ElasticSearch / Kibana

- JDEV 2015 Plénière : Donnez du sens à vos données (Elasticsearch, un moteur de recherche open source pour implémenter vos services de recherche big data et d'analyse) - David Pilato

- JDEV 2015 Atelier : Prise en main d'elasticsearch et de Kibana - - David Pilato
ElasticSearch / Kibana

- [https://www.elastic.co](https://www.elastic.co)

- ElasticSearch = real-time distributed search and analytics engine.
  - Permet de stocker, d'indexer des documents et de les rechercher en quasi temps réel (latence 1 s)
  - Basé sur Apache Lucene (bibliothèque/moteur de recherche full text)
  - Expose l'ensemble de ces services en HTTP REST/JSON

- Kibana = a data visualization engine, allows you to natively interact with all your data in Elasticsearch via custom dashboards
  - Portail Web : permet d'explorer les données (soumettre des recherches, filtrer les résultats de recherche) et de visualiser les résultats de recherche sous forme graphique (histogramme, camembert, carte, …)
ElasticSearch : Basic Concepts

- **Cluster**
  - = collection of one or more servers (nodes) that together holds your entire data and provides federated indexing and search capabilities across all nodes.
  - Name by default = "elasticsearch".

- **Node**
  - = a single server that stores your data, and participates in the cluster’s indexing and search capabilities.
  - Name by default = a random name that is assigned to the node at startup

- **Document**
  - = a JSON object . key-value pairs
  - like a row in a table in a relational database.
    - Core field types (string, numbers, booleans)
    - Complex field types (arrays, objects)
    - Additional field types (geo points, geo shapes)
ElasticSearch : Basic Concepts

- **Index**
  - = collection of documents that have somewhat similar characteristics
  - is like a database in a relational database.
  - identified by a customized name, used when performing CRUD operations against the documents in it.

- **Type**
  - = a logical category/partition of your index whose semantics is completely up to you. In general, a type is defined for documents that have a set of common fields.
  - like a table in a relational database. Each type has a list of fields that can be specified for documents of that type.
  - Holds the mapping (schema definition in a relational database)

- **ID**: identifies a document.
- **The index/type/id of a document must be unique.**
- **Each document is stored in an index and has a type and an id.**
ElasticSearch : Basic Concepts

- **Shard**
  - A part of a index
  - An index is a logical namespace which points to primary and replica shards
  - **primary shard** : Each document is stored in a single primary shard. When you index a document, it is indexed first on the primary shard, then on all replicas of the primary shard.
  - **replica shard** : A replica is a copy of the primary shard, and has two purposes :
    - increase failover
    - increase performance
  - Each shard is in itself a fully-functional and independent "index" that can be hosted on any node in the cluster.
  - The mechanics of how a shard is distributed and also how its documents are aggregated back into search requests are completely managed by Elasticsearch and is transparent to you as the user.
ElasticSearch : The REST API

- **What can be done ?**
  - Check your cluster, node, and index health, status, and statistics
  - Administer your cluster, node, and index data and metadata
  - Perform CRUD (Create, Read, Update, and Delete) and search operations against your indexes
  - Execute advanced search operations such as paging, sorting, filtering, scripting, aggregations, and many others

- **The document API : CRUD API (index, get, delete, update)**

- **Ex : Index and Query a Document**

  **Index a JSON document:**

  ```
  PUT localhost:9200/index/type/ID
twitter/tweet/1
  ```

  **The result of the above index operation is:**

  ```
  {
    "_index": "twitter",
    "_type": "tweet",
    "_id": "1",
    "_version": 1,
    "created": true
  }
  ```
ElasticSearch : The document API

- The document API : CRUD API (index, get, delete, update)
- Ex : Index and Query a Document

**Query**

- GET localhost:9200/index/type/ID
twitter/tweet/1

```curl
-XGET 'http://localhost:9200/twitter/tweet/1'
```

The result of the above get operation is:

```json
{
  
  "_index": "twitter",
  "_type": "tweet",
  "_id": "1",
  "_version": 1,
  "found": true,
  "_source": {
    
    "user": "kimchy",
    "postDate": "2009-11-15T14:12:12",
    "message": "trying out Elasticsearch"
  }
}
```
ElasticSearch : Marvel

- **Marvel** = Web management and monitoring tool for Elasticsearch
- Outil qui collecte des métriques sur les nodes, index, ...
- free for development use.
ElasticSearch : Marvel / Sense

- Sense = interactive console which makes it easy to talk to Elasticsearch directly from your browser.

- Open with Marvel Menu : Dashboard/Sense
ElasticSearch : The Search API

- Allows to execute a search query and get back search hits that match the query.

- The query:
  - using a simple query string as a parameter ->
  ```
  $ curl -X GET 'http://localhost:9200/twitter/tweet/_search?q=user:kimchy'
  
  And here is a sample response:
  ```
  ```json
  {
    "_shards":{
      "total" : 5,
      "successful" : 5,
      "failed" : 0
    },
    "hits":{
      "total" : 1,
      "hits" : [
        {
          "_index" : "twitter",
          "_type" : "tweet",
          "_id" : "1",
          "_source" : {
            "user" : "kimchy",
            "postDate" : "2009-11-15T12:12:12",
            "message" : "trying out Elasticsearch"
          }
        }
      ]
    }
  }
  ```
  - Or using a request body
ElasticSearch : The Search API

- The query:
  - Or using a request body based on JSON ->

- Différents types:
  - basic:
    - « term »,
    - « match »,
    - « prefix »
  - Compound: « bool » using one or more boolean clause:
    - Must (=and)
    - Should (=or)

```bash
$ curl -X GET 'http://localhost:9200/twitter/tweet/_search' -d '{
  "query" : {
    "term" : { "user" : "kimchy" }
  }
}

And here is a sample response:

```json
{
  "_shards":{
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits":{
    "total" : 1,
    "hits" : [
      {
        "_index" : "twitter",
        "_type" : "tweet",
        "_id" : "1",
        "_source" : {
          "user" : "kimchy",
          "postDate" : "2009-11-15T14:12:12",
          "message" : "trying out Elasticsearch"
        }
      }
    ]
  }
}
```
ElasticSearch : The Search API

- make sense of your data: make aggregation

  « aggs » - Ex 1 Atelier

```json
PUT /person/person/1
{
  "name":"Anaelle Alessio",
  "dateOfBirth":"2009-09-05",
  "gender":"female",
  "marketing":{
    "shoes":1000,
    "fashion":1200,
    "music":800
  },
  "address":{
    "country":"England",
    "zipcode":"5226",
    "city":"Plymouth",
    "countrycode":"GB"
  }
}
```

```json
GET /person/person/_search
{
  "aggs": {
    "by_country": {
      "terms": {
        "field": "address.country"
      }
    }
  }
}
```

```json
{
  ...
  "aggregations": {
    "by_country": {
      "buckets": [{
        "key": "England",
        "doc_count": 30051
      }, {
        "key": "Germany",
        "doc_count": 30004
      }, {
        "key": "France",
        "doc_count": 15034
      }, {
        "key": "Spain",
        "doc_count": 14912
      }]
    }
  }
}
```
ElasticSearch : The Search API

- make sense of your data: make aggregation
  - « aggs » - Ex 2 Atelier
Kibana

- Kibana: Web analytics and visualization platform
  - Use to search, view, and interact with data stored in Elasticsearch indices.
  - Can easily perform advanced data analysis and visualize your data in a variety of charts, tables, and maps.
  - Assemble visualizations into a Dashboard
Kibana

- Ex dashboard atelier