Hatoholのログ蓄積・検索機能2014/12版

須藤功平

株式会社クリアコード

2014/12/09

内容

- ログ蓄積・検索機能の概要
  - ただし2014年12月時点での情報
- 詳細はWikiを参照
  - https://github.com/project-hatohol/hatohol/wiki/Log-archive
  - https://github.com/project-hatohol/hatohol/wiki/Log-search

目的

- ✓ 現状を共有すること

蓄積：解決したい問題

- ✓ 大量のログを蓄積したい
  - ✓ 目的：ルールを満たすため
    例：一定期間の過去ログ蓄積が義務
  - ✓ 目的：問題発生時期の特定
    問題発覚時にいつから問題が起こっていたかを確認

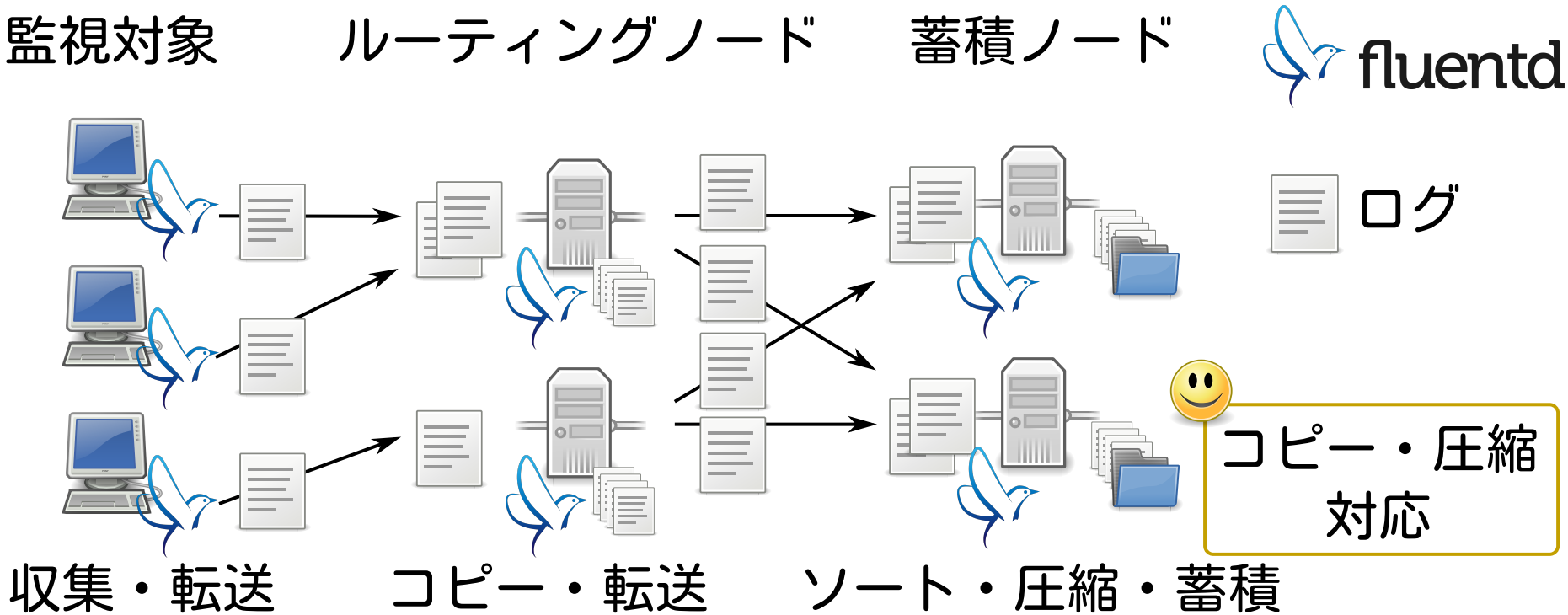
蓄積：解決方法の問題

- ZabbixでやるならRDBMSへ蓄積
- RDBMSへ蓄積：オーバースペック
  - リソース消費が割にあわない
    - めったに参照しない
    - 処理量が多い
  - レプリケーション必須
    - 失いたくないデータだから

蓄積：解決方針

- ファイルに保存
  - オーバーヘッドが少ない
  - 扱いやすい（コピー・圧縮）
- Fluentdと連携
  - データ配送システム

蓄積：構成


監視対象
ルーティングノード
蓄積ノード
fluentd
ログ
コピー・圧縮
対応
収集・転送
コピー・転送
ソート・圧縮・蓄積

蓄積：ポイント（1）

- 監視対象は生ログを収集・転送
  - 直接蓄積ノードに送らない
- ルーティングノードの導入
  - 目的：構成変更耐性の強化
  - 変更：蓄積ノードの増減
  - 全監視対象を変更するより楽

蓄積：ポイント（2）

- ✓ 蓄積ノードでのソート
  - ✓ 時間順にログが届くとは限らない
  - ✓ 一定時間バッファリング
  - ✓ バッファ内をソート後に書き込み
  - ✓ ↑用プラグインは新規開発
    (fluent-plugin-sort)

蓄積：タグ設計

- 監視対象
  - raw.${type}.log.${host_name}
  - 例：raw.messages.log.node1
- 蓄積ノード
  - raw....→ソート→sorted.raw....
  - →蓄積

蓄積：設定：監視対象（1）

```
<source>
  type tail
  path /var/log/messages
  pos_file /var/log/td-agent/messages.pos
  tag "raw.messages.log.#{Socket.gethostname}"
  format none
</source>
```

蓄積：設定：監視対象（2）

```
<match raw.*.log.**>
  type copy
  <store>
    type secure_forward
    shared_key fluentd-secret
    self_hostname "#{Socket.gethostname}"
    # バッファー設定は省略
    <server>
      host router1.example.com
    </server> # ←ルーティングノードの数だけ書く
  </store>
</match>
```

蓄積：設定：ルーター（1）

```
# 受信
<source>
  type secure_forward
  shared_key fluentd-secret
  self_hostname "#{Socket.gethostname}"
  cert_auto_generate yes
</source>
```

蓄積：設定：ルーター（2）

```
<match raw.*.log.**> # 転送
  type forest
  subtype secure_forward
  <template> # 共通設定
    shared_key fluentd-secret
    self_hostname "#{Socket.gethostname}"
    # バッファー設定は省略
  </template>
  # 個別ケースは後述
  <case **> # デフォルト
    <server>
      host archiver2.example.com
    </server>
  </case>
</match>
```

蓄積：設定：ルーター（3）

```
# 個別設定
<case raw.*.log.node1.example.com>
  <server>
    host archiver1.example.com
  </server>
</case>
<case raw.*.log.node2.example.com>
  <server>
    host archiver2.example.com
  </server>
</case>
```

蓄積：設定：蓄積（1）

```
# 受信
<source>
  type secure_forward
  shared_key fluentd-secret
  self_hostname "#{Socket.gethostname}"
  cert_auto_generate yes
</source>
```

蓄積：設定：蓄積（2）

```
# ソート
<match raw.**>
  type sort
  add_tag_prefix sorted.
  buffer_type file
  buffer_path /var/spool/td-agent/buffer/sort
  flush_interval 60
</match>
```

蓄積：設定：蓄積（3）

```
# タグの並び替え（次の処理をしやすくするためだけ）
# sorted.raw.${type}.log.${host_name}
# ↓
# archive.raw.log.${host_name}.${type}
<match sorted.raw.**>
  type record_reformer
  enable_ruby false

  tag archive.${tag_parts[1]}.${tag_parts[3]}.${tag_suffix[4]}.${tag_parts[2]}
</match>
```

蓄積：設定：蓄積（4）

```
<match archive.raw.log.**> # ←蓄積
  type forest
  remove_prefix archive.raw.log
  # ↓タグの区切りでディレクトリーを掘る
  escape_tag_separator /
  subtype file
  <template>
    path /var/log/archive/${escaped_tag}
    compress gz # ← 圧縮
    format single_value
    append true
    flush_interval 60
  </template>
</match>
```

蓄積：Hatohol連携

なし

蓄積：Hatohol連携案（1）

- 蓄積ノードへリンク
  - ログは単なるファイル
  - HTTPで公開することは簡単
- 課題
  - アクセス権限はどうする？
  - Hatoholが認可サーバーになる？
  - HatoholをLDAP対応させる？

蓄積：Hatohol連携案（2）

- ログ検索システムへのロードUI
  - 検索→ログ検索システム（後述）へロードが必要
  - ロードはfluent-catで可能
- 課題
  - 蓄積ノードでのコマンド実行が必要
  - ホスト管理機能との連携で可能？

検索：解決したい問題

- ログの確認が面倒
  - 対象ホスト数が多い
    （Hatoholは大規模システム用のソフトウェア）
    - 個別にログインするのが面倒
  - 対象ログ数が多い
    - 該当ログを探すのが面倒

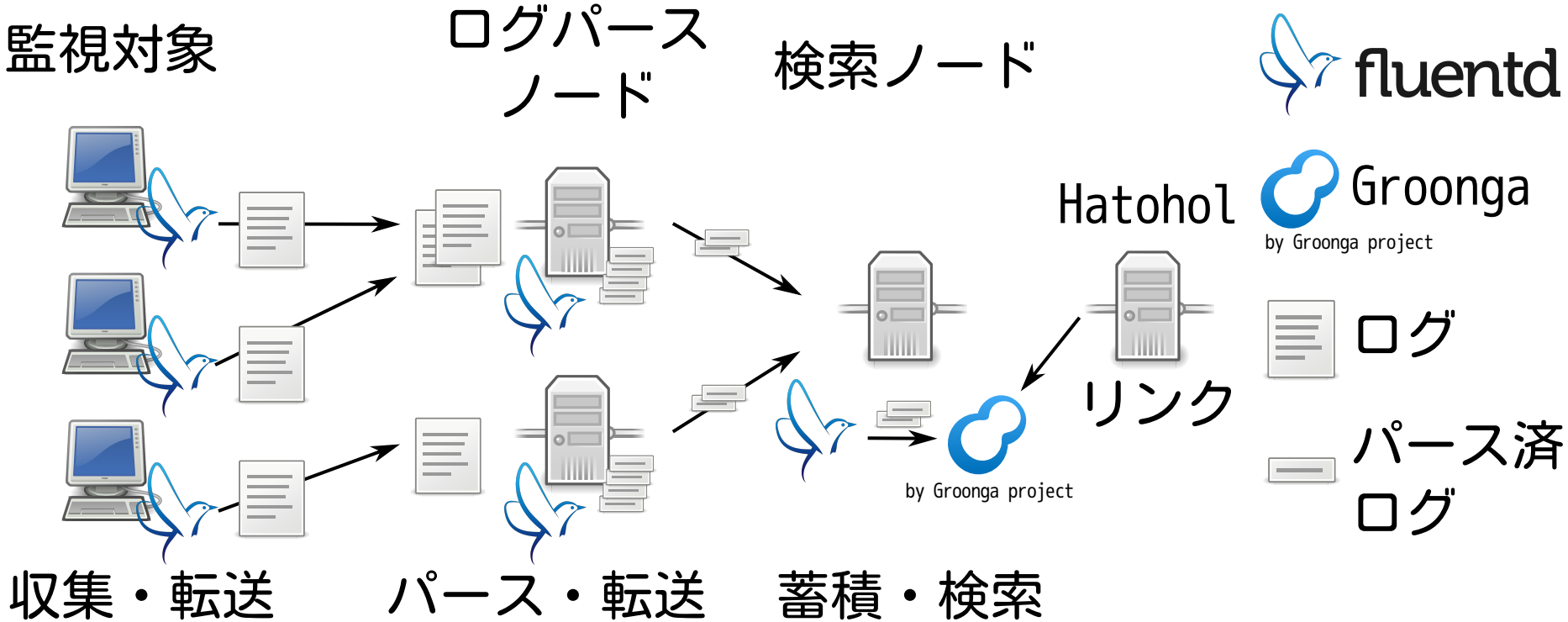
検索：解決方法

- ✓ 全文検索システムの利用
- ✓ 要件
  - ✓ 最新のログをすぐに検索できる
  - ✓ 十分高速
  - ✓ 検索対象は最新数週間から数ヶ月

検索：全文検索システム

- ✓ Groongaを採用
  - ✓ 即時検索可能・十分高速
  - ✓ 追加中も検索性能が落ちない
    （ログは常に追加される）
  - ✓ nginxのモジュールとしても使える
    - ✓ nginxの機能・ノウハウを使える
      （HTTPSや認証周りなど）
    - ✓ 運用が容易

検索：構成

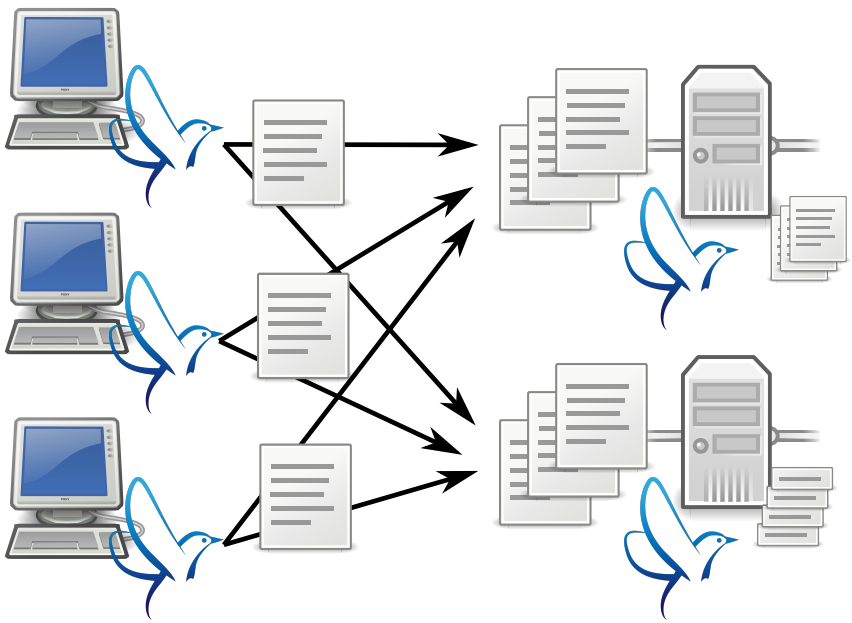

監視対象
ログパース
ノード
検索ノード
fluentd
Groonga
by Groonga project
Hatohol
ログ
リンク
パース済
ログ
by Groonga project
収集・転送
パース・転送
蓄積・検索

検索：ポイント

- 監視対象は生ログを収集・転送
  - パースしない
  - ログ蓄積と同じ設定→共通化可能
- ログパースノードの導入
  - 監視対象の負荷を下げる

検索：構成：共通化

監視対象



収集・コピー・転送

ルーティング
ノード
（蓄積）

ログパース
ノード
（検索）

fluentd

ログ

パース済
ログ

検索：タグ設計（1）

- 監視対象
  - raw.${type}.log.${host_name}
  - 例：raw.messages.log.node1
- ログパースノード
  - raw.${type}....→パース→
  - ${type}....→フォーマット→log

検索：タグ設計（2）

- ✓ 検索ノード
  - ✓ log→Groonga

検索：設定：監視対象（1）

```
<source>
  type tail
  path /var/log/messages
  pos_file /var/log/td-agent/messages.pos
  tag "raw.messages.log.#{Socket.gethostname}"
  format none
</source>
```

検索：設定：監視対象（2）

```
<match raw.*.log.**>
  type copy
  <store>
    # 蓄積用設定
  </store>
  <store>
    type secure_forward
    # 蓄積用と同じなので設定は省略
    <server>
      host parser1.example.com
    </server> # ←ログパースノードの数だけ書く
  </store>
</match>
```

検索：設定：パース（1）

```
# 受信
<source>
  type secure_forward
  shared_key fluentd-secret
  self_hostname "#{Socket.gethostname}"
  cert_auto_generate yes
</source>
```

検索：設定：パース（2）

```
# パース
<match raw.*.log.**>
  type forest
  subtype parser
  <template>
    key_name message
  </template>
  <case raw.messages.log.**>
    remove_prefix raw
    format syslog # ← messages型はsyslogとしてパース
  </case>
</match>
```

検索：設定：パース（3）

```
# フォーマット
<match *.log.*.**>
  type record_reformer
  enable_ruby false
  tag ${tag_parts[1]} # ←tagをlogだけにする
  <record> # メタデータをデータにする
    host ${tag_suffix[2]}
    type ${tag_parts[0]}
    timestamp ${time}
  </record>
</match>
```

検索：設定：パース（4）

```
# 転送
<match log>
  type secure_forward
  shared_key fluentd-secret
  self_hostname "#{Socket.gethostname}"
  # バッファー設定は省略
  <server>
    host search.example.com
  </server>
</match>
```

検索：設定：検索（1）

```
# 受信
<source>
  type secure_forward
  shared_key fluentd-secret
  self_hostname "#{Socket.gethostname}"
  cert_auto_generate yes
</source>
```

検索：設定：検索（2）

```
# 保存
<match log>
  type groonga
  store_table Logs
  # バッファー設定は省略
  protocol http
  host 127.0.0.1
  # スキーマ定義
</match>
```

検索：設定：検索（3）

```
# スキーマ定義
<match log>
  <table> # 全文検索用語彙表
    name Terms
    flags TABLE_PAT_KEY
    key_type ShortText
    default_tokenizer TokenBigram
    normalizer NormalizerAuto
  </table>
</match>
```

検索：設定：検索（4）

```
# スキーマ定義
<match log>
  <table> # ホスト名データを共有
    name Hosts
    flags TABLE_PAT_KEY
    key_type ShortText
    # normalizer NormalizerAuto
  </table>
</match>
```

検索：設定：検索（5）

```
# スキーマ定義
<match log>
  <table> # タイムスタンプ用語彙表
    name Timestamps
    flags TABLE_PAT_KEY
    key_type Time
  </table>
</match>
```

検索：設定：検索（6）

```
# スキーマ定義
<match log>
  <mapping> # ホスト名データを共有
    name host
    type Hosts
    <index> # 高速検索のための索引定義
      table Hosts
      name logs_index
    </index>
  </mapping>
</match>
```

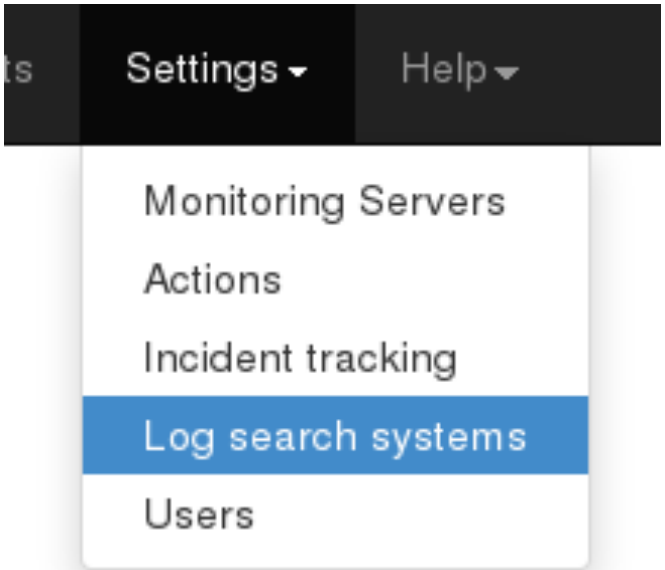
検索：設定：検索（7）

```
# スキーマ定義
<match log>
  <mapping>
    name timestamp
    type Time
    <index> # 高速検索のための索引定義
      table Timestamps
      name logs_index
    </index>
  </mapping>
</mapping>
```

検索：設定：検索（7）

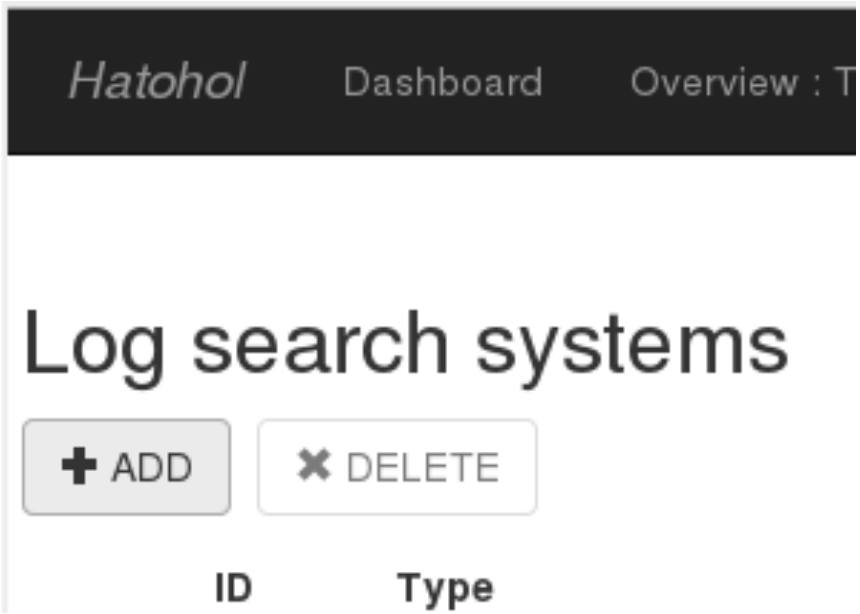
```
# スキーマ定義
<match log>
  <mapping>
    name message
    type Text
    <index> # 全文検索用索引定義
      table Terms
      name logs_message_index
    </index>
  </mapping>
</match>
```

検索：設定：Hatohol（1）



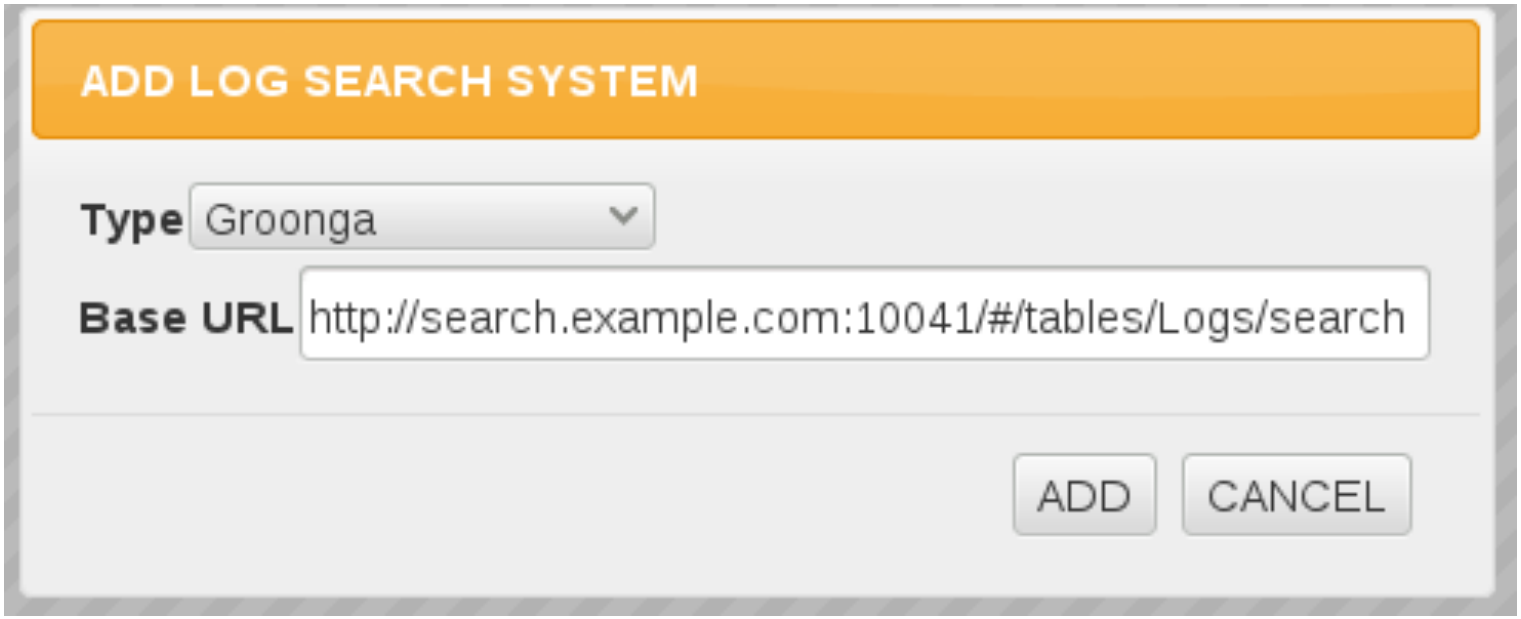
検索UI管理画面に移動

検索：設定：Hatohol（2）



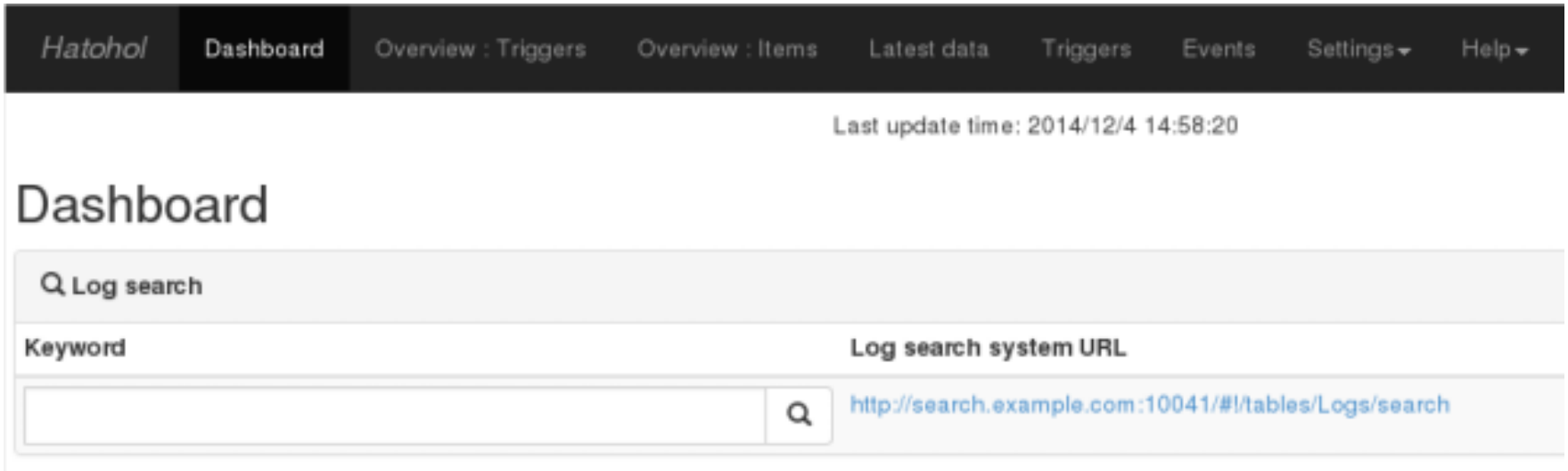
検索UIの追加フォームを出す

検索：設定：Hatohol（3）



検索UIの追加フォームを入力

検索：Hatohol連携



ダッシュボードに検索フォーム

検索：Hatohol連携案（1）

- イベントページにリンク
  - 障害情報から検索できる
  - 時間情報から検索できる
- 課題
  - ログ以外の情報が多い
  - ↑検索可能ではない情報が多い

検索：Hatohol連携案（2）

- ログ以外も検索可能にする
  - 例：イベントの説明
  - 例：Redmineのコメント
- 課題
  - 検索結果の見せ方はどうなる？
  - 見つかった後の動線は？
  - Hatoholに戻る？Redmineに行く？

検索：Hatohol連携案（3）

- 検索キーワードを事前登録
  - クリックで検索できて楽
- 課題
  - 誰が登録するのか
  - 既存ログから拾ってこれるとよさそう？

検索：課題

- ユーザー毎に権限設定できない
  - ログ検索UIのURLをサーバーではなくUIに保存しているため
- 検索UIからHatoholに戻ってこれない
  - Redmine連携も同じ？