# Elastic Stack ver.5
# ハンズオン

Acroquest Technology 株式会社

樋口 慎

Acroquest Technology

# 自己紹介

## 樋口 慎 (Acroquest Technology 株式会社)

- Elasticテクニカルワークショップ講師
  (https://info.elastic.co/japan-technical-workshop.html)
- Data Analytics Showcase
  (http://www.db-tech-showcase.com/data-analytics-showcase)
- elasticsearch勉強会
- JJUG CCC 2016 spring

Twitter : @shin0higuchi

# ハンズオン概要

1. **Elastic Stack**を用いて、ログデータの分析を行う

2. **Elastic Stack 5.0**に触れる

3. **X-pack**を使ってみる

# ハンズオン概要

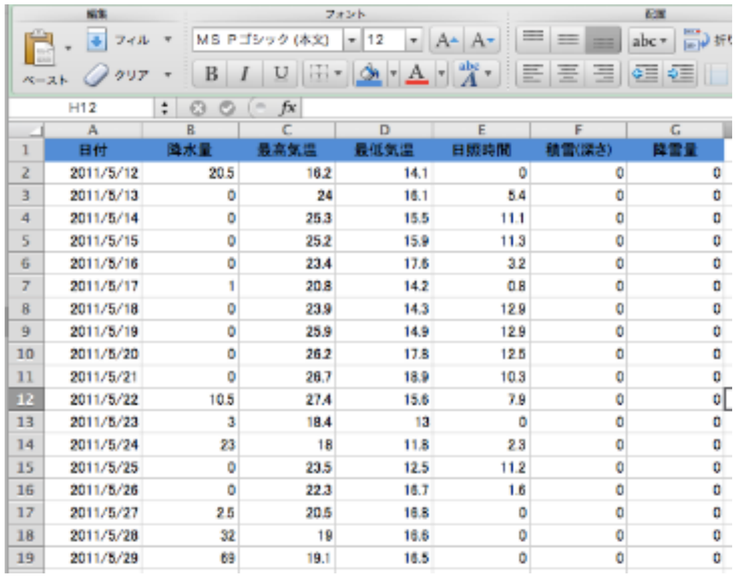
1. **Elastic Stack**を用いて、ログデータの分析を行う

2. **Elastic Stack 5.0**に触れる

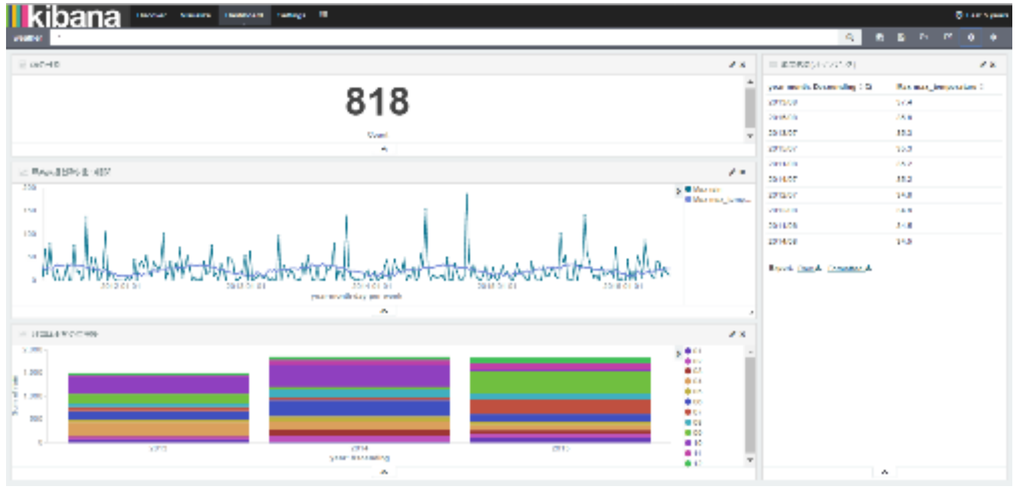
3. **X-pack**を使ってみる

# ハンズオン概要

- **Elastic Stack**を用いて、ログデータの分析を行う。

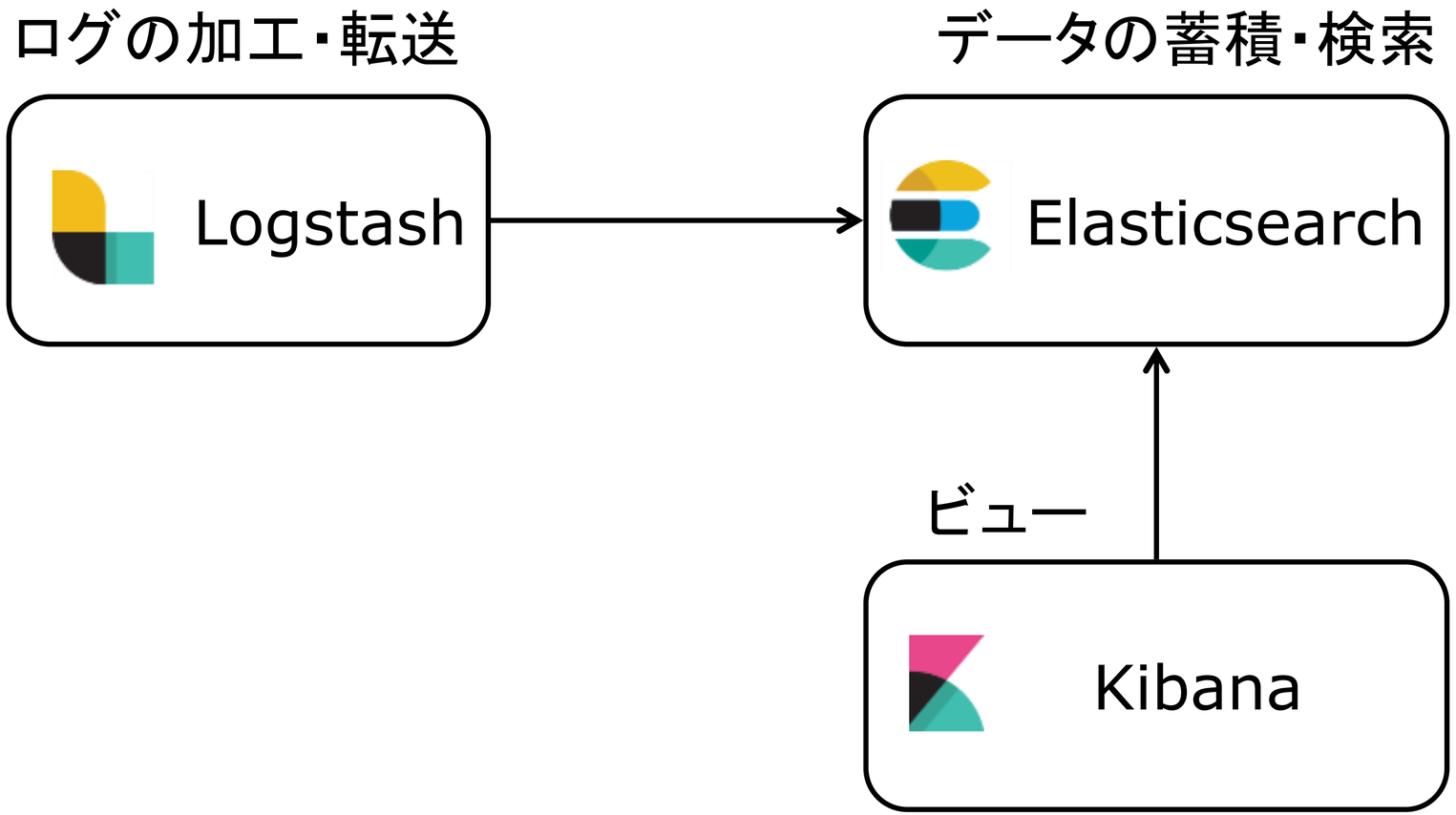


| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | 日付 | 降水量 | 最高気温 | 最低気温 | 日照時間 | 積雪(深さ) | 降雪量 |
| 2 | 2011/5/12 | 20.5 | 16.2 | 14.1 | 0 | 0 | 0 |
| 3 | 2011/5/13 | 0 | 24 | 16.1 | 5.4 | 0 | 0 |
| 4 | 2011/5/14 | 0 | 25.3 | 15.5 | 11.1 | 0 | 0 |
| 5 | 2011/5/15 | 0 | 25.2 | 15.9 | 11.3 | 0 | 0 |
| 6 | 2011/5/16 | 0 | 23.4 | 17.6 | 3.2 | 0 | 0 |
| 7 | 2011/5/17 | 1 | 20.8 | 14.2 | 0.8 | 0 | 0 |
| 8 | 2011/5/18 | 0 | 23.9 | 14.3 | 12.9 | 0 | 0 |
| 9 | 2011/5/19 | 0 | 25.9 | 14.9 | 12.9 | 0 | 0 |
| 10 | 2011/5/20 | 0 | 26.2 | 17.8 | 12.5 | 0 | 0 |
| 11 | 2011/5/21 | 0 | 26.7 | 18.9 | 10.3 | 0 | 0 |
| 12 | 2011/5/22 | 10.5 | 27.4 | 15.6 | 7.9 | 0 | 0 |
| 13 | 2011/5/23 | 3 | 18.4 | 13 | 0 | 0 | 0 |
| 14 | 2011/5/24 | 23 | 18 | 11.8 | 2.3 | 0 | 0 |
| 15 | 2011/5/25 | 0 | 23.5 | 12.5 | 11.2 | 0 | 0 |
| 16 | 2011/5/26 | 0 | 22.3 | 16.7 | 1.6 | 0 | 0 |
| 17 | 2011/5/27 | 2.5 | 20.5 | 16.8 | 0 | 0 | 0 |
| 18 | 2011/5/28 | 32 | 19 | 16.6 | 0 | 0 | 0 |
| 19 | 2011/5/29 | 69 | 19.1 | 16.5 | 0 | 0 | 0 |



# Elastic Stackとは

**Elastic**のオープンソース製品のこと。



# Elastic Stackとは

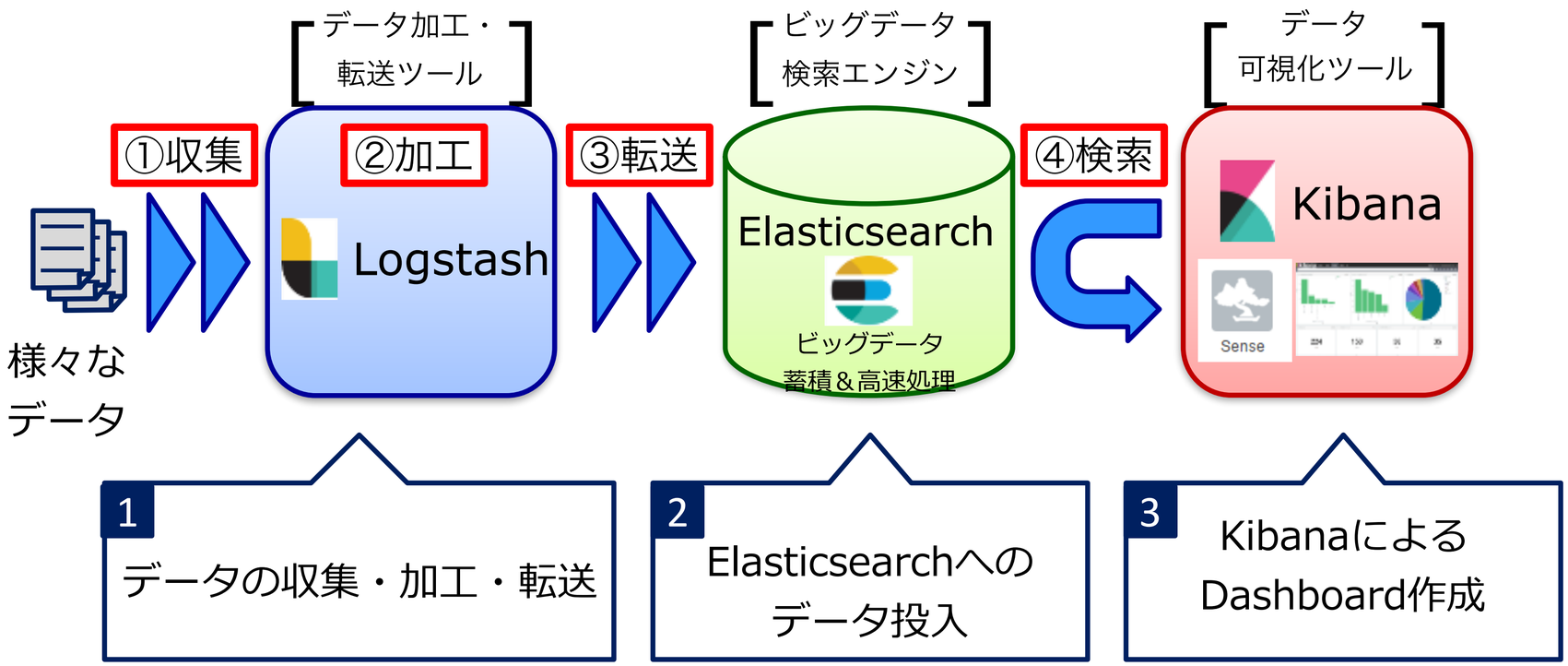
- **Beats** 

  データ収集に特化したサーバ常駐エージェント。
  GO言語で書かれており、軽量。

- **Elastic Cloud** 

  ElasticsearchをAWS上で構築・提供するサービス。
  商用プラグインなども利用することができる。

# ハンズオン概要



# ハンズオンの目的

1. **Elastic Stack**を用いて、ログデータの分析を行う

2. **Elastic Stack 5.0**に触れる

3. **X-pack**を使ってみる

# ハンズオンの目的

1. **Elastic Stack**を用いて、ログデータの分析を行う

2. **Elastic Stack 5.0**に触れる

3. **X-pack**を使ってみる

# ハンズオンの目的

- **X-packを使ってみる** 

拡張プラグインのパッケージ

**Security**：**ElasticStack**へのアクセスを制御するセキュリティプラグイン

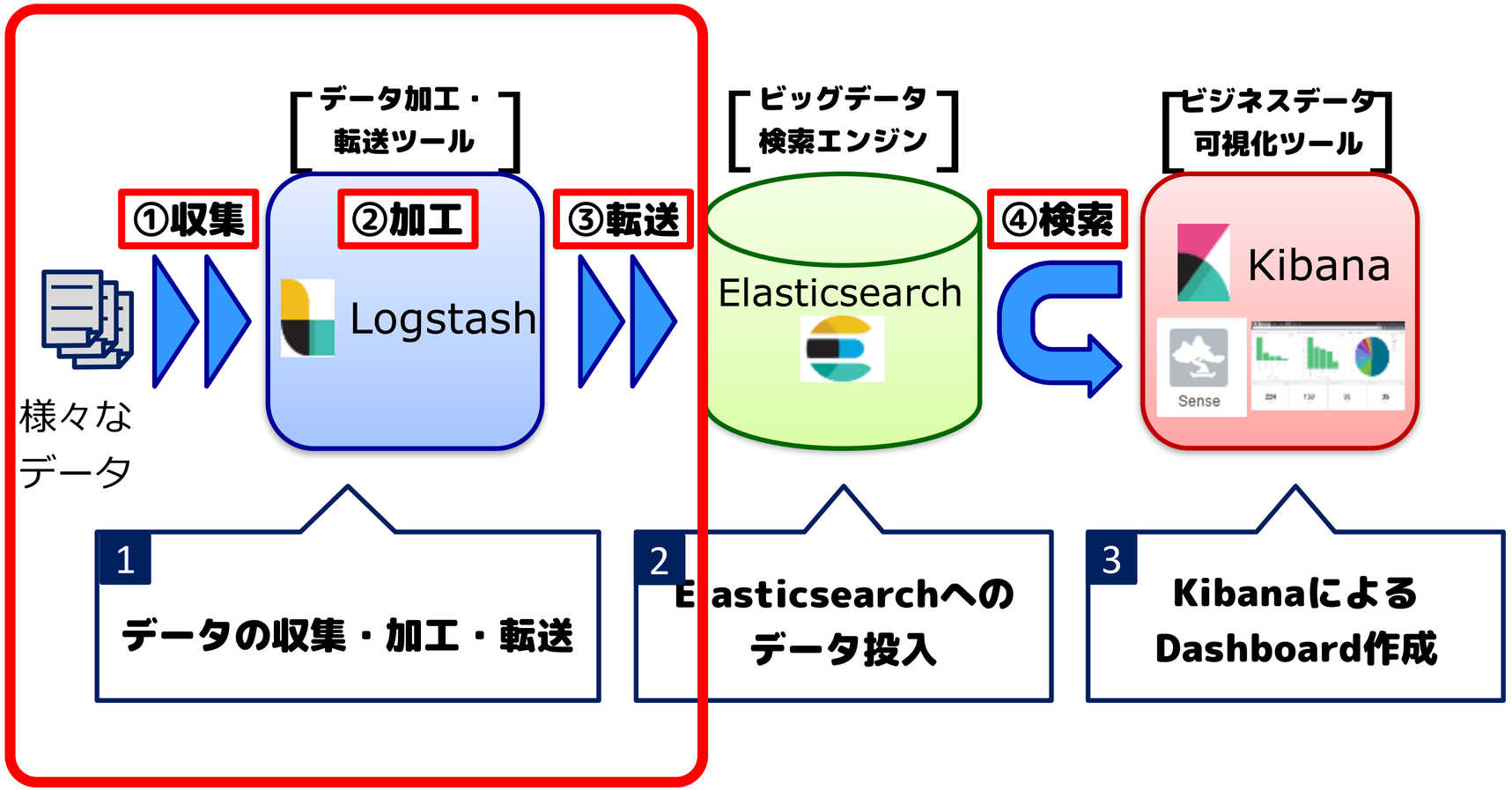
**Monitoring**：**ElasticStack**の稼働状況を監視する

**Alerting**：特定の条件を満たしたとき、メールなどでアラートをあげる

**Graph**：要素同士の関連を可視化する

**Reporting**：**Dashboard**などからレポートを作成する

# 【再掲】全体フロー図



# Logstashによる
# データの収集・加工・転送

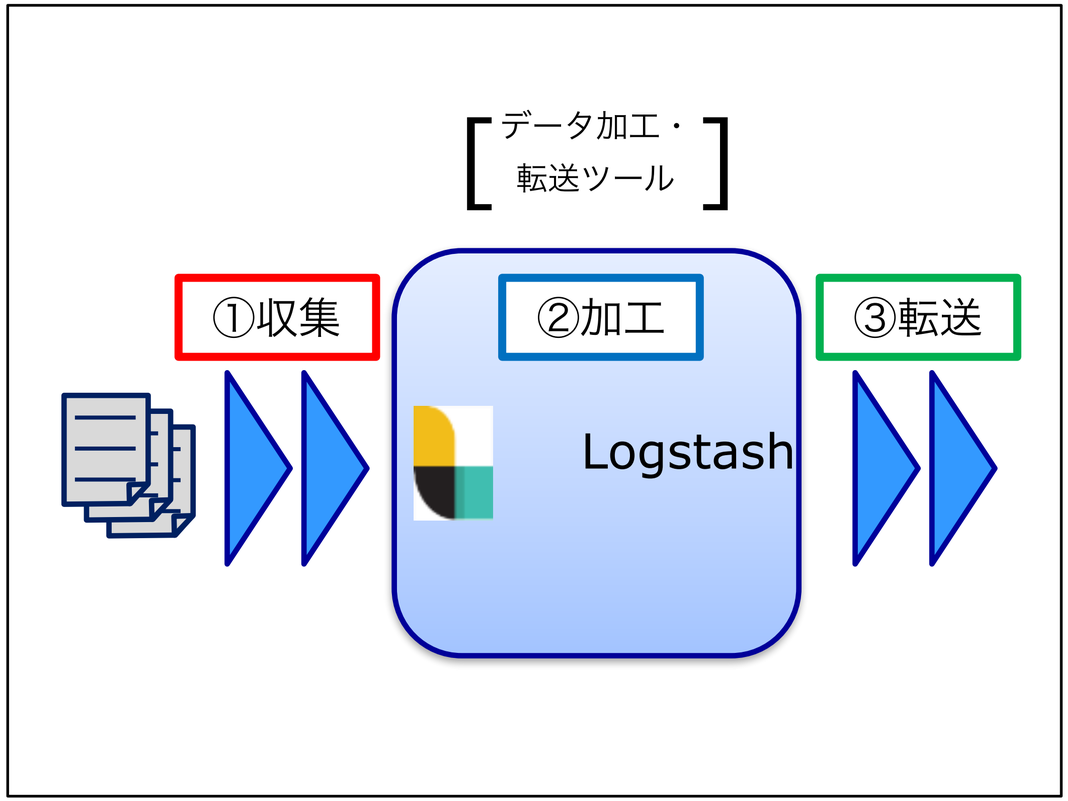
# Logstashとは

- Logstashは、ログを収集・加工し、転送する。(Elasticsearch以外への出力も可)

- 200以上もの公式プラグインが用意されており、DBやHDFS、S3、Twitterなどとも連携できる。

# 1-2. Logstashの処理の流れ

## フロー図（Logstash部分）




## 設定ファイルの例

```
input { stdin {} }                    ①

filter {}                             ②

output {                              ③
  stdout {
    codec => rubydebug
  }
}
```

# 設定例

```
input {
  stdin { }
}
filter {
  grok {
    match => {
      "message" => '%{IPORHOST:clientip} %{USER:ident} %{USER:auth} \[%{HTTPDATE:timestamp}\] "%
{WORD:verb} %{DATA:request} HTTP/%{NUMBER:httpversion}" %{NUMBER:response:int} (?:-|%
{NUMBER:bytes:int}) %{QS:referrer} %{QS:agent}'
    }
  }
  date {
    match => [ "timestamp", "dd/MMM/YYYY:HH:mm:ss Z" ]
  }
  geoip {
    source => "clientip"
  }
  useragent {
    source => "agent"
    target => "useragent"
  }
}
output {
  elasticsearch {
    hosts => "localhost"
  }
}
```

# Input

**Inputの設定例を以下に示す。**

- input {stdin{}}

- input {
  plugin {
    setting_1 => "value"
    array_2 => ["value1","value2"]
    hash_3 => { key => "value" }
    # comment
  }
}

# filter

```
filter {
  grok {
    match => {
      "message" => '%{IPORHOST:clientip} %{USER:ident} %{USER:auth} \[%
{HTTPDATE:timestamp}\] "%
{WORD:verb} %{DATA:request} HTTP/%{NUMBER:httpversion}" %{NUMBER:response:int} (?:-|%
{NUMBER:bytes:int}) %{QS:referrer} %{QS:agent}'
    }
  }
  date {
    match => [ "timestamp", "dd/MMM/YYYY:HH:mm:ss Z" ]
  }
  geoip {
    source => "clientip"
  }
  useragent {
    source => "agent"
    target => "useragent"
  }
}
```

# 演習１
# Logstashを使ってみよう！

# 演習1. Step2

- テキストエディタなどで**logstash1.conf**を開く

※ファイルのエンコードは**UTF-8**(BOM無し)にする。

# 演習1. Step2(設定ファイル内容)

```
input { stdin {} }


filter {}


output {
  stdout {
    codec => rubydebug
  }
}
```

# 演習1. Step3

作成したファイルでテスト実行を行う。

1. コマンドラインツールを起動する

2. **logstash-5.0.0**に移動する。
   - ① **cd ~/JJUG_fall/logstash-5.0.0 (Mac,Linux)**
   - ② **cd C:\JJUG_fall\logstash-5.0.0 (Windows)**
3. **bin/logstash -f logstash1.conf** と打ち込んで実行する

4. 最終行に
   **"Successfully started Logstash API endpoint {:port=>9600} "**
   と表示されれば**OK**

# 演習1. Step3(実行)

実行すると
" Successfully started Logstash API endpoint {:port=>9600}"
と表示された状態で出力がとまる(標準入力からの入力待ち状態)

画面例

```
------------------------------------------------------------------------
SHIN /Users/SHIN/AcroWorks/hands-on-workshop_temp/logstash-5.0.0-beta1 16-10-06 0:56:09
bin/logstash -f logstash.conf
The stdin plugin is now waiting for input:
[2016-10-06T00:56:29,679][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>"main", "p
ipeline.workers"=>4, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max_infli
ght"=>500}
[2016-10-06T00:56:29,702][INFO ][logstash.pipeline        ] Pipeline main started
[2016-10-06T00:56:29,833][INFO ][logstash.agent           ] Successfully started Logstash API e
ndpoint {:port=>9600}
```

# 演習1. Step3

文字列を入力してみる:

1. **Hello, world!** と入力して、**return**キーを押す
2. 以下のような結果が表示されれば成功

```
{
   "@timestamp" => 2016-10-05T16:08:32.486Z,
    "@version" => "1",
      "host" => “$HOST",
     "message" => "Hello, world!"
}
```

# 演習2. Filterの設定(grok filter)

目標:

ログの特定部分を抜き出すための**grok filter**を定義し、動作を確認する。

① 設定に**grok filter**を追加する。

② **grok filter**がどのような挙動を行うのか確認する。

# 演習2. Step1

- logstash2.confを開く

```
filter {
  grok {
    match => {
      "message" =>'%{HTTPDATE:timestamp} %{IP:ip} <%{DATA:msg}>'
    }
  }
}
```

# 演習2. Step2

作成したファイルでテスト実行を行う:

1. **logstash**が実行中の場合、**Ctrl+C**で終了する

2. **bin/logstash -f logstash2.conf**
   と打ち込んで実行する

3. **22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>**
   と打ち込んで実行する

4. 次ページのような結果が表示されれば成功

# 演習2. Step2

入力した文字列：22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>

出力例：

```
{
        "msg" => "This",
        "@timestamp" => 2016-10-08T02:21:37.761Z,
        "ip" => "183.60.215.50",
        "@version" => "1",
        "host" => "$HOST",
        "message" => "22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>",
        "timestamp" => "22/Mar/2014:16:38:00 -0700"
   }
```

# 演習2. Step2

入力した文字列：**22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>**


**grok**フィルターのパターン一覧：

**https://github.com/elastic/logstash/blob/v1.4.2/patterns/grok-patterns**

# 演習3. date

目標：

ログに記載されている時刻を利用して、
**timestamp**フィールドを設定する。

①**date filter**を設定する。

②**remove_field** の設定を追加し、不要なカラムを削除する。

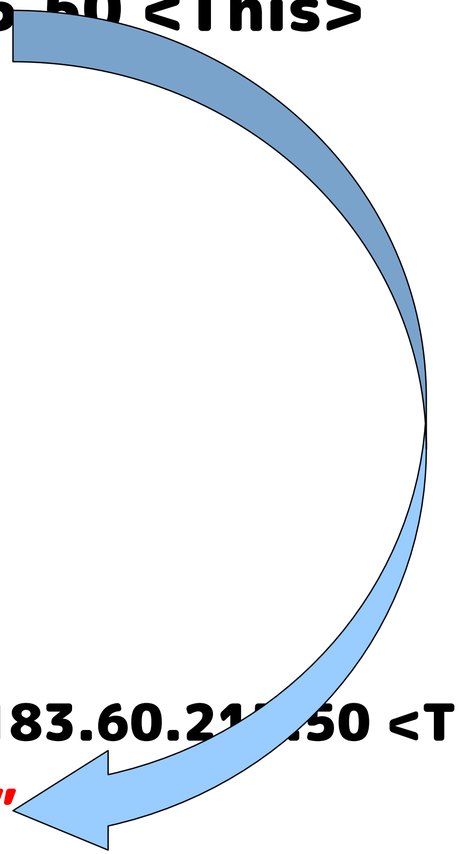
# 演習3. Step1

## logstash3.confを開く

```
date {
  match => [ "timestamp", "dd/MMM/YYYY:HH:mm:ss Z" ]
  locale => jp
}
```

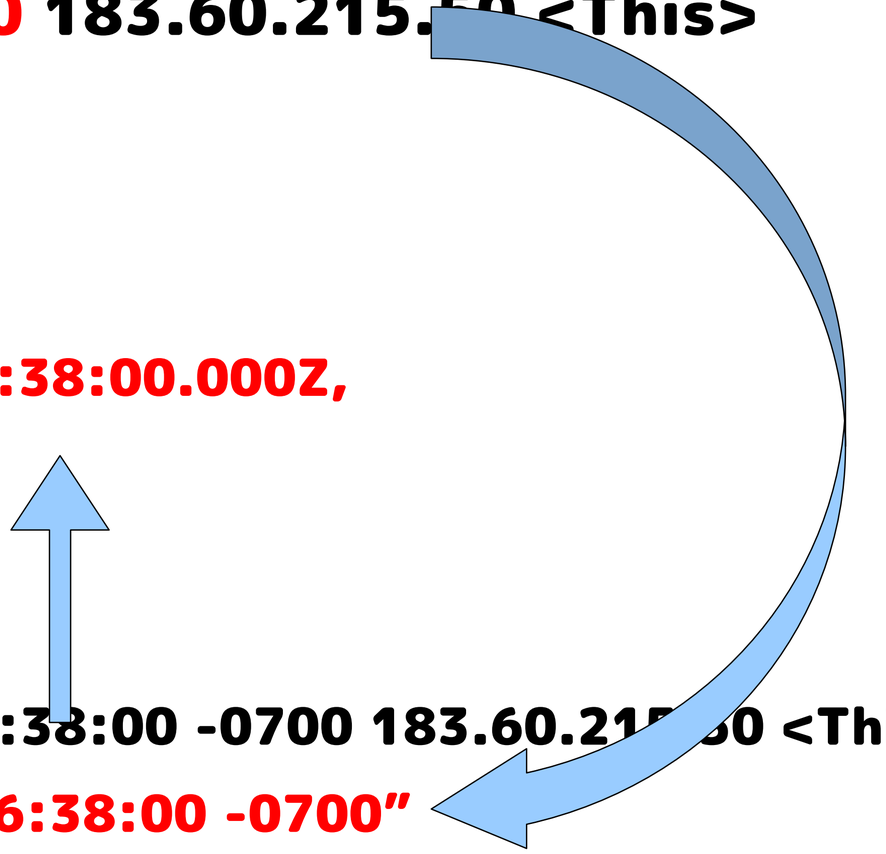
# 演習3. Step2

```
% bin/logstash -f logstash.conf
  22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>

{
          “msg” => “This”,
   “@timestamp” => 2014-03-22T23:38:00.000Z,
           “ip” => “183.60.215.50”,
     “@version” => “1”,
         “host” => “HIGUCHI-2.local”,
      “message” => “22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>”,
    “timestamp” => “22/Mar/2014:16:38:00 -0700”
}
```



# 演習3. Step2

```
% bin/logstash -f logstash.conf
  22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>

{
          “msg” => “This”,
   “@timestamp” => 2014-03-22T23:38:00.000Z,
           “ip” => “183.60.215.50”,
     “@version” => “1”,
         “host” => “HIGUCHI-2.local”,
      “message” => “22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>”,
    “timestamp” => “22/Mar/2014:16:38:00 -0700”
}
```



# 演習4. geoip

目標：

**geoip filter**を設定し、**IP**アドレスから地図情報を設定する。

①**geoip filter**を設定する。

②**geoip filter**を設定することで、緯度・経度・地域情報が取得できることを確認する。

# 演習4. Step1

**logstash4.confを開く。**

```
geoip {
  source => “ip”
}
```

# 演習4. Step1

実行した結果を以下に示します。

```
{
   "msg" => "This",
   "@timestamp" => 2014-03-22T23:38:00.000Z,
   "geoip" => {
      ... <this part is on next slide> ...
   }
   "ip" => "183.60.215.50",
   "@version" => "1",
   "host" => "$HOST",
   "message" => "22/Mar/2014:16:38:00 -0700 183.60.215.50 <This>"
}
```

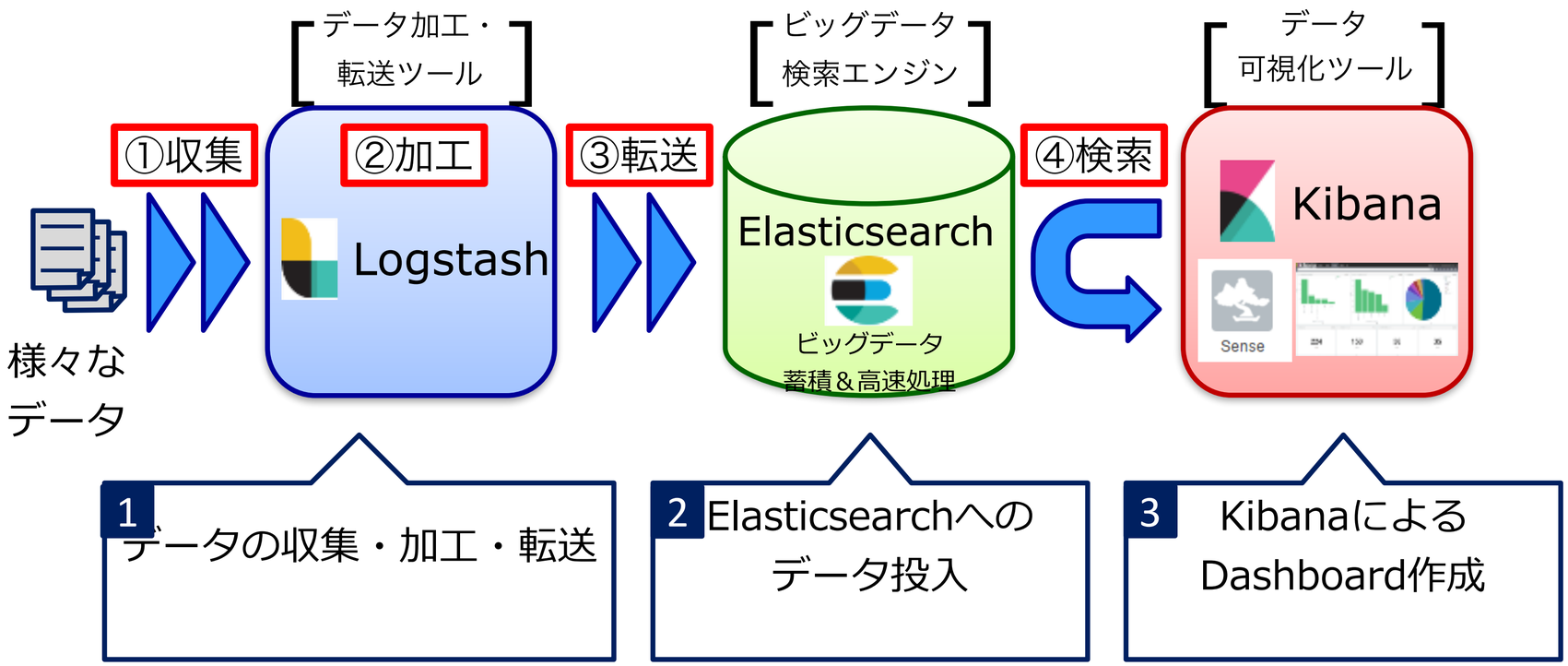
# 演習4. Step1 出力例

```
      "geoip"  => {
            "timezone"  => "Asia/Shanghai",
                  "ip"  => "183.60.215.50",
            "latitude"  => 23.1167,
      "continent_code"  => "AS",
           "city_name"  => "Guangzhou",
        "country_code2"  => "CN",
         "country_name"  => "China",
            "dma_code"  => nil,
        "country_code3"  => "CN",
          "region_name"  => "Guangdong",
            "location"  => [
          [0] 113.25,
          [1] 23.1167
      ],
         "postal_code"  => nil,
           "longitude"  => 113.25,
         "region_code"  => "44"
},
```

# Elasticsearch

## ~Store, Search, Analyze~

# ハンズオン概要



# 演習

# Elasticsearch,Kibanaのセットアップ

# Elasticsearchの起動

1. カレントディレクトリを移動する

   コマンド例：

   **$ cd ~/JJUG_fall/elasticsearch-5.0.0**

2. **Elasticsearchを起動する**

   **$ bin/elasticsearch**

# Elasticsearchの起動確認

- ブラウザ（**Chrome or Safari**）でhttp://localhost:9200を開く。

  **user:elastic , password:changeme**

- 以下の内容が表示されること。

```
{
  "name" : "1_IyK7k",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "0Lwk8ZDITCOTXLfm9mO5Eg",
  "version" : {
    "number" : "5.0.0",
    "build_hash" : "7eb6260",
    "build_date" : "2016-09-20T23:10:37.942Z",
    "build_snapshot" : false,
    "lucene_version" : "6.2.0"
  },
  "tagline" : "You Know, for Search"
}
```

# Kibana

# Kibanaの起動

- 新しくコマンドラインツールのウィンドウを開き、
カレントディレクトリを移動する

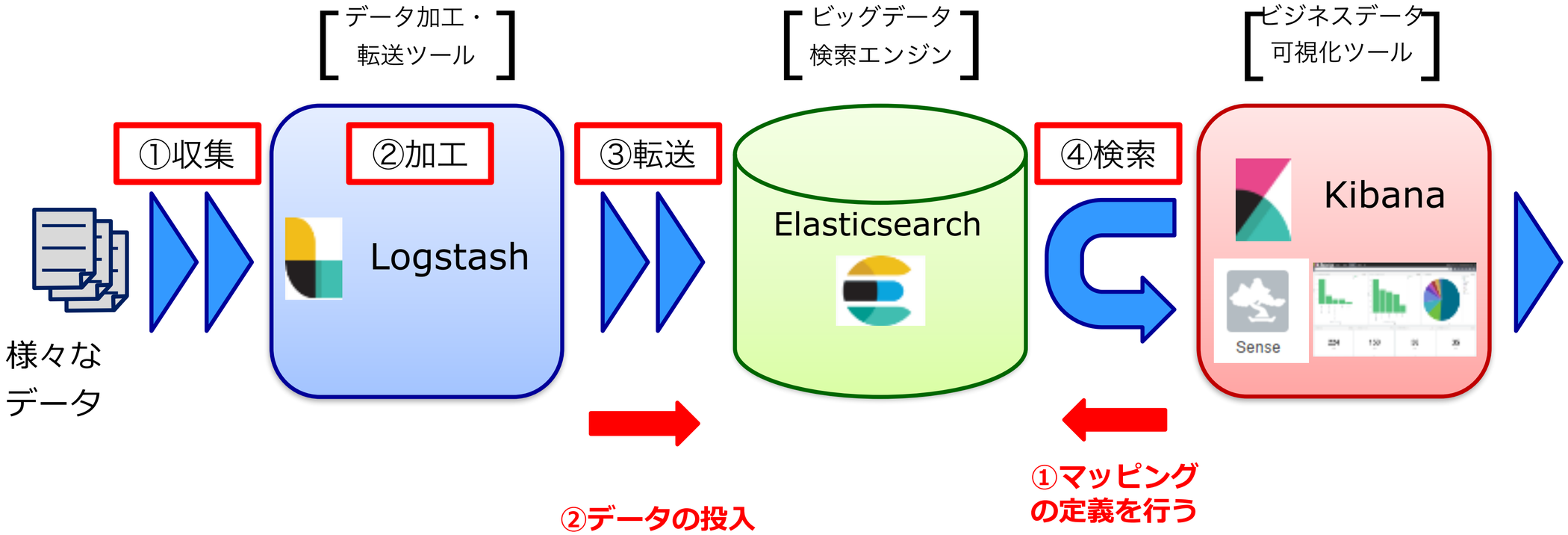
**$ cd ~/JJUG_fall/kibana-5.0.0-darwin-x86_64**

**※Macの場合のコマンド例**

- **Kibana**を起動する

**$ bin/kibana**

# 全体の流れの確認



# Kibanaとは?

1. **Log**データの**Dashboard**
2. さまざまな種類のグラフが用意されている
3. 分かりやすい**UI**で、初心者でも可視化できる
4. 作成した**Dashboard**は、共有できる
5. **Web**ブラウザでアクセス可能
   - モダンなブラウザが必要
   - **Chrome/Safari** 推奨

# 解析するログの説明


## Quick Draw

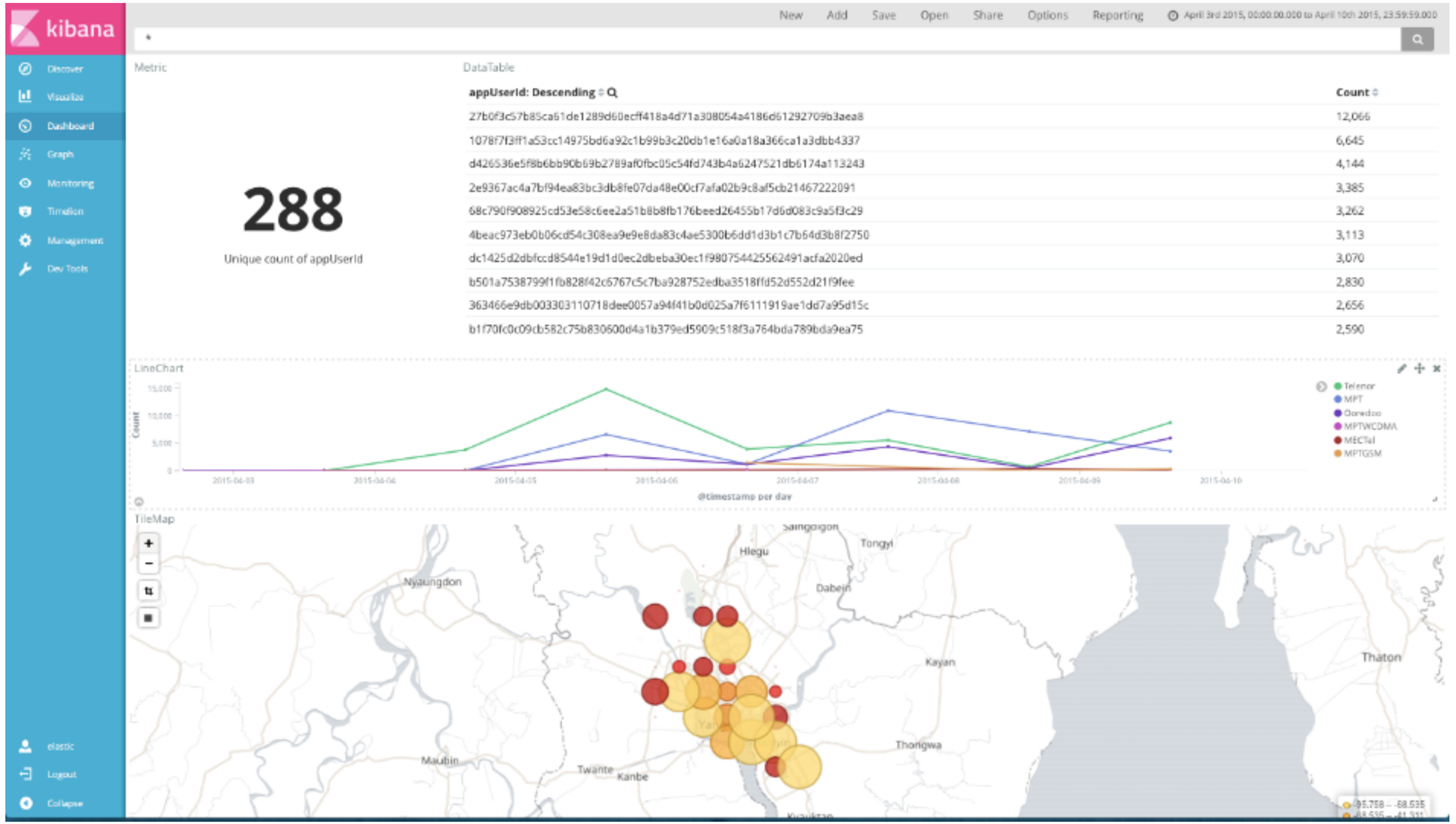
① ミャンマーで利用できる、スマホアプリ。
② スマホを操作すると、現在地の通信状況をサーバに送付する。
③ ユーザは、データを送付することで、景品が取得できるくじを引くことができる。

# ログのデータ形式

## 1. Quick Drawで取得できる項目の一部

| 項目名 | 内容 |
| --- | --- |
| appUserId | アプリケーションのユーザID |
| @timestamp | ログ時刻 |
| gender | 性別 |
| carrier | 通信キャリア<br>(Ooredoo, MPT, Telenor, 取得不可(null)) |
| networkBean3G_neighboringCellInfo_rssi | 3G回線の通信強度 |

# Kibana Dashboard



# マッピング定義を行う

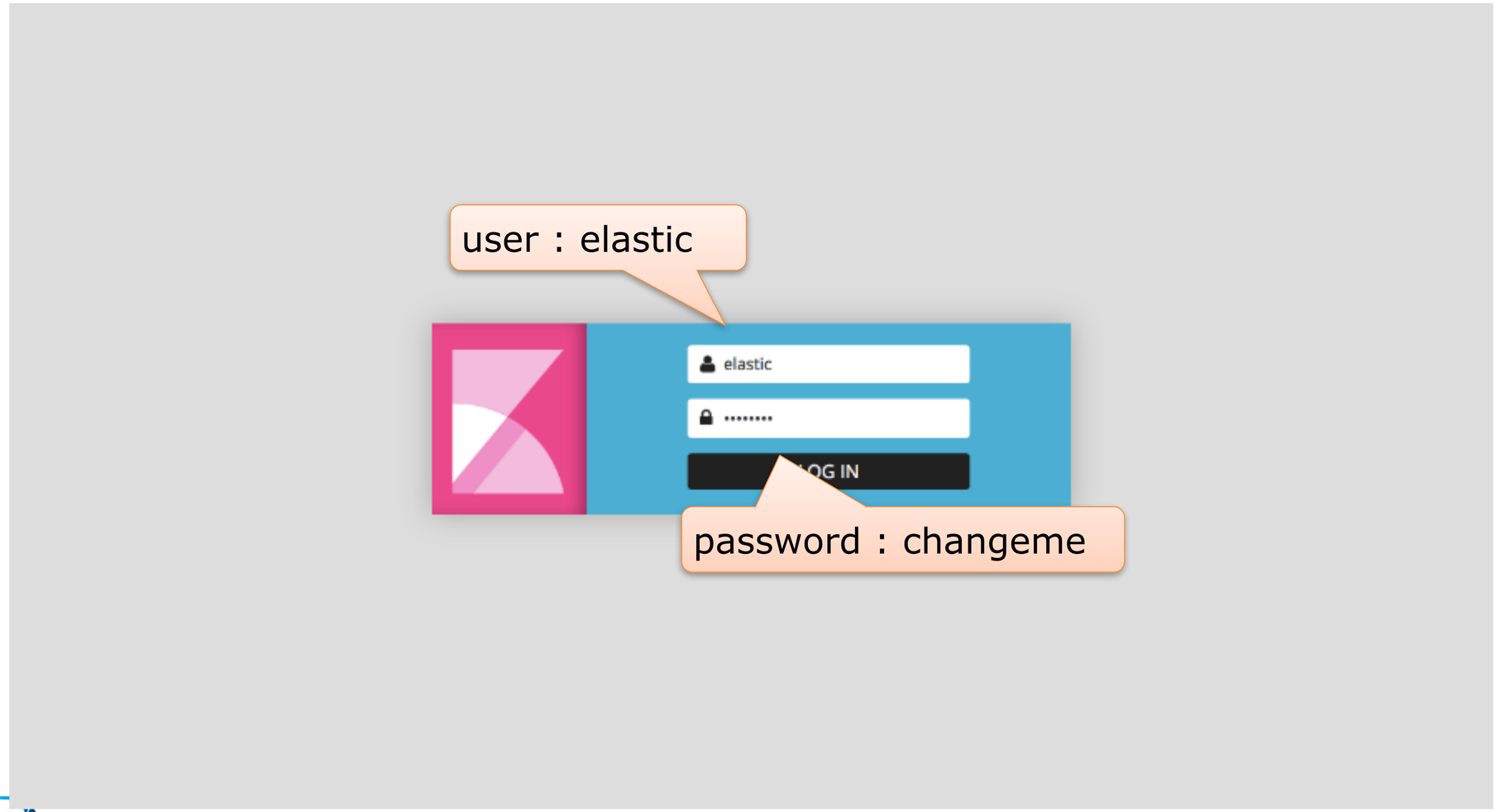
- マッピング定義は、**Elasticsearch**に保存するドキュメントのデータ構造を設定するもの。

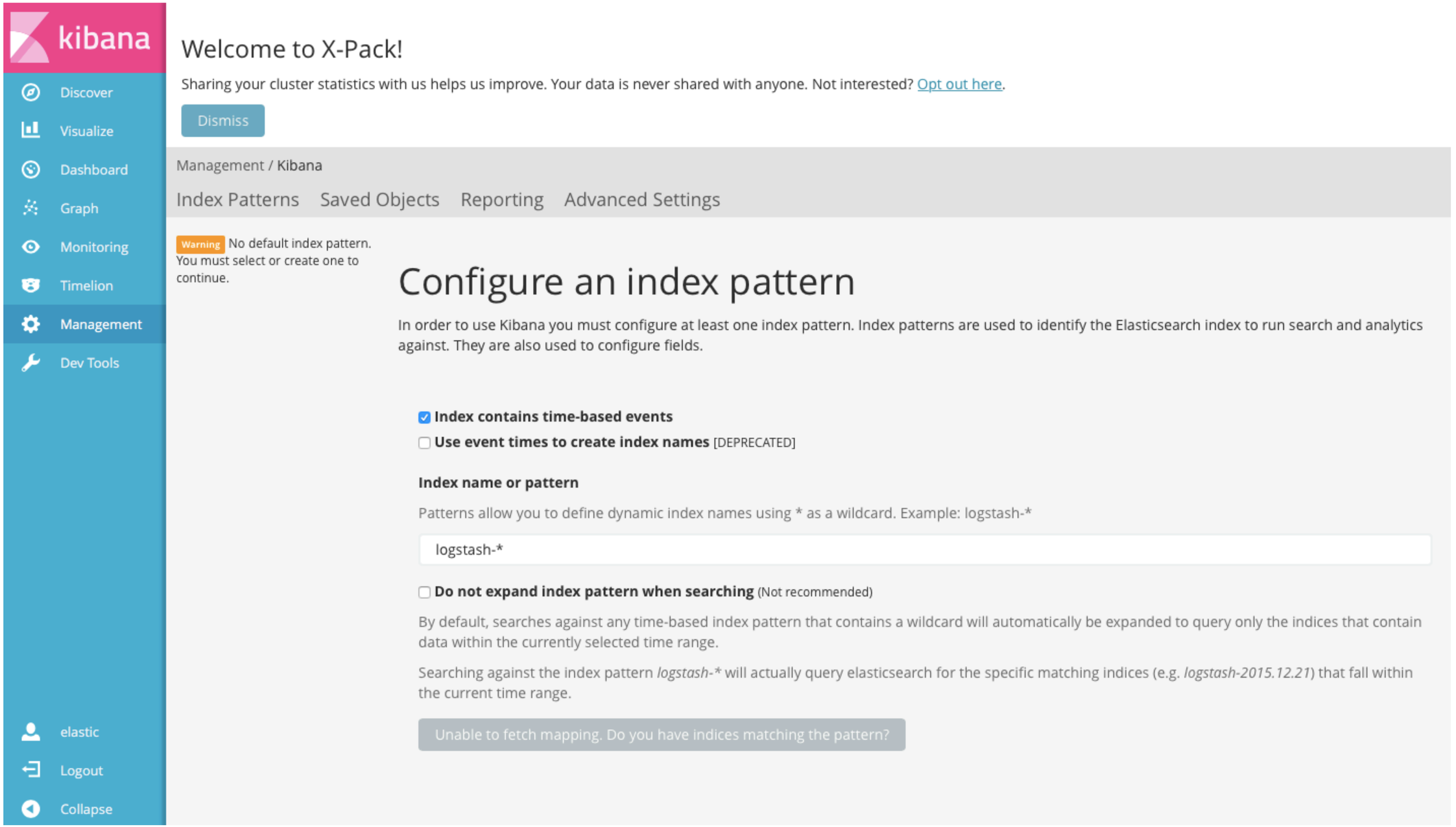
- 以下の**URL**から**Kibana**にアクセスし**Console**を用いてマッピング定義を行う。

**http://localhost:5601**

# Kibanaログイン

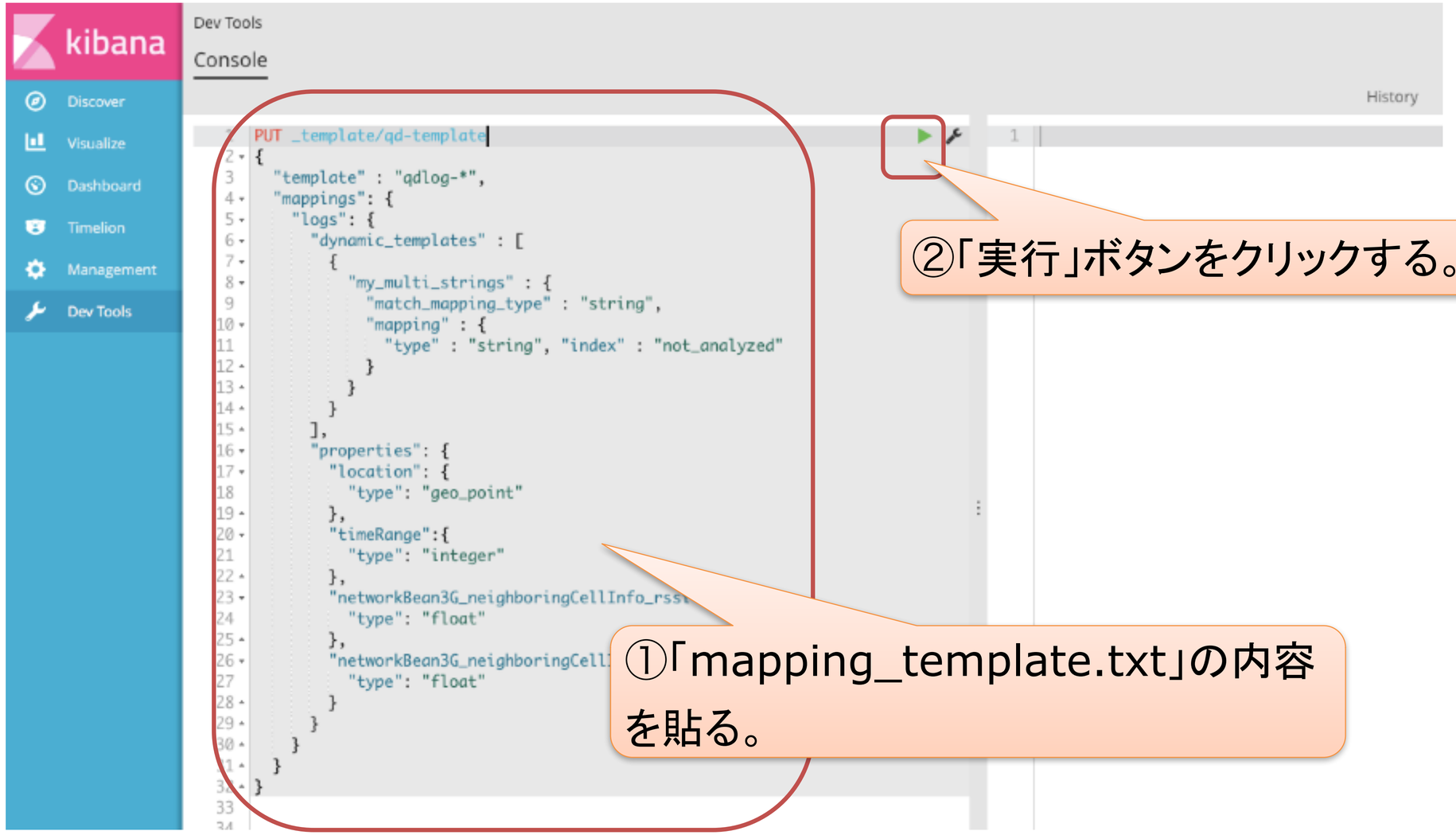
・x-packをインストールすると認証がかかる




kibana
Discover
Visualize
Dashboard
Graph
Monitoring
Timelion
Management
Dev Tools
elastic
Logout
Collapse
Welcome to X-Pack!
Sharing your cluster statistics with us helps us improve. Your data is never shared with anyone. Not interested? Opt out here.
Dismiss
Management / Kibana
Index Patterns
Saved Objects
Reporting
Advanced Settings
Warning No default index pattern. You must select or create one to continue.
Configure an index pattern
In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.
Index contains time-based events
Use event times to create index names [DEPRECATED]
Index name or pattern
Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*
logstash-*
Do not expand index pattern when searching (Not recommended)
By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.
Searching against the index pattern logstash-* will actually query elasticsearch for the specific matching indices (e.g. logstash-2015.12.21) that fall within the current time range.
Unable to fetch mapping. Do you have indices matching the pattern?

# LogをElasticsearchに送る(マッピング定義)

# LogをElasticsearchに送る(マッピング定義)



# LogをElasticsearchに送る(マッピング定義)

# LogをElasticsearchに送る

- **Logstashの設定ファイルは"complete.conf"を使用**

※JJUG_fall直下に移動してから以下のコマンドを実行する

**※Linux, Macの方**

```
% cat qdlog.csv | logstash-5.0.0/bin/logstash -f complete.conf
```

**※Windowsの方**

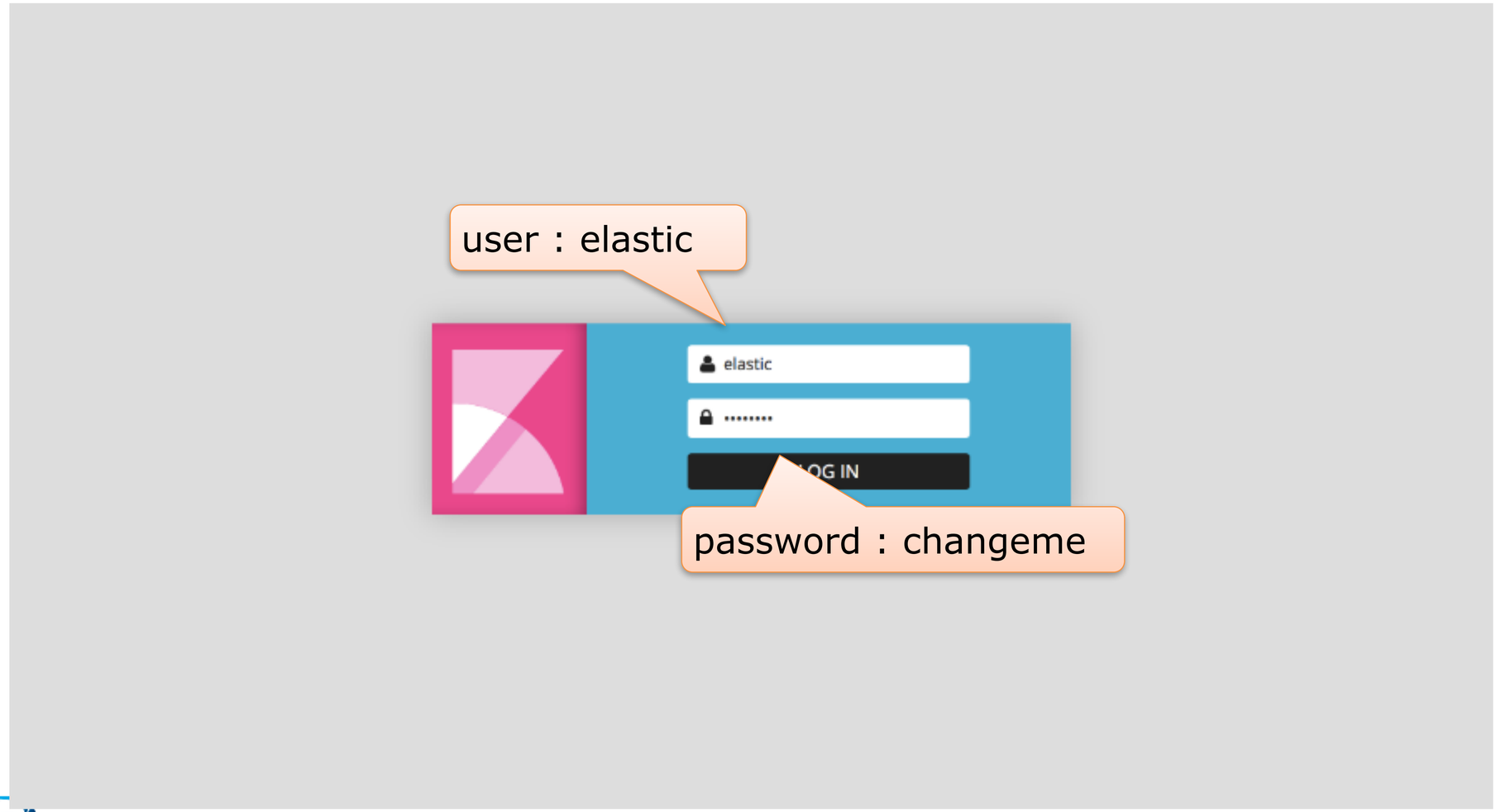
```
% type qdlog.csv | logstash-5.0.0\bin\logstash -f complete.conf
```

# 演習

# KibanaでDashboardを作成してみよう！

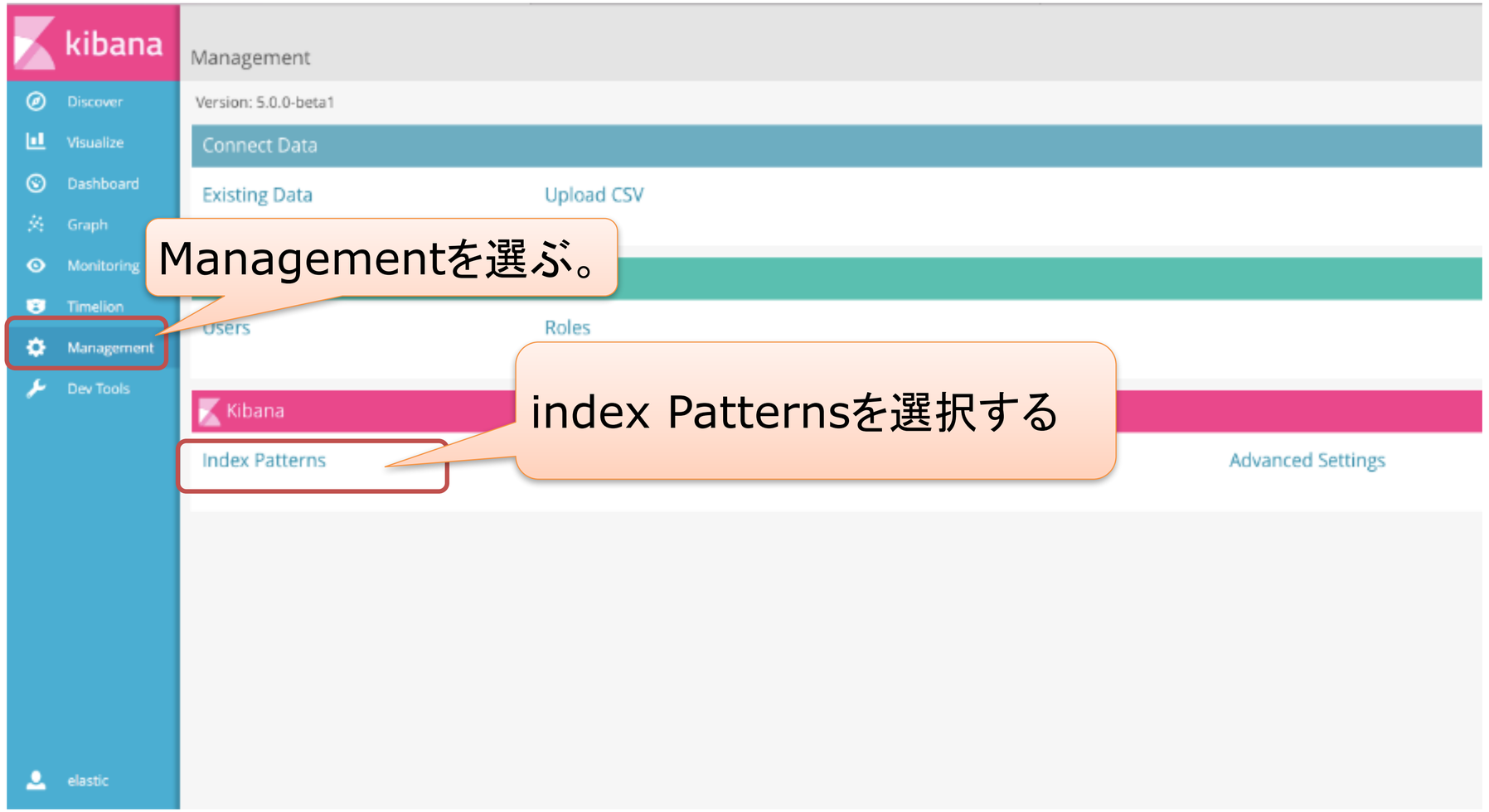
# Security

・x-packをインストールすると認証がかかる



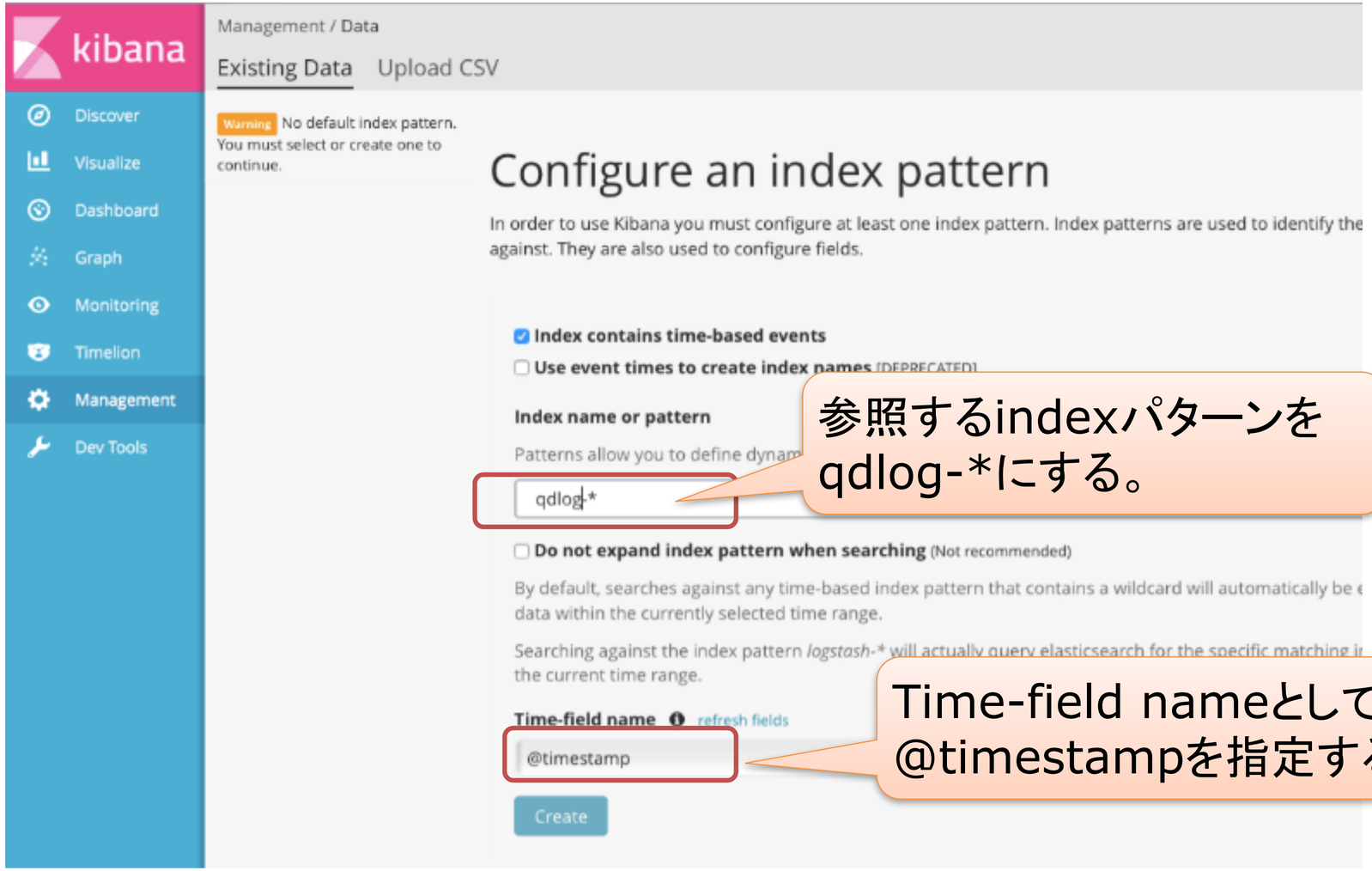
# Settings

index patternを設定する



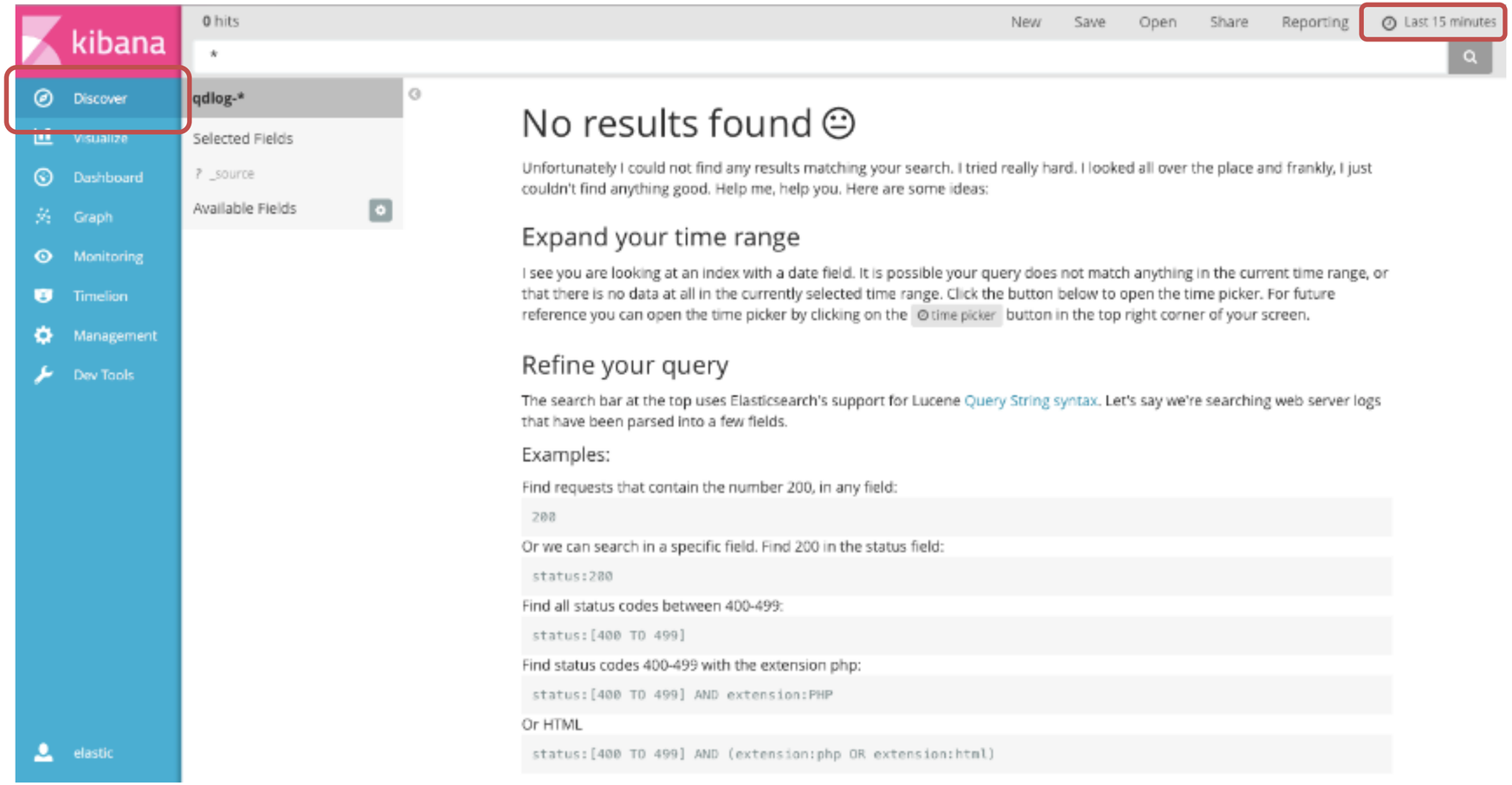
# Settings

index patternを設定する



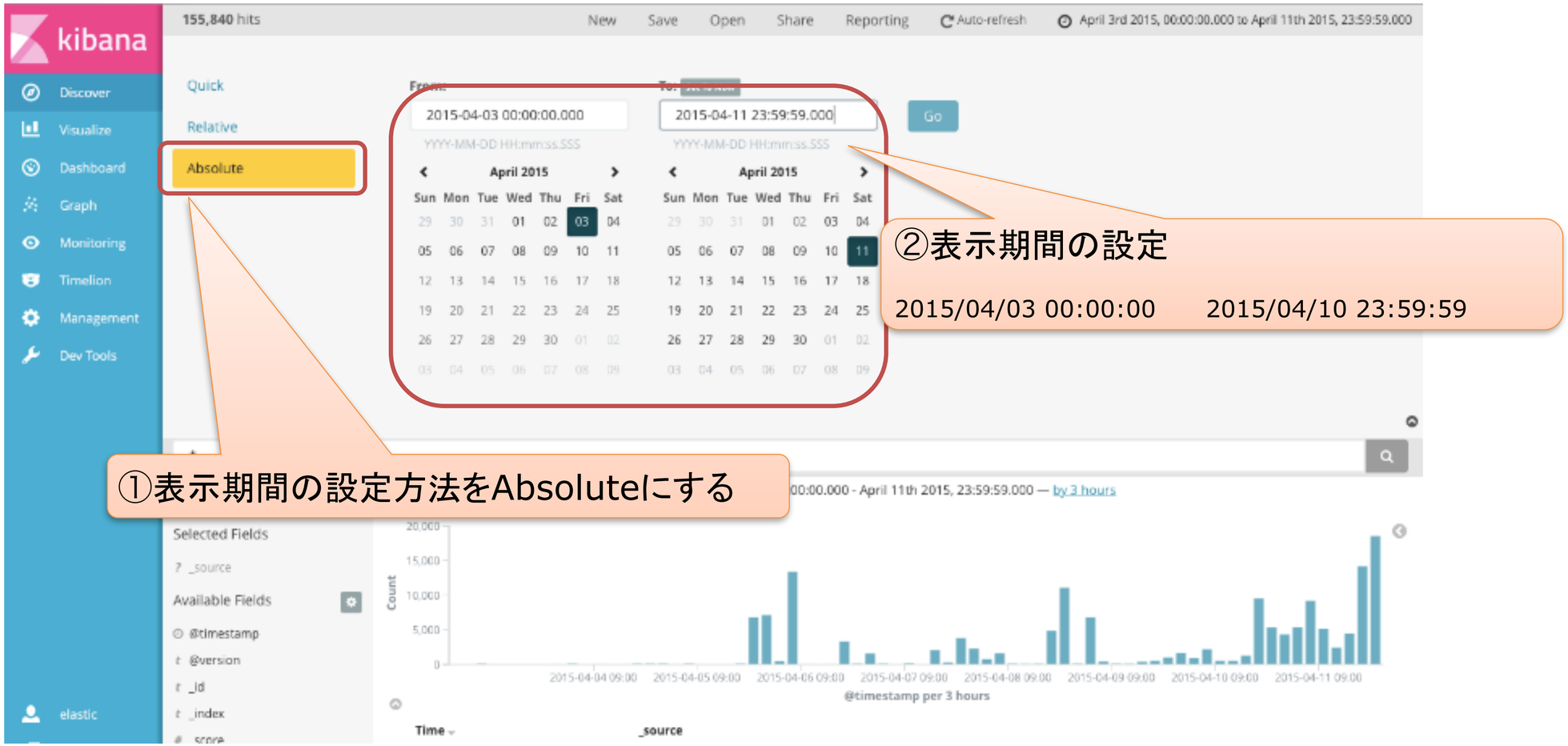
# Discover

1.Discoverを選択し、画面右上の時計マークをクリックする



# Discover

## 2.データの表示期間を設定する



# Visualizationの作成

## 1. 以下のVisualizationを作成する。

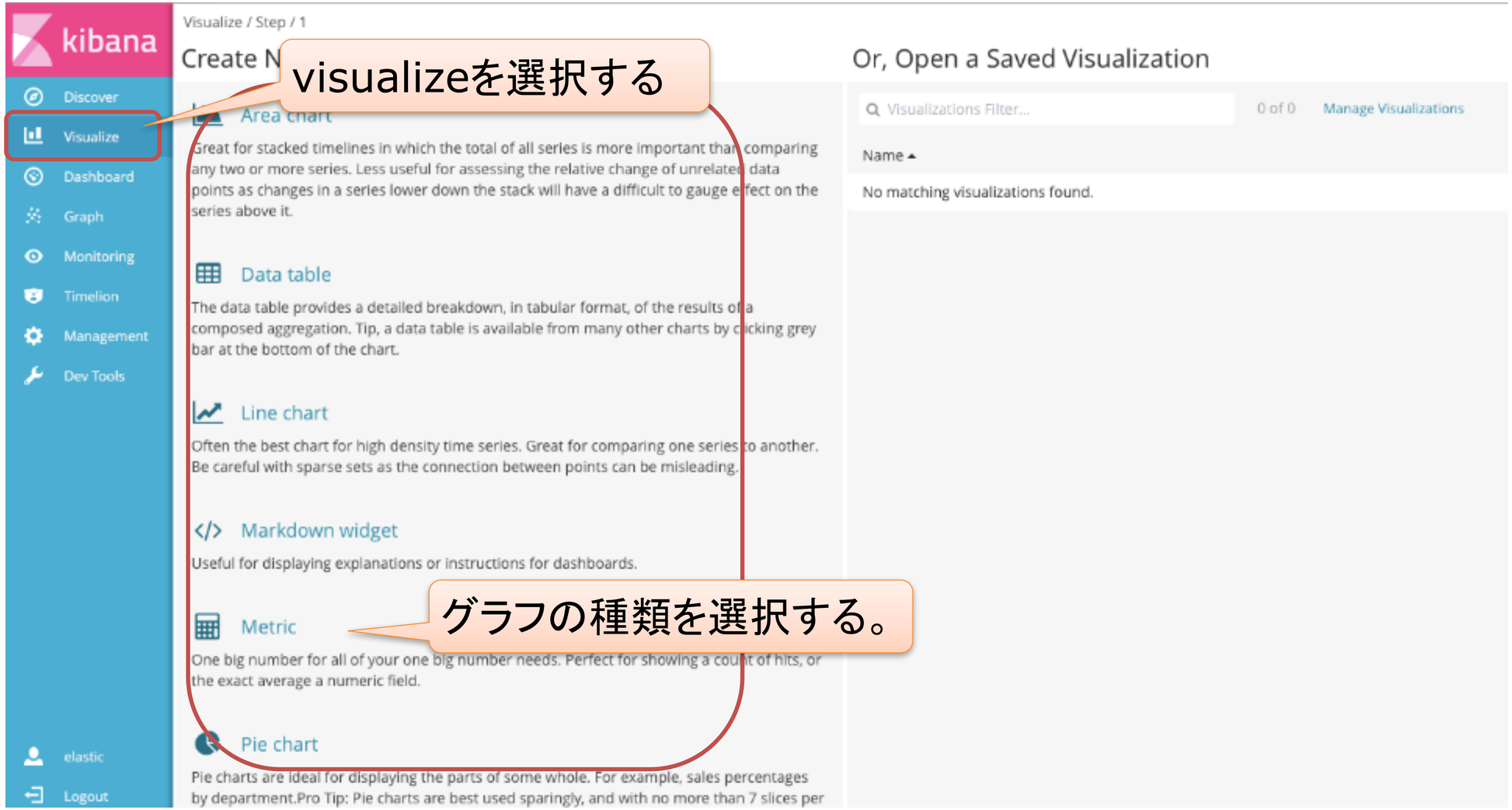
① 「**Metrics**」を利用し、ユニークユーザ数を表示する。

② 「**Data Tables**」を利用し、データ数の多い上位10ユーザの一覧を表示する。

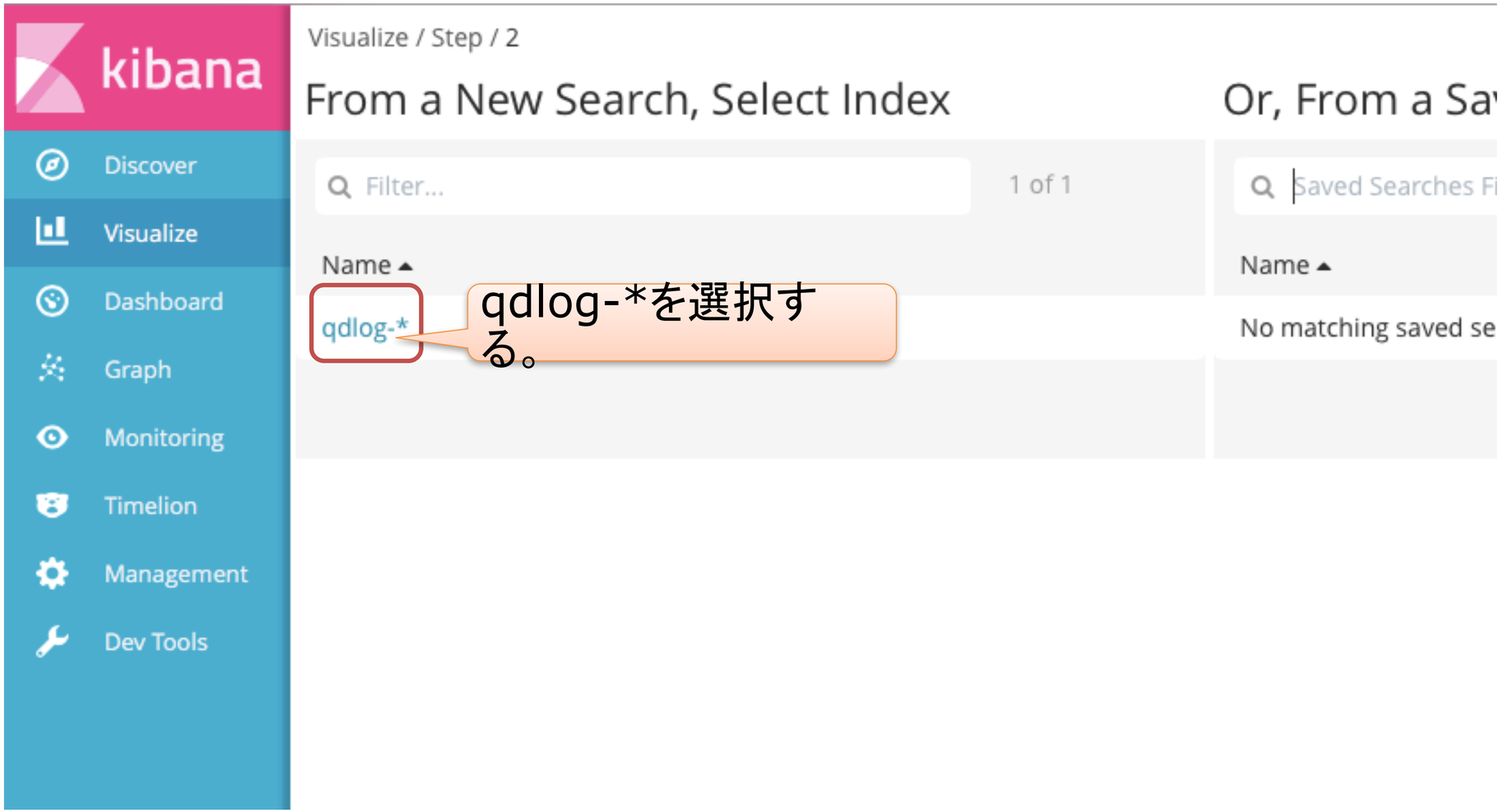
③ 「**Line Chart**」を利用し、日毎・キャリア毎のアクセス数を表示する。

④ 「**Tile Map**」を利用し、キャリア**MPT**の**3G**回線通信強度を地図上に可視化する

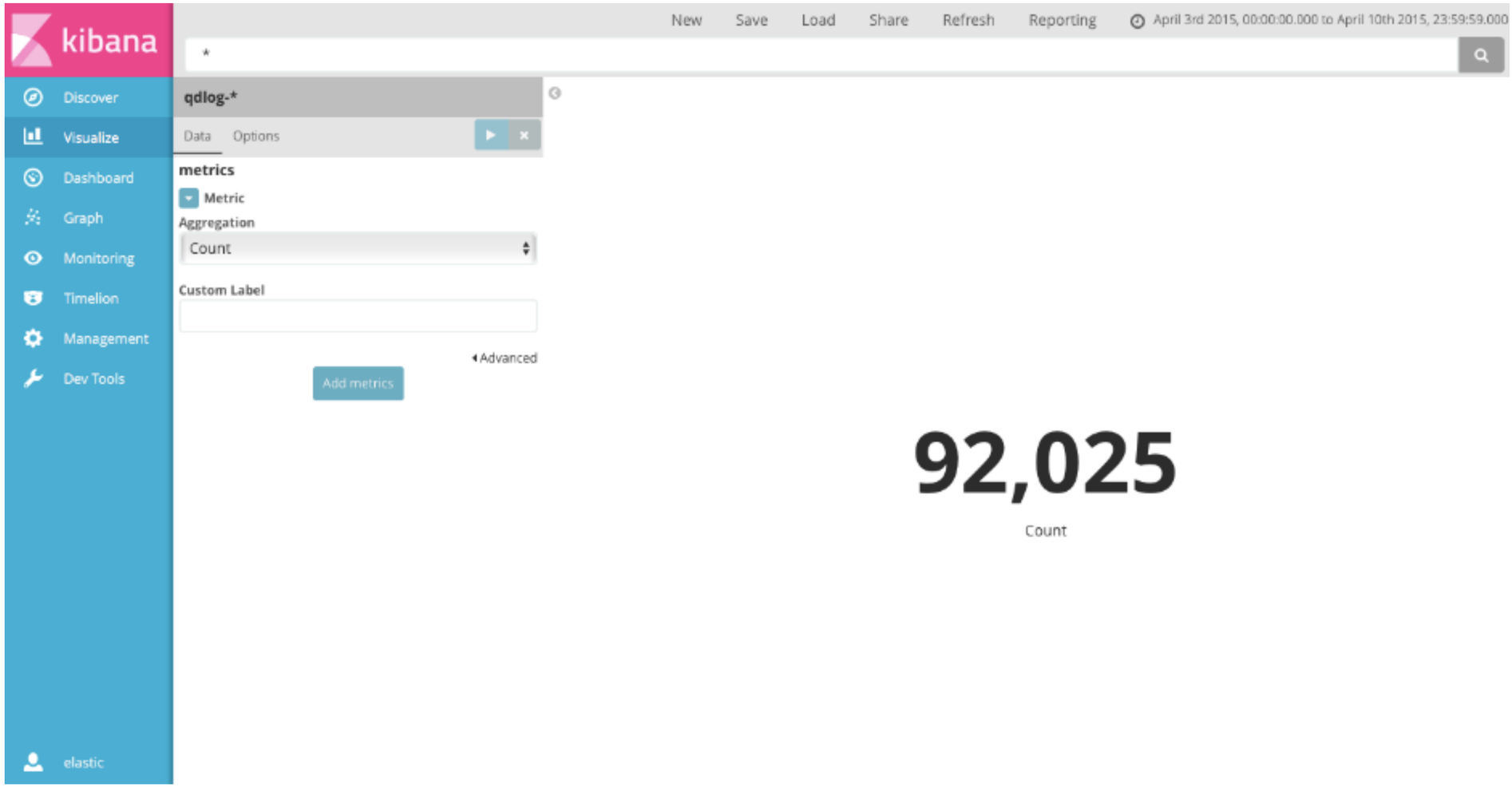
# Visualizeの作成方法



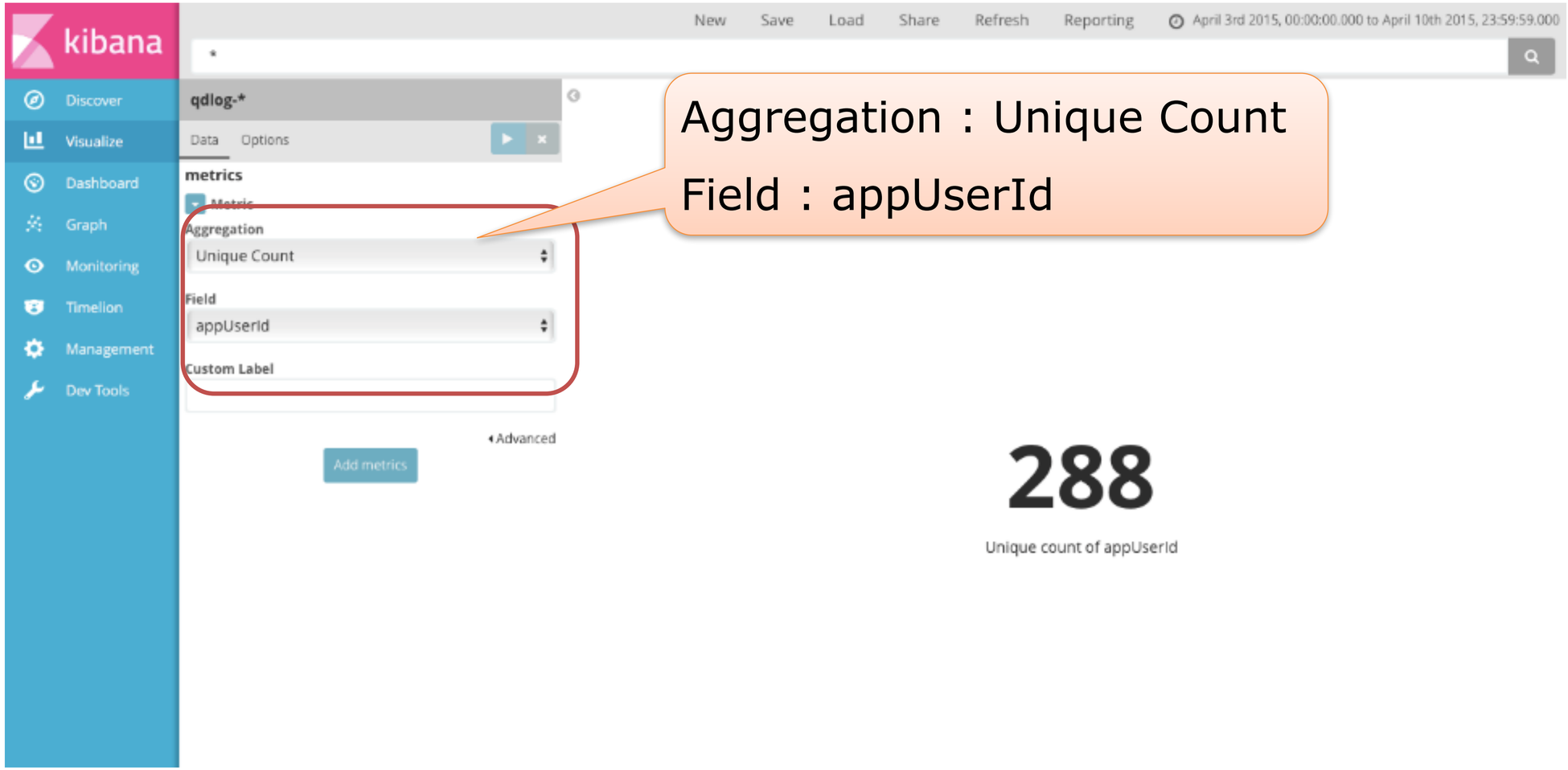
# Visualizationの作成



# Visualizationの作成



# Visualizationの作成



# Visualizationの作成

# Visualizationの作成

## 1. 以下のVisualizationを作成する。

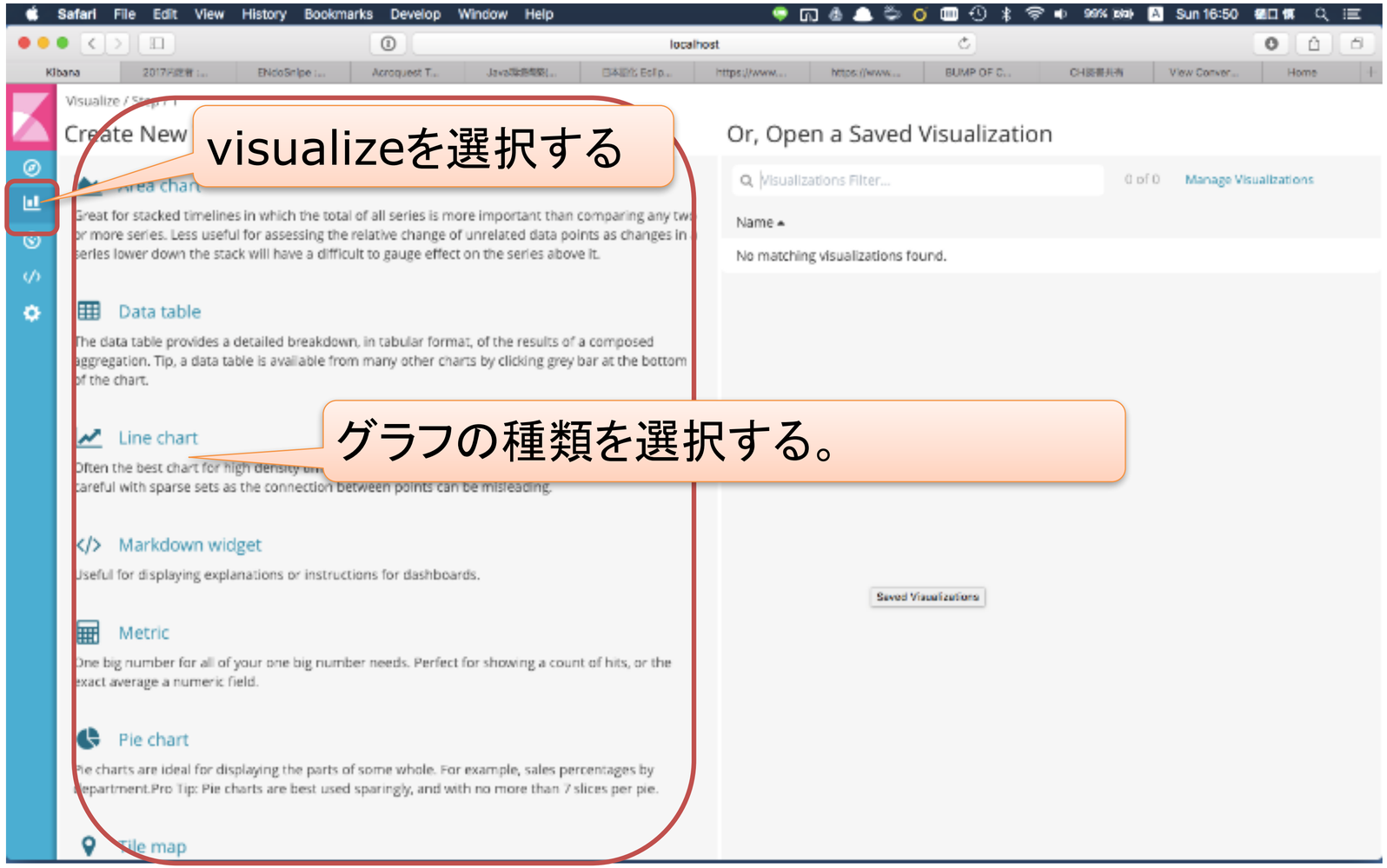
① 「**Metrics**」を利用し、ユニークユーザ数を表示する。

② 「**Data Tables**」を利用し、データ数の多い上位10ユーザの一覧を表示する。

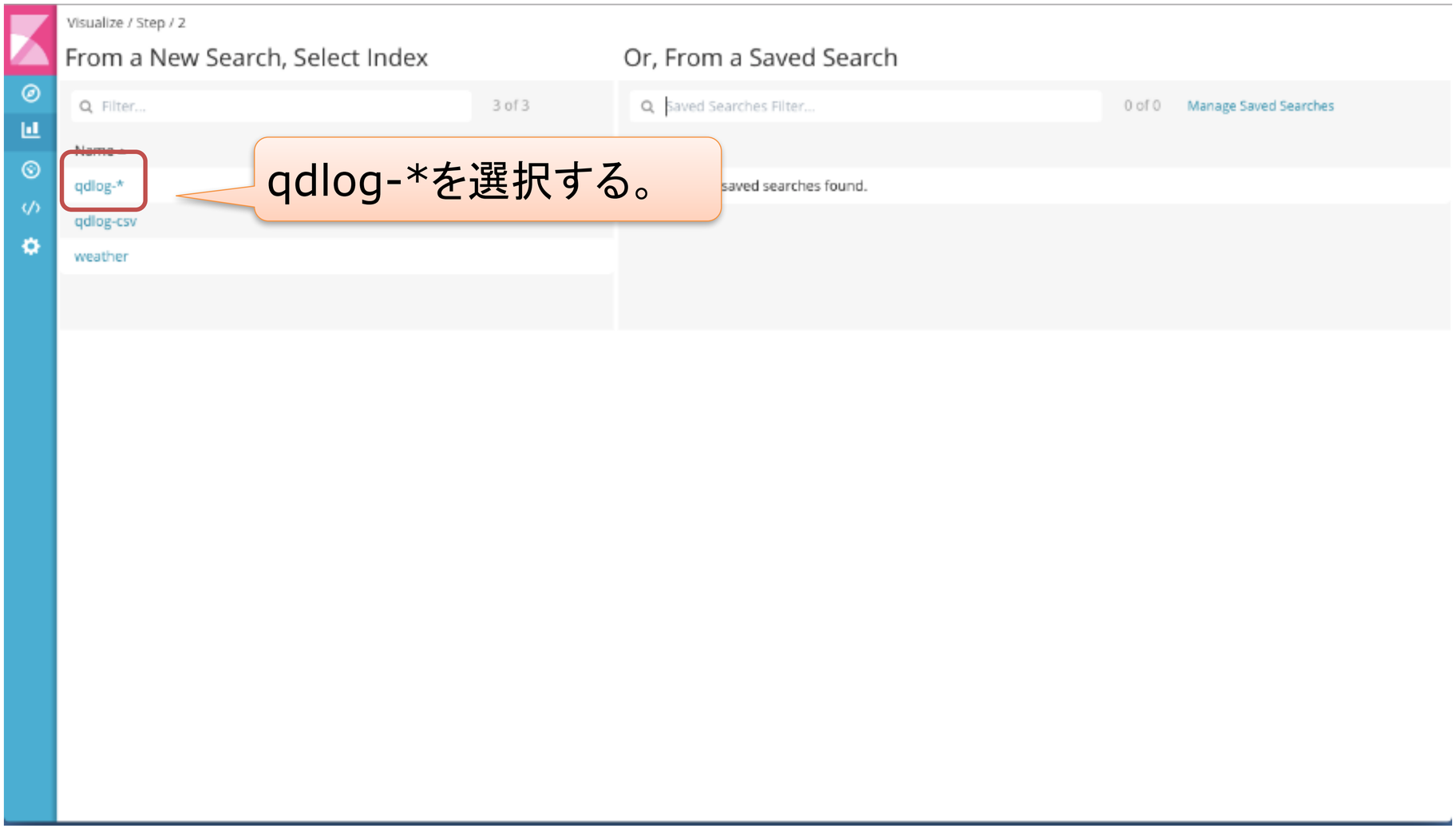
③ 「**Line Chart**」を利用し、日毎・キャリア毎のアクセス数を表示する。

④ 「**Tile Map**」を利用し、キャリア**MPT**の**3G**回線通信強度を地図上に可視化する。

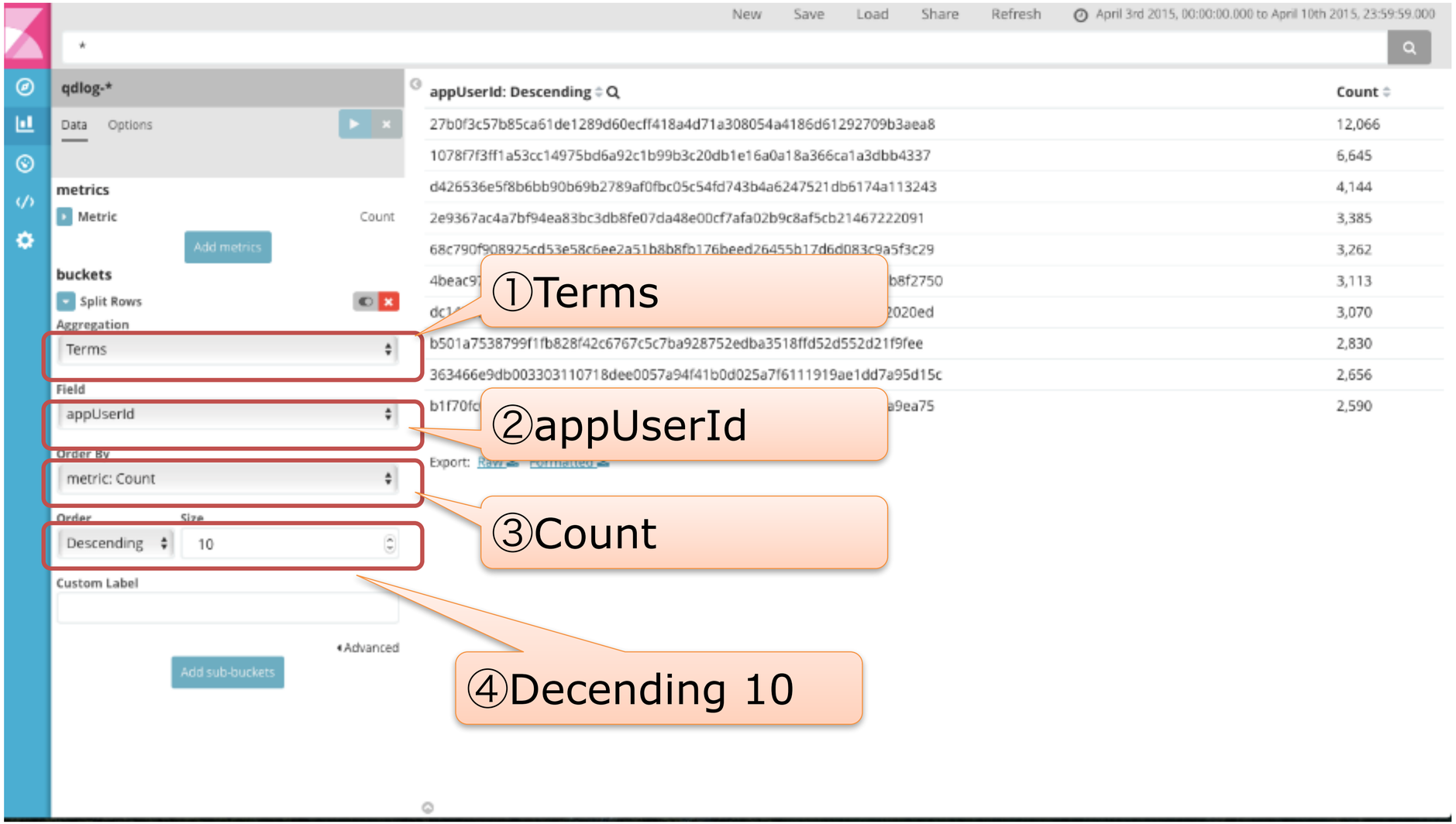
# Visualizationの作成



# Visualizationの作成



# Visualizationの作成(Data Tables)



# Visualizationの作成

## 1. 以下のVisualizationを作成する。

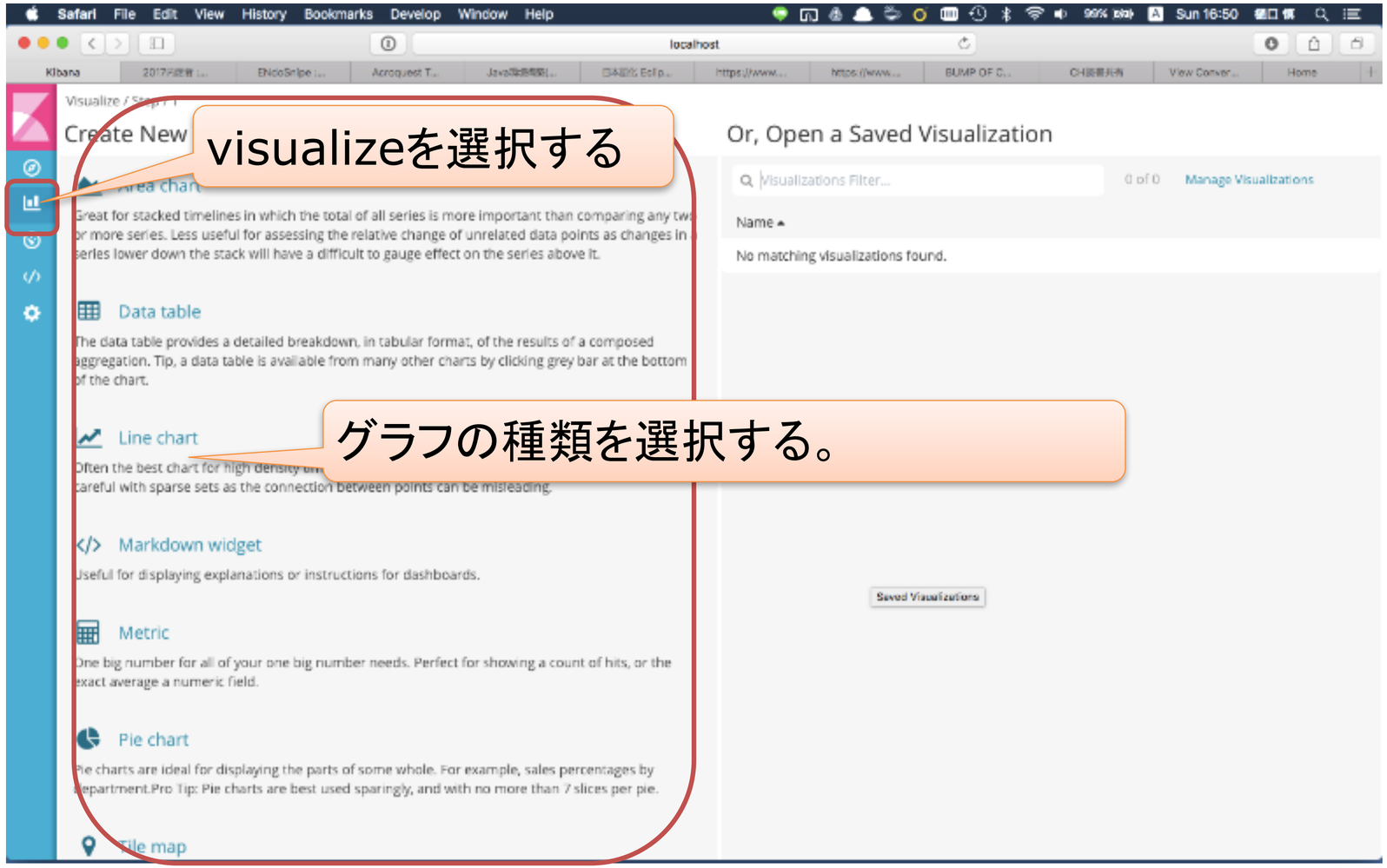
① 「**Metrics**」を利用し、ユニークユーザ数を表示する。

② 「**Data Tables**」を利用し、データ数の多い上位10ユーザの一覧を表示する。

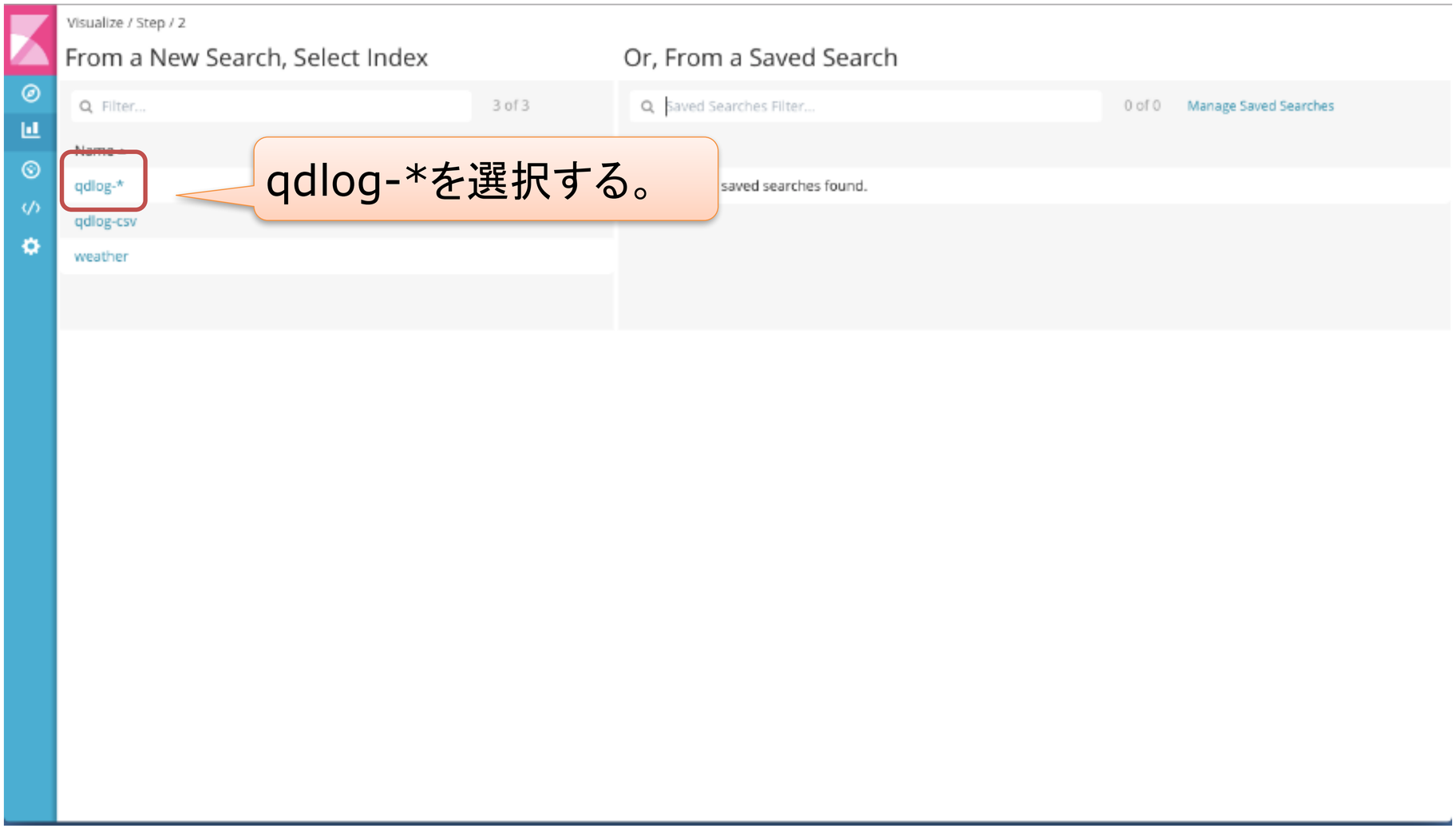
③ 「**Line Chart**」を利用し、日毎・キャリア毎のアクセス数を表示する。

④ 「**Tile Map**」を利用し、キャリア**MPT**の**3G**回線通信強度を地図上に可視化する。

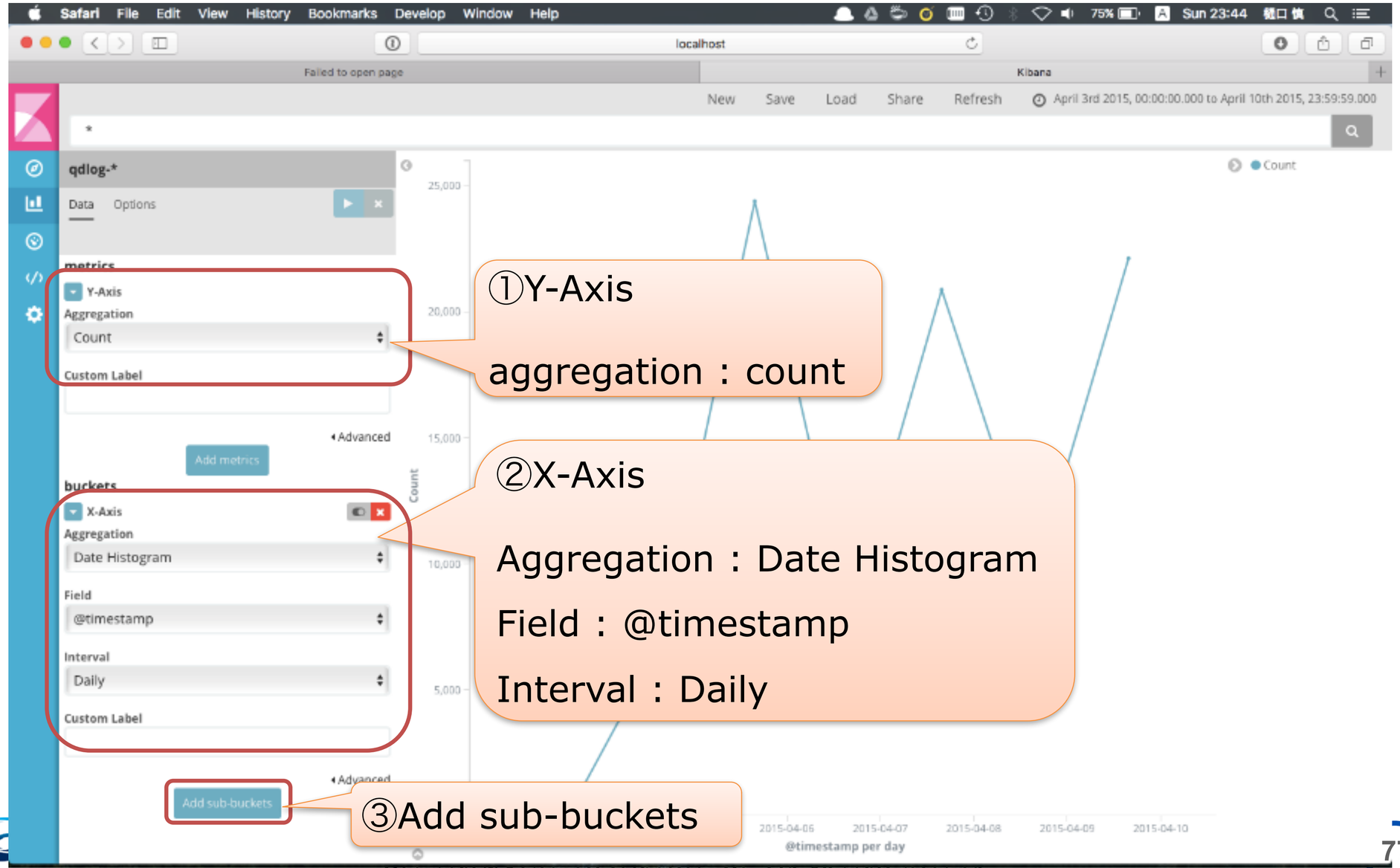
# Visualizationの作成



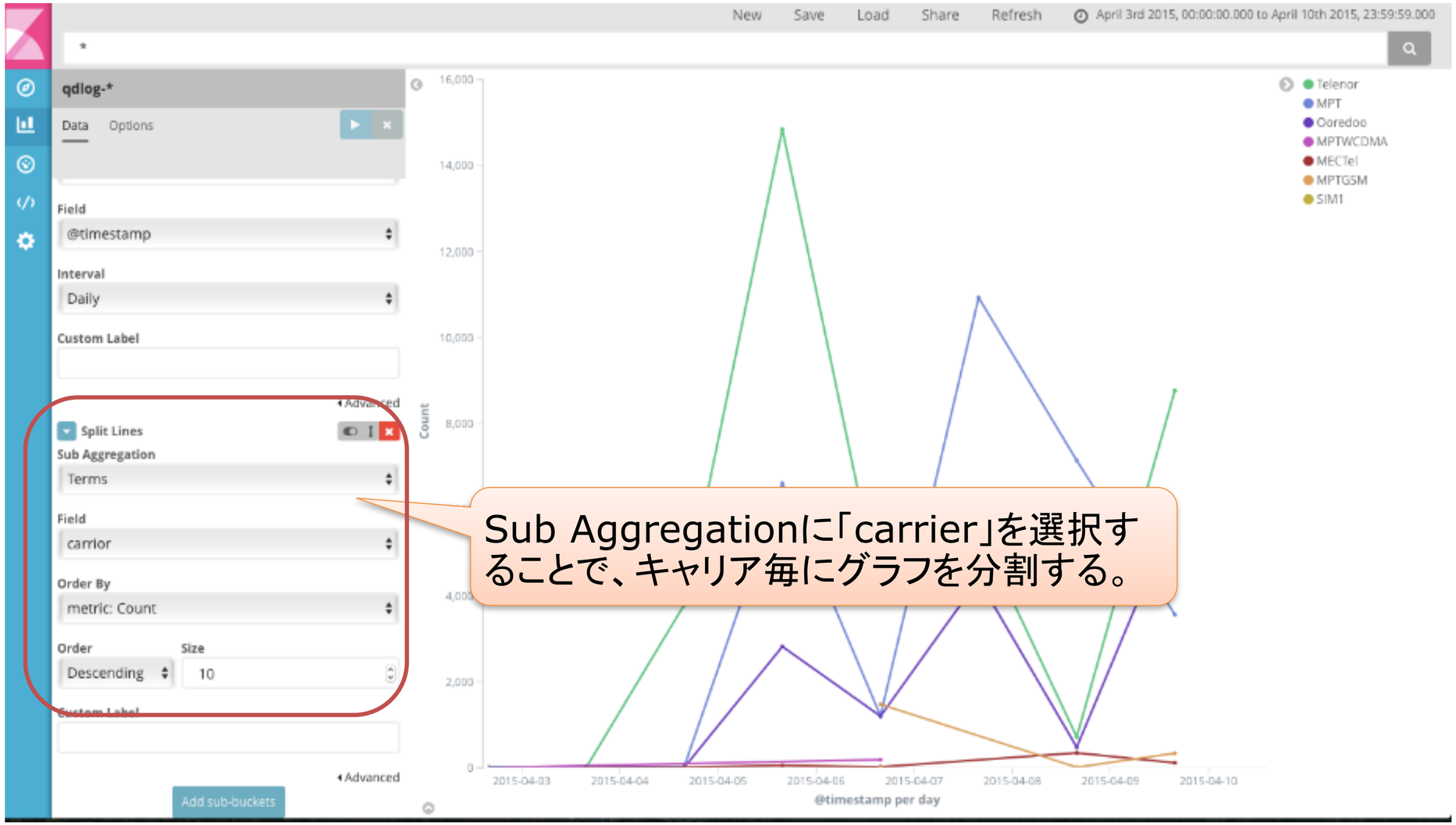
# Visualizationの作成



# Visualizationの作成(Line Chart)



# Visualizationの作成(Line Chart)



# Visualizationの作成

## 1. 以下のVisualizationを作成する。

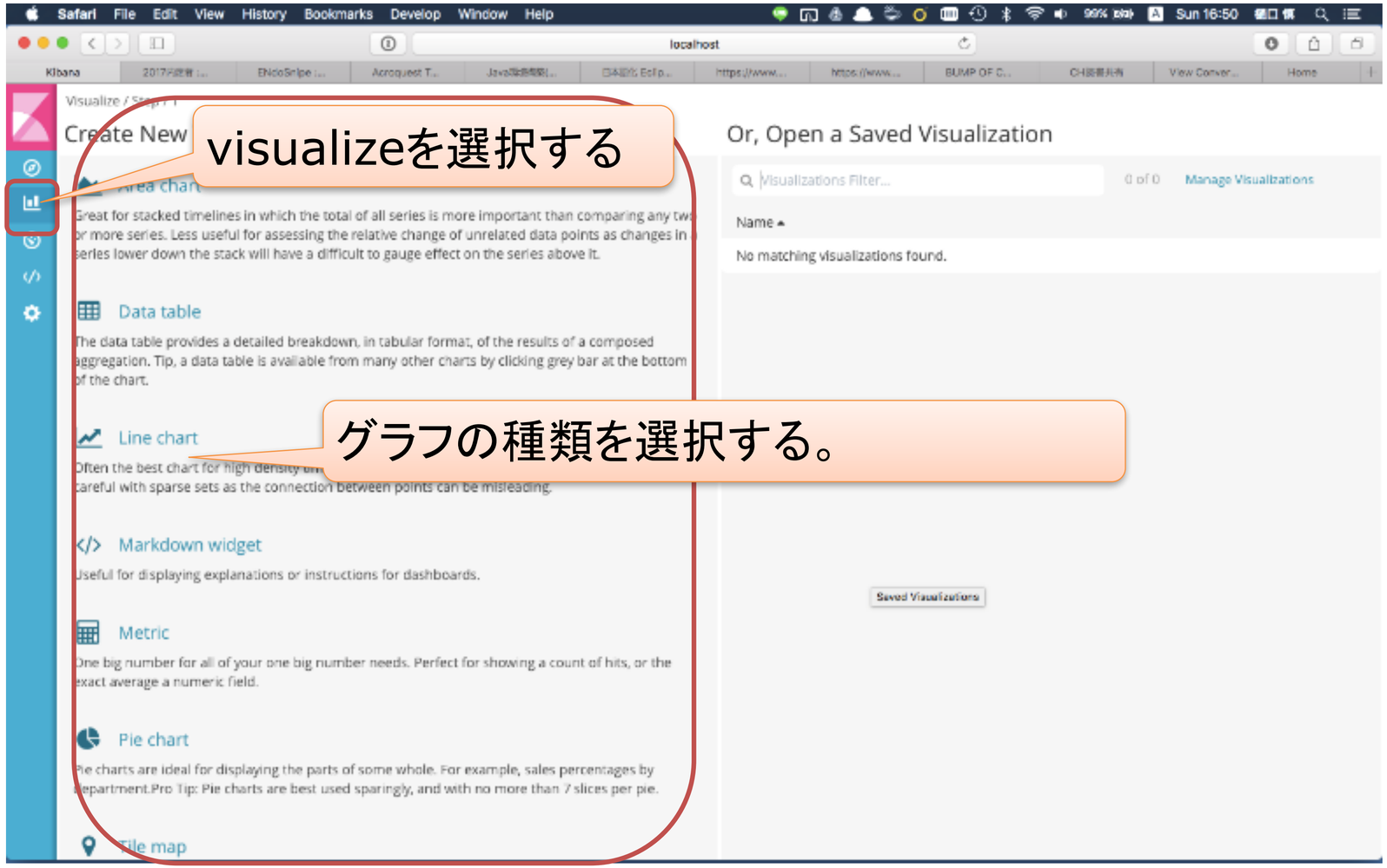
① 「**Metrics**」を利用し、ユニークユーザ数を表示する。

② 「**Data Tables**」を利用し、データ数の多い上位10ユーザの一覧を表示する。

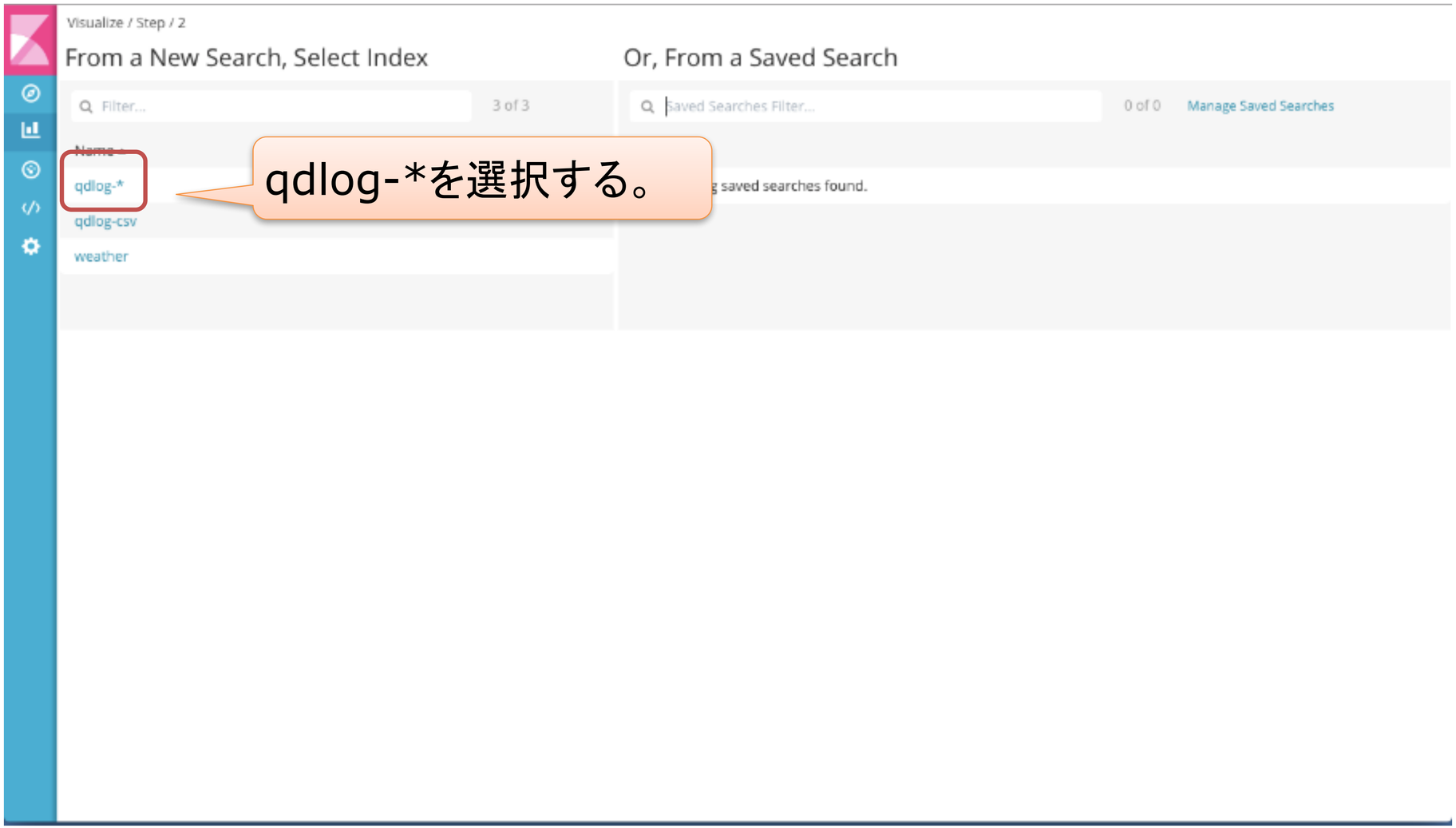
③ 「**Line Chart**」を利用し、日毎・キャリア毎のアクセス数を表示する。

④ 「**Tile Map**」を利用し、キャリア**MPT**の**3G**回線通信強度を地図上に可視化する。

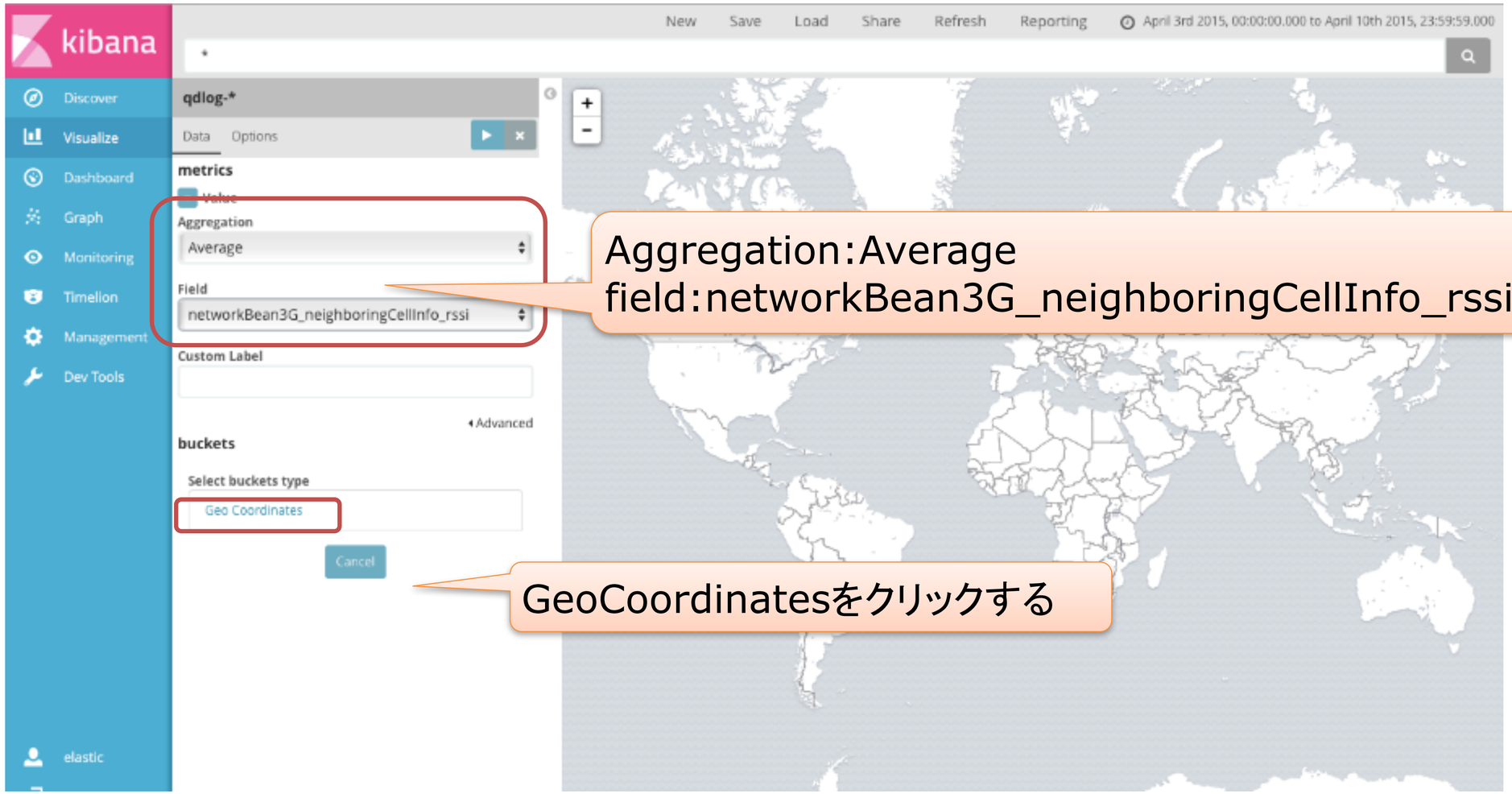
# Visualizationの作成



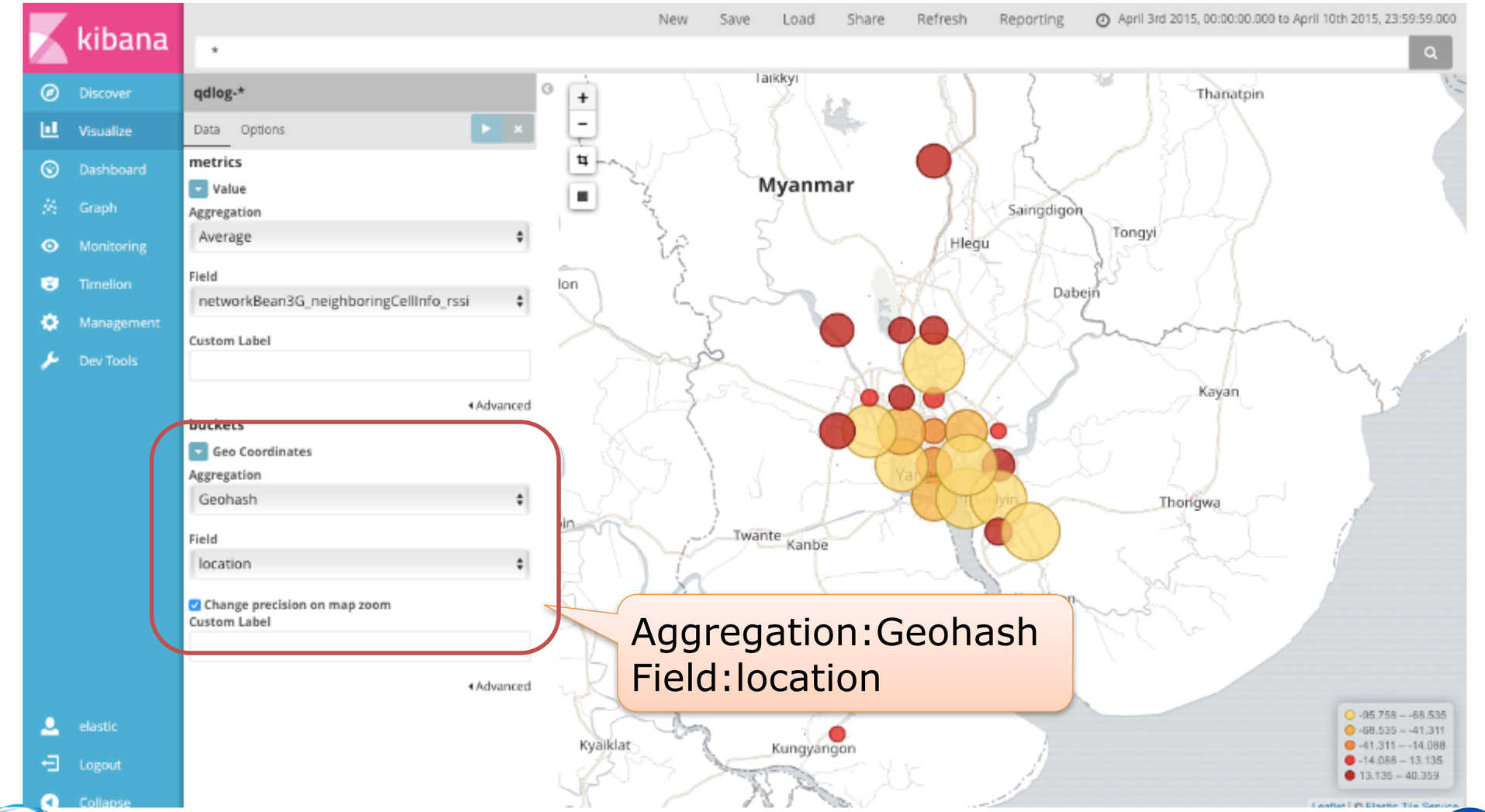
# Visualizationの作成



# Visualizationの作成(Line Chart)



# Visualizationの作成(Line Chart)

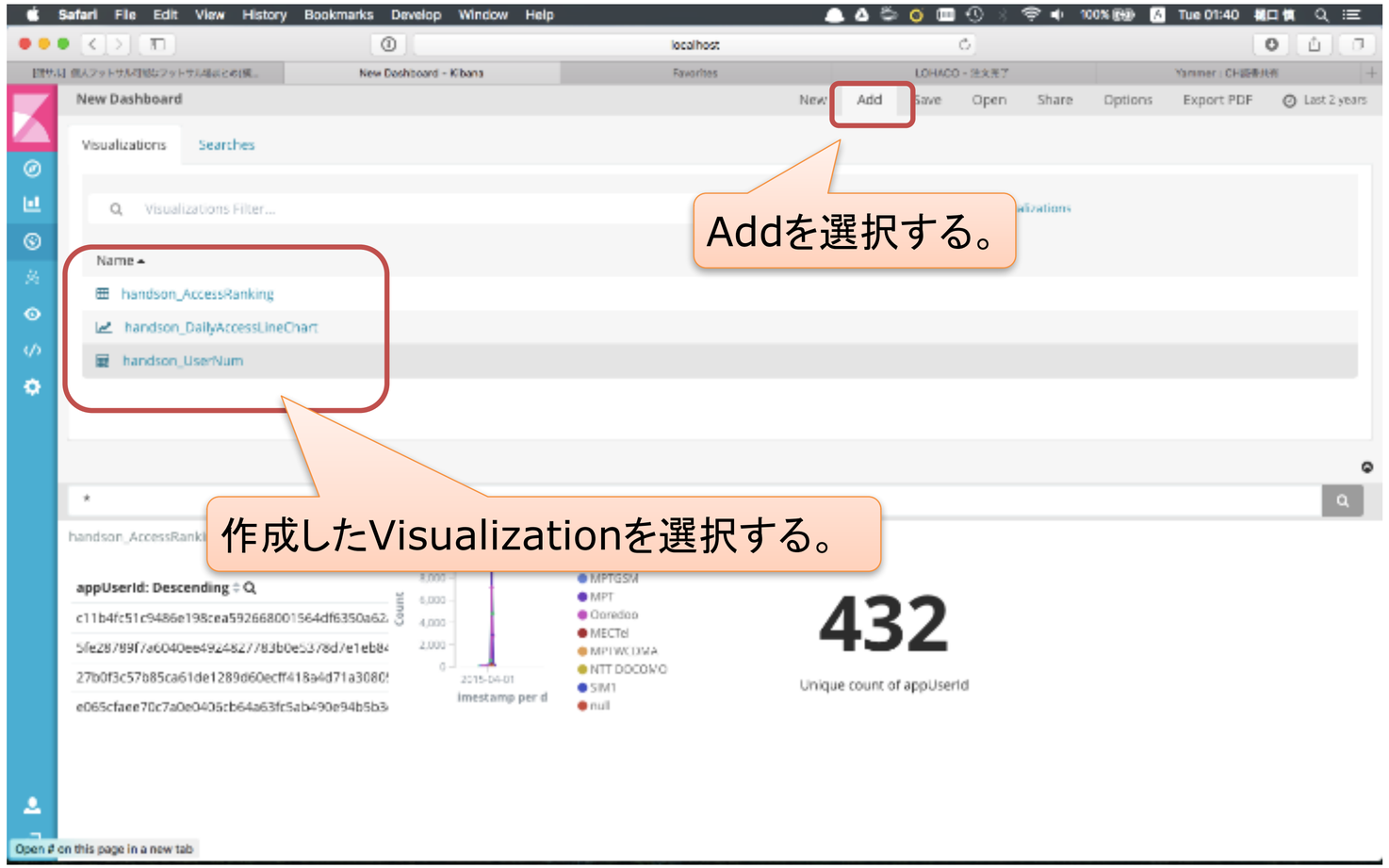


# Dashboardタブ

1. 演習2で作成したVisualizationを利用し、Dashboardを作成する。

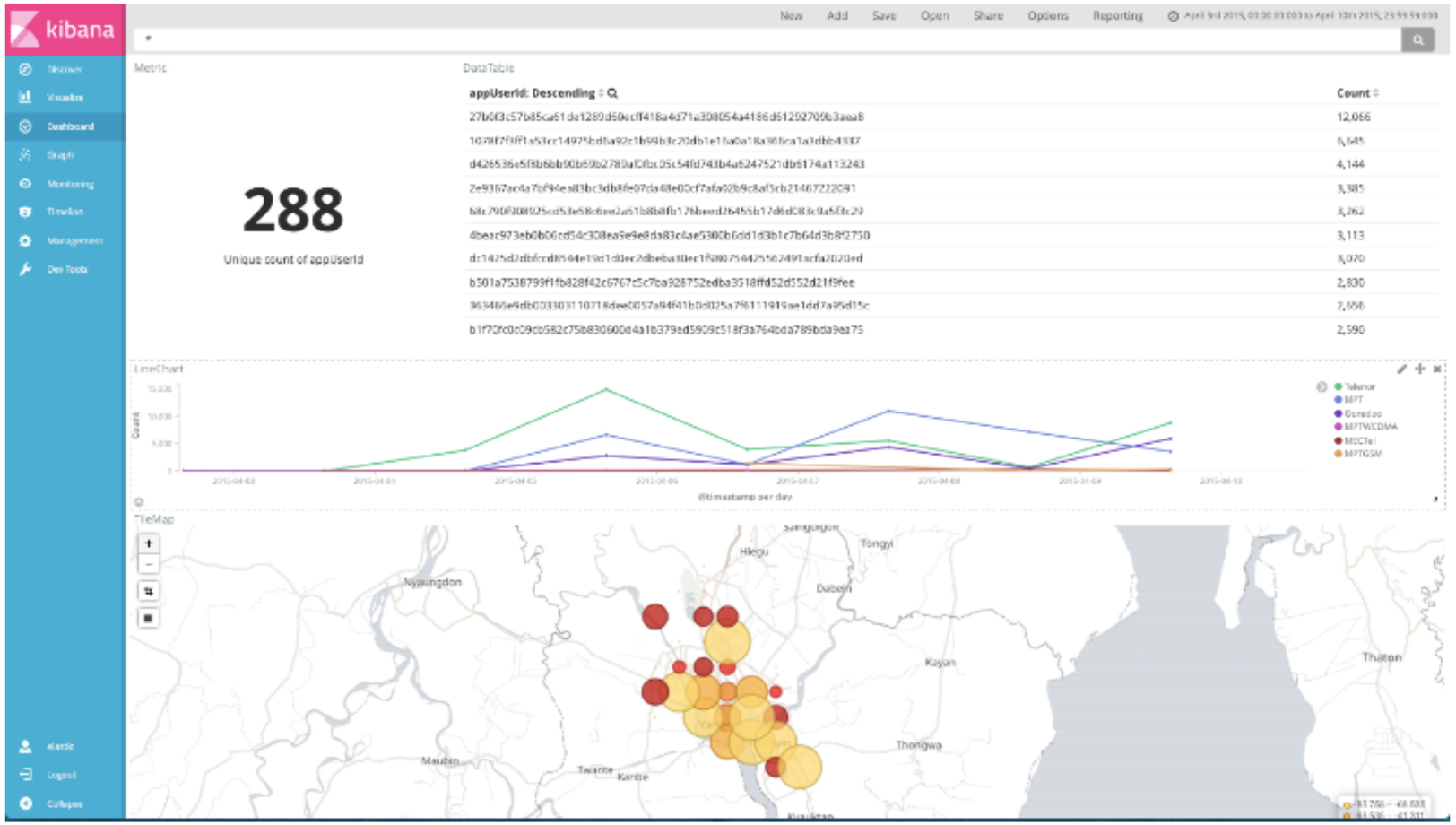
2. Dashboardを作成したら、Dashboardで表示した内容から、どのような情報が得られるか確認する。

# Dashboardタブ



# Dashboardタブ

・ドラッグで位置変更やサイズ変更ができる



# 発展課題

1. **Visualize**で、様々な種類のグラフを試しに作ってみましょう。

2. **Percentile Area Chart**や**Heatmap**なども作れるので、試してみましょう。

# X-Packの機能
# Security

# Security

1. ユーザ認証機能を持つ

   **LDAP/Active Directory/**ファイルベース

2. 認可機能を持つ

   ロールベースの**ACL**

   インデックス、操作単位で設定可

   **Document**および**Field**レベルのセキュリティ対応

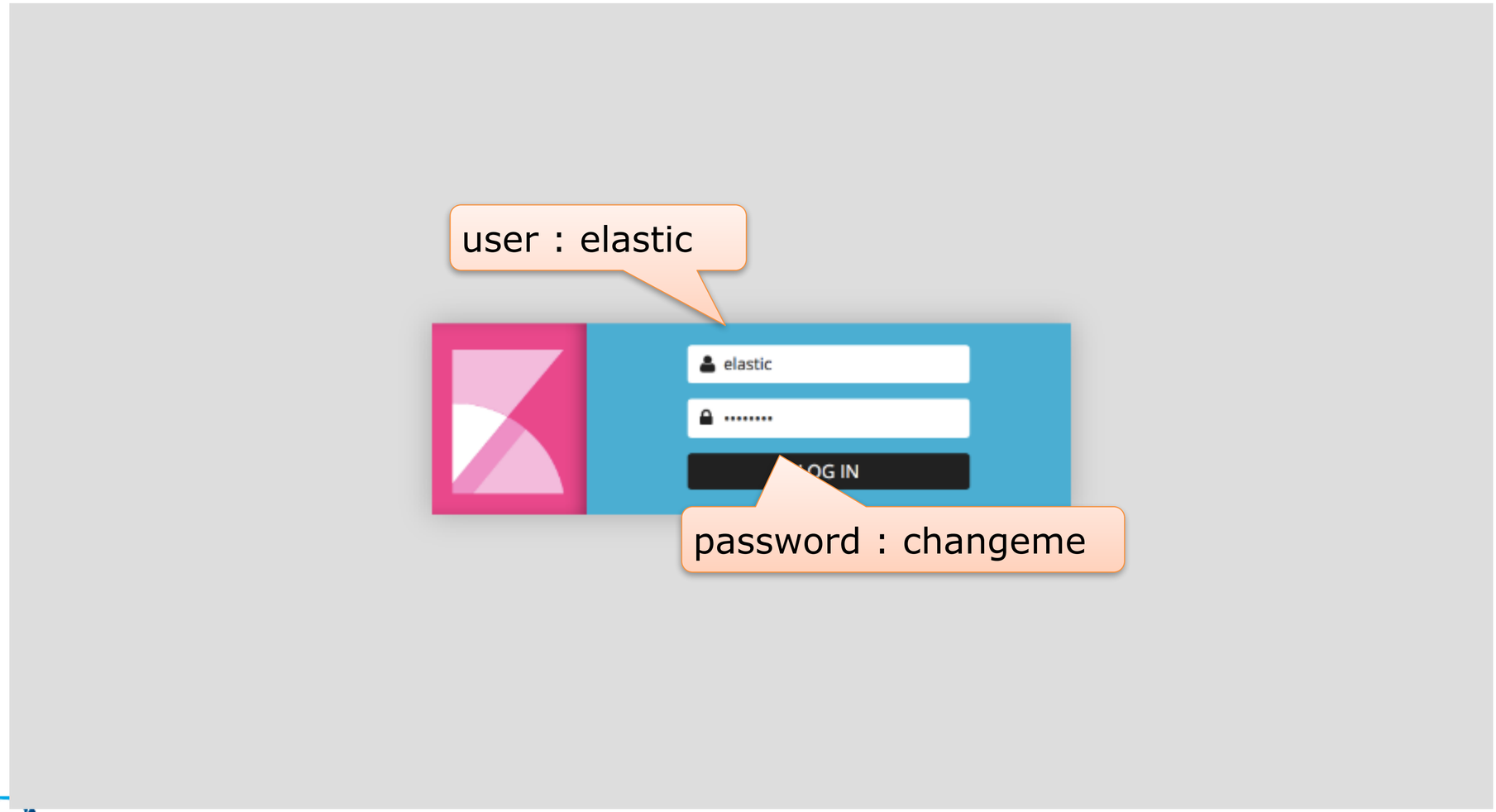
3. セキュアな通信が可能となる

   ノードおよびクライアント間、**IP**フィルタリング

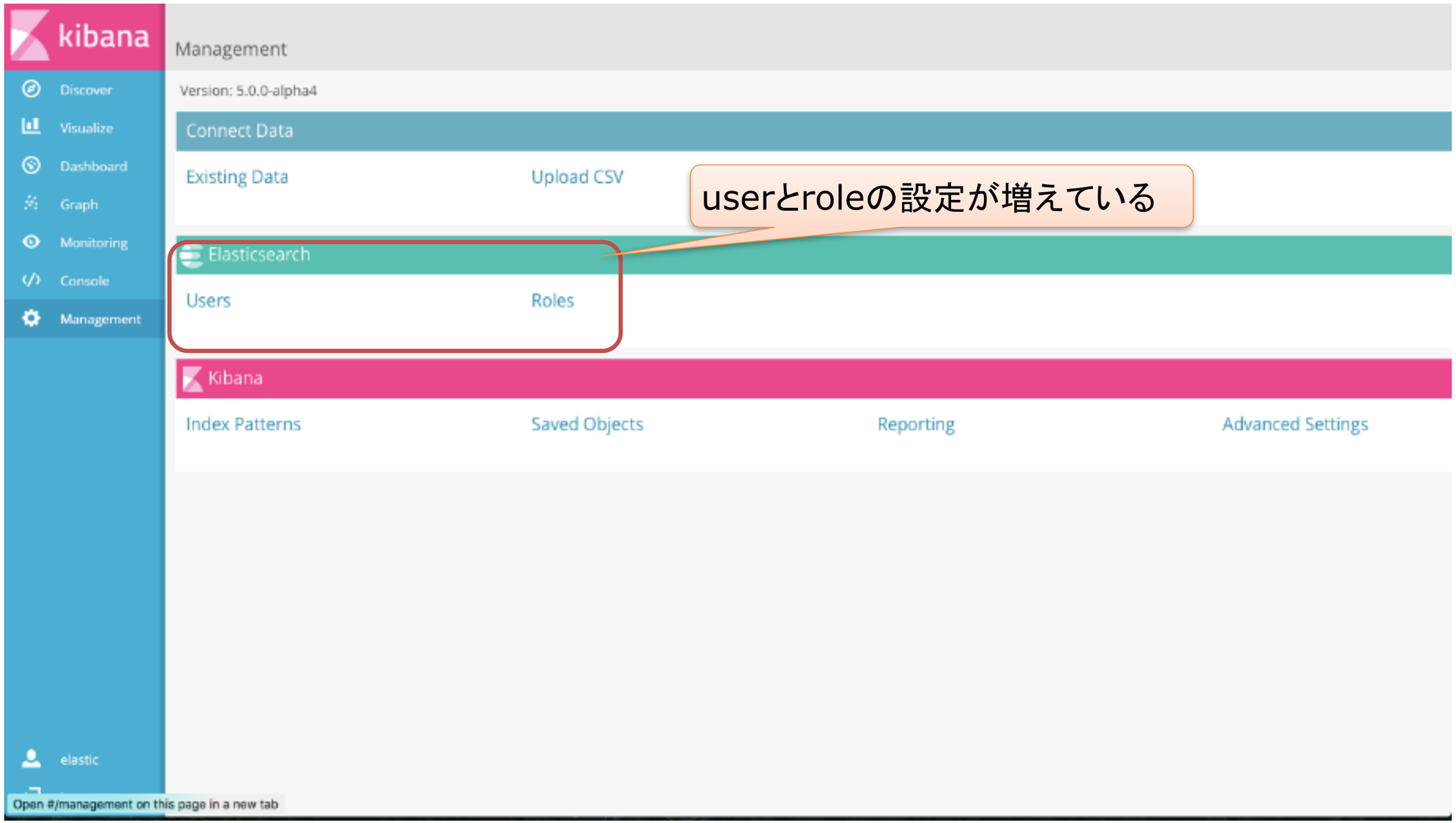
4. 監査ログを残すことができる

# Security

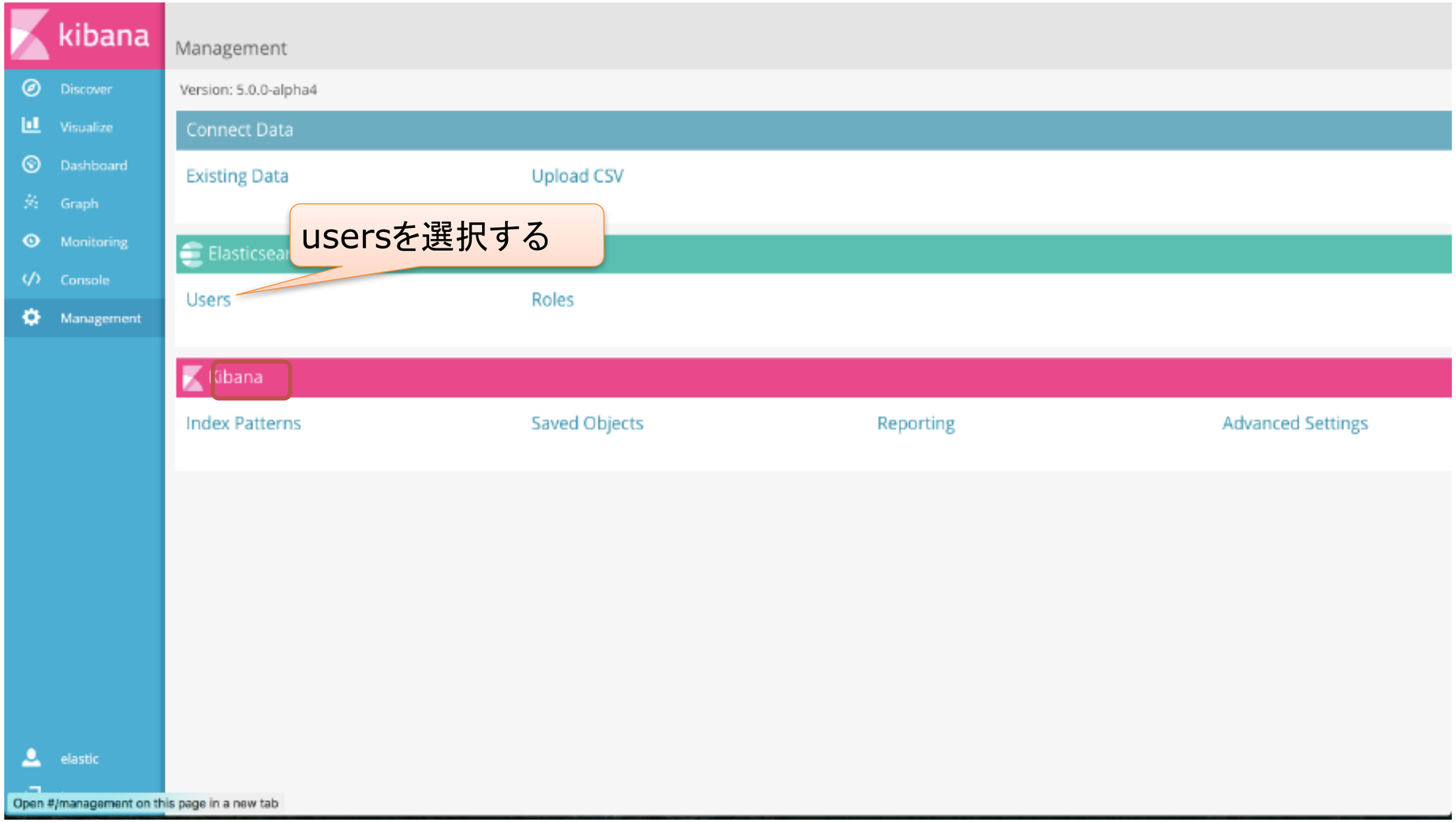
・x-packをインストールすると認証がかかる



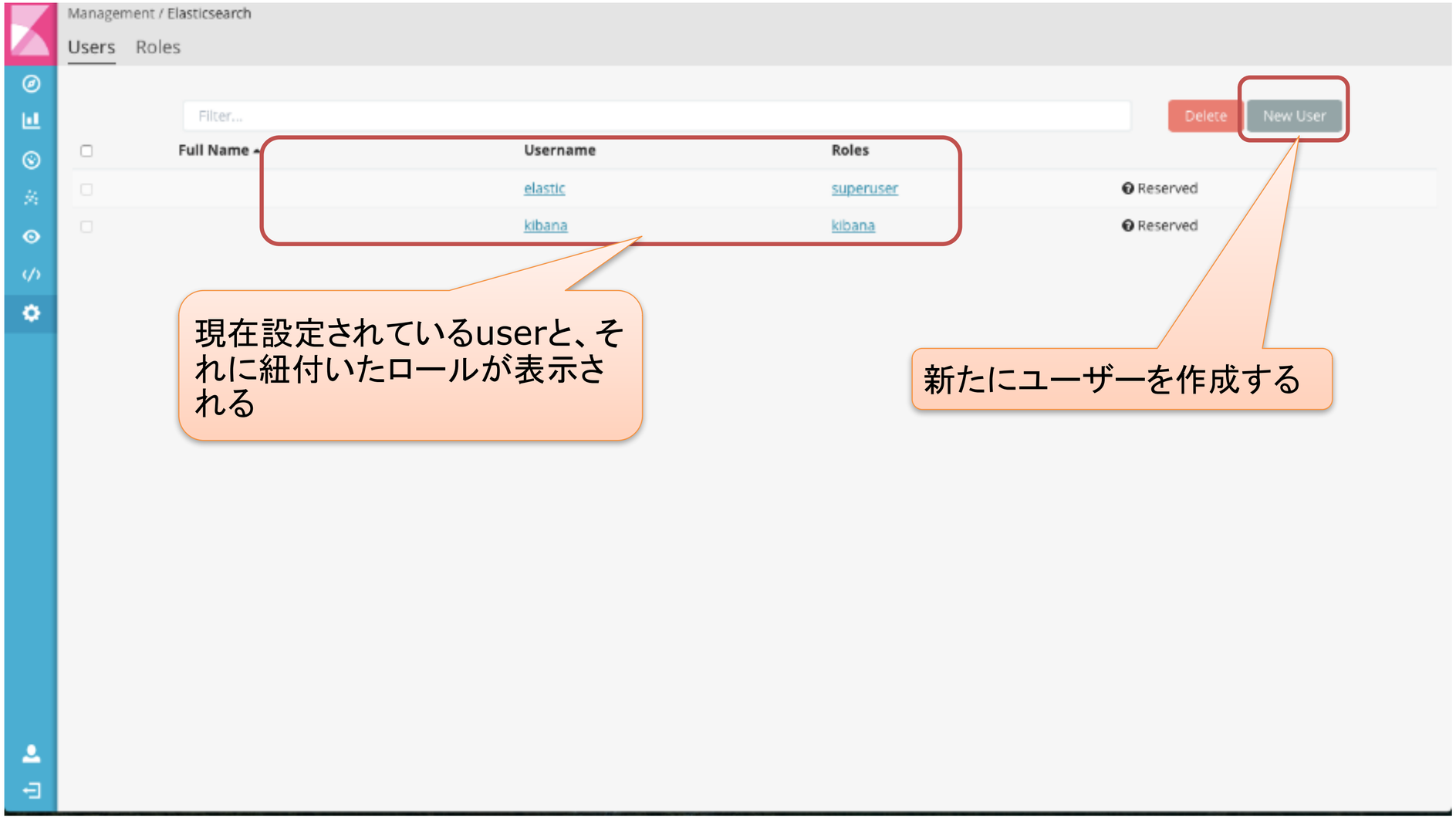
# Security



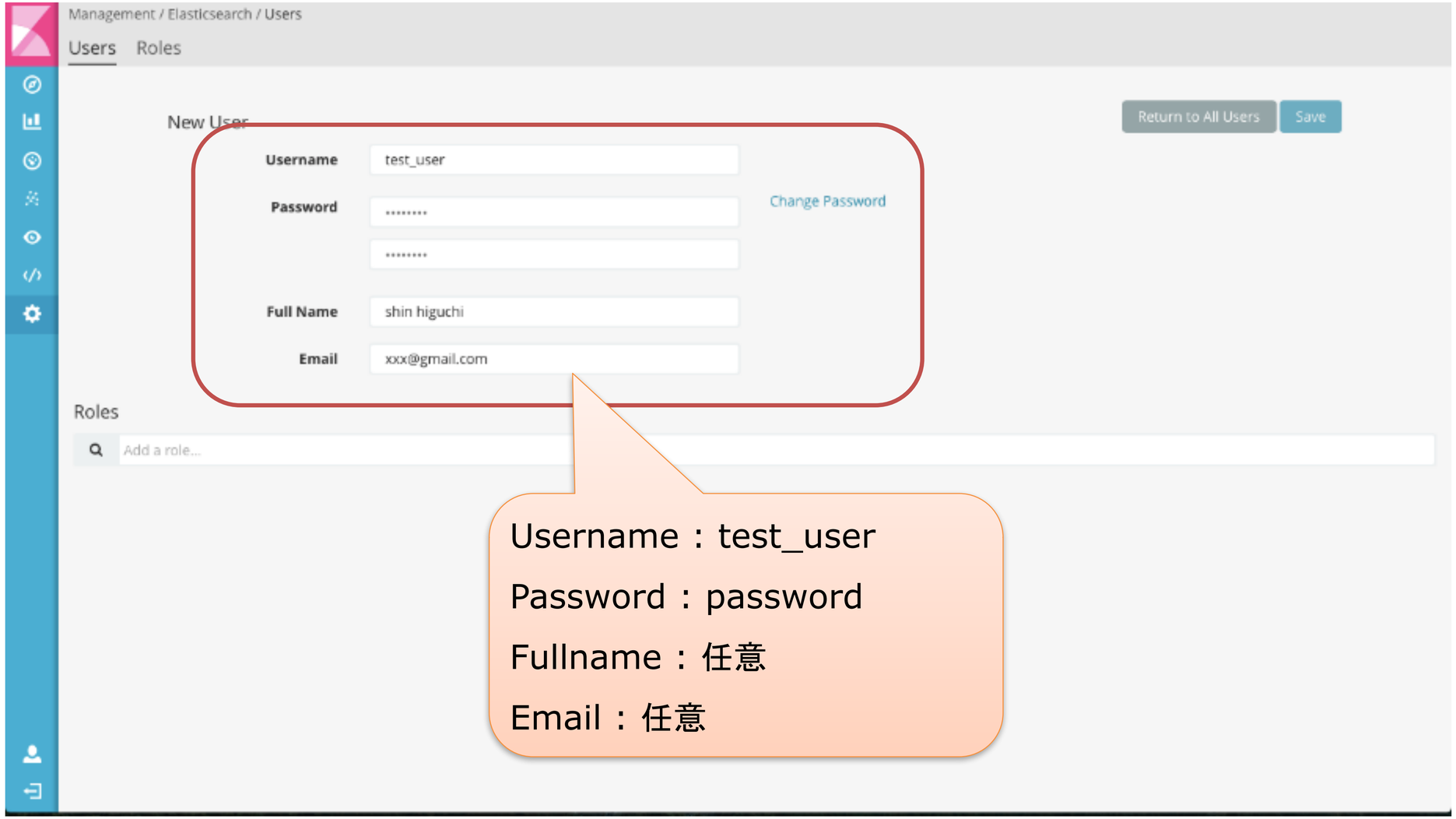
# Security



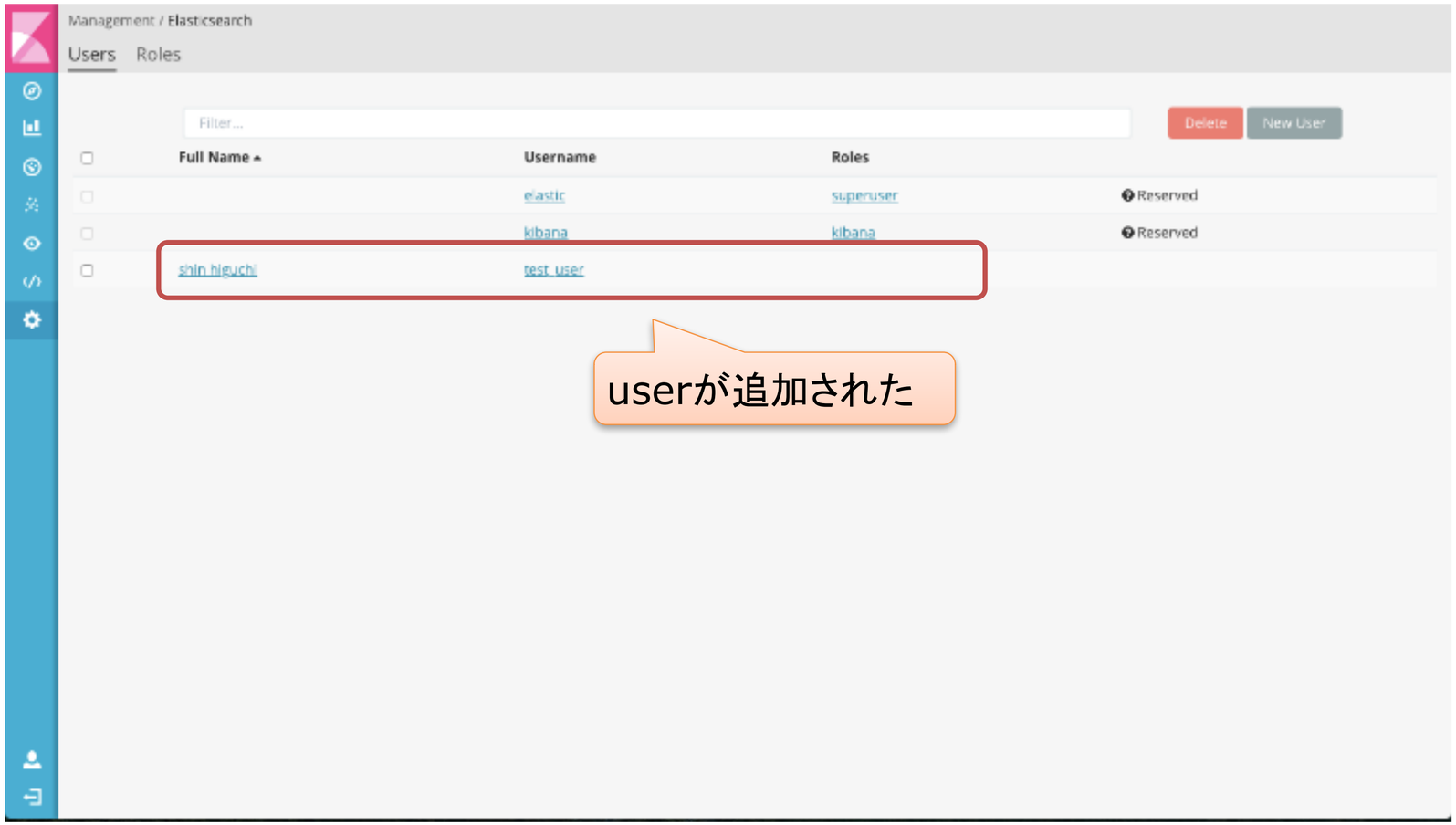
# Security



# Security

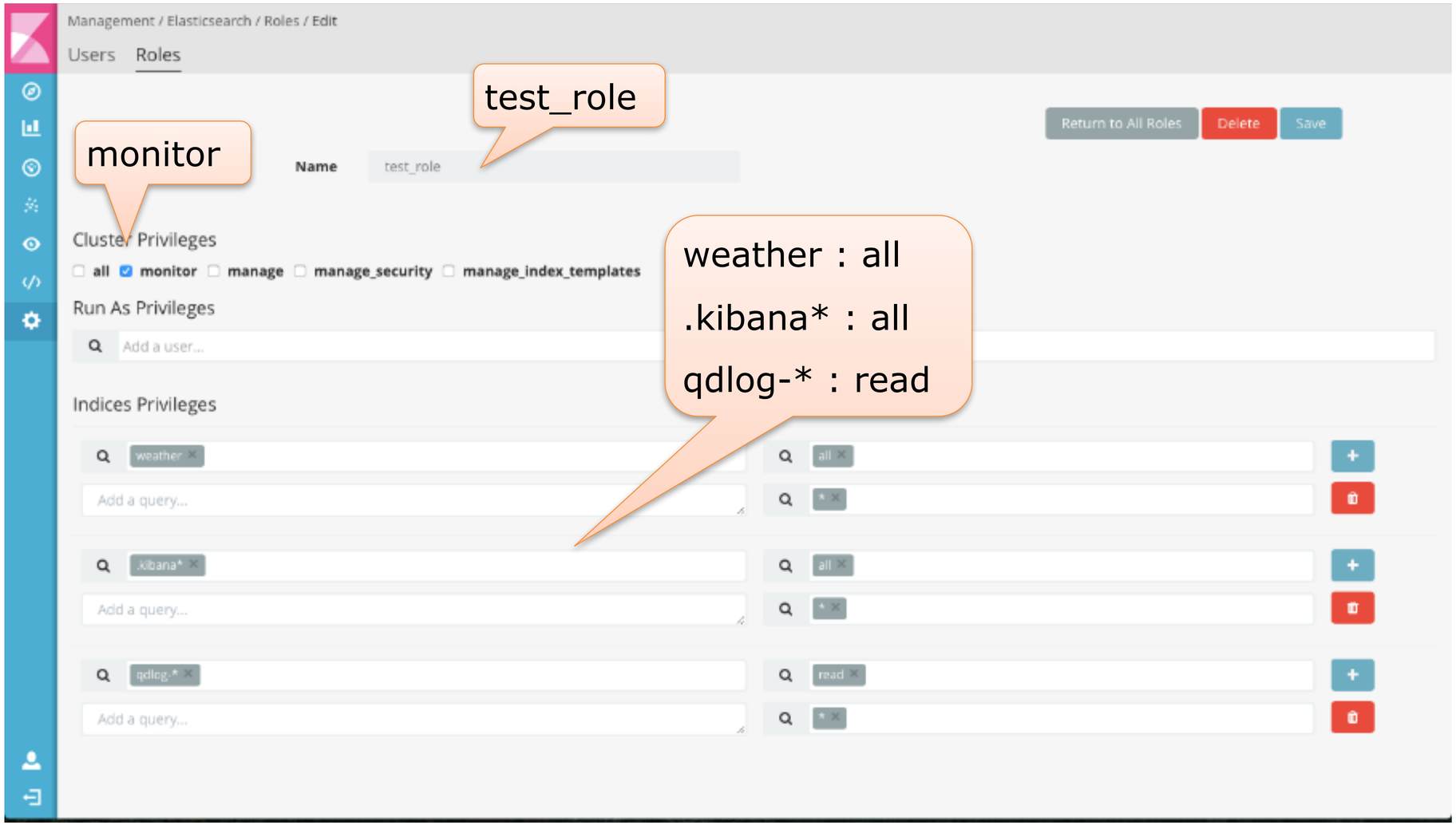


# Security



# Security

# Security



# Security

# Security

1. **test_user**でログインした時に**qdlog-***にマッチする**index**が作成できないことを確認してみましょう

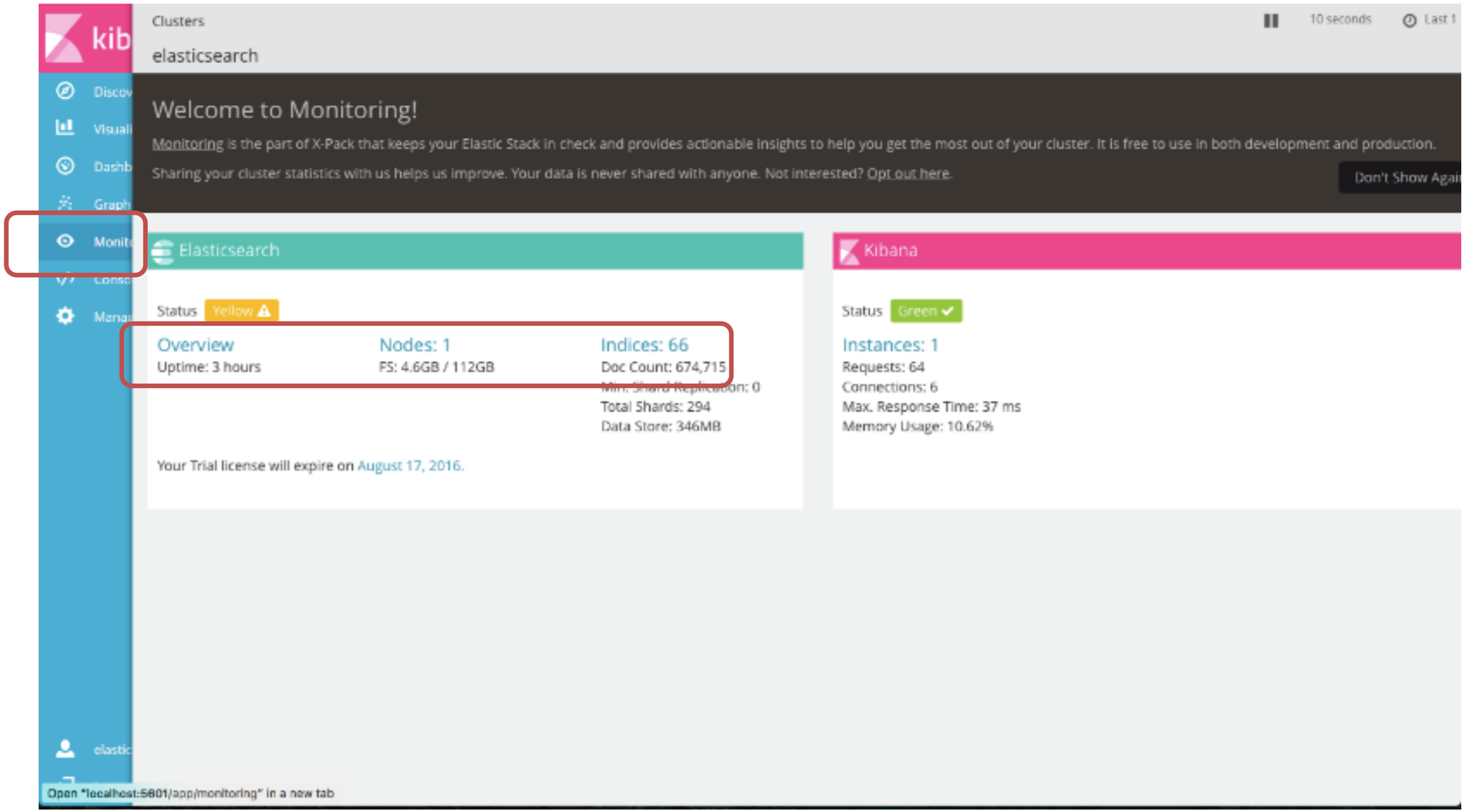
2. 他のユーザーも作成して、いろいろなアクセス制御を試してみましょう

# Monitoring

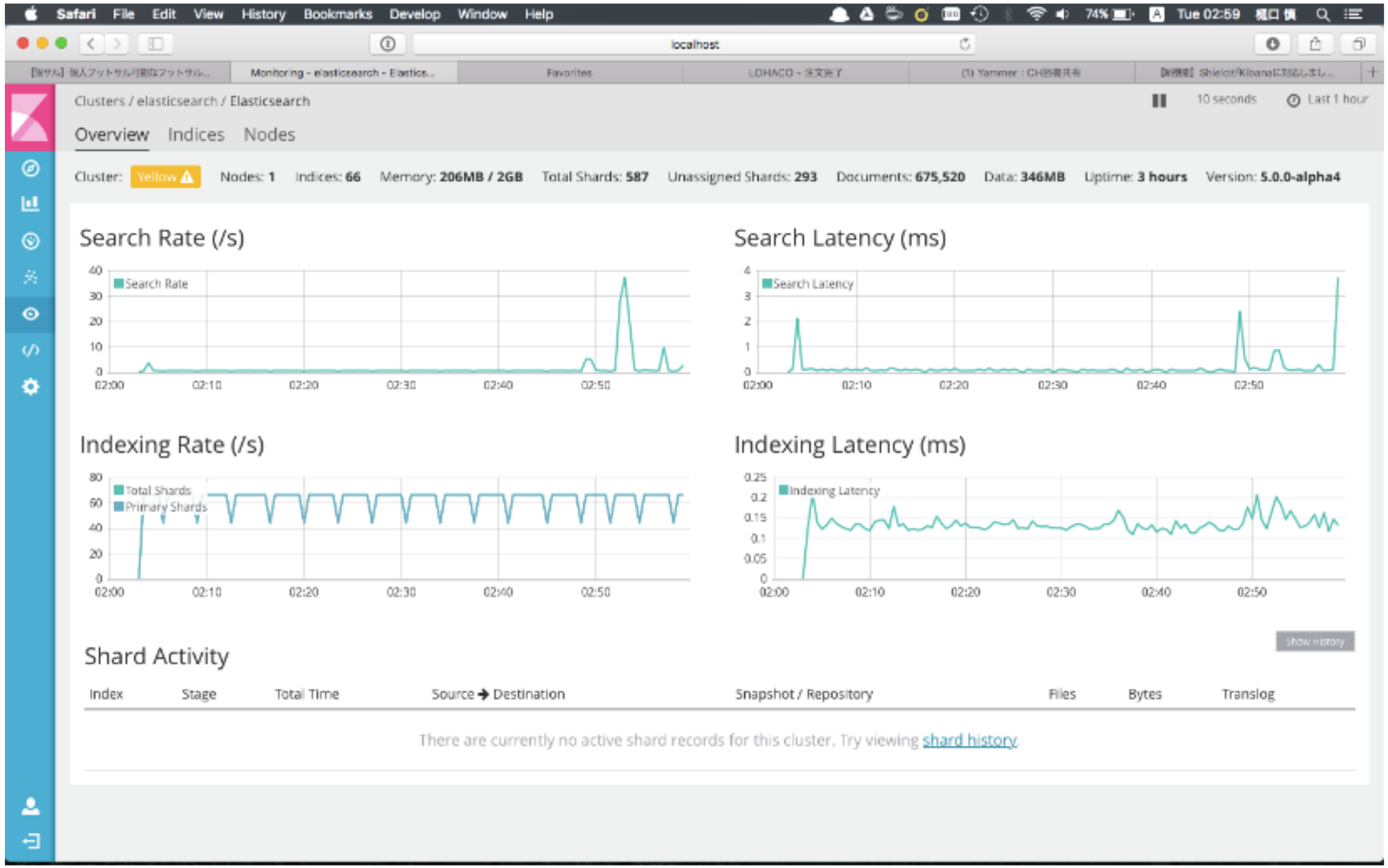
# Monitoring

- **kibana**の画面上で**elasticsearch**や**kibana**の稼働状況を監視できる。

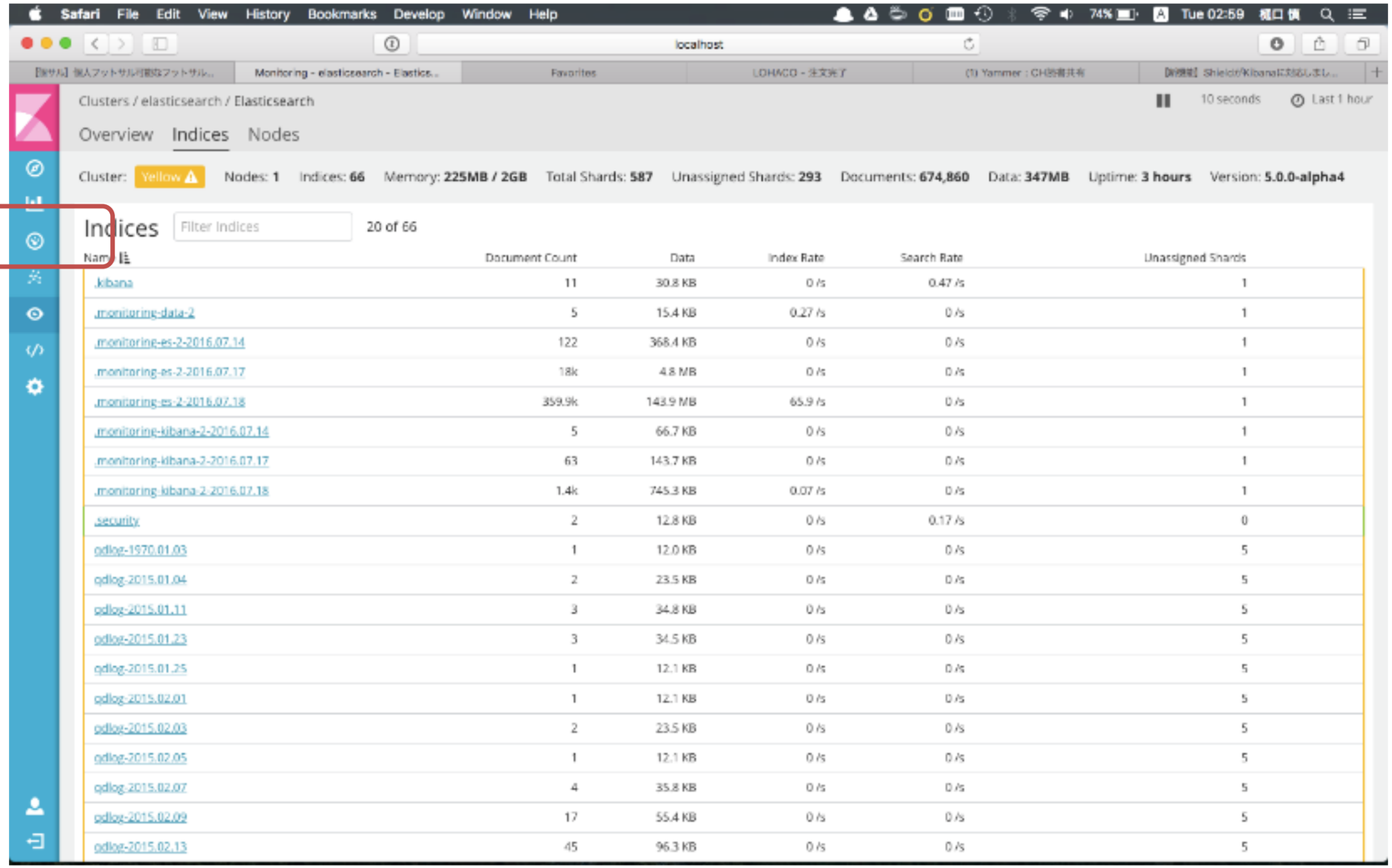
# Monitoring



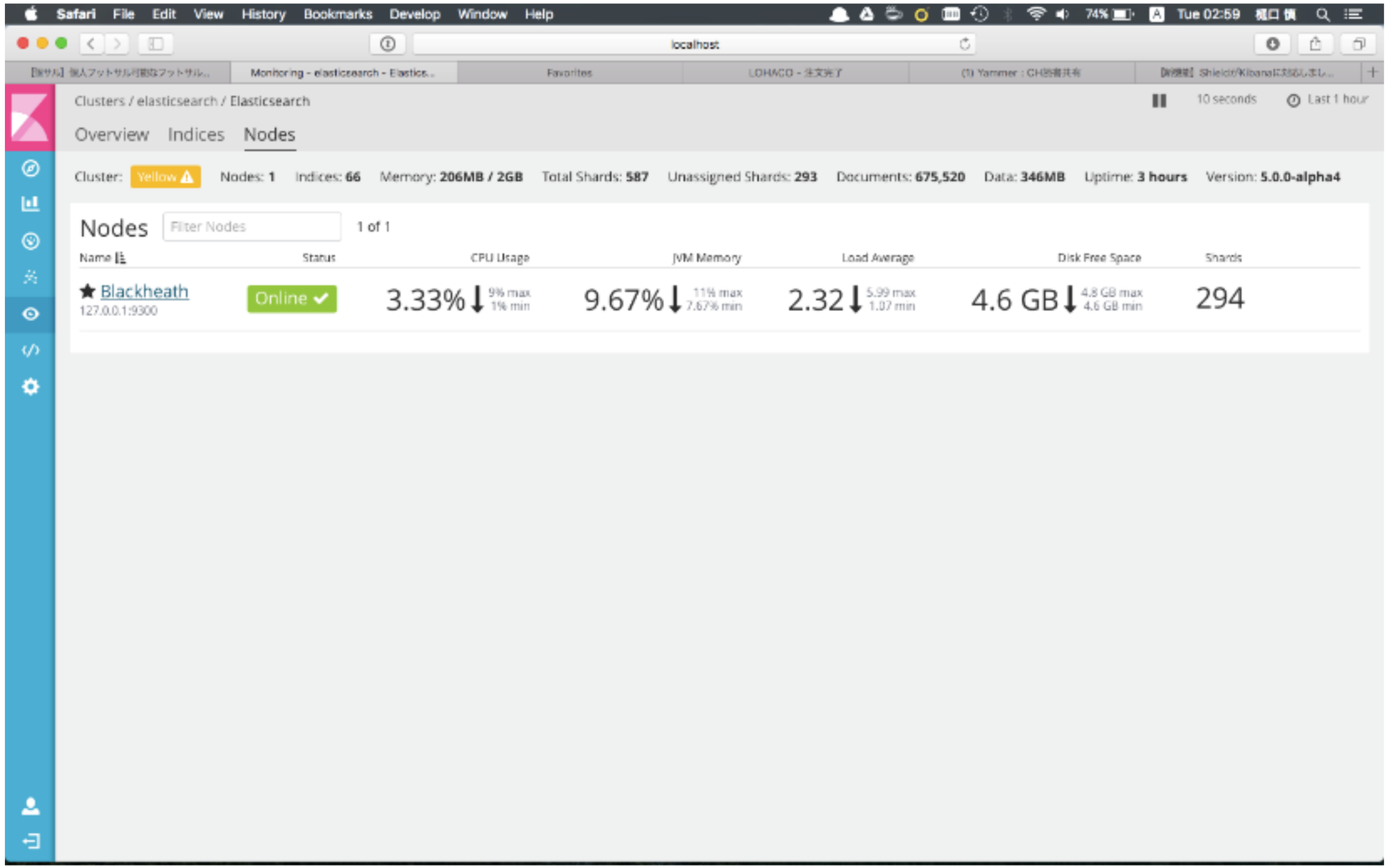
# Monitoring (Overview)



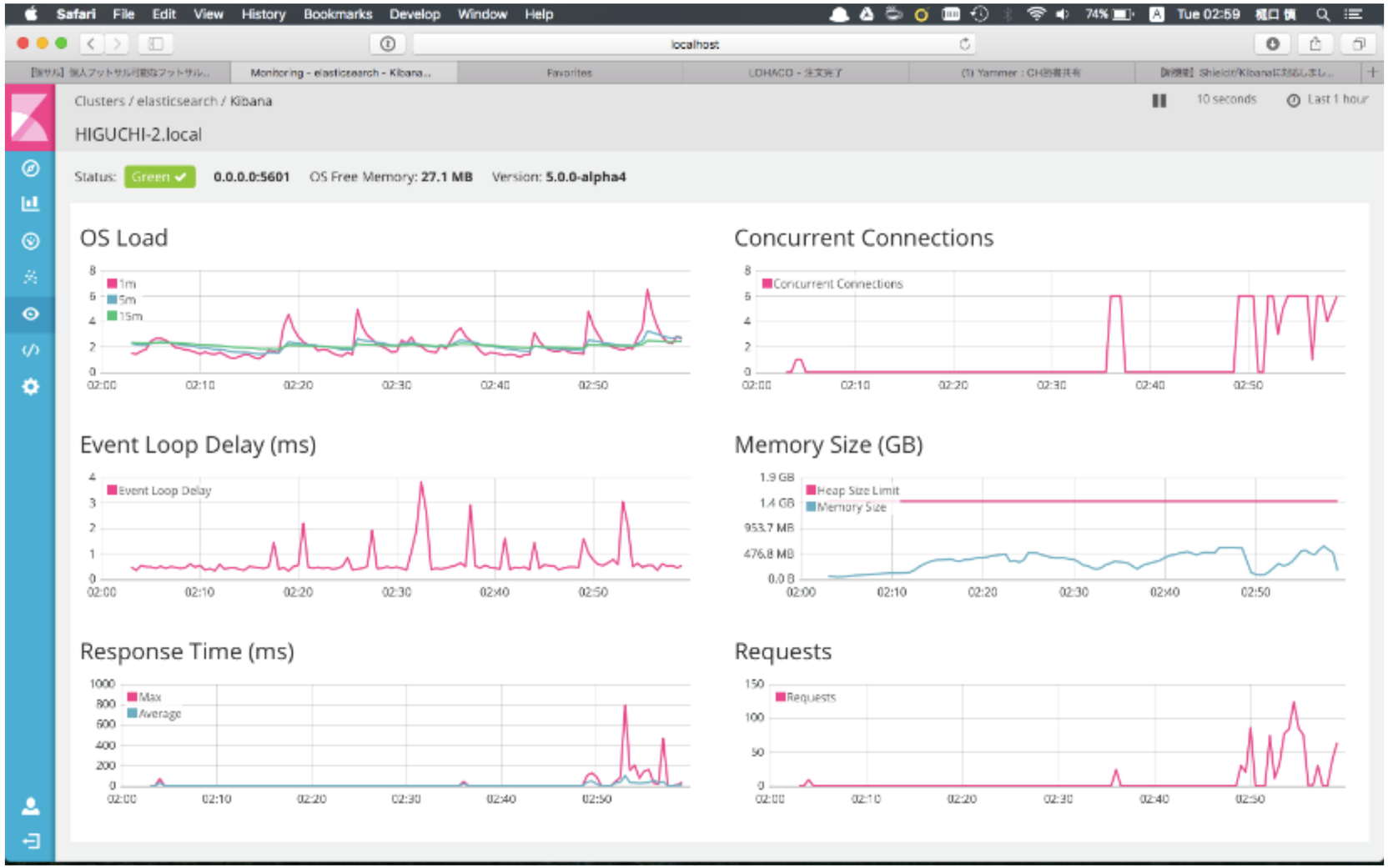
# Monitoring (indices)



# Monitoring (Nodes)



# Monitoring (Kibana)



# X-Packのその他機能

# Watcher

# Watcherとは

1. クエリを登録し、条件に当てはまった場合に、通知を行う。

   （例）
   - 一定時間内に、特定のURLに大量にアクセスがあった。
   - CPU使用率が閾値を超えた。
   - 特定の商品に在庫ができた。
2. 通知方法
   - メール送信
   - 外部APIの実行
   - ログ出力

   など

# Watcher、Monitoringとの組み合わせ

- **Watcher**、**Monitoring**を組み合わせると以下のことが実現できる。
  - クラスタのチェック

    直近**60**秒以内にクラスタの状態が**red**になったらメールを送る。
  - メモリ使用量のチェック

    平均ヒープ使用量が**75%**以上になったらメールを送る。
  - クラスタの構成チェック

    クラスタにノードが追加・削除されたらメールを送る。

# まとめ

1. Logstash → Elasticsearch → Kibana
   を使って、簡単なデータ分析を体験した。

2. 拡張機能(X-pack)の一部を体験した。

→身近なデータを使って、実際に分析・検索
を試してみてください。

# We're hiring!

# ご清聴ありがとうございました。

Acroquest Technology

*Infrastructures Evolution*