# ログ・係数集約と可視化・分析

# Agenda

- Fluentd
- Embulk
- ElasticStack6.0

# Fluentdとは

— Fluentdはシンプルにログ収集ができる

— データ連携のハブ

— 拡張性の高いストリーミングログコレクタ

# Fluentdとは

— 「CNCF(Cloud Native Computing Foundation)」の管理するプロジェクト

— Kubernetes や Prometheus といったクラウドネイティブな OSS 技術の推進を行う団体

— Kubernetes環境におけるログ収集ツールの **標準** となった

# Fluentdとtd-agent

- Fluentd
  - 最新バージョンはv1.1.3 - 2018/04/03
  - Fluentdのコアソフトウェア
  - プラグインは個別に利用者側でインストール
  - 最新を試したい場合はFluentd本体を使う
- td-agent
  - Fluentdに各種プラグインやRuby環境を組み込んだパッケージ
  - 主要環境でgemコマンドによるインストールが可能
    - Linuxへのインストールも容易
  - サポートOSに制約があるが依存関係の問題がほぼなし
    - トレジャーデータ側で検証済なので本番、安定稼働がいいのはtd-agent

# ユースケース

1. ログの収集
   ログをローカルディスクから、RDB等に渡すことができる
   ログの欠損は高可用性を維持することで防げる
2. 簡単なリアルタイム集計
   プラグインを利用することで、リアルタイムでステータスコードを含んだログを遅れる
   可視化ツールでグラフ化などもできる。
3. センサーログ収集
   センサー(ラズパイ)からゲートウェイに集めて、ログサーバに集計する。

# 利用しない方がいいケース

- ログの欠損も重複も許さず、確実に書き込む必要があるというケース
  - 課金データなど

# 非同期メッセージングサービスQoS

ネットワーク上で提供する機能を安定的に稼働させるために行う、サービス品質管理技術

| At Most Once(デフォルト) | At Least Once(オプション) | Exactly Once(サポートされていない) |
|---|---|---|
| 到達保証なし | 到達保証あり | 到達保証あり |
| 投げる側が一度で投げること | 投げる側が一度で投げること | 投げる側、受け取る側ともに一度で配信されること |
| 欠損の可能性あり | 欠損しない | 欠損しない |
| 重複しない | 重複の可能性 | 重複しない |

# v0.12バージョン

old stable
- プラグイン: Input, Parser, Filter, Output,

Formatter,Buffer
- 以下のような問題がある
  - 秒単位のみ
  - windows未対応
  - multi core未対応
  - プラグインが貧弱

# v0.14以降のバージョン

v0.14 v1の開発バージョン

- プラグイン: Input, Parser, Filter, Output, Formatter, Storage, Buffer
- 改善点
  - New Plugin APIs
  - ミリ秒対応
  - windows対応
  - multi core対応
  - New Plugin Helpers & Plugin Storage

# v0.14以降のバージョン

v1.0はv0.14と機能が同じでstableバージョン。
名前を変更しただけ。
最新バージョンはv1.1.3 - 2018/04/03

td-agent3は2017年の12月からstable版がでており、
Fluentd v1ベースになっている。

# v0.12とv1

- v0.12 APIを使用するプラグインは、Fluentd v0.14とv1の間でサポートされる（v2で廃止される予定）
- Fluentd v1は、起動時に自動的にv0.12スタイルをv1.0スタイルに変換するので、v0.12の設定をv1で再利用可能
- Fluentd v1.0の新機能は、新しいAPIを使用するプラグインでのみ使用可能
  - flexible chunk keys
  - placeholders
- Fluentd v0.12.xでは新しいAPIを使用するプラグインは動作しない

# v0.12とv1の設定の違い

v1はoutputのバッファパラメータに<buffer>sectionを使っている

```
# v1
<match pattern>
  @type foo
  database db1
  apikey foobarbaz
  # buffer parameters
  <buffer>
    @type file
    path /path/to/buffer
    flush_interval 10s
  </buffer>
</match>

# v0.12
<match pattern>
  @type foo
  database db1
  apikey foobarbaz
  # buffer parameters
  buffer_type file
  buffer_path /path/to/buffer
  flush_interval 10s
</match>
```

# fluent-plugin-bigquery

- 最新v2.0.0.beta
- schemeが間違っていると無限にretryしていた。
- v0.2.13以降ではデータがinvalidなのにretryかけても意味無いので、retryableな例外の時だけ例外上げ直して、
  その他の例外の時にはretry_stateを弄ってリトライを強制停止している。

# fluent-plugin-bigquery(v.1.2.0)のoutbigqueryinsert.rb

```ruby
def insert(project, dataset, table_id, rows, schema, template_suffix)
  writer.insert_rows(project, dataset, table_id, rows, template_suffix: template_suffix)
rescue Fluent::BigQuery::Error => e
  if @auto_create_table && e.status_code == 404 && /Not Found: Table/i =~ e.message
    # Table Not Found: Auto Create Table
    writer.create_table(project, dataset, table_id, schema)
    raise "table created. send rows next time."
  end

  raise if e.retryable?

  if @secondary
    # TODO: find better way
    @retry = retry_state_create(
      :output_retries, @buffer_config.retry_type, @buffer_config.retry_wait, @buffer_config.retry_timeout,
      forever: false, max_steps: @buffer_config.retry_max_times, backoff_base: @buffer_config.retry_exponential_backoff_base,
      max_interval: @buffer_config.retry_max_interval,
      secondary: true, secondary_threshold: Float::EPSILON,
      randomize: @buffer_config.retry_randomize
    )
  else
    @retry = retry_state_create(
      :output_retries, @buffer_config.retry_type, @buffer_config.retry_wait, @buffer_config.retry_timeout,
      forever: false, max_steps: 0, backoff_base: @buffer_config.retry_exponential_backoff_base,
      max_interval: @buffer_config.retry_max_interval,
      randomize: @buffer_config.retry_randomize
    )
  end

  raise
end
```

# Fluentd v1.2にはretryの挙動が変わりそう。

Fluentdのoutput oluginは、chunk flush中に復帰不可能なエラーを発生するが、
これらのチャンクを処理するために retry limit と secondary を使っている。

— 再開時に破損したfilechunkをskipして削除

https://github.com/fluent/fluentd/pull/1874

— chunkのflush中にoutput pluginが回復不可能なエラーを

# バッファ設計(v0.12)

InputからOutputへ情報が渡される仕組みにおいて
Output側ではBufferとQueueという仕組みがある。
これがログの欠損をしない仕組みにしている。

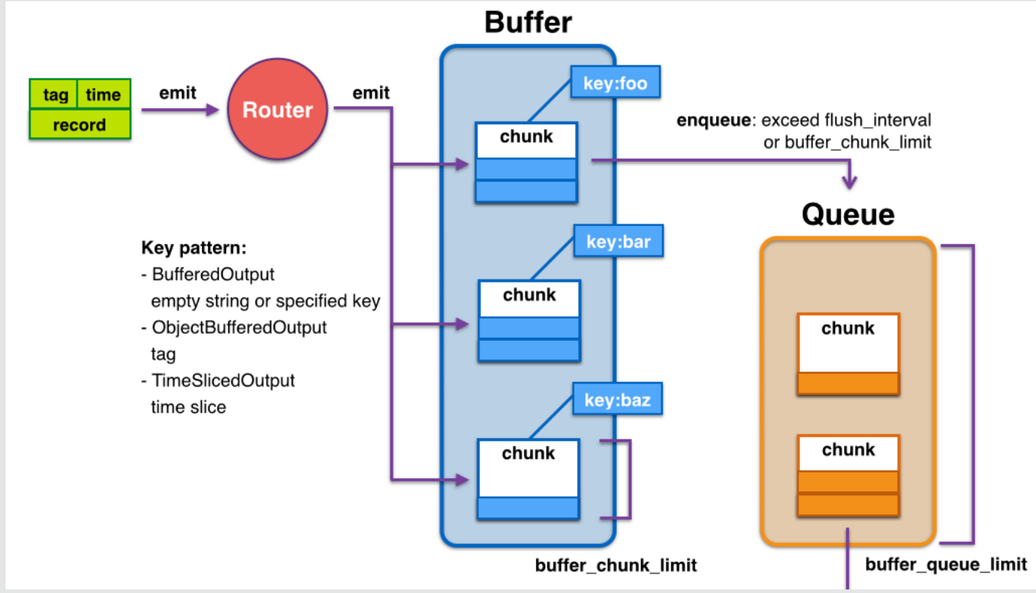
# バッファ設計(v0.12)

— 最初に情報が入ってくるBufferという機能の最大サイズ: `buffer_chunk_limit`

— 次にQueueという部分にchunkが押し出されるがQueueで何個までchunkを蓄えられるか: `buffer_queue_limit`

— enqueue: `buffer_chunk_limit` を超えた場合押し出される場合と `flush_interval` を経過した場合渡されるケース

— それぞれ設定すれば柔軟にログを送ることができる



# バッファ設計(v0.12)

— Outputのパラメータ

| パラメータ | 内容 |
| --- | --- |
| buffer_type | バッファの種類(file,memory) |
| buffer_path | ファイルバッファの格納先 |
| bufferchunklimit | chunck最大サイズ |
| bufferqueuelimit | Queue内chunck最大数 |
| flush_interval | バッファフラッシュ間隔 |

# バッファ設計(v0.12)

```
<match access.**>
  @type forward
  buffer_type file
  buffer_path /var/log/td-agent.buffer
  buffer_chunk_limit 8m #8MBを保持する
  buffer_queue_limit 64 #64個まで蓄える
  flush_interval 60s # BufferからQueueに渡される場合60秒たったら中身のchunckをQueueに渡す
  <server>
    name test_server
    host 192.168.33.11
    port 24224
  </server>
</match>
```

蓄えるDiskの容量やメモリのサイズは　buffer_chunk_limit x buffer_queue_limit をかけ合わせたものが領域としてBufferのサイズとして必要になる

matchの数だけこのかけ合わせた値が必要になるので注意が必要。

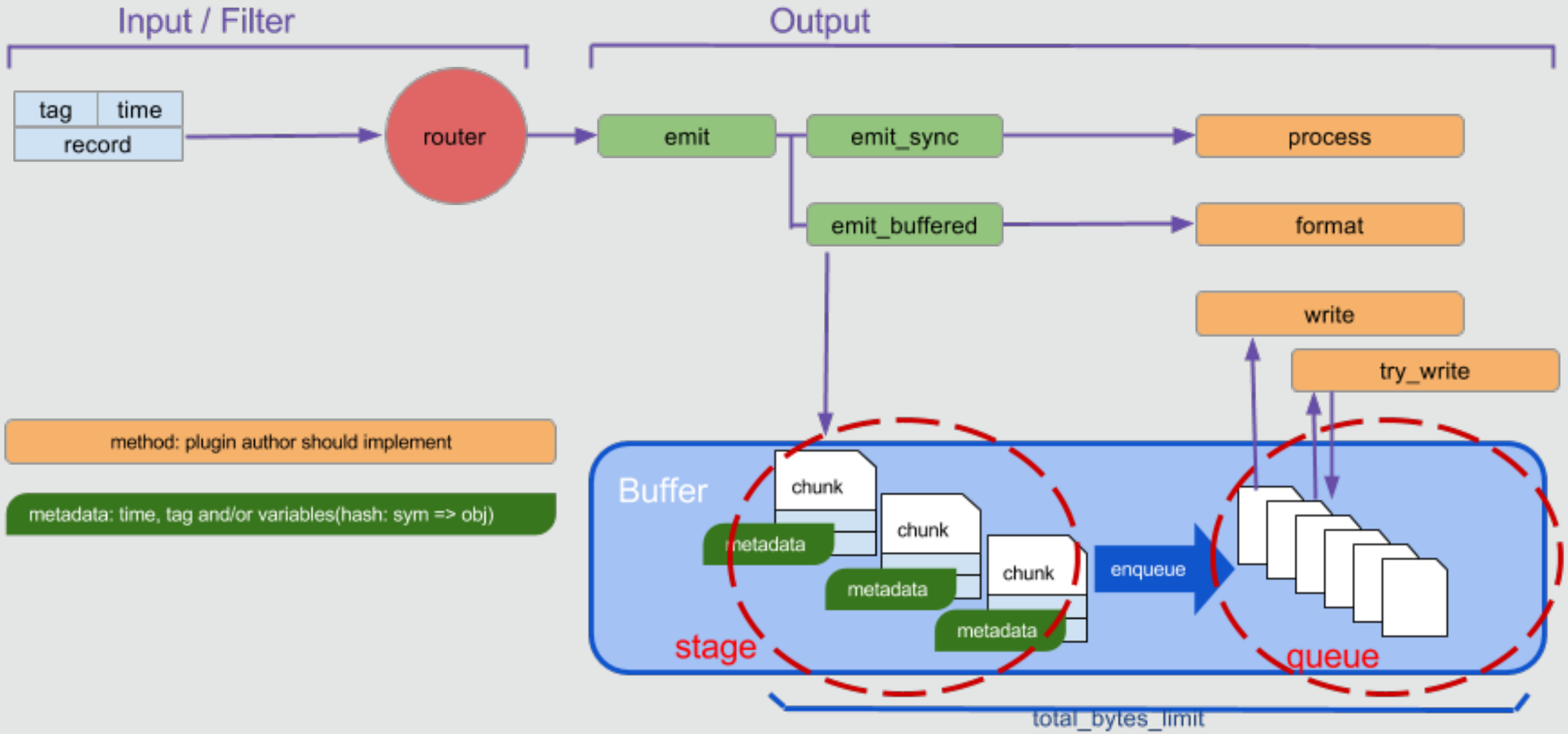
# バッファ設計(v1)

内部的には、バッファプラグインには、チャンクがイベントでいっぱいになる「ステージ」と、
チャンクが転送前に待機する「キュー」という 2つの分離された場所があります。
新しく作成されたすべてのチャンクは、ステージから開始し、時間内にキューに入れられます（その後、宛先に転送されます）。

— staged:buffering 状態

— queued:flush待ちのqueueに入っている状態

# バッファ設計(v1)


Input / Filter
Output
tag
time
record
router
emit
emit_sync
process
emit_buffered
format
write
try_write
method: plugin author should implement
metadata: time, tag and/or variables(hash: sym => obj)
Buffer
chunk
chunk
chunk
metadata
metadata
metadata
enqueue
stage
queue
total_bytes_limit

# Embulk

最新バージョン0.9.7(2018-04-16)

バルク版のFluentd

バッチ的な転送

— オープンソースの並列分散処理バルクローダー

— プラグインアーキテクチャ

— 容易なデータインテグレーションの実現

# Embulkのユースケース

- 過去分の情報を解析したい
- バッチ的にデータを転送したい
- 異なるストレージにデータを同期したい
- 大きな1ファイルだけを転送したい

# FluentdとEmbulkの使い分け

- Fluentd
  - WEB/APPサーバのログ収集
  - 監視、モニタリング
  - 流量の大きいログ収集
  - リアルタイム性の高い分析用途
  - バッチで溜め込むと送れないもの
- Embulk
  - マスタデータの同期
  - 一日ごとのデータ移動(バッチ的)
  - S3などからの並列データダウンロード
  - DWHへのデータロード

# バージョン

0.9.0 (2018-01-30)
- Java 8
  - Lambda
  - Stream
  - Time
  - Async File IO
  - FileSystem
- Oracle Java SEサポート・ロードマップ
  - LTS バージョンが、3年ごとのリリースを目標
  - 機能リリースは、6ヶ月ごとを目標

# バージョン

0.9.3(2018-02-13)
- JRubyベースのプラグインが使用されていない場合、JRubyの初期化を停止
- プラグインのロードと起動が速くなっている

0.9.7(2018-04-16)
- 最新バージョン

# embulk-announce

Embulkの新バージョンのリリース通知、互換性に関する通知など開発者からのアナウンス専用ML

https://t.co/w8TFtr30u0

# ElasticStack6.0

6.2.0リリース(2018-02-06)

検索と分析のスタックとして機能するコンポーネントのエコシステム

- Kibana
- Logstash
- Beats
- X-Pack
- Elasticsearch

# 各コンポーネントの役割

- Elasticsearch
  すべてのデータを格納し、検索機能と分析機能をスケーラブルに提供
- Logstash
  ログ、メトリックなどのイベントデータを任意の形式で集中管理
- Beats
  Filebeatは、サーバーからLogstashやElasticsearchにログファイルを配信するために構築されたBeat
  Metricbeatは、サーバー上で実行されているOSやサービスから定期的にメトリックを収集するサーバー監視エージェント
- kibana
  Elastic Searchの視覚化ツール
- X-Pack
  Elastic Stackにセキュリティ、監視、アラート、レポート、およびグラフ機能を追加
  コードが公開。

# お役立ち情報まとめ

- Fluentd
  Fluentd v1 and future at techtalk
  プラグイン開発者から見るfluentd v1.0の活用法
  fluentd の基礎知識
- Embulk
  Embulk v0.9
  Embulk
- Bigdam
  Bigdam
- ElasticStack
  discuss.elastic.co