ログ収集基盤 fluentdのご紹介

SRA OSS, Inc. 日本支社
盛　宣陽

ログの管理

システムを安定して運用していくために重要な要素

- ログ管理・運用の問題点
  - 肥大化問題
    - 収集の問題
    - ログの加工の問題
  - 多種多様なミドル・アプリ・機器が存在
    - ログのフォーマットが異なる
  - 利用方法も多種多様
    - 障害・解析に利用したい
    - ログデータを迅速に解析してビジネスに利用したい

ログの取得方法（バッチ/syslog）

- 各サーバのログを夜間バッチで取得する
  - 問題点
    - 大規模なシステムでは困難
- syslogサーバで取得
  - 問題点
    - 信頼性の欠如　UDPプロトコルによるログの取りこぼし
    - テキストファイル形式
      - サイズが大きいと確認が困難

ログの取得方法(rsyslog)

- rsyslogで取得
  - RHEL6標準
  - 信頼性の確保 TCP TLS
  - モジュール構成 (Input,Output..)
    ログデータをRDBMSやHDFSに格納することもできる
  - 特定の文字列を含んだログを収集できるフィルタ機能

ログの取得方法

- fluentdで取得
  - 簡単なインストール
  - 信頼性の確保 TCP SSL
  - プラグイン構成(Input,Output..）
    rsyslog 約50種 fluentd 250種以上
    プラグインを活用し柔軟にログを活用できるようになった
  - 負荷分散や可用性にも対応
  - ログのJSON化
    - 構造を持たせログをより扱いやすくなった

fluentdでログデータの活用

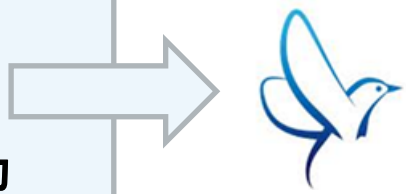
- OS、ミドルウェア、アプリケーションから出力されるログをデータベースで管理
  - 障害解析のスピードアップ
    - 各種ログ・リソース値を時系列で串刺しして事象を確認
    - JSON要素にインデックスを定義して高速検索
    - 障害事象をメールで通知したり統合監視ツールと連携
  - フィルタリング機能
    - 重要なログだけ残して容量削減
  - ログファイルから集計・解析
  - ログから数値データを抜き出してログの視覚化
  - 監査ログ

- ログを収集するツール
  http://fluentd.org/
- 開発
  - 米国Treasure Data社の開発者
  - コミュニティベース
  - 言語：Ruby + C言語
- ライセンス
  - Apache License Version 2.0

シンプルなイメージ

入力：Input

アプリケーションのログ
Webサーバのログ
データベースのログ
Syslog
http入力
Unixドメインソケット入力
コマンド実行結果



出力：Output

fluentdの特徴

- 簡単なインストール
- ログの半構造化
- プラグイン機能
- バッファリング機能
- ログ転送

簡単なインストール

- コマンド一発　例)RHEL(CentOS)

```
curl -L http://toolbelt.treasuredata.com/sh/install-redhat.sh | sh
```

注　fluentdの安定版の配布パッケージであるtd-agentがインストールされます
　　稼働に必要なRuby 1.9.xもインストールされます

- 起動・停止

```
/etc/init.d/td-agent start
/etc/init.d/td-agent stop
```

- プロセス

```
/usr/lib64/fluent/ruby/bin/ruby/usr/sbin/td-agent--grouptd-agent--log/var/log/td-agent/td-agent.log--daemon/var/run/td-agent/td-agent.pid
```

ログの半構造化①

- 一つのログを「タグ」「時刻」「ログの内容」で管理

- 「タグ」 ・・利用者 :ログの種類を指定
  fluentd:タグ単位で処理
  タグは階層を持つ

<親タグ>.<子タグ>.<孫タグ>..
例) ホスト名.ミドルウェア名.ログの種類
Host.Apache.access

ログの半構造化②

- 「時刻」 ‥ ログに記載されている
  時刻フォーマットを指定
- 「ログの内容」 ‥ ログの内容をJSON化

```
value1  value2  value3
=>
{“key1":“value1”,“key2":“value2",“key3",“value3“}
```

- 厳密なフォーマットを持たずJSONで柔軟に定義
- DBを利用してJSON要素にIndexを張って高速検索

ログの半構造化③

具体例

- 入力:Input (Apacheのアクセスログ)

```
192.168.8.48 - - [20/Nov/2013:19:05:36 +0900] "GET /10.html
HTTP/1.0" 200 3 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:25.0) Gecko/20100101 Firefox/25.0"
```

- 出力:Output

```
2013-11-20T19:05:36+09:00          ・・・・・時刻データ (注 apacheログと比較)
Host.Apache.access                 ・・・・・タグ: 利用者が定義
{"host":"192.168.8.48","user":"-
","method":"GET","path":"/10.html","code":"200","size":"3","referer":"-
","agent":"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:25.0)
Gecko/20100101 Firefox/25.0"}      ・・・・・レコード JSON形式のログ内容
```

プラグイン機能①

- 3種類のプラグイン
  - Input プラグイン
    ログデータの入力元となるプラグイン
  - Outputプラグイン
    fluentdで取得したログデータの出力先
  - バッファプラグイン
    - 後述
- プラグイン数 267種類

2014年3月時点
http://fluentd.org/plugin/に紹介があるもの
個人のGitHubで公開しているものも多数あり

プラグイン機能②

- プラグインの入手方法
  ruby gemでインストール 例) mail通知プラグイン

```
#/usr/lib64/fluent/ruby/bin/fluent-gem install fluent-plugin-mail
```

- 自作のプラグイン

```
/etc/td-agent/plugin/fluent-easy-plugin.rb
```

プラグイン機能③ Input

- Inputプラグインの基本
  - tailプラグイン
    - tailコマンドのようにログファイルの行を取得する
  - 設定ファイル(/etc/td-agent/td-agentd.conf)

```
<source>      #Inputプラグインは<source>ディレクティブで指定
   type tail
   tag          タグ名
   path         ログファイルパス
   pos_file     指定ファイルに最後に読み込んだ位置を記録します。
   format       ログのフォーマット '/'で囲まれた正規表現
   time_format  時間フィールドのフォーマット(strftime形式)
</source>
```

プラグイン機能④ Input

- ログ生成スクリプト　1秒ごとにログを出す

```
#!/bin/bash
export LANG=C
while :
do
        echo `date` ERROR テスト >> /tmp/test.log
        sleep 1
done
```

- ログ出力サンプル（□スペース）

```
Mon□Mar□18□00:56:03□JST□2014□ERROR□テスト
```

fluentdの形式(時刻、タグ名、ログ内容)で取る

時刻 タグ名{"ERRORLEVEL":"ERROR" ,"MESSAGE":"テスト"}

プラグイン機能⑤ Input

- 設定ファイルの中身（□スペース）

```
<source>
  type      tail
  tag       localhost.tail
  path      /tmp/test.log
  pos_file /tmp/test.pos
  format□/^(?<time>[^ ]*□[^ ]*□[^ ]*□[^ ]*□[^ ]*
   □[^ ]*)□(?<ERRORLEVEL>[^ ]*)□(?<MESSAGE>.*)$/
  time_format□%a□%b□%d□%H:%M:%S□%Z□%Y
</source>
```

このままでも動きますが
Outputプラグインを指定していないので
出力されません。

プラグイン機能⑥ Output

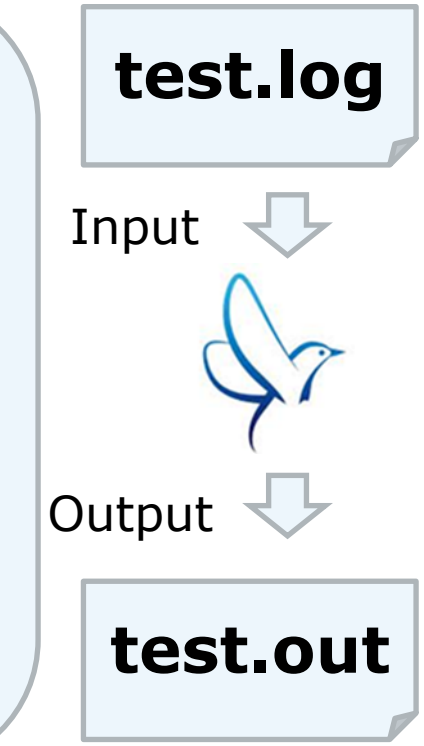
- Outputプラグイン基本
  - fileプラグイン
    - fluentdのログ形式でファイルに出力
  - 設定ファイル(/etc/td-agent/td-agentd.conf)

```
<match タグ名>      #outputプラグインで指定するディレクティブ
                    #inputプラグインに指定したタグ名を記述する
  type file         #fileプラグインの設定
  path                出力ログファイルパス
  time_slice_format  出力ファイル名＋付与される名前
                              (デフォルト%Y%m%d)
  time_wait  出力ファイルがローテーションする前に待つ時間
                                    (デフォルト10分)
  <バッファの設定>  後述
</match>
```

プラグイン機能⑦Output

- 設定ファイルの中身（□スペース）InputとOutput

```
<source>
  type      tail
  tag       localhost.tail
  path      /tmp/test.log
  pos_file /tmp/test.pos
  format□/^(?<time>[^ ]*□[^ ]*□[^ ]*□[^ ]*□[^ ]*
   □[^ ]*)□(?<ERRORLEVEL>[^ ]*)□(?<MESSAGE>.*)$/
  time_format□%a□%b□%d□%H:%M:%S□%Z□%Y
</source>
<match localhost.tail>
  type file
  path /tmp/test.out
</match>
```



プラグイン機能⑧Output

- Inputのログ形式

```
Mon□Mar□18□00:56:03□JST□2014□ERROR□テスト
```

- Outputファイルのログ形式

```
2014-03-18T00:56:03+09:00
localhost.tail
{"ERRORLEVEL":"ERROR","MESSAGE":"テスト"}
```

プラグイン機能⑨

Inputプラグイン

tail ..単一行のログ
http ..http経由
syslog ..syslog(UDP)
(in)exec ..スクリプト実行結果
tail-multiline ..複数行のログ
dstat ..リソースメトリック
cloudwatch .. AWSメトリックス
(in)forward ..flunedからの転送

目的に合ったプラグインを使って運用コスト削減

outputプラグイン

file ..ファイル出力
Exec ..出力をスクリプトに渡す
copy ..入力を複数のouputにcopy
Mongo ..MongoDBへの出力
notifier ..数値範囲や文字列検査
Mail ..メール通知
Zabbix ..Zabbixへの通知
pgjson ..PostgreSQL JSON型へ出力
rewrite_tag_filter ..filterを通してtagの書き換え
datacounter .. 集計処理
Null ..データの破棄
forward .. fluentdへ転送

プラグイン機能⑩

```
class EasyInput < Fluent::Input
  Fluent::Plugin.register_input('easy',
self)

  def configure(conf)
    //設定でtagが利用できる
    super
    @tag = conf['tag']
  end

  def start
     @thread =
Thread.new(&method(:run))
  end

  def run
    loop do
      //入力内容
      Fluent::Engine.emit(@tag,
         Fluent::Engine.now, {"message"
=>
         "easy-test"})
      sleep 1
    end
  end

  def shutdown
  //ソケットやファイルをcloseする処理
  end
end
```

- 簡単な自作プラグイン

  設定からタグ名を読み込んで、

  タグ名

  時刻

  {"message"=>"easy-test"}

  を1秒ごとに入力するプラグイン

- 保存先

  /etc/td-agent/plugin/fluent-easy-plugin.rb

- 設定

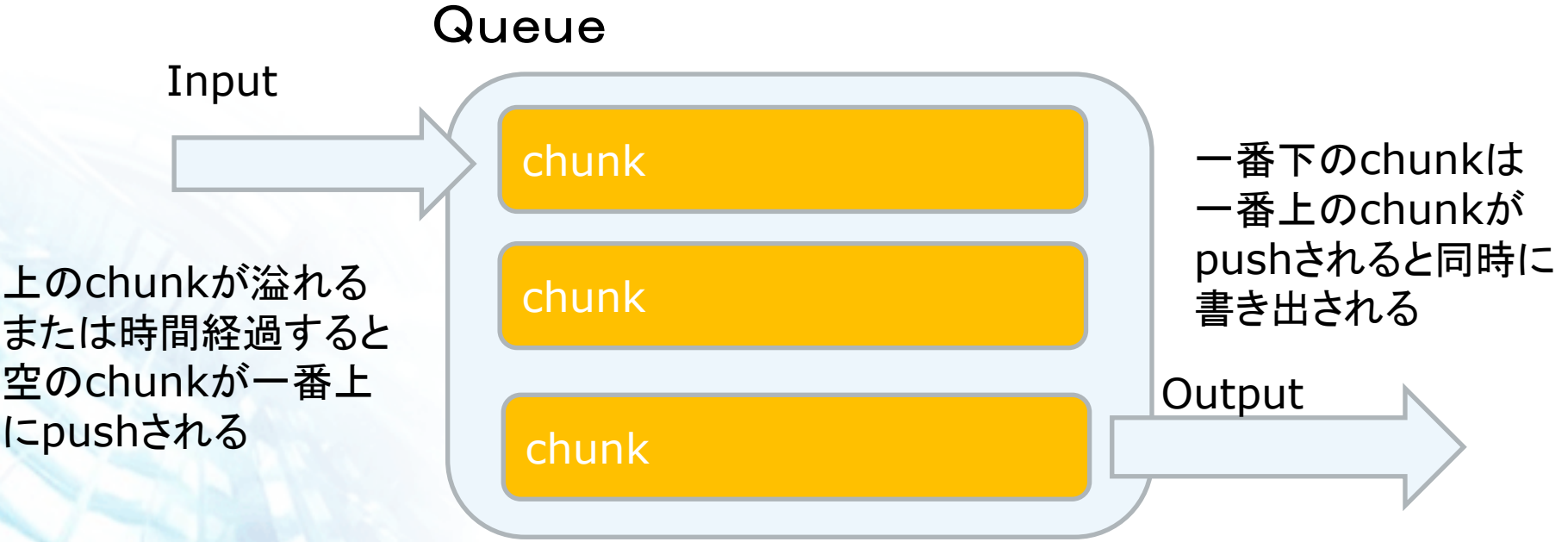
```
<source>
  type easy
  tag localhost.easy
</source>
```

バッファリング機能①

- バッファプラグイン
  - 性能と信頼性向上
    - まとめて出力
    - 受け取ったデータを保管して出力先の一時的な障害に対応
  - バッファリングするOuputプラグインで利用
    (ファイル・DB出力や転送処理)
  - オンメモリとファイルの指定が可能

バッファリング機能②

- バッファの構造 (Queueとchunk)
  - chunkのサイズとQueue内のchunk数は設定可能



バッファリング機能③

- バッファ関連設定パラメータ
  - buffer_type ‥memoryまたはfileを指定
    fileの場合にはbuffer_pathにファイル名を指定
  - buffer_queue_limit ‥queue内のchunk数
  - buffer_chunk_limit ‥chunk一個当たりのサイズ
  - flush_interval ‥新しいchunkをpushする時間間隔
    (書き出しが行われる時間間隔)

バッファ使用量は?
buffer_queue_limit x buffer_chunk_limit

バッファリング機能④

- バッファ関連設定パラメータ　続き
  - retry_wait　・・chunkの書き出しに失敗した場合に
    指定秒間待ってから再試行
  - retry_limit　・・再試行回数の限度を指定する
    超えてしまうとchunkを破棄する
    再試行を待つ時間は前回の2倍

書き出しができずにqueueが溢れると?

→入力を受け付けない

バッファリング機能⑤

設定例

```
<source>
  type      tail
  tag       localhost.tail
  path      /tmp/test.log
  pos_file /tmp/test.pos
  format□/^(?<time>[^ ]*□[^ ]*□[^ ]*□[^ ]*□[^ ]*
   □[^ ]*)□(?<ERRORLEVEL>[^ ]*)□(?<MESSAGE>.*)$/
  time_format□%a□%b□%d□%H:%M:%S□%Z□%Y
</source>
<match localhost.tail>
  type file
  path /tmp/test.out
  buffer_type  memory
  buffer_queue_limit 64
  buffer_chunk_limit 8m
  flush_interval  60s
  retry_wait 1s
  retry_limit 10
</match>
```

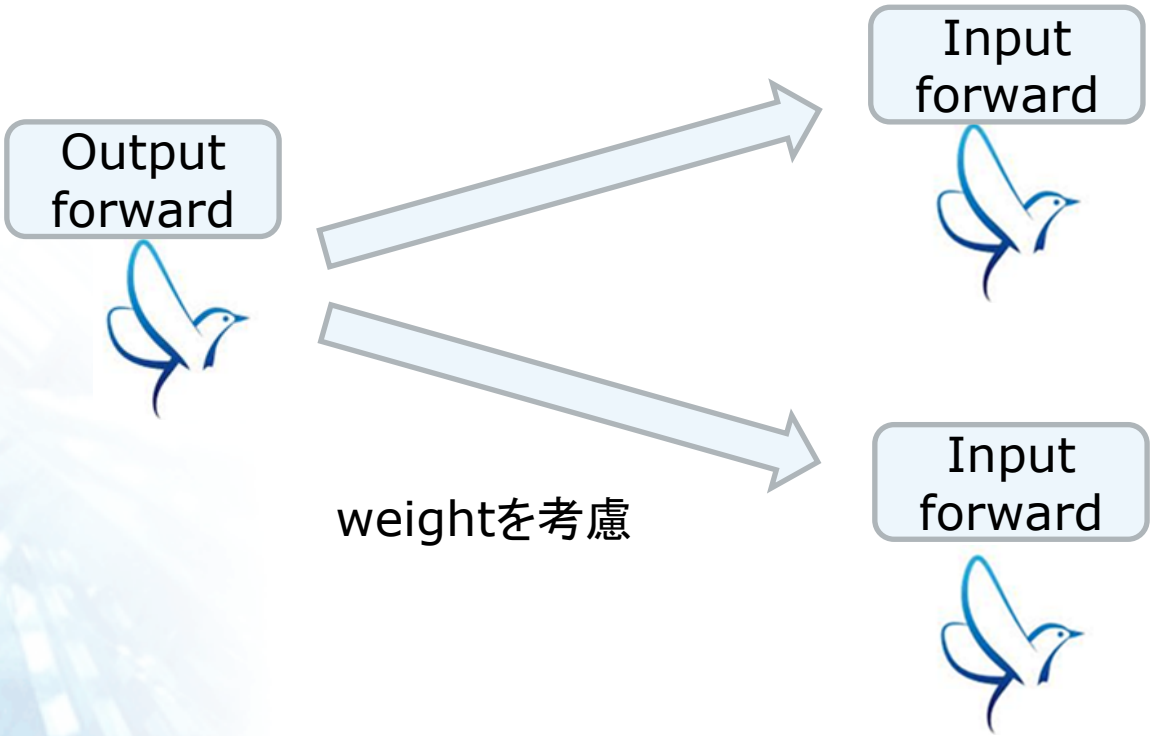
outputプラグイン単位でバッファの設定

ログ転送①

- forwardプラグインによるログ転送
  タグ単位で設定可

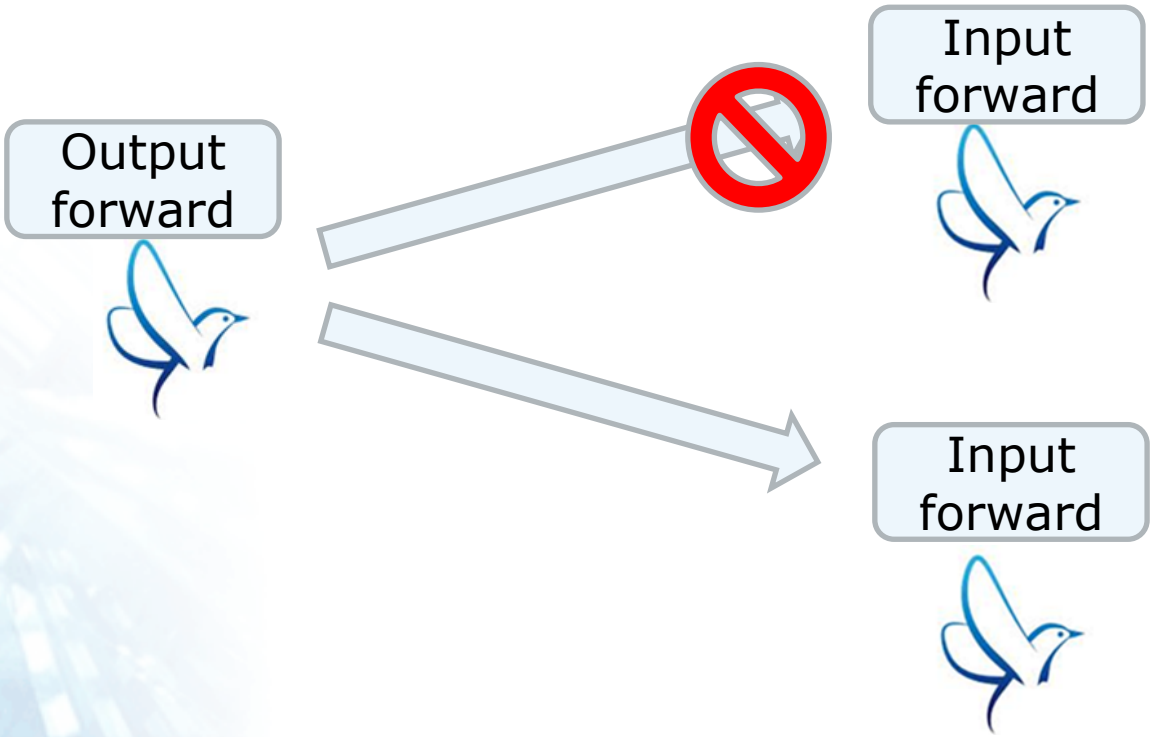
ログ転送②

- forwardプラグインによる負荷分散



ログ転送③

- forwardプラグインによるHA構成



ログ転送④

設定例

```
<match localhost.tail>
  type forward
  <server>
        host 192.168.0.1
        port 24224
  </server>
  <server>
        host 192.168.0.2
        port 24224
        standby                    #待機系
   </server>
   バッファの設定
</match>
```

分かりやすい設定

fluentdのまとめ

- 簡単なインストール
  rpmで一発
- ログの半構造化
  構造を持たせてログを扱いやすく
- プラグイン機能
  目的にあったプラグインを活用して運用コスト削減
- バッファリング機能
  性能と信頼性の確保
- ログ転送
  負荷分散と可用性も確保

おまけ

- fluentd + PostgreSQL(9.3推奨)
  - fluent-plugin-pgjson
  - インストール(PostgreSQL 9.0以降のlibpqが必要)

```
#/usr/lib64/fluent/ruby/bin/fluent-gem install fluent-plugin-pgjson
```

  - PostgreSQLテーブル定義

```
fluentd=# ¥d fluentd
        テーブル "public.fluentd"
 カラム |           型             | 修飾語
--------+--------------------------+--------
 tag    | text                     |
 time   | timestamp with time zone |
 record | json                     |

fluentd=# SELECT * from fluentd limit 1;
-------------------------------
tag      | apache.access
time     | 2013-11-20 12:57:06+09
record |{"host":"133.137.177.177","user":"-","method":"GET","path":"/","code":"403","size":"5039","referer":"-","agent":"ApacheBench/2.3"}
```

#CREATE INDEX fluentd_idx on fluentd ((record ->> 'code'));

```
#SELECT count(*) from fluentd;
  count
---------
 5804348
(1 行)

# EXPLAIN ANALYZE  SELECT *  from fluentd where record->>'code'= '403';
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on fluentd  (cost=545.36..72223.05 rows=29022 width=54) (actual time=0.070..0.078
rows=23 loops=1)
   Recheck Cond: ((record ->> 'code'::text) = '403'::text)
   ->  Bitmap Index Scan on fluentd_idx  (cost=0.00..538.10 rows=29022 width=0) (actual time=0.055..0.055
rows=23 loops=1)
         Index Cond: ((record ->> 'code'::text) = '403'::text)
 Total runtime: 0.130 ms
(5 行)

# set enable_bitmapscan to off;
# EXPLAIN ANALYZE SELECT *  from fluentd where record->>'code'= '403';
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Index Scan using fluentd_idx on fluentd  (cost=0.43..108072.32 rows=29022 width=54) (actual
time=0.055..0.077 rows=23 loops=1)
   Index Cond: ((record ->> 'code'::text) = '403'::text)
 Total runtime: 0.122 ms
(3 行)
```