
- **record** is key-value pairs of the event record.

[2] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#message-mode

Message Mode protocol

Message Mode[2]

It carries just an event.

- **tag** is a string separated with '.' (e.g. myapp.access) to categorize events.
- **time** is a number of seconds since Unix epoch.
- **record** is key-value pairs of the event record.
- **option** is optional key-value pairs, to bring data to control servers' behavior.

[2] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#message-mode

Message Mode protocol

Message Mode[3]

It carries just an event.

name	type	content	mandatory?
tag	str	tag name	yes
time	Integer/EventTime(ext type)	Unix Time/Unix Time with nano seconds	yes
record	Object(like HashMap)	pairs of keys(String) and values(Object)	yes
option	Object(like HashMap)	pairs of keys(String) and values(Object)	no

[3] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#message-mode

Forward Mode protocol

Forward Mode[4]

It carries a series of events as a msgpack array on a single request.

- **tag** is a string separated with '.' (e.g. myapp.access) to categorize events.
- **entries**[5] is an array of Entries which are pairs of Unix epoch and record.

[4] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#forward-mode

[5] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#entry

Forward Mode protocol

Forward Mode[4]

It carries a series of events as a msgpack array on a single request.

- **tag** is a string separated with '.' (e.g. myapp.access) to categorize events.
- **entries**[5] is an array of Entries which are pairs of Unix epoch and record.
- **option** is optional key-value pairs, to bring data to control servers' behavior.

---

[4] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#forward-mode

[5] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#entry

Forward Mode protocol

Forward Mode[6]

It carries a series of events as a msgpack array on a single request.

name	type	content	mandatory?
tag	str	tag name	yes
entries	Array	an array of Entries which are pairs of Unix epoch and record.	yes
option	Object(like HashMap)	pairs of keys(String) and values(Object)	no

[6] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#forward-mode

PackedForward Mode protocol

PackedForward Mode[7]

It carries a series of events as a msgpack binary on a single request.

- **tag** is a string separated with '.' (e.g. myapp.access) to categorize events.
- **entries** is a msgpack stream of Entry which contains pairs of Unix epoch and record msgpack binary.

[7] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#packedforward-mode

PackedForward Mode protocol

PackedForward Mode[7]

It carries a series of events as a msgpack binary on a single request.

- **tag** is a string separated with '.' (e.g. myapp.access) to categorize events.
- **entries** is a msgpack stream of Entry which contains pairs of Unix epoch and record msgpack binary.
- **option** is optional key-value pairs, to bring data to control servers' behavior.

[7] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#packedforward-mode

PackedForward Mode protocol

PackedForward Mode[8]

It carries a series of events as a msgpack binary on a single request.

name	type	content	mandatory?
tag	str	tag name	yes
entries	str — bin	a msgpack stream of Entry which contains pairs of Unix epoch and record.	yes
option	Object(like HashMap)	pairs of keys(String) and values(Object)	no

[8] https://github.com/fluent/fluentd/wiki/Forward-Protocol-Specification-v0#packedforward-mode

How to implement minimal Fluentd logger?

Minimal Fluentd logger implementation strategy

- **Fluentd logger** needs to send an event to Fluentd.

How to implement minimal Fluentd logger?

Minimal Fluentd logger implementation strategy

- **Fluentd logger** needs to send an event to Fluentd.
  - **Fluentd logger** need **not** to add optional elements in sending events to Fluentd.

How to implement minimal Fluentd logger?

Minimal Fluentd logger implementation strategy

- **Fluentd logger** needs to send an event to Fluentd.
  - **Fluentd logger** need **not** to add optional elements in sending events to Fluentd.
  - **Fluentd logger should** use message mode to send events to Fluentd.

How to implement minimal Fluentd logger?

Minimal Fluentd logger implementation

- **Fluentd logger** needs to connect to Fluentd with TCP.

How to implement minimal Fluentd logger?

Minimal Fluentd logger implementation

- **Fluentd logger** needs to connect to Fluentd with TCP.
- **Fluentd logger** send an event which contains tag, timespec, and record, to Fluentd.

How to implement minimal Fluentd logger?

Minimal Fluentd logger implementation

- **Fluentd logger** needs to connect to Fluentd with TCP.
- **Fluentd logger** send an event which contains tag, timespec, and record, to Fluentd.
- **Fluentd logger** should disconnect connection against Fluentd immediately when sending an event.

A short characteristics of Rust[9]language

Characteristics

Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

- Guaranteed memory safety

[9]https://www.rust-lang.org/

A short characteristics of Rust[9]language

Characteristics

Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

- Guaranteed memory safety
- Trait-based generics

[9]https://www.rust-lang.org/

A short characteristics of Rust[9]language

Characteristics

Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

- Guaranteed memory safety
- Trait-based generics
- Type inference etc.

[9]https://www.rust-lang.org/

About Rust language – what is the weakness points of Rust language?

Weakness of Rust language

Rust does not have...

- Dynamic typing

About Rust language – what is the weakness points of Rust language?

Weakness of Rust language

Rust does not have...

- Dynamic typing
- Code simplicity

About Rust language – what is the weakness points of Rust language?

Weakness of Rust language

Rust does not have...

- Dynamic typing
- Code simplicity
- Easy to write code

A short exploration of Rust language

Fun things of Rust

Rust makes you to benefit static type system because ...

- Flexible type by trait based generics

A short exploration of Rust language

Fun things of Rust

Rust makes you to benefit static type system because ...

- Flexible type by trait based generics
- To clarify software specification

A short exploration of Rust language

Fun things of Rust

Rust makes you to benefit static type system because ...

- Flexible type by trait based generics
- To clarify software specification
- Well documented ecosystem etc. (e.g. You don't create document when there is wrong code. Amazing!)

fruently example:

```
extern crate fruently;
use fruently::fluent::Fluent;
use std::collections::HashMap;
use fruently::forwardable::JsonForwardable;

fn main() {
    // create record
    let mut obj: HashMap<String, String> = HashMap::new();
    obj.insert("name".to_string(), "fruently".to_string());
    // establish a TCP connection with Fluentd.
    let fruently = Fluent::new("0.0.0.0:24224", "test");
    // send json-encoded record to Fluentd
    // and disconnecting automatically.
    let _ = fruently.post(&obj);
}
```

Demo

Any questions?