ELK Stack: Elasticsearch, Logstash and Kibana

Kajorn Pathomkeerati (IAI)
Samuel Ambroj Peréz (SCC)
Extra: Sense (Beta)

- Extension for Google Chrome, available in Chrome Web Store
- User-friendly console
- Designed for Elasticsearch

![Image of Sense (Beta) features: Individual Request, Text Highlight, Auto Complete, Indentation, Request, Response]
Visualization Tool For Elasticsearch

KIBANA
Kibana - Overview

- **Full integration with Elasticsearch**
  - Easy Configuration
- **Import & Discovery**
  - Time-based Data
  - Real-time Discovery
- **Visualization**
  - Easy to customize
  - Fast analytic
- **Dashboard**
  - Connecting the visualizations
  - Easy to share as <iframe> or URL

- Open source. Community driven. Apache licensed.
- More Info
  
  https://www.elastic.co/products/kibana
Kibana- Live Demo

KIBANA GOES LIVE
Kibana - Summary

- **Advantages**
  - Easy visualizing
  - Various visualizations available
  - Fully integrated with Elasticsearch

- **Limitations**
  - No custom aggregation supported
  - No custom request
  - Event-based data only
  - Elasticsearch data only
  - Dashboard built on saved visualizations
  - Dashboard filter affects all visualizations
Use Case

ELASTICSEARCH & SQL DATABASE
Overview

- Relational Database:
  - Traditional SQL Databases
  - Complex SQL-Statements needed for some analytics
  - Still widely used

- Elasticsearch:
  - Non-relational Databases - NoSQL
  - As-a-service
  - Accessible via HTTP

- Relational DB → Non-relational DB
  - Data Migration
  - Using plugin: **JDBC River plugin**
JDBC River

- An Elasticsearch Plugin
- Enabling data migration
- SQL Database → Elasticsearch
  - Import – function
  - Using SQL statement to filter data
- Using a JDBC connector
  - Supports native connectors
    - MySQL, Postgresql, ...

Note:
- River plugin is deprecated since ES 1.5 (Currently ~1.7)
- Still supported by community
JDBC River - Parameters

```bash
$ curl -XPUT 'localhost:9200/_river/type_name/_meta' -d '
{
  "type" : "jdbc",
  "jdbc" : {
    "url" : "jdbc:mysql://localhost",
    "user" : "db_user",
    "password" : "db_user_password",
    "sql" : "SELECT * FROM table_name",
    "index" : "es_index",
    "type" : "es_type",
    "type_mapping" : { ... }
  }
}
'
```

- Easy import
- Filter Data by SQL-Statement
- More Info:
  [https://github.com/jprante/elasticsearch-jdbc](https://github.com/jprante/elasticsearch-jdbc)
Experimental Scenario

- **Input:**
  - A dump file for MySQL

- **Output:**
  - Visualizations in Kibana

- **Question: How?**

- **Answer: following instructions**
  1. Prepare MySQL Server
  2. Prepare JDBC River plugin for MySQL
  3. Import data to Elasticsearch
  4. Visualizing with Kibana
MySQL Server - Installation

- Required Components / File:
  - MySQL Server
  - MySQL Client
  - An SQL dump File

- Instructions:
  1. Set up MySQL Server & Client
     $ aptitude install mysql-server mysql-client
  2. Create a database
     $ mysql -u root -p
     > create database db_name;
     > exit
  3. Restore the database with dump file
     $ mysql -u root -p db_name < dump_file.sql
JDBC River - Installation

Required Components:
- JDBC River Plugin
- JDBC Driver (Connector)

Instructions:
1. Download & install JDBC River plugin
2. Download & install a JDBC driver
3. Restart Elasticsearch
JDBC River - Installation (2)

- Download & install JDBC River plugin
  
  ```
  $ cd /usr/share/elasticsearch
  $ ./bin/plugin --install river-jdbc --url 'http://xbib.org/repository/org/xbib/elasticsearch/plugin/elasticsearch-river-jdbc/1.5.0.5/elasticsearch-river-jdbc-1.5.0.5-plugin.zip'
  ```

- Download & install a JDBC connector (found in MySQL JDBC driver)
  
  ```
  $ cd /usr/share/elasticsearch/plugins/
  $ wget http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.36.tar.gz
  $ tar -zxvf mysql-connector-java-5.1.36.tar.gz --wildcards '*.jar'
  $ mv mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar /river-jdbc/
  $ rm -rf mysql-*
  ```

- Restart Elasticsearch Service
  
  ```
  $ /etc/init.d/elasticsearch stop
  $ /etc/init.d/elasticsearch start
  ```
Import Data

Define JDBC River parameters

```bash
$ curl -XPUT 'localhost:9200/_river/tweet/_meta' -d
'{
    "type": "jdbc",
    "jdbc": {
        "url": "jdbc:mysql://localhost:3306/tweetsql",
        "user": "root",
        "password": "root",
        "sql": "select tid as _id, tweet, hashtag.hashtag, lang, created_at
            from tweet
            left join hashtag_tweet on tweet.id = hashtag_tweet.tweet_id
            left join hashtag on hashtag.id = hashtag_tweet.hashtag_id",
        "index": "tweetsql",
        "type": "tweet",
        "type_mapping": {
            "tweet": {
                "dynamic": true,
                "properties": {
                    "created_at": {
                        "type": "date",
                        "format": "EEE MMM dd HH:mm:ss Z yyyy"
                    }
                }
            }
        }
    }
}'
```
Visualization in Kibana
Plugins

- Other River Plugins:
  - Google Drive River Plugin
  - Dropbox River Plugin
  - Wikipedia River Plugin
  - …

- Cloud Service Discovery Plugins
  - AWS Cloud Plugin, GCE Cloud Plugin, …

- Analysis Plugins
  - ICU Analysis Plugin, Stempel Analysis Plugin, …

- There are more plugins:
Exercise - MySQL Database

- Given:
  - A dump file contains tweet data
- Goal:
  - Visualizations in Kibana

- Example for visualizations
  - Number of tweets in total
  - Number of tweets by a language
  - Top hashtags / tweet-languages
  - etc.

- Create a dashboard with various visualizations
Question ?