La Rochelle Université

Géoluciole

Documentation technique suite elastic (ELK)

Master 2 Informatique
Année Universitaire 2019-2020
La Rochelle Université, France

Sommaire

Introduction

Ce document détaille la configuration du serveur et indique comment la modifier. Comme nous avons utilisé un environnement de test (serveur interne sur une machine distante) et un environnement de production (le serveur de l'Université), la configuration de chaque environnement et ses spécificités sont indiquées.

Serveur Université - datamuseum.univ-lr.fr

Prérequis d'utilisation :

- être connecté en wifi à **eduroam** ;
- avoir récupéré la clé SSH privée de Cyril Faucher et se situer dans le même répertoire que la clé.

Connexion au serveur

Pour se connecter au serveur de l'Université en SSH, exécuter la commande suivante :

```
ssh -i id_rsa_cyril root@datamuseum.univ-lr.fr
```

Un mot de passe sera demandé. Le mot de passe est : *csadminmte*.

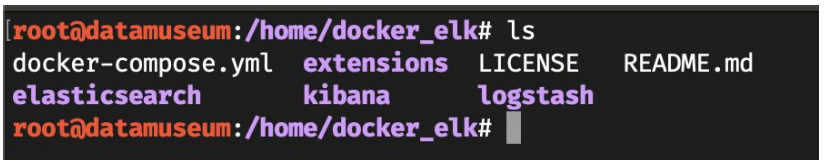
En cas de message d'erreur **SSH Key: "Permissions 0644 for 'ma_cle_ssh' are too open." on mac**, il faut exécuter la commande chmod 400 sur le fichier contenant la clé et ré-exécuter la commande de connexion.

Configuration

La suite elastic est configurée pour être lancée avec docker. Le répertoire de configuration du serveur est le suivant :

```
cd /home/docker_elk
```

L'arborescence du répertoire est la suivante :



Le fichier *docker-compose.yml* contient toute la configuration pour démarrer les conteneurs dockers regroupant les services de la suite elastic (ici, elasticsearch, kibana et logstash). Dans ce fichier, vous pourrez définir le nom des conteneurs, les répertoires de travail mais aussi les redirections de ports pour y accéder sur votre machine. Il existe ensuite un dossier de configuration par conteneur.

Elasticsearch est un serveur d'API qui va permettre la collecte de données. C'est sur ce serveur que l'application mobile Géoluciole va envoyer ces données.

Kibana est une interface web qui va permettre de visualiser les données reçues par Elasticsearch. Il va être possible de requêter les index ou bien de visualiser les données sous différentes formes (graphes, map, …). Pour mieux comprendre comment utiliser l'outil, aller à la section Fonctionnement Kibana.

Logstash, quant à lui, va permettre de collecter des données provenant de différentes sources, modifier ces données si nécessaire, et les envoyer vers une source unique. Il sert également d'outil de journalisation.

ATTENTION : Seul le port *80* est accessible sur le serveur de l'Université actuellement. Ce port est utilisé par Elasticsearch. Si voulez accéder à un autre service comme *Kibana*, il faut modifier le fichier *docker-compose.yml* et rediriger le conteneur *auquel vous souhaitez accéder* sur le port *80*.

Gestion des conteneurs avec la commande docker-compose

Cette partie détaille les commandes essentielles pour gérer les conteneurs dockers :

```
# Lancement des conteneurs
docker-compose up
# ajouter -d à la fin de la commande pour lancer les conteneurs en mode
arrière plan (ne bloque pas le terminal courant)

# Affichage de l'état de fonctionnement des conteneurs
docker-compose ps

# Arrêt de tous les conteneurs
docker-compose stop
```

Format de données

Schéma URL requête elasticsearch :

```
http://<host>:<port>/[index]/_doc/[_action/id]
```

Index

Cette partie détaille les index qui ont été créés pour le fonctionnement de l'application mobile Géoluciole :

Données gps

da3t_gps (id autogénéré)
- id_user - latitude - longitude - altitude - timestamp - precision - vitesse - date_str - location (auto-généré)

Données utilisateurs

da3t_compte (id manuel)
- id_user - date_gps - date_gps_str - consentement_gps - type (ios/android) - version - device _____ - mail - nom - prenom - phone - consentement_form - date_form - date_form_str

Questions du formulaire

da3t_question (id manuel)
- label

Réponses au formulaire

da3t_formulaire (id autogénéré)
- id_user - id_question - reponse

Fonctionnement Kibana

Accès à l'interface web

ATTENTION : Le port par défaut de Kibana est le port *5601*. Comme le port *80* est le seul accessible depuis le serveur de l'Université, il faut modifier le fichier *docker-compose.yml* pour rediriger Kibana sur le port *80*.

Le Kibana de l'Université est accessible depuis un navigateur à l'url suivante : http://datamuseum.univ-lr.fr/.

Détails des onglets

Onglet Discover

Cette partie permet de visualiser les données des index de manière brute. Vous pouvez sélectionner l'index à visualiser et vous accéderez à la structure de données de votre index ainsi qu'aux derniers enregistrements.

Exemple de visualisation de l'index *da3t_question*

Pour avoir accès à la visualisation d'un index, un Index Pattern doit être créé. Pour savoir comment créer un Index Pattern, aller à la section Management > Index Pattern.

Onglet Visualize

Cette partie permet de créer des représentations plus visuelles de vos index. Vous pouvez afficher vos données sous forme de graphe pour faire des statistiques ou bien afficher des points sur une carte si vous faites du tracking de position.

ATTENTION : Pour pouvoir afficher des points sur une carte, il faut que l'un des champs de votre index soit de type *geo_point*. Ce type n'étant pas pris en charge par défaut, il faut ajouter manuellement un champs de type *geo_point*. Cette manipulation est possible grâce au mécanisme de pipeline. Pour en savoir plus sur le fonctionnement des pipelines et comment les utiliser avec vos index, aller à la section Création de pipeline.

Onglet Dashboard

Cette partie permet de regrouper plusieurs visualisations créées via l'onglet **Visualize**. Cette interface a pour intérêt principal de créer une représentation personnalisée de nos données.

N'ayant pas approfondi son fonctionnement, vous trouverez plus d'informations sur la documentation officielle.

Onglet Timelion

Cette partie permet de visualiser des données provenant de sources de données multiples. Il est donc complémentaire à l'utilisation de Logstash. N'ayant pas approfondi son fonctionnement, vous trouverez plus d'informations sur la documentation officielle.

Onglet Dev Tools

Cette partie permet de requêter les données stockées dans les index via un mécanisme d'appel REST simplifié. Voici un exemple de requête REST

```
# Visualisation de la structure de données de l'index da3t_gps
GET da3t_gps/_mapping
```

Résultat de la commande de mapping


```
{
  "da3t_gps" : {
    "mappings" : {
      "_doc" : {
        "properties" : {
          "altitude" : {
            "type" : "float"
          },
          "id" : {
            "type" : "text",
            "fields" : {
              "keyword" : {
                "type" : "keyword",
                "ignore_above" : 256
              }
            }
          },
          "latitude" : {
            "type" : "float"
          },
          "location" : {
            "type" : "geo_point"
          },
          "longitude" : {
            "type" : "float"
          },
          "precision" : {
            "type" : "float"
          },
          "timestamp" : {
            "type" : "float"
          },
          "vitesse" : {
            "type" : "float"
          }
        }
      }
    }
  }
```


Comme illustré ci-dessus, pour requêter un index, il suffit d'indiquer le type de requête que l'on souhaite (GET), l'index que l'on souhaite interroger (da3t_gps) et le type d'opération à effectuer (mapping).

Globalement, les requêtes réalisées via cette interface sont identiques aux requêtes CURL classiques hormis que l'on ne préfixe par notre requête par l'url de la machine (dans notre cas *datamuseum.univ-lr.fr* pour une requête CURL vers le serveur de l'Université).

Requêtage des index

Vous pouvez faire des requêtes pour avoir des informations plus spécifiques sur vos index :

Exemple d'une requête où l'on veut récupérer les enregistrements de l'index da3t_gps compris entre 2 dates avec une limite de 500 résultats

```
GET da3t_gps/_search
{
      "sort" : [
        { "timestamp" : "desc" }
    ],
    "query": {
        "range" : {
            "timestamp" : {
              "gte" : 1579186800,
              "lte" : 1579190400
            }
        }
    },
    "size": 500
}
```

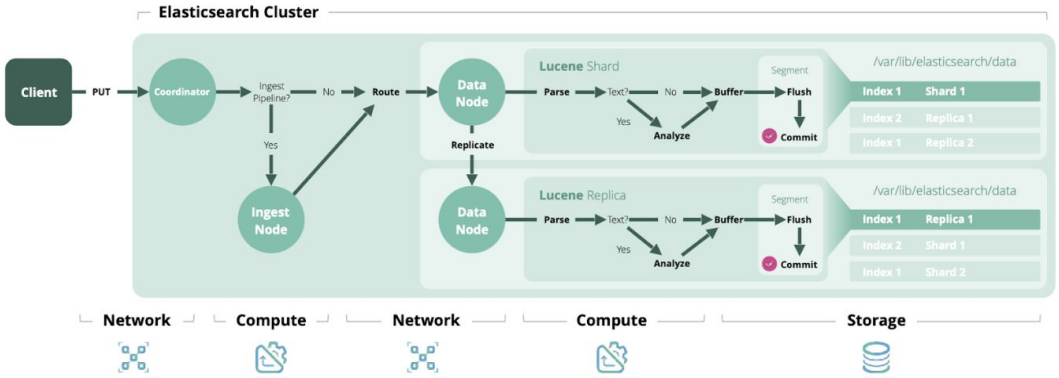
ATTENTION : Par défaut, une requête ne retourne que 10 résultats. Si vous voulez en afficher davantage, il faut indiquer la limite avec l'attribut *size* comme montré ci-dessus

Exemple d'une requête où l'on veut récupérer les enregistrements de l'index da3t_gps compris entre 2 dates pour un index donné

```
GET da3t_gps/_search
{
   "query": {
      "bool" : {
            "must" : [ {"match": {"id": 1919640340} } ],
            "filter" : [ {
                        "range" : {
                              "timestamp" : {
                                "gte" : 1579968000000,
                                 "lte" : 1579978800000
                              }
                         } } ]
      }
   }
}
```

Création de pipeline

Lorsque des données sont envoyées à Elasticsearch, ces données peuvent être modifiées via le mécanisme de pipeline. Voici un schéma expliquant la chaîne de traitement d'Elasticsearch pour indexer les données :



Source : elastic.co - diapo 24

Comme vous pouvez le constater, le mécanisme de pipeline intervient au tout début de la chaîne de traitement. Il est donc relativement simple de modifier les données reçues avant de les indexer.

Voici un exemple de pipeline :

```
PUT _ingest/pipeline/geopoint-pipeline
{
  "description": "Create geopoint field called 'location'"
  , "processors": [
    {
      "set": {
        "field": "location",
        "value": "{{latitude}},{{longitude}}"
      }
    }
  ]
}
```

Cette pipeline a pour but de récupérer les champs *longitude* et *latitude* pour créer un nouveau champs *location*.

Toutes les pipelines nécessitent d'être ajouté dans un index nommé *_ingest*. Pour pouvoir utiliser des pipelines dans un index, il faut exécuter cette commande :

```
PUT da3t_gps
```

```
{
  "mappings": {
    "_doc": {
      "properties": {
        "location": {
          "type": "geo_point"
        }
      }
    }
  },
  "settings": {
    "index.default_pipeline": "geopoint-pipeline"
  }
}
```

Nous avons définis la pipeline créée précédemment comme pipeline par défaut. Comme l'index n'était pas créé et que le type *geo_point* n'est pas pris en charge par défaut, nous avons créé le champ manuellement dans l'index. Si on envoi des données au serveur, on voit qu'une propriété 'location' est créée

Il est également possible de tester une pipeline avant de l'utiliser dans un index. Voici un exemple

```
POST _ingest/pipeline/geopoint-pipeline/_simulate
{
  "docs": [
    {
      "_source": {
        "latitude": 46.1667,
        "longitude": -1.15,
       }
    }
  ]
}
```

Voici le résultat du pipeline

Pour que le test fonctionne, il suffit de faire une requête POST terminant par *_simulate*. Il faut néanmoins spécifier un élément docs donner des éléments en entrée (équivalent au contenu d'une requête POST classique).

Si nous voulons supprimer cette pipeline par la suite, il suffit d'exécuter cette commande

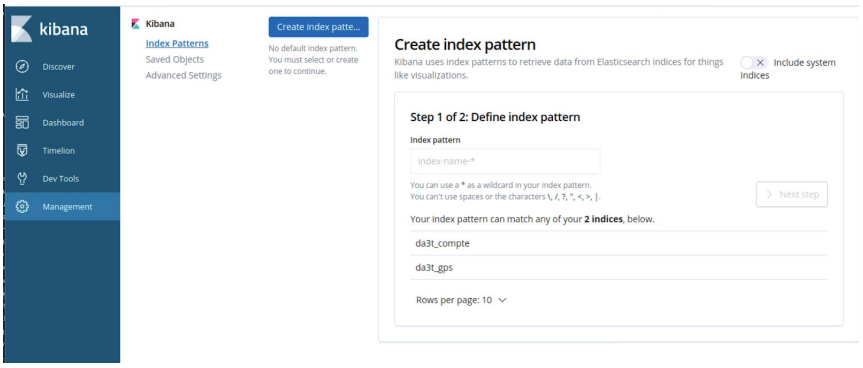
```
DELETE da3t_gps
{
  "settings": {
    "index.default_pipeline": "geopoint-pipeline"
  }
}
```

Onglet Management

Cette partie permet de gérer les index patterns qui nous permettent de visualiser les index dans la section **Discover** ou **VIsualize** par exemple. D'autres fonctionnalités existent, comme persister les données de notre dashboard ou d'un index. N'ayant pas expérimenté ces types d'utilisation, vous trouverez plus d'informations dans la documentation officielle.

Index Pattern

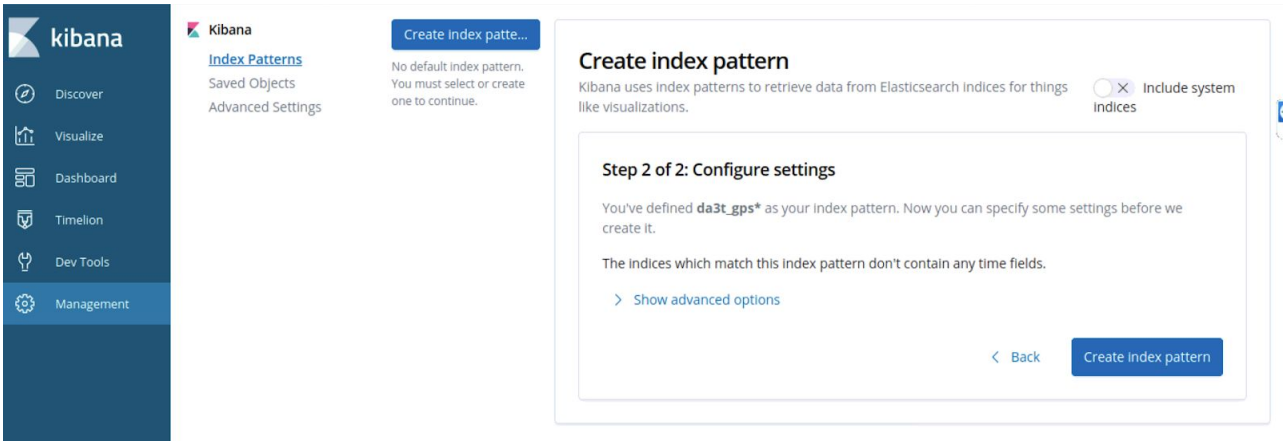
Les index pattern sont essentiels afin de visualiser nos données dans le **Dashboard** ou la section **Visualize**. Cliquer sur le bouton “Créer un Index Pattern” pour en créer un. Vous aurez ce genre d’interface :



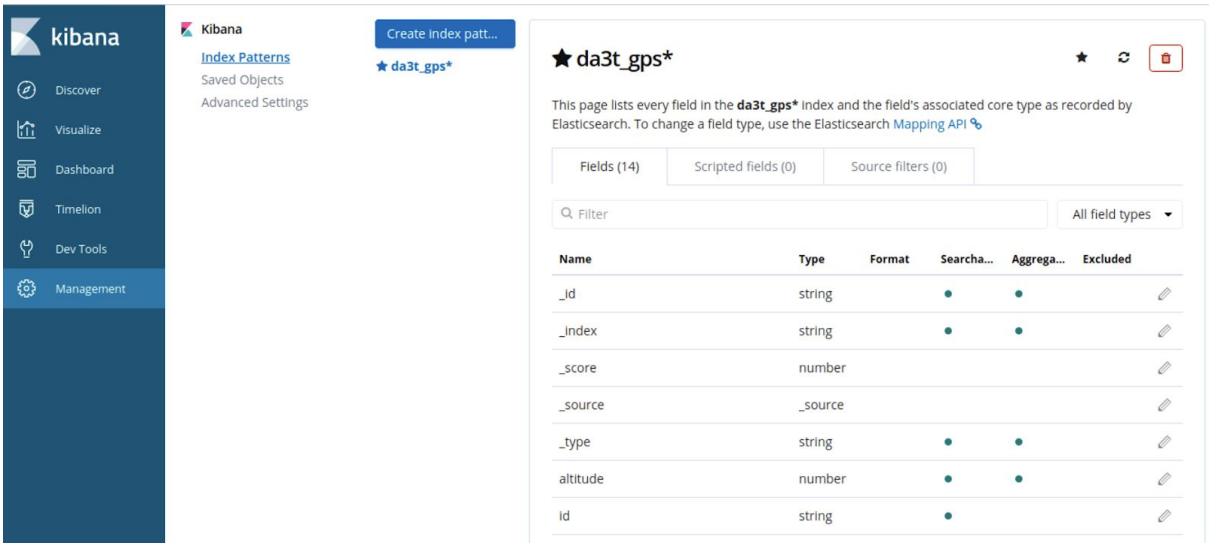
Il faudra ensuite renseigner le nom de l’index que vous souhaitez visualiser et cliquer sur le bouton “Next Step” lorsqu’il ne sera plus désactivé. Les index disponibles sont renseignés juste en dessous du champs de recherche pour faciliter la recherche d’un index.

ATTENTION : Il est important d’écrire le nom complet de l’index dans le champs de recherche. Sinon, le bouton ne s’activera pas.

Une fois appuyé sur le bouton “Next Step”, vous pourrez cliquer sur le bouton “Create Index pattern” pour valider la création



Votre index pattern apparaît désormais dans la liste des index pattern disponibles :


kibana
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
Kibana
Index Patterns
Saved Objects
Advanced Settings
Create index patt...
da3t_gps*
da3t_gps*
This page lists every field in the da3t_gps* index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API
Fields (14)
Scripted fields (0)
Source filters (0)
Filter
All field types
Name
Type
Format
Searcha...
Aggrega...
Excluded
_id
string
_index
string
_score
number
_source
_source
_type
string
altitude
number
id
string

Requêtes vers Elasticsearch

Prérequis d'utilisation :

- Le conteneur Elasticsearch doit être redirigé vers le port *80* pour que les requêtes fonctionnent.

Cette partie a pour but de montrer des exemples de requête HTTP que l'on peut faire vers Elasticsearch via l'outil CURL. L'idée est de montrer les urls à renseigner pour pouvoir communiquer avec Elasticsearch selon les besoins que vous avez.

GET

```
# Liste les index du serveur
curl -XGET datamuseum.univ-lr.fr:80/_cat/indices?v

# Retourne les éléments présents dans l'index da3t_gps (10 éléments
retournés par défaut)
curl -XGET http://datamuseum.univ-lr.fr:80/da3t_gps/_search

# Retourne le nombre d'éléments présent dans l'index da3t_gps
curl -XGET http://datamuseum.univ-lr.fr:80/da3t_gps/_count

# Retourne l'élément avec l'identifiant 1 de l'index da3t_gps
curl -XGET http://datamuseum.univ-lr.fr:80/da3t_gps/_doc/1
```

ASTUCE : Pour avoir un retour plus lisible en ligne de commande (indentation, …), rajouter *?pretty* à la fin de la commande :

```
# sur windows
curl -XGET http://datamuseum.univ-lr.fr:80/da3t_gps/_search?pretty

# sur mac, si un \ apparait après _search, laisser le (normal)
```

POST

```
# Ajout d'un élément dans l'index da3t_compte
curl -XPOST http://datamuseum.univ-lr.fr:80/da3t_compte/_doc/<id> -H
"Content-Type: application/json" -d "{\"consentement\":\"isokay\",
\"date\":\"15/01/2020\", \"nom\":\"jean\", \"prenom\": \"michel\",
\"mail\": \"mail@gmail.com\"}"

# Ajout d'un élément dans l'index da3t_gps
curl -XPOST http://datamuseum.univ-lr.fr:80/da3t_gps/_doc -H
"Content-Type: application/json" -d "{\"id_user\":\"...\",
\"latitude\":..., \"longitude\":..., \"altitude\": ..., \"timestamp\":
...}"

# Ajout d'un élément dans l'index da3t_question
curl -XPOST http://datamuseum.univ-lr.fr:80/da3t_question/_doc/<id> -H
"Content-Type: application/json" -d "{\"label\":\"...\"}"

# Ajout d'un élément dans l'index da3t_formulaire
curl -XPOST http://datamuseum.univ-lr.fr:80/da3t_formulaire/_doc -H
"Content-Type: application/json" -d "{\"id_user\":\"...\",
\"id_question\":\"...\", \"reponse\":\"...\"}"
```

ATTENTION : Le paramètre **id** n'est pas obligatoire. Si celui-ci n'est pas renseigné, il sera généré automatiquement par Elasticsearch. Si en revanche l'id est renseigné, il sera pris en compte. Dans le cas où l'on envoi des données plusieurs fois sur le même id, les données seront écrasées.

Bulk - envoie de plusieurs enregistrements en une requête

Il est possible d'envoyer plusieurs enregistrements à la fois à Elasticsearch via des *bulks*. Nous pouvons envoyer des données manuellement ou bien envoyer le contenu d'un fichier

```
# Ajoute les enregistrements du fichier « gps_bulk.json » dans l'index
da3t_gps
curl -XPOST datamuseum.univ-lr.fr:80/da3t_gps/_doc/_bulk -H
"Content-Type: application/json" --data-binary @gps_bulk.json
```

```
# Ajout de données manuellement dans l’index da3t_gps
curl -XPOST datamuseum.univ-lr.fr:80/da3t_gps/_doc/_bulk -H
"Content-Type: application/json" -d "{\"index\":
{}\n{\"id_user\":\"...\", \"latitude\":..., \"longitude\":...,
\"altitude\": ..., \"timestamp\": ...}\n{\"index\":
{}\n{\"id_user\":\"...\", \"latitude\":..., \"longitude\":...,
\"altitude\": ..., \"timestamp\": ...}"
# Chaque enregistrement doit commencer par la ligne index, un retour à
la ligne puis la ligne à insérer
```

Pour que les enregistrements soient pris en compte lors d’un bulk, il faut que chaque enregistrement à insérer respecte la sémantique suivante :

```
"index": {}
{"id_user": "...", "latitude": ..., "longitude": ..., "altitude": ...,
"timestamp": ...}
```

ATTENTION : Il faut spécifier un retour à la ligne après chaque ligne comme illustré ci-dessus.

PUT

Les requêtes PUT sont identiques aux requêtes POST. L’identifiant est néanmoins obligatoire car il s’agit d’une modification. Si l’identifiant n’existe pas, l’enregistrement est créé.

DELETE

```
# Supprime l’index « da3t_question » et tout son contenu
curl -XDELETE datamuseum.univ-lr.fr:80/da3t_question

# supprime la question avec l’id spécifié dans l’index da3t_question
curl -XDELETE datamuseum.univ-lr.fr:80/da3t_question/_doc/<id>
```