Elasticsearch, stocker une masse de données texte

La tête la première dans Kibana

- Accédez à l'ip de votre machine virtuelle Kibana

Vue Discover

Un exemple de données : des vols d'avions

- Des **évènements** très similaires à des **logs** mais plus facile à imaginer;
  - une date
  - un lieu
  - des informations spécifiques
- Avec des données de géolocalisation (ce qui est pas forcément le cas pour des logs)

Les différents champs décrivant chaque vol

- *DestCityName* et *OriginCityName* - Ville de départ et d'arrivé
- *timestamp* - Heure de départ
- *AvgTicketPrice* - le prix moyen des places pour le vol
- *FlightTimeHour* - Durée en Heure
- *DestWeather*, *OriginWeather* - La météo au départ et à l'arrivée.

Une première recherche

- Faites une petite recherche : "New"

Résultat:

- Resultats exacts : New York
- Résultats partiels: New Chitose Airport, Louis Armstrong New Orleans International Airport

Régler la Période

En haut à droite de la vue discover il y a le selecteur `Time Range`

- Combien de vols ?
  - depuis 24h ?
  - depuis une semaine ?
  - depuis 30 jours ?

Vue visualisations

- On peut créer des **visualisations** c'est à dire différents type de graphes et cartes.
- Permet de voir une proportion ou un changement en un coup d'oeil (quand on sait de quoi ça parle).

Vue Dashboard

- Vue globale d'un ensemble de **visualisation** pour comprendre rapidement les données
- Tout est dynamique: vous pouvez ajouter un filtre et les informations se mettent à jour

Dev tools

- Pour exécuter directement des requêtes *REST JSON* (on y reviendra)
- taper: TODO même requête que la recherche au dessus
- Ctrl+Entrée ou "Play" pour lancer la commande selectionnée.

Dev Tools

- Elle montre mieux les dessous des requêtes.
- L'API REST JSON permet de développer des applications interagissant avec elasticsearch.

Lucene et la recherche **Full text**

- **Elasticsearch** est un moteur de base de données utilisant le moteur d'indexation et de recherche **Lucene**.
- La principale force d'elasticsearch et de la stack ELK est de pouvoir gérer du texte brut en grande quantité grace à une indexation spécifique

Indexation Elasticsearch

La recherche "full text" fonctionne grâce à une indexation de chaque mot de chaque document de la base de données. En version simplifiée cela donne :

- Les champs `"text"` sont découpés en mots suivant les espaces
- Puis les mots sont mis en minuscules
- Ensuite comme elasticsearch connait les mots des langages naturels il "radicalise" (tokenize) les mots: `chercher` et `cherchant` deviennent `cherch`.
- Les radicaux resultants sont rangés dans un grand tableau à deux colonnes qu'on appelle l'index.
  - colonne de gauche les radicaux (tokens).
  - colonne de droite les documents dans lesquel le radical à été trouvé.
  - on peut ainsi récupérer tous les documents contenant un mot assez rapidement.

Recherche dans les champs **"text"** et **"keyword"**

- Exemple fulltext: `Destfull:"New"` -> champ `text` -> OK

  ```
  New chitose airport
  ```

  `Dest:"New"` -> champ `keyword` -> 0 hits

- Exemple recherche exacte: `DestCityName: "New York"` -> keyword pour la ville `Dest`:" John F Kennedy International Airport" -> keyword pour l'aéroport de destination

class: impact

Recherche Complexes

filtres et aggregations

Des requêtes complexes pour l'analyse

Elasticsearch est puissant pour l'analyse car il permet:

- de combiner un grande quantité de critères de recherche différent en même temps
- de transformer et afficher les données récupérées pour les rendre significatives

Exemple

- Ajouter un filtre avec le bouton "+ Add a Filter"
- Utiliser le filtre pour ne garder que les avions en retard

Imaginons qu'on veuille chercher tous les avions qui ont décollé de New York sous la pluie depuis un mois et qui ont un prix moyen supérieur à 800$. Par exemple pour créer une mesure du risque économique que le dérèglement climatique fait peser sur une companie ?

- On va devoir écrire une requête complexe.

requêtes composées

- Des requête avec des `AND` des `OR` et des `NOT` :
- Des filtres d'intervale avec : `Price:[500 TO 800]`
- Tous les vols qui concernent tel aéroport **et** qui contiennent le nom airways.

Filtres de requêtes

- En partant des résultats d'une recherche **fulltext** :
- On récupère les documents renvoyés par une requête (ce qu'Elastic appel des **hits**) et on ne va en garder qu'une partie*.
- Garder que les vols dont le prix est entre 300 et 1000 €:
  - `FlightDelayMin:[30 TO 50]`
  - rajouter un filtre avec le bouton "+ Add a Filter"
- La période de temps en haut à droite de kibana est aussi un filtre

Aggrégations des résultats de requêtes

- Très proche d'un **group by** en SQL.
- Grouper les documents/évènements par thème et faire des calculs transformations sur ces groupes.
- Pour calculer le prix moyen d'un ticket par compagnie par exemple : On va aggréger les vols de chaque compagnie et calculer la moyenne des prix des billets.

3 types principaux d'aggrégations:

- **Bucket** (faire des groupes)
  - Grouper les vols par prix.
- **Metric** (travailler sur une dimension des données)
  - calculer la moyenne des prix, ou du retard des vols
- **Geographique** (grouper par zone géographique)
  - Basés sur des coordonnées de localisation.
  - Le prix moyen au départ d'un pays.
- On peut **combiner** les aggrégations

Une métrique des données

- **Métrique** = caractéristique **chiffrée** des données.
  - # dans liste des propriétés dans Kibana. (pour faire une moyenne par exemple, il faut une quantité)