

Elastic Stack

David THIBAU – 2024

david.thibau@gmail.com

Agenda

- **Introduction**

- L'offre ELK et ses cas d'usage
- API ELS, Dev console de Kibana
- Concepts de Clustering

- **Alimentation des Index**

- Qu'est ce qu'un Document ?
- Document API et Routing
- Pipeline ElasticSearch
- Ingestion massive avec Beats et Logstash
- API Search Lite

- **Configuration d'index**

- Types de données
- Contrôle du mapping
- Configuration d'index
- Gabarit d'index

- **Recherche avancée avec DSL**

- Syntaxe DSL et combinaison de clauses
- Recherche filtre
- Recherche full-text
- Agrégations
- Géolocalisation

- **Analyse temps-réel avec Kibana**

- Présentation
- Mise au point de tableau de bord
- Types de visualisation
- Management
-

Introduction

L'offre Elastic Stack

L'API Rest

Concepts de clustering

Introduction

- **Elastic Stack** (avant ELK) est un groupe d'outils facilitant l'analyse et la visualisation temps-réel de données volumineuses
- Ces outils libres sont développés par la même structure, la société *Elastic*, qui encadre le développement communautaire et propose des services complémentaires (support, formation, intégration et hébergement cloud)

Principaux composants de la pile

- Elastic Stack est composé des outils suivants :
 - **Elasticsearch** : Base documentaire NoSQL basée sur le moteur de recherche Lucene
 - **Logstash** : Outil de traitement des traces qui exécutent différentes transformations, et exporte les données vers des destinations diverses dont les index Elasticsearch
 - **Beats** : Agents spécialisés permettant de fournir les données à logstash
 - **Kibana** est une application Angular permettant de visualiser les données présentes dans Elasticsearch

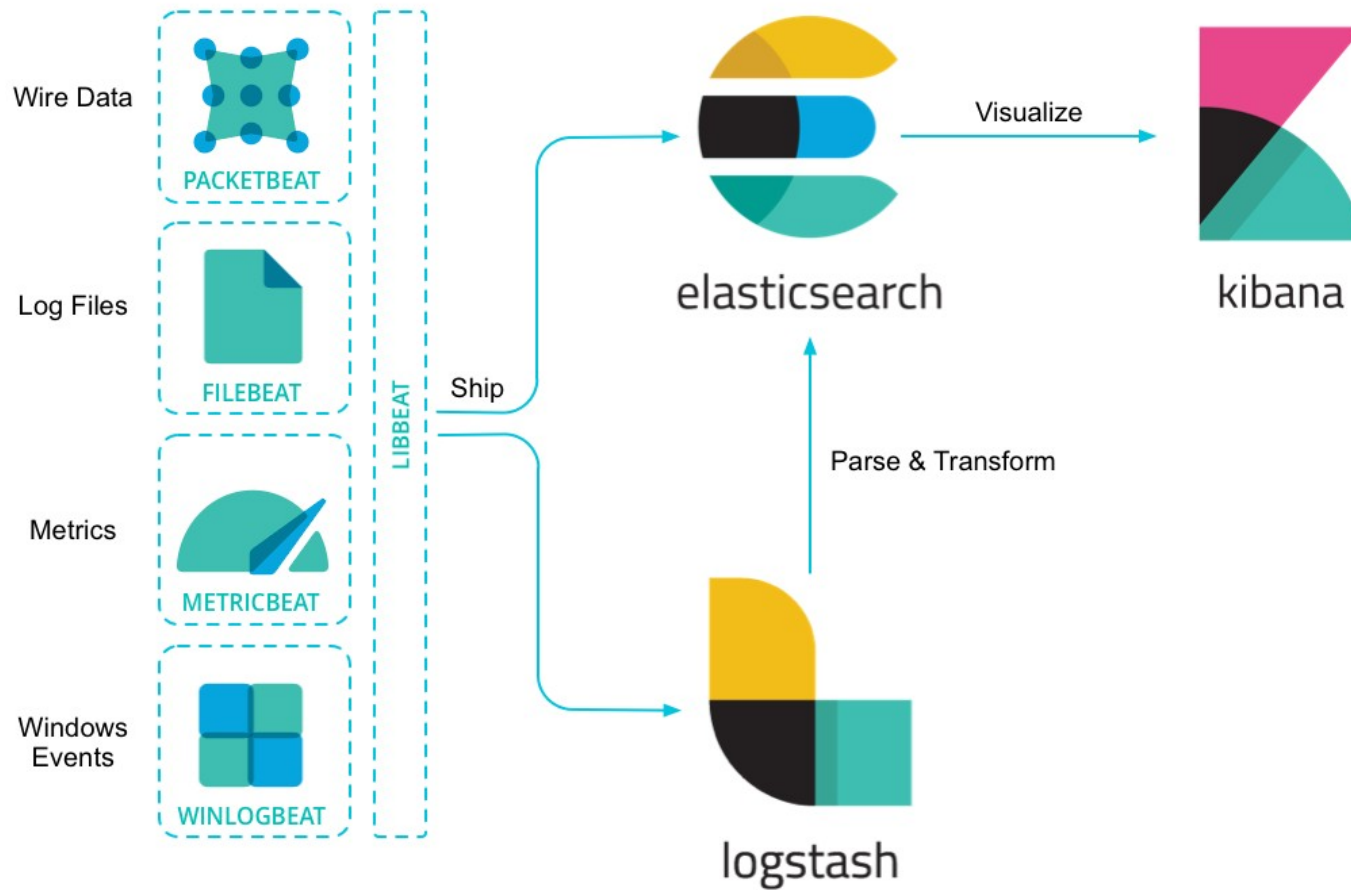
ElasticSearch

- ElasticSearch est un serveur offrant une API RestFul qui permet de :
 - Stocker et indexer des données
(Contenu web ou bureautique, tweets, fichiers de traces, métriques de performance, données structurées RDBMS...)
 - D'exécuter des requêtes de recherche
(structuré, full-text, langage naturel, geo-localisation)
 - Aggréger les données afin de calculer des KPI ou métriques

Caractéristiques

- ✓ Architecture massivement **distribuée et scalable** en volume et en charge
=> Adapté au Big Data, performance remarquable performance
- ✓ **Haute disponibilité** grâce à la réplication
- ✓ **Muti-tenancy** ou multi-index. Les données sont stockés dans des indexes dont les cycle de vie sont complètement indépendants
- ✓ Basé sur la librairie de référence **Lucene**
=> Recherche Full-text, prise en compte des langues, recherche floue, etc.
- ✓ **Flexibilité** : Stockage structuré des documents sans schéma préalable, possibilité d'ajouter des champs sur un schéma existant
- ✓ **API RESTFuL** très cohérente et complète
- ✓ **Open Source**

Architecture



Raisons du succès

- ELK est une plate-forme simple robuste et opensource. Ses principales atouts sont
 - Accès aux données en temps-réel
 - Scalable en ressources (CPU, RAM) et en volume (Disque)
 - API REST
 - Intégration aisée avec langages et framework. Nombreux clients, (Java, Javascript, PHP, Python, ...)
 - Documentation disponible et complète
 - Importante communauté
 - Netflix, Facebook, Microsoft, LinkedIn, Salesforce et Cisco l'ont choisi !

Offres connexes

- La société Elastic propose également des services payants :
 - ***X-Pack*** : fonctionnalités d'entreprise
 - Sécurité, Monitoring (avant release 8.x)
 - Alertes, Machine Learning
 - Application Performance Management
 - Reporting et planification
 - ***Elastic Cloud*** : Delegation de l'exploitation d'un cluster à Elastic

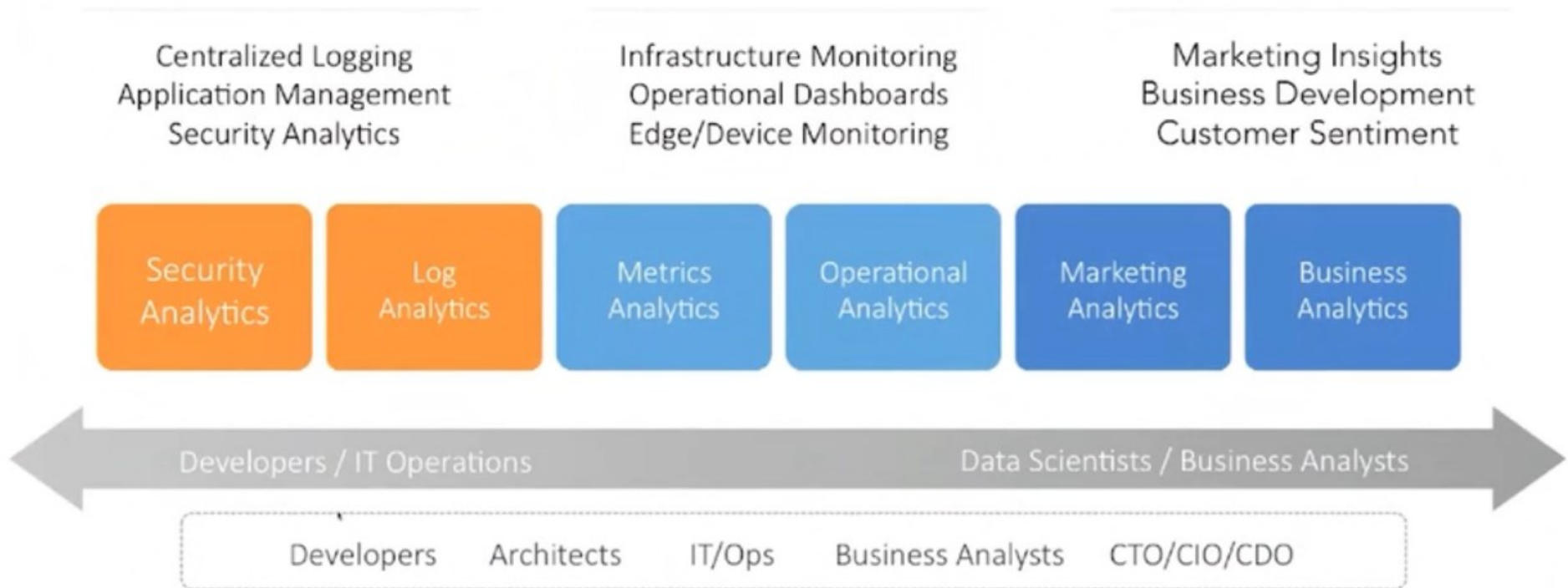
Solutions sur étagère

- ***Enterprise Search*** : Définir les sources de contenu (web, autre..), indexer le contenu et fournir des fonctionnalités de recherche
- ***Elastic Observability*** : Surveillez la consommation de CPU et de mémoire, agrégez les fichiers journaux, inspectez le trafic réseau, la gestion des performances des applications.
Machine Learning pour détecter les anomalies et anticiper le futur
- ***Elastic Security*** : Détecter et répondre aux menaces

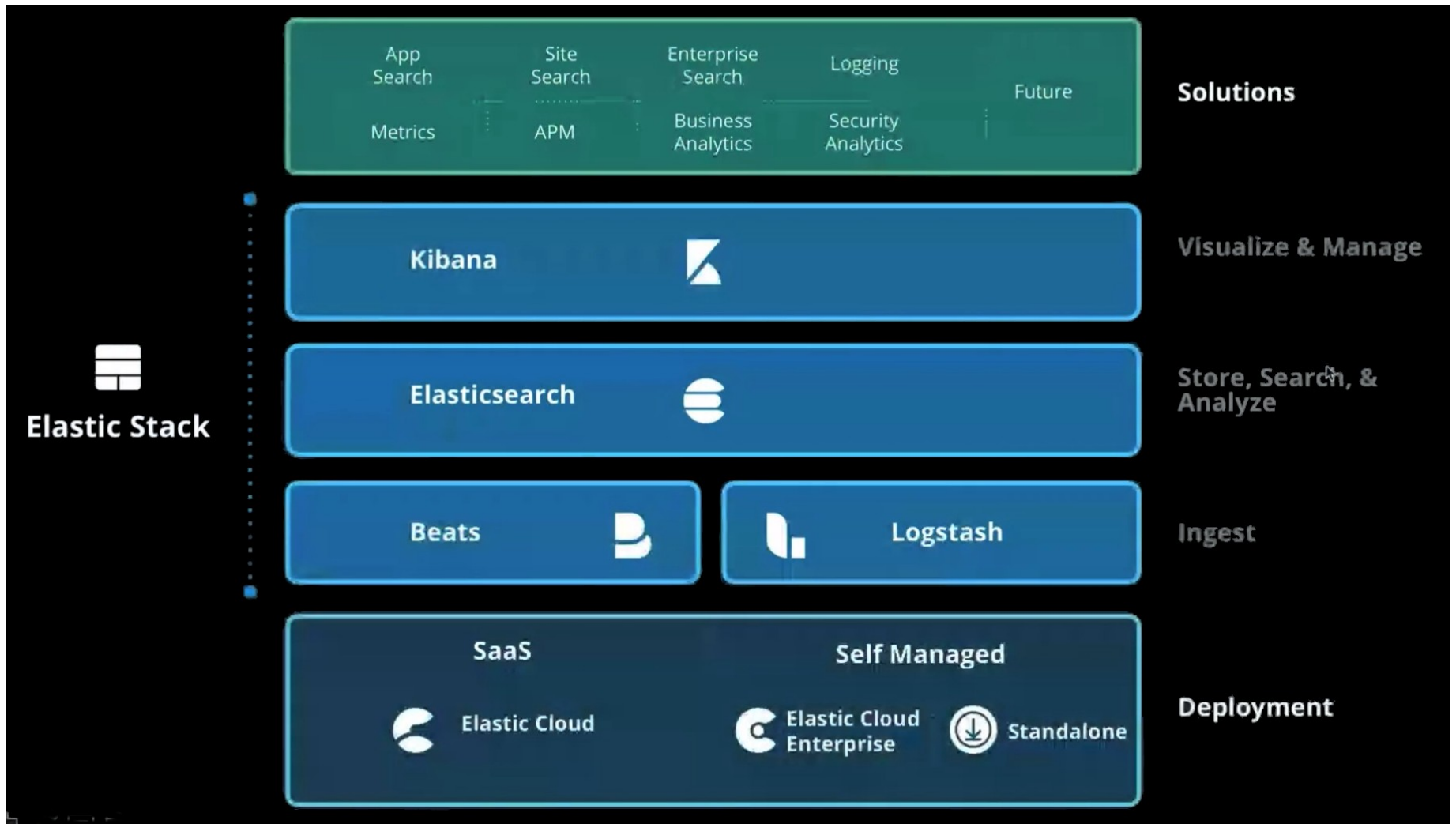
Autres Cas d'usage

- La pile a également d'autres cas d'usage
 - BI : Elle permet d'avoir une vue métier à 360°
 - Solution de recherche pour les applications webs ou mobiles
 - Comportement utilisateur, fréquentation, segmentation marketing
 - Gestion de risque métier et détection de fraude
 - L'Internet des objets connectés, (capteurs de santé, ...)
 - Big Data

Elastic Use cases



En résumé



Introduction

L'offre Elastic Stack

L'API Rest

Concepts de clustering

API d'ELS

- ELS expose donc une API REST utilisant JSON
- L'API est divisée en catégorie
 - API **d'index** (CRUD sur Index)
 - API **document** (CRUD sur document)
 - API **Ingest, Logstash** (Ingestion massive de documents)
 - API **Data Stream** (Données temporelles sur plusieurs index)
 - API de **recherche** (*_search*)
 - API **cluster** : gestion de l'architecture (*_cluster*)
 - ...

Introduction

- Les différentes API REST d'ELS respectent un ensemble de conventions
 - Pour les APIs travaillant sur les index, Possibilité de travailler sur plusieurs indexs, plusieurs datastream
 - Support des *Date*, *Math* dans les noms d'index.
(Utilisation de timestamp dans les noms d'index et calculs de date)
=> Je veux travailler sur mes données du mois dernier
 - Options/paramètres communs

Multiple index

```
# Tous les documents des index test1, test2 et test3  
test1,test2,test3/_search
```

```
# Tous les documents des index se terminant par test  
*test/_search
```

```
# Tous les documents des index se terminant par test  
# sauf test3  
+test*, -test3/_search
```

```
# Tous les documents de tous les index  
_all/_search
```

Noms d'index avec Date Math

- La résolution d'index avec Date Math permet de restreindre les index utilisés en fonction d'une date.
- Il faut que les index soient nommés avec des dates
- La syntaxe est :

<static_name{date_math_expr{date_format|time_zone}}>

- date_math_expr : Expression qui calcul une date
- date_format : Format de rendu
- time_zone : Fuseau horaire

- Exemples :

GET /<logstash-{now/d}>/_search => **logstash-2017.03.05**

GET /<logstash-{now/d-1d}>/_search => **logstash-2017.03.04**

GET /<logstash-{now/M-1M}>/_search => **logstash-2017.02**

Options communes (1)

- Les paramètres utilisent l'**underscore casing**
- **?pretty=true** : JSON bien formaté
- **?format=yaml** : Format yaml
- **?human=false** : Si la réponse est traitée par un outil
- **?filter_path** : Filtre de réponse, permettant de spécifier les données de la réponse
Ex : GET `/_cluster/state?pretty&filter_path=nodes`

Options communes (2)

- **?flat_settings=true** : Les settings sont renvoyés en utilisant la notation « . » plutôt que la notation imbriquée
- **?error_trace=true** : Inclut la stack trace dans la réponse
- Unités de temps : **d, h, m, s, ms, micros, nanos**
- Unité de taille : **b, kb, mb, gb, tb, pb**
- Nombre sans unités : **k, m, g, t, p**
- Unités de distance : **km, m, cm, mi, yd, ft**

Clients possibles

- Il est possible d'utiliser les clients REST classiques :
curl, navigateurs avec add-ons, *SOAPUI*, ...
- Elastic fournit des librairies dans la plupart des langages permettant d'intégrer les appels ELS dans vos applications.
- Enfin, le client le plus confortable est la **Dev Console de Kibana**

DevConsole Kibana

- Kibana, via sa ***DevConsole*** offre une interface utilisateur permettant d'interagir avec l'API REST *d'Elasticsearch*.
- Elle est composée de 2 onglets :
 - L'éditeur permettant de composer les requêtes
 - L'onglet de réponse
- La console comprend une syntaxe proche de Curl

```
GET /_search
{
  "query": {
    "match_all": {}
  }
}
```

Fonctionnalités

- La console permet une traduction des commandes CURL dans sa syntaxe
- Elle permet l'auto-complétion
- L'auto indentation
- Passage sur une seule ligne de requête (utile pour les requêtes BULK)
- Permet d'exécuter plusieurs requêtes
- Peut changer de serveur Elasticsearch
- Raccourcis clavier
- Historique des recherche
- Elle peut être désactivée (*console.enabled: false*)

Atelier 1.1 : Démarrage Cluster et DevConsole

Introduction

L'offre Elastic Stack

L'API Rest

Concepts de clustering

Cluster

- Un **cluster** est un ensemble de serveurs (nœuds) qui contient l'intégralité des données et offre des capacités de recherche sur les différents nœuds
 - Il est identifié par son nom unique sur le réseau local (par défaut : "*elasticsearch*").
- => Un cluster peut être constitué que d'un seul nœud
- => Un nœud ne peut pas appartenir à 2 clusters distincts

Nœud

- Un **nœud** est un simple serveur faisant partie d'un cluster.
- Un nœud stocke des données, et participe aux fonctionnalités d'indexation et recherche du cluster.
 - Un nœud est également identifié par un nom unique (généré automatiquement si pas renseigné)
- Le nombre de nœuds dans un cluster n'est pas limité

Nœud maître

- Dans un cluster un nœud est élu comme **nœud maître**, c'est lui qui est en charge de gérer la configuration du cluster comme la création d'index, l'ajout de nœud dans le cluster
- Pour toutes les opérations sur les documents (indexation, recherche), chaque nœud du cluster est **interchangeable** et un client peut s'adresser à n'importe lequel des nœuds

Index

- Un **index** est une collection de documents qui ont des caractéristiques similaires
 - Par exemple un index pour les données client, un autre pour le catalogue produits et encore un autre pour les commandes
- Un index est identifié par un nom (en minuscule)
 - Le nom est utilisé pour les opérations de mise à jour ou de recherche
- Dans un cluster, on peut définir autant d'index que l'on veut

Document

- Un **document** est l'unité basique d'information qui peut être indexée.
 - Le document est un ensemble de champs (clé/valeur) exprimé avec le format JSON
- A l'intérieur d'un index/type, on peut stocker autant de documents que l'on veut
- Un document d'un index doit être assigné à un type

Shard

- Un index peut stocker une très grande quantité de documents qui peuvent excéder les limites d'un simple nœud.
- Pour pallier ce problème, ELS permet de sous-diviser un index en plusieurs parties nommées *shards*
 - A la création de l'index, il est possible de définir le nombre de shards
- Chaque *shard* est un index indépendant qui peut être hébergé sur un des nœuds du cluster

Apports du sharding

- Le sharding permet :
 - De **scaler** le volume de contenu
 - De **distribuer** et paralléliser les opérations
=> augmenter les performances
- La mécanique interne de distribution lors de l'indexation et d'agrégation de résultat lors d'une recherche est complètement gérée par ELS et donc transparente pour l'utilisateur

Répliques

- Pour pallier à toute défaillance, il est recommandé d'utiliser des mécanismes de failover dans le cas où un nœud défaille
- ELS permet de mettre en place des copies des shards : les **répliques**
- La réplication permet
 - La **haute-disponibilité** dans le cas d'une défaillance d'un nœud (Une réplique ne réside jamais sur le nœud hébergeant le shard primaire)
 - Il permet de **scaler** le volume des requêtes car les recherches peuvent être exécutées sur toutes les répliques en parallèle .

-

Résumé

- En résumé chaque index peut être divisé sur plusieurs shards.
- Il peut être répliqué plusieurs fois
- Une fois répliqué, chaque index a des shards primaires et des shards de réplication
- Le nombre de shards et de répliques peuvent être spécifiés au moment de la création de l'index
- Après sa création, le nombre de répliques peut être changé dynamiquement mais pas le nombre de shards
- Par défaut, ELS alloue 1 shards et une réplique

Indexation et Documents

Qu'est-ce qu'un document

Document API

Pipeline d'ingestion

Ingestion massive avec Beats et Logstash

API Search Lite

Introduction

- *ElasticSearch* est une base documentaire distribuée.
- Il est capable de stocker et retrouver des structures de données (sérialisées en documents JSON) en temps réel
- Les documents sont constitués de champs.
- Chaque champ a un type
- Certains champs de type texte sont indexés

Structure de données

- Un document est donc une **structure de données**.
 - Il contient des champs et chaque champ a une ou plusieurs valeurs (tableau)
- Un champ peut également être une structure de données. (Imbrication)
- Le format utilisé par ELS est le format JSON

Exemple

```
{
  "name": "John Smith",      // String
  "age": 42,                 // Nombre
  "confirmed": true,        // Booléen
  "join_date": "2014-06-01", // Date
  "home": {                 // Imbrication
    "lat": 51.5,
    "lon": 0.1
  },
  "accounts": [              // tableau de données
    {
      "type": "facebook",
      "id": "johnsmith"
    }, {
      "type": "twitter",
      "id": "johnsmith"
    }
  ]
}
```

Méta-données

- Des méta-données sont également associées à chaque document.
- Les principales méta-données sont :
 - ***_index*** : L'emplacement où est stocké le document
 - ***_id*** : L'identifiant unique
 - ...

Indexation et Documents

Qu'est-ce qu'un document

Document API

Pipeline d'ingestion

Ingestion massive avec Beats et Logstash

API Search Lite

Introduction

- L'API document est l'API permettant les opérations CRUD sur la base documentaire
 - Création : *POST*
 - Récupération à partir de l'ID : *GET*
 - Mise à jour : *PUT*
 - Suppression : *DELETE*

Indexation et *id* de document

- Un document est identifié par ses méta-données *_index* , *_type* et *_id*.
- Lors de l'indexation (insertion dans la base Elastic), il est possible de fournir l'ID ou de laisser ELS le générer

Exemple avec Id

POST /{index}/_doc/{id}

```
{  
  "field": "value",  
  ...  
}  
...  
{  
  "_index": {index},  
  "_type": "_doc",  
  "_id": {id},  
  "_version": 1,  
  "created": true  
}
```

Exemple sans Id

POST *{index}/_doc*

```
{
  "field": "value",
  ...
}
...
{
  "_index": {index},
  "_type": "_doc",
  "_id": "wM00SFhDQXGZAWDf0-drSA",
  "_version": 1,
  "created": true
}
```

Récupération d'un document

- La récupération d'un document peut s'effectuer en fournissant l'identifiant complet :

GET /{index}/_doc/{id}?pretty

- La réponse contient le document et ses méta-données.
Exemple :

```
{  "_index" : "website",
  "_type": "_doc",
  "_id" : "123",
  "_version" : 1,
  "found" : true,
  "_source" : {
    "title": "My first blog entry",
    "text": "Just trying this out...",
    "date": "2014/01/01"  } }
```

Partie d'un document

- Il est possible de ne récupérer qu'une partie des données.
- Exemples :

Les méta-données + les champs title et text

GET /website/_doc/123?_source=title,text

Juste la partie source sans les méta-données

GET /website/_doc/123/_source

Vérifier qu'un document existe (Retour 200)

HEAD /website/_doc/123

Mise à jour

- Les documents stockés par ELS sont immuables.
 - => La mise à jour d'un document consiste à indexer une nouvelle version et à supprimer l'ancienne.
- La suppression est asynchrone et s'effectue en mode batch (suppression de toutes les répliques)

Mise à jour d'un document

PUT /website/_doc/123

```
{  
  "title": "My first blog entry",  
  "text": "I am starting to get the hang of this...",  
  "date": "2014/01/02"  
}  
...  
{  
  "_index" : "website",  
  "_type": "_doc",  
  "_id" : "123",  
  "_version" : 2,  
  "created": false  
}
```


Suppression d'un document

```
DELETE /website/_doc/123
```

```
...
```

```
{
```

```
  "found" : true,
```

```
  "_index" : "website",
```

```
  "_type": "_doc",
```

```
  "_id" : "123",
```

```
  "_version" : 3
```

```
}
```

Autres opérations de l'API document

- Mise à jour partielle : **_update** permet de fusionner les champs fournis avec les champs existants
- Scripts : Permet de mettre à jour des champs avec un script (PainLess ou Groovy)
- Mise à jour multiples : API **bulk** permet de faire plusieurs opérations en 1 seule requête

Indexation et Documents

Qu'est-ce qu'un document

Document API

Pipeline d'ingestion

Ingestion massive avec Beats et Logstash

API Search Lite

Pipeline

- Les pipelines d'ingestion permettent d'effectuer des transformations sur les données avant l'indexation.
Ex : Supprimer des champs, extraire des valeurs , enrichir les données, ...
- Une **pipeline** est définie par 2 champs :
 - Une description
 - Une liste de **processeurs**
- Les processeurs correspondent au pré-traitements effectués dans leur ordre de déclaration



Ingest API

- L'API ***_ingest*** permet de gérer les pipelines :
 - ***PUT*** pour ajouter ou mettre à jour une pipeline
 - ***GET*** pour retourner une pipeline
 - ***DELETE*** pour supprimer
 - ***SIMULATE*** pour simuler un appel à une pipeline

Kibana

- Kibana permet de gérer les pipelines via :
Stack Management > Ingest Pipelines
- On peut alors :
 - Affichez une liste de vos pipelines et accédez aux détails
 - Modifier ou cloner des pipelines existants
 - Supprimer des pipelines

Usage

#Création de la pipeline nommée attachment

```
PUT _ingest/pipeline/attachment
```

```
{
  "description" : "Extrait le texte de document bureautique",
  "processors" : [
    { "attachment" : { "field" : "data" } }
  ]
}
```

#Utilisation de la pipeline lors de l'indexation d'un doc.

```
PUT my_index/my_type/my_id?pipeline=attachment
```

```
{
  "data":
  "e1xydGYxXGFuc2kNCkxvcmVtIGlwc3VtIGRvbG9yIHNPdCBhbWV0DQpccGFyIH0="
}
```

Résultat

```
GET my_index/my_type/my_id
{
  "found": true,
  "_index": "my_index",
  "_type": "my_type",
  "_id": "my_id",
  "_version": 1,
  "_source": {
    "data":
    "e1xydGYxXGFuc2kNCkxvcmVtIGlwc3VtIGRvbG9yIHNpdCBhbWV0DQpccGFyIH0=",
    "attachment": {
      "content_type": "application/rtf",
      "language": "ro",
      "content": "Lorem ipsum dolor sit amet",
      "content_length": 28
    }
  }
}
```


Processeurs disponibles

- Enrichissement de données :
 - Ajouter une valeur à un champ
 - Enrichir les données avec un autre index
 - Utiliser le ML pour classifier un champ
 - Extraire le texte de documents bureautique
 - Convertir une IP en localisation géographique
 - ...
- Transformation de données
 - Conversion de types, Renommage, Remplacement de chaînes
 - Splitter une donnée d'entrée en plusieurs champs.
Ex : une ligne de log, un fichier CSV, ..
- Filtrage
 - Suppression d'un champ
 - Suppression du document selon certaines conditions

Indexation et Documents

Qu'est-ce qu'un document

Document API

Pipeline d'ingestion

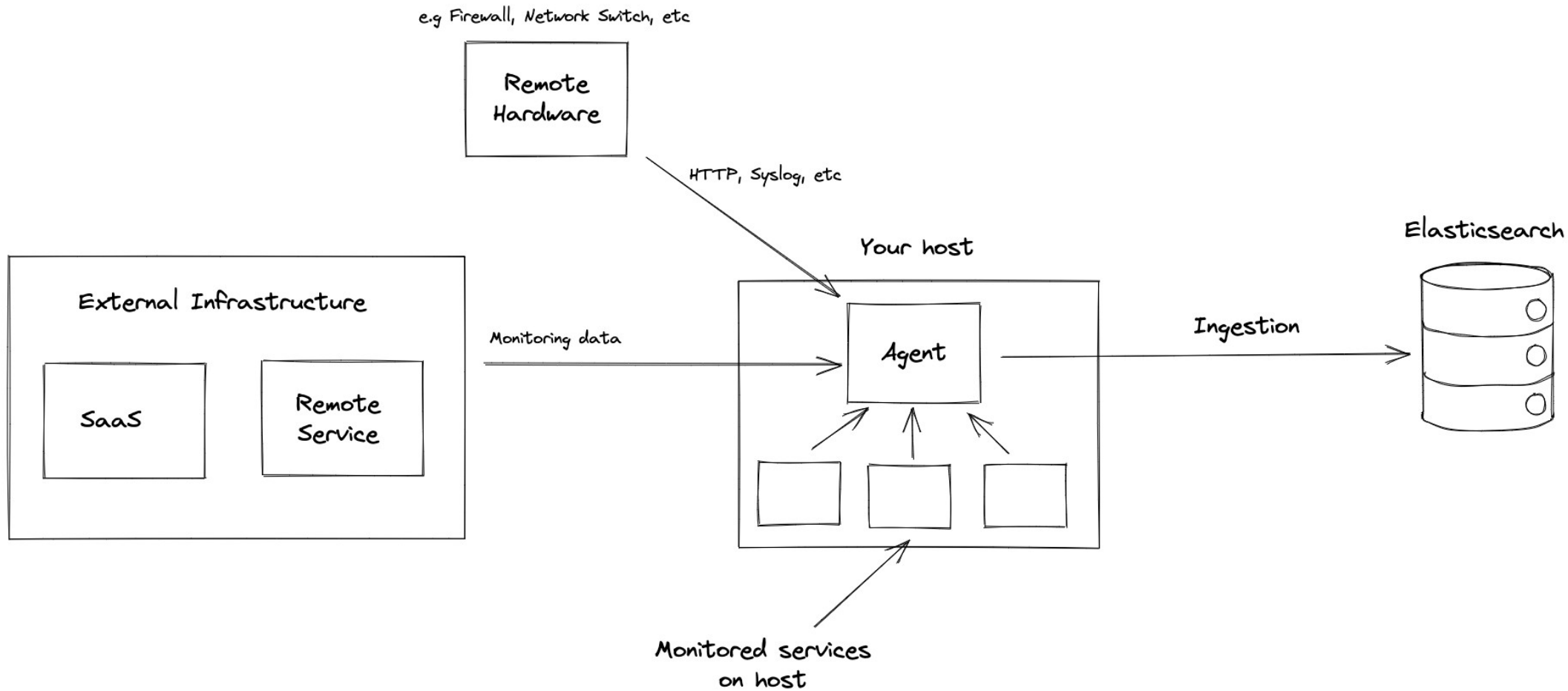
**Ingestion massive avec Beats et
Logstash**

API Search Lite

Ingestion de données

- Différentes alternatives existent pour importer des données dans Elasticsearch
 - **Elastic Agent / Integrations**, un process capable de surveiller des métriques locaux ou distants aux formats standards (Recommandé dans les versions 8.x)
 - **Beats** : Des convoyeurs de données installés sur les machines cibles
 - **Logstash** : Permet de mettre au point des pipelines de transformations très performante
 - Les **librairies clientes** fournies par Elastic permet de développer des programmes qui alimentent Elasticsearch
 - Les applications sur étagère d'Elastic (Web Enterprise Search par ex.)

Elastic Agent



Elastic Integrations

- Elastic Integrations offrent un moyen simple de connecter Elastic à des services et systèmes externes.
 - Livrés avec des ressources prêtes à l'emploi telles que des tableaux de bord, des visualisations Kibana et des pipelines pour extraire des champs structurés.
 - Supportent les services et plates-formes populaires tels que Nginx ou AWS, ainsi que pour de nombreux types d'entrées génériques tels que les fichiers journaux.
 - Kibana fournit une interface utilisateur Web pour ajouter et gérer des intégrations.

Vous pouvez parcourir une vue unifiée des intégrations disponibles qui montre à la fois les intégrations Elastic Agent et Beats.

All categories	384
APM	1
AWS	41
Azure	25
Cloud	9
Containers	15
Custom	44
Database	39
Elastic Stack	51
Elasticsearch SDK	9
Search	37
Google Cloud	20
Network	56
Observability	124
Operating Systems	6

☐ Display beta integrations

If an integration is available for **Elastic Agent and Beats**, show:

- ☒ Recommended ⓘ
- ☐ Elastic Agent only
- ☐ Beats only



Amazon VPC

Collect Amazon VPC flow logs with Elastic Agent



Amazon VPN

Collect VPN metrics with Elastic Agent



Anomali

Ingest threat intelligence indicators from Anomali with Elastic Agent.



Apache HTTP Server

Collect logs and metrics from Apache servers with Elastic Agent.



Apache Spark

Collect metrics from Apache Spark with Elastic Agent.



Apache Tomcat

Collect and parse logs and metrics from Apache Tomcat servers with Elastic Agent.



API

Add search to your application with Elasticsearch's robust APIs.



Arbor Peakflow Logs

Collect and parse logs from Netscout Arbor Peakflow SP with Filebeat.



Arista NG Firewall

Collect logs and metrics from Arista NG Firewall.



Atlassian Bitbucket

Collect logs from Atlassian Bitbucket with Elastic Agent.



Atlassian Confluence

Collect logs from Atlassian Confluence with Elastic Agent.



Atlassian Jira

Collect logs from Atlassian Jira with Elastic Agent.



Auditbeat Events

Collect events from your servers with Auditbeat.



Auditd Logs

Collect logs from Linux audit daemon with Elastic Agent.



Auditd Manager

The Auditd Manager Integration receives audit events from the Linux Audit Framework that is a part of the Linux kernel.

Beats

- Les **beats** sont des convoyeurs de données
- Ils sont installés comme agents sur les différents serveurs et envoient leur données
 - Directement à *ElasticSearch*
 - Ou à des instances *logstash* qui peuvent effectuer certaines transformations sur les données
- De nombreux beats ont été remplacés par des Integrations

Beats fournis par Elastic

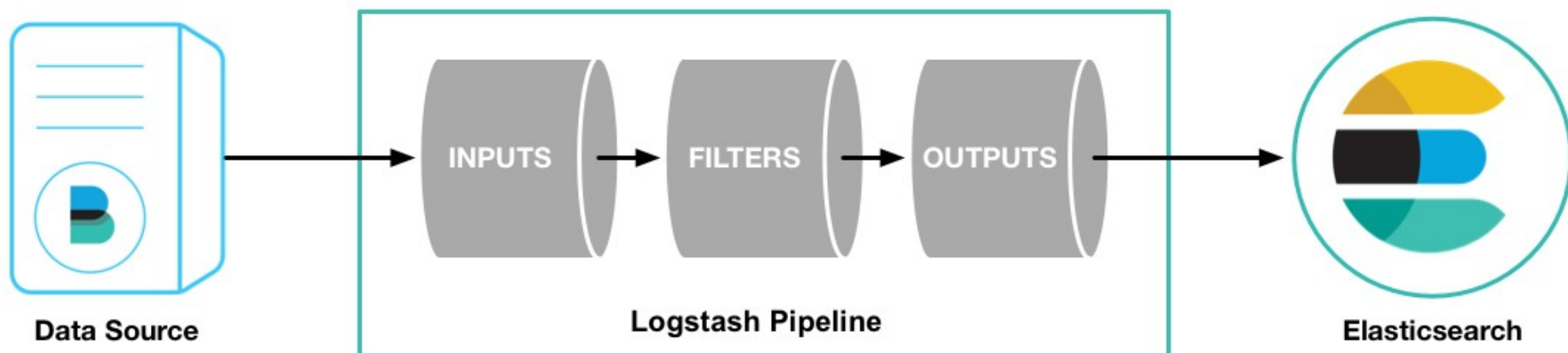
- Elastic fourni les beats suivants :
 - ***Packetbeat*** : Un analyseur de paquets réseau qui convoie des informations sur les transactions entre les différents serveurs applicatifs
 - ***Filebeat*** : Convoie les fichiers de trace des serveurs.
 - ***Metricbeat*** : Un agent de monitoring qui collecte des métriques sur l'OS et les services qui s'y exécutent
 - ***Winlogbeat*** : Événements Windows
 - ***Auditbeat*** : Convoie les données d'audit
 - ***Heartbeat*** : Convoie les données de disponibilité
- Il est possible de créer ses propre beats. ELK offre la librairie *libbeat* (écrit en Golang) comme support

Logstash

- *Logstash* est un outil de **collecte de données temps-réel capable d'unifier et de normaliser** les données provenant de sources différentes
 - A l'origine centré sur les fichiers de traces, il peut s'adapter à d'autres types d'événements
- *Logstash* est basé sur la notion de **pipeline** de traitement.
Il permet de filtrer, mixer et orchestrer différentes entrées
- Extensible via des plugins, il se connecte à de nombreuses sources de données

Pipeline Logstash

- Une **pipeline** logstash a au minimum :
 - Un plugin d'entrée pour lire les données
 - Une plugin de sortie pour écrire les données
 - Optionnellement des filtres entre les 2



Exemple de pipeline

Les pipelines sont définies dans des fichiers de configuration à la syntaxe logstash.

```
input {
  beats { port => "5043" }
}
filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
  geoip {
    source => "clientip"
  }
}
output {
  elasticsearch {
    hosts => [ "localhost:9200" ]
  }
}
```

Principaux Inputs

- Quelques inputs :
 - **beats** : Événements d'un beat
 - **elasticsearch** : Lit des résultats de requêtes à partir d'un cluster Elasticsearch
 - **exec** : Sortie d'une commande shell
 - **file** : Lecture de lignes de fichier
 - **generator, heartbeat** : Génère des événements pour les test
 - **jdbc** : Événements à partir de JDBC
 - **Kafka, redis, rabbitmq, jms** : Messaging
 - **Tcp, udp, log4j, unix, syslog** : Évènements sockets bas niveau
 - ...

Exemples

```
input {
  beats {
    port => 5044
  }
  file {
    path => "/var/log/*"
    exclude => "*.gz"
  }
  generator {
    lines => [
      "line 1",
      "line 2",
      "line 3"
    ]
    # Emet toutes les lignes 3 fois.
    count => 3
  }
}
```

Quelques filtres

- ***cidr*** : Vérifie des adresses IP
- ***csv*** : Parse le format csv
- ***date*** : Parse des champs afin de déterminer le timestamp de l'événement
- ***dns*** : Effectue un lookup DNS
- ***drop*** : Supprime un événement
- ***elapsed*** : Calcul le temps entre 2 événements
- ***environment*** : Stocke des variables d'environnement comme champ metadata
- ***extractnumbers*** : Extrait des nombres à partir d'une string

Quelques filtres (2)

- **geoip** : Ajoute des informations latitude/longitude à partir d'une IP
- **grok** : Divise un champ en d'autres champs
- **json** : Parse les événements JSON
- **metrics** : Calcul des agrégations sur un évènements (Comptage, cadence)
- **mutate** : Transforme un champ
- **prune** : Filtre des événements à partir d'une liste rouge ou blanche
- **range** : Vérifie que la longueur d'un champ est dans un intervalle
- **translate** : Remplace les valeurs des champs à partir d'une table de hash ou fichier YAML
- **truncate** : Tronque les champ supérieur à une certaine longueur
- **urldecode** : Décode les champs URL-encoded
- **xml** : Parse le format XML

Les plugins outputs

- Quelques outputs
 - **csv** : Ecrit sur le disque au format CSV
 - **elasticsearch** : Stocke dans Elasticsearch
 - **email** : Envoie un email à une adresse spécifiée
 - **exec** : Exécute une commande si un évènement correspond
 - **file** : Ecrit dans un fichier
 - **stdout** : Ecrit sur la sortie standard
 - **pipe** : Envoie vers l'entrée standard d'un autre programme
 - **http** : Envoie sur un endpoint HTTP ou HTTPS
 - **nagios, nagios_nasca** : Envoi vers Nagios
 - **tcp, udp, syslog, statsd, websocket** : Sockets
 - **kafka, redis, rabbitMQ** : Messaging
 - ...

Output *elasticsearch*

- L'output ***elasticsearch*** contient de nombreuses options. Les plus importantes sont :
 - ***hosts*** : Les hôtes du cluster
 - ***index*** : Le nom de l'index. Exemple : logstash-%{+YYYY.MM.dd}. Attention, le nom influe sur le gabarit d'index utilisé
 - ***template*** : Un chemin vers le fichier template
 - ***template_name*** : Le nom du template côté Elasticsearch
 - ***template_overwrite*** (false) : Permet de mettre à jour le template utilisé
 - ***document_id*** : L'id du document (permet les mises à jour d'événements)
 - ***parent*** : Le document parent de l'événement (nested document)
 - ***pipeline*** : La pipeline côté Elasticsearch à utiliser
 - ***routing*** : Peut spécifier le shard à utiliser

Exemple complet log Apache

```
input {
  file {
    path => "/var/apache/*_log"
  }
}
# Traitements différents en fonction du type de log
filter {
  if [path] =~ "access" {
    mutate { replace => { type => "apache_access" } }
    grok {
      #gabarit connu par grok
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
    date {
      # Normalisation de la date
      match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
    }
  } else if [path] =~ "error" {
    mutate { replace => { type => "apache_error" } }
  } else {
    mutate { replace => { type => "random_logs" } }
  }
}

output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Résultat pour une ligne de log d'accès

```
{
    "message" => "127.0.0.1 - - [11/Dec/2013:00:01:45 -0800] \"GET
/xampp/status.php HTTP/1.1\" 200 3891 \"http://cadenza/xampp/navi.php\" \"Mozilla/5.0
(Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0\"",
    "@timestamp" => "2013-12-11T08:01:45.000Z",
    "@version" => "1",
    "host" => "cadenza",
    "clientip" => "127.0.0.1",
    "ident" => "-",
    "auth" => "-",
    "timestamp" => "11/Dec/2013:00:01:45 -0800",
    "verb" => "GET",
    "request" => "/xampp/status.php",
    "httpversion" => "1.1",
    "response" => "200",
    "bytes" => "3891",
    "referrer" => "\"http://cadenza/xampp/navi.php\"",
    "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0)
Gecko/20100101 Firefox/25.0\""
}
```

Support Kibana

- Dans sa version commerciale, ElasticStack propose la gestion centralisée des pipeline logstash dans Kibana
- Il est alors possible :
 - De contrôler et surveiller plusieurs instances Logstash
 - Ajouter, modifier et supprimer des configurations de pipeline

Management → Stack Management

Ateliers 2.3 : Traitements de logs applicatifs

Indexation et Documents

Qu'est-ce qu'un document

Document API

Pipeline d'ingestion

Ingestion massive avec Beats et Logstash

API Search Lite

APIs de recherche

- Il y a donc 2 API de recherche :
 - Une version **simple** qui attend que tous ses paramètres soient passés dans la chaîne de requête
 - La version **complète** composée d'un corps de requête JSON qui utilise un langage riche de requête appelé DSL

Search Lite

- La version *lite* est cependant très puissante, elle permet à n'importe quel utilisateur d'exécuter des requêtes lourdes portant sur l'ensemble des champs des index.
- Ce type de requêtes peut être un trou de sécurité permettant à des utilisateurs d'accéder à des données confidentielles ou de faire tomber le cluster.
- => En production, on interdit généralement ce type d'API au profit de DSL

Recherche vide

GET /_search

- Retourne tous les documents de tous les index du cluster

Réponse

```
{
  "hits" : {
    "total" : 14,
    "hits" : [ {
      "_index": "us",
      "_type": "tweet",
      "_id": "7",
      "_score": 1,
      "_source": {
        "date": "2014-09-17",
        "name": "John Smith",
        "tweet": "The Query DSL is really powerful and flexible",
        "user_id": 2
      }
    }, ... RESULTS REMOVED ... ],
    "max_score" : 1
  }, tooks : 4,
  "_shards" : {
    "failed" : 0,
    "successful" : 10,
    "total" : 10
  },
  "timed_out" : false
}
```

Champs de la réponse

- ***hits*** : Le nombre de document qui répondent à la requête, suivi d'un tableau contenant l'intégralité des 10 premiers documents. Chaque document a un élément `_score` qui indique sa pertinence. Par défaut, les documents sont triés par pertinence
- ***took*** : Le nombre de millisecondes pris par la requête
- ***shards*** : Le nombre total de shards ayant pris part à la requête. Certains peuvent avoir échoués
- ***timeout*** : Indique si la requête est tombée en timeout. Il faut avoir lancé une requête de type :
GET `/_search?timeout=10ms`

Limitation à un index

- ***_search*** : Tous les index, tous les types
- ***/gb/_search*** : Tous les types de l'index gb
- ***/gb,us/_search*** : Tous les types de l'index gb et us
- ***/g*,u*/_search*** : Tous les types des index commençant par g ou u

Pagination

- Par défaut, seul les 10 premiers documents sont retournés.
- ELS accepte les paramètres *from* et *size* pour contrôler la pagination :
 - ***size*** : indique le nombre de documents devant être retournés
 - ***from*** : indique l'indice du premier document retourné

Paramètre *q*

- L'API search lite prend le paramètre *q* qui indique la chaîne de requête
- La chaîne est parsée en une série de
 - Termes : (Mot ou phrase)
 - Et d'opérateurs (AND/OR)
- La syntaxe support :
 - Les caractères joker et les expressions régulières
 - Le regroupement (parenthèses)
 - Les opérateurs booléens (OR par défaut, + : AND, - : AND NOT)
 - Les intervalles : Ex : [1 TO 5]
 - Les opérateurs de comparaison

Exemples search lite

GET /_all/_search?q=tweet:elasticsearch

Tous les documents de type *tweet* dont le champ *full text match* « *elasticsearch* »

+name:john +tweet:mary

GET /_search?q=%2Bname%3Ajohn+%2Btweet%3Amary

- Tous les documents dont le champ full-text *name* correspond à « *john* » et le champ *tweet* à « *mary* »
- Le préfixe *+* indique des conditions qui ne doivent pas matcher

Champ *_all*

- Lors de l'indexation d'un document, ELS concatène toutes les valeurs de type string dans un champ full-text nommé ***_all***
- C'est ce champ qui est utilisé si la requête ne précise pas de champ

GET /_search?q=mary

- Si le champ *_all* n'est pas utile, il est possible de le désactiver

Exemple plus complexe

- La recherche suivante utilise les critères suivants :
- Le champ *name* contient « mary » ou « john »
- La date est plus grande que « 2014-09-10 »
- Le champ *_all* contient soit les mots « aggregations » ou « geo »

+name:(mary john) +date:>2014-09-10 +(aggregations geo)

- Voir doc complète :
<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html#query-string-syntax>

Autres Paramètres de la query string

- Les autres paramètres disponibles sont :
 - **df** : Le champ par défaut, utilisé lorsque aucun champ n'est précisé dans la requête
 - **default_operator** (AND/OR) : L'opérateur par défaut. Par défaut OR
 - **explain** : Une explication du score ou de l'erreur pour chaque hit
 - **_source** : *false* pour désactiver la récupération du champ *_source*.
Possibilité de ne récupérer que des parties du document avec *_source_include* & *_source_exclude*
 - **sort** : Le tri. Par exemple *title:desc,_score*
 - **timeout** : Timeout pour la recherche. A l'expiration du timeout, les hits trouvés sont retournés

Mapping et types de données

Types de données
Contrôle du mapping
Gabarits d'index

Types simples supportés

- ELS supporte :
 - Les chaînes de caractères : *string*
 - *text* ou *keyword* (pas analysé)
 - Les numériques : *byte* , *short* , *integer* , *long*, *float* , *double*, *token_count*
 - Les booléens : *boolean*
 - Les dates : *date*
 - Les octets : *binary*
 - Les intervalles : *integer_range*, *float_range*, *long_range*, *double_range*, *date_range*
 - Les adresses IP : *IPV4* ou *IPV6*
 - Les données de géo-localisation : *geo_point*, *geo_shape*
 - Une requête (structure JSON) : *percolator*

Valeur exacte ou full-text

- Les chaînes de caractère indexées par ELS peuvent être de deux types :
 - **keyword** : La valeur est prise telle quelle, des opérateurs de type filtre peuvent être utilisés lors de la recherche
Foo!= foo
 - **text** : La valeur est analysée et découpée en termes ou token.
Des opérateurs de recherche full-text peuvent être utilisés lors des recherche. Cela concerne des données en langage naturelS

Index inversé

- Afin d'accélérer les recherches sur les champs texte, ELS utilise une structure de donnée nommée **index inversé**
- Cela consiste en une liste de mots unique où chaque mot est associé aux documents dans lequel il apparaît.

	A	B
1	term	docs
2	pizza	3, 5
3	solr	2
4	lucene	2, 3
5	sourcesense	2, 4
6	paris	1, 10
7	tomorrow	1, 2, 4, 10
8	caffè	3, 5
9	big	6
10	brown	6
11	fox	6
12	jump	6
13	the	1, 2, 4, 5, 6, 8, 9

Analyseurs

- Les **analyseurs** utilisés à l'indexation et lors de la recherche, transforment un champ texte en un flux de “token” ou mots qui constituent l’index inversé.
- ELS propose des analyseurs prédéfinis :
 - **Analyseur Standard** : C'est l'analyseur par défaut. Le meilleur choix lorsque le champ texte est dans des langues diverses. Il consiste à :
 - Séparer le texte en mots
 - Supprimer la ponctuation
 - Passer tous les mots en minuscule
 - **Analyseurs de langues** : Ce sont des analyseurs spécifiques à la langue. Ils incluent les « stop words » (enlève les mots les plus courant) et extrait la racine d'un mot. C'est le meilleur choix si le champ texte est dans une langue fixe

Test des analyseurs

GET /_analyze?analyzer=standard
Text to analyze

Réponse :

```
{  
  "tokens": [ {  
    "token": "text",  
    "start_offset": 0,  
    "end_offset": 4,  
    "type": "<ALPHANUM>",  
    "position": 1  
  }, {  
    "token": "to",  
    "start_offset": 5,  
    "end_offset": 7,  
    "type": "<ALPHANUM>",  
    "position": 2  
  }, {  
    "token": "analyze",  
    "start_offset": 8,  
    "end_offset": 15,  
    "type": "<ALPHANUM>",  
    "position": 3  
  } ] }
```

Comportement par défaut

- Lorsque ELS détecte un nouveau champ *String* dans un type de document, il le configure automatiquement comme 2 champs :
 - 1 champ *text* utilisant l'analyseur standard.
 - 1 champ *keyword*
- Si ce n'est pas le comportement voulu, il faut explicitement spécifier le **mapping** lors de la création de l'index

Types complexes

- En plus des types simples, ELS supporte
 - Les **tableaux** : Il n'y a pas de mapping spécial pour les tableaux. Chaque champ peut contenir 0, 1 ou n valeurs `{ "tag": ["search", "nosql"] }`
 - Les valeurs d'un tableau doivent être de même type
 - Un champ à *null* est traité comme un tableau vide
 - Les **objets** : Il s'agit d'une structure de données embarquées dans un champ
 - Les **nested** : Il s'agit d'un tableau de structures de données embarquées dans un champ. Chaque élément du tableau est stocké séparément dans un document Elasticsearch (~Jointure)

Mapping et types de données

Types de données
Contrôle du mapping
Gabarits d'index

Mapping

- ELS est capable de générer dynamiquement le mapping
 - Il devine alors le type des champs
- On peut voir le mapping d'un type de données dans un index par :

GET /gb/_mapping/tweet

Réponse *_mapping*

```
{ "gb": {  
  "mappings": {  
    "properties": {  
      "date": {  
        "type": "date",  
        "format": "dateOptionalTime"  
      },  
      "name": {  
        "type": "text"  
      },  
      "tweet": {  
        "type": "text"  
      },  
      "user_id": {  
        "type": "long"  
      }  
    }  
  }  
}
```

Mapping

- Un mapping définit pour chaque champ d'un type de document
 - Le type de donnée
 - Si champ est de type *text*, l'analyseur associé
 - Les méta-données associées

Mapping personnalisé

- Un mapping personnalisé permet (entre autre) de :
 - Spécifier le type d'un champ
 - Faire une distinction entre les champs string de type full-text ou valeur exacte (*keyword*)
 - Utiliser des analyseurs spécifiques
 - Définir plusieurs types et analyseurs pour le même champ
 - Spécifier des formats de dates personnalisés
 - ...

Spécification du mapping

- On peut spécifier le mapping :
 - Lors de la création d'un index
 - Lors de l'ajout d'un nouveau champ dans un index existant

=> On ne peut pas modifier un champ déjà indexé

Le point d'entrée de l'API est ***_mapping***

Création manuelle de l'index

Lors de la création d'un, on fournit 2 blocs de configuration :

- **settings** : Principalement répliques et shards
- **mappings** : Les types de champs et les analyseurs

```
PUT /my_index
{
  "settings": { ... any settings ... },
  "mappings": {
    "type_one": { ... any mappings ... },
    ...
  }
}
```


Exemple (création)

```
PUT /gb
{
  "settings": {
    "number_of_shards" : 20,
    "number_of_replicas" : 2
  },
  "mappings": {
    "properties" : {
      "tweet" : {
        "type" : "text",
        "analyzer": "english"
      },
      "date" : {
        "type" : "date"
      },
      "name" : {
        "type" : "keyword"
      },
      "user_id" : {
        "type" : "long"
      }
    }
  }
}
```

Exemple : Ajout d'un nouveau champ

```
PUT /gb/_mapping/
{
  "properties" : {
    "tag" : {
      "type" : "text",
      "index": "not_analyzed"
    }
  }
}
```

Mapping et types de données

Types de données
Contrôle du mapping
Gabarits d'index

Introduction

- Les gabarits d'index permettent de définir des gabarits qui s'appliquent lors de la création d'index, lorsque son nom respecte un pattern (propriété ***index-pattern***)
- Ils définissent 2 aspects:
 - ***settings*** : Nombre de shards, de répliques, etc
 - ***mappings*** : Types des champs

API Rest

- Les gabarits d'index utilise l'API Rest ***template***. Ex :

```
PUT _template/template_1
{
  "index_patterns": ["te*", "bar*"],
  "settings": { "number_of_shards": 1 },
  "mappings": {
    "properties": {
      "host_name": {
        "type": "keyword"
      },
      "created_at": {
        "type": "date",
        "format": "EEE MMM dd HH:mm:ss Z YYYY" } } } }
```

Exemple pattern logstash

```
"mappings" : {
  "dynamic_templates" : [
    {
      "message_field" : {
        "path_match" : "message",
        "match_mapping_type" : "string",
        "mapping" : {
          "type" : "text",
          "norms" : false } } },
    {
      "string_fields" : {
        "match" : "*",
        "match_mapping_type" : "string",
        "mapping" : {
          "type" : "text",
          "norms" : false,
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256 } } } } }
  ],
  "properties" : {
    "@timestamp" : {"type" : "date" },
    "@version" : {"type" : "keyword"},
    "geoip" : {
      "dynamic" : true,
      "properties" : {
        "ip" : { "type" : "ip"},
        "location" : { "type" : "geo_point" },
        "latitude" : { "type" : "half_float" }, "longitude" : {"type" : "half_float" } } } }
  }
}
```

Recherche avec DSL

Syntaxe DSL

Principaux opérateurs

Agrégations

Géolocalisation

Introduction DSL

- Les recherches avec un corps de requête offrent plus de fonctionnalités qu'une recherche simple.
- En particulier, elles permettent :
 - De combiner des clauses de requêtes plus facilement
 - D'influencer le score
 - De mettre en surbrillance des parties du résultat
 - D'agréger des sous-ensemble de résultats
 - De retourner des suggestions à l'utilisateur
 - ...

GET ou POST

- La RFC 7231 qui traite de la sémantique HTTP ne définit pas des requêtes GET avec un corps
 - => Seuls certains serveurs HTTP le supportent
- ELS préfère cependant utiliser le verbe GET car cela décrit mieux l'action de récupération de documents
- Cependant, afin que tout type de serveur HTTP puisse être mis en frontal de ELS, les requêtes GET avec un corps peuvent également être effectuées avec le verbe POST

Recherches vides

```
GET /_search
```

```
{}
```

```
GET /index_2014*/_search
```

```
{}
```

```
GET /_search
```

```
{
```

```
"from": 30,
```

```
"size": 10
```

```
}
```

Requête DSL

- Le langage DSL permet d'exposer toute la puissance du moteur Lucene avec une interface JSON
- Pour utiliser DSL, il faut passer une requête dans le paramètre *query* :

```
GET /_search
```

```
{ "query": YOUR_QUERY_HERE }
```

Principe DSL

- DSL peut être vu comme un arbre syntaxique qui contient :
 - Des clauses de requête **feuille**.
Elles correspondent à un type de requête (*match*, *term*, *range*) s'appliquant à un champ. Elles peuvent s'exécuter seules
 - Des clauses **composées**.
Elles combinent d'autres clauses (feuille ou composée) avec des opérateurs logiques (*bool*, *dis_max*) ou altèrent leurs comportements (*constant_score*)
- De plus, les clauses peuvent être utilisées dans 2 contextes différents qui modifient leur comportement
 - Contexte **requête ou full-text**
 - Contexte **filtre**

Exemple Combinaison

```
"query" {  
  "bool": {  
    "must": { "match": { "tweet": "elastic" }},  
    "must_not": { "match": { "name": "mary" }},  
    "should": { "match": { "tweet": "full text" }}  
  }  
}
```

Distinction entre requête et filtre

- DSL permet d'exprimer deux types de requête
 - Les **filtres** sont utilisés pour les champs contenant des valeurs exactes. Leur résultat est de type booléen. Un document satisfait un filtre ou pas
 - Les **recherches** calculent un score de pertinence pour chaque document trouvé. Le résultat est trié par le score de pertinence

Activation des contextes

- Le contexte requête est activée dès lors qu'une clause est fournie en paramètre au mot-clé *query*
- Le contexte filtre est activée dès lors qu'une clause est fournie en paramètre au mot-clé *filter* ou *must_not*

Exemple

```
GET /_search
{
  "query": { // activation du contexte requête
    "bool": { // Les clauses bool, must et match sont exécutées dans un contexte requête
      "must": [
        { "match": { "title": "Search" } },
        { "match": { "content": "Elasticsearch" } }
      ],
      "filter": [ // activation du contexte filtre
        { "term": { "status": "published" } }, // exécuté dans un contexte filtre
        { "range": { "publish_date": { "gte": "2015-01-01" } } } // contexte filtre
      ]
    }
  }
}
```


Recherche avec DSL

Syntaxe DSL

Principaux opérateurs

Agrégations

Géolocalisation

Opérateurs dans le contexte filtres

- **term** : Utilisé pour filtrer des valeurs exactes :

```
{ "term": { "age": 26 } }
```
- **terms** : Permet de spécifier plusieurs valeurs :

```
{ "terms": { "tag": [ "search", "full_text", "nosql" ] } }
```
- **range** : Permet de spécifier un intervalle de date ou nombre :

```
{ "range": { "age": { "gte": 20, "lt": 30 } } }
```
- **exists** et **missing** : Permet de tester si un document contient ou pas un champ

```
{ "exists": { "field": "title" } }
```
- **bool** : Permet de combiner des clauses avec :
 - **must** équivalent à ET
 - **must_not** équivalent à NOT
 - **should** équivalent à OU

match

- La recherche *match* est la recherche standard pour effectuer une recherche exacte ou full-text sur presque tous les champs.
 - Si la requête porte sur un champ full-text, il analyse la chaîne de recherche en utilisant le même analyseur que le champ,
 - si la recherche porte sur un champ à valeur exacte, il recherche pour la valeur exacte
- { "match": { "tweet": "About Search" } }

OR par défaut

- *match* est une requête booléenne qui par défaut analyse les mots passés en paramètres et construit une requête de type OR. Elle peut être composée de :
 - ***operator*** (and/or) :
 - ***minimum_should_match*** : le nombre de clauses devant matcher
 - ***analyzer*** : l'analyseur à utiliser pour la chaîne de recherche
 - ***lenient*** : Pour ignorer les erreurs de types de données

Example

```
GET /_search
{
  "query": {
    "match" : {
      "message" : {
        "query" : "this is a test",
        "operator" : "and"
      }
    }
  }
}
```

multi_match

- La requête ***multi_match*** permet d'exécuter la même requête sur plusieurs champs :

```
{"multi_match": { "query": "full text search", "fields": [ "title", "body" ] } }
```

- Les caractères joker peuvent être utilisés pour les champs
- Les champs peuvent être boostés avec la notation ^

```
GET /_search
{
  "query": {
    "multi_match" : {
      "query" : "this is a test",
      "fields" : [ "subject^3", "text*" ]
    }
  }
}
```

Filtre *prefix*

- Le filtre *prefix* est une recherche s'effectuant sur le terme. Il n'analyse pas la chaîne de recherche et assume que l'on a fourni le préfixe exact

```
GET /my_index/address/_search
{
  "query": {
    "prefix": { "postcode": "W1" }
  }
}
```

Wildcard et regexp

- Les recherche **wildcard** ou **regexp** sont similaires à *prefix* mais permet d'utiliser des caractère joker ou des expressions régulières

```
GET /my_index/address/_search
```

```
{  
  "query": {  
    "wildcard": { "postcode": "W?F*HW" }  
  }  
}
```

```
GET /my_index/address/_search
```

```
{  
  "query": {  
    "regexp": { "postcode": "W[0-9].+" }  
  }  
}
```


match_phrase

- La recherche ***match_phrase*** analyse la chaîne pour produire une liste de termes mais ne garde que les documents qui contiennent tous les termes dans la même position.

```
GET /my_index/my_type/_search
{
  "query": {
    "match_phrase": { "title": "quick brown
fox" }
  }
}
```

Proximité avec *slop*

- Il est possible d'introduire de la flexibilité au phrase matching en utilisant le paramètre **slop** qui indique à quelle distance les termes peuvent se trouver.
- Cependant, les documents ayant ces termes les plus rapprochés auront une meilleur pertinence.

```
GET /my_index/my_type/_search
{
  "query": {
    "match_phrase": {
      "title": {
        "query": "quick fox",
        "slop": 20 // ~ distance en mots
      }
    }
  }
}
```

match_phrase_prefix

- La recherche ***match_phrase_prefix*** se comporte comme *match_phrase*, sauf qu'il traite le dernier mot comme un préfixe
- Il est possible de limiter le nombre d'expansions en positionnant *max_expansions*

```
{  
  "match_phrase_prefix" : {  
    "brand" : {  
      "query": "johnnie walker bl",  
      "max_expansions": 50  
    }  
  }  
}
```

Requête fuzzy

- La recherche fuzzy ou recherche floue permet de gérer des erreurs de typo en récupérant les termes approchant (Distance de Levenshtein)
- Elle est équivalente à une recherche par terme

```
GET /my_index/my_type/_search
{
  "query": {
    "fuzzy": {"text": "surprize" }
  }
}
```

2 paramètres peuvent être utilisées pour limiter l'impact de performance :

- ***prefix_length*** : Le nombre de caractères initiaux qui ne seront pas modifiés.
- ***max_expansions*** : Limiter les options que la recherche floue génère. La recherche fuzzy arrête de rassembler les termes proches quand elle atteint cette limite

Recherche avec DSL

Syntaxe DSL

Principaux opérateurs

Agrégations

Géolocalisation

Introduction

- Les agrégations sont extrêmement puissantes pour le reporting et les tableaux de bord
- Des énormes volumes de données peuvent être visualisées en temps-réel
 - Le reporting change au fur et à mesure que les données changent
- L'utilisation de la stack Elastic, Logstash et Kibana démontre bien tout ce que l'on peut faire avec les agrégations.

Syntaxe DSL

- Une agrégation peut être vue comme une unité de travail qui construit des informations analytiques sur un ensemble de documents.
- En fonction de son positionnement dans l'arbre DSL, il s'applique sur l'ensemble des résultats de la recherche ou sur des sous-ensembles
- Dans la syntaxe DSL, un bloc d'agrégation utilise le mot-clé **aggs**

// Le max du champ *price* dans tous les documents

```
POST /sales/_search?size=0
```

```
{
  "aggs" : {
    "max_price" : { "max" : { "field" : "price" } }
  }
}
```

Types d'agrégations

- Plusieurs concepts sont relatifs aux agrégations :
 - **Groupe ou Buckets** : Ensemble de document qui ont un champ à la même valeur ou partagent un même critère. Les groupes peuvent être imbriqués. ELS propose des syntaxes pour définir les groupes et compter le nombre de documents dans chaque catégorie
 - **Métriques** : Calculs de métriques sur un groupe de documents (min, max, avg, ..)
 - **Matrice** : Opérations de classification selon différents champs, produisant une matrice des différentes possibilités. Le scripting n'est pas supporté pour ce type d'agrégation
 - **Pipeline** : Une agrégation s'effectuant sur le résultat d'une agrégation

Example Bucket

```
GET /cars/transactions/_search
{
  "aggs" : {
    "colors" : {
      "terms" : { "field" : "color.keyword" }
    } } }
}
```

Réponse

```
{
...
"hits": { "hits": [] },
"aggregations": {
"colors": {
  "doc_count_error_upper_bound": 0, // incertitude
  "sum_other_doc_count": 0,
  "buckets": [
    { "key": "red", "doc_count": 4 },
    { "key": "blue", "doc_count": 2 },
    { "key": "green", "doc_count": 2 }
  ]
} } }
```

Exemple métrique

GET /cars/transactions/_search

```
{
  "size": 0,
  "aggs" : {
    "avg_price" : {
      "avg" : {"field" : "price"}
    }
  }
}
```

Réponse

```
"hits": {  
  "total": 8,  
  "max_score": 0,  
  "hits": []  
},  
"aggregations": {  
  "avg_price": {  
    "value": 26500  
  }  
}
```

Juxtaposition Bucket/Métriques

GET /cars/transactions/_search

```
{
  "size": 0,
  "aggs" : {
    "colors" : {
      "terms" : { "field" :
"color.keyword" }
    },
    "avg_price" : {
      "avg" : {"field" : "price"}
    }
  }
}
```

Réponse

```
{  
  ...  
  "hits": { "hits": [] },  
  "aggregations": {  
    "avg_price": {  
      "value": 26500  
    },  
    "colors": {  
      "doc_count_error_upper_bound": 0,  
      "sum_other_doc_count": 0,  
      "buckets": [  
        { "key": "red", "doc_count": 4 },  
        { "key": "blue", "doc_count": 2 },  
        { "key": "green", "doc_count": 2 }  
      ]  
    }  
  }  
}}
```

Imbrication

```
GET /cars/transactions/_search {  
  "aggs": {  
    "colors": {  
      "terms": { "field": "color" },  
      "aggs": {  
        "avg_price": {  
          "avg": { "field": "price" }  
        }, "make": {  
          "terms": { "field": "make" }  
        }  
      }  
    } } } }  
}
```

Réponse

```
{
  ...
  "aggregations": {
    "colors": {
      "buckets": [
        { "key": "red",
          "doc_count": 4,
          "avg_price": {
            "value": 32500
          },
          "make": { "buckets": [
            { "key": "honda", "doc_count": 3 },
            { "key": "bmw", "doc_count": 1 }
          ]
        }
      ],
    },
    ...
  }
}
```


Agrégation et recherche

- En général, une agrégation est combinée avec une recherche. Les buckets sont alors déduits des seuls documents qui matchent.

```
GET /cars/transactions/_search
```

```
{  
  "query" : {  
    "match" : { "make" : "ford" }  
  }, "aggs" : {  
    "colors" : {  
      "terms" : { "field" : "color" }  
    }  
  }  
}
```

Spécification du tri

GET /cars/transactions/_search

```
{  
  "aggs" : {  
    "colors" : {  
      "terms" : { "field" : "color",  
                  "order": { "avg_price" : "asc" }  
      }, "aggs": {  
        "avg_price": {  
          "avg": {"field": "price"}  
        }  
      }  
    }  
  }  
}
```

Types de bucket

- Différents types de regroupement sont proposés par ELS
 - Par terme, par filtre : Nécessite une tokenization du champ
 - Par intervalle de valeurs
 - Par intervalle de dates, par histogramme
 - Par intervalle d'IP
 - Par absence d'un champ
 - Par le document parent
 - Significant terms
 - Par géo-localisation
 - ...

Intervalle de valeur

```
GET /cars/transactions/_search
{
  "aggs": {
    "price": {
      "histogram": {
        "field": "price",
        "interval": 20000
      },
      "aggs": {
        "revenue": {
          "sum": { "field" : "price" }
        }
      }
    }
  }
}
```

Histogramme de date

```
GET /cars/transactions/_search
{
  "aggs": {
    "sales": {
      "date_histogram": {
        "field": "sold",
        "interval": "month",
        "format": "yyyy-MM-dd"
      }
    }
  }
}
```

significant_terms

- L'agrégation ***significant_terms*** est plus subtile mais peut donner des résultats intéressants (proche du machine-learning).
- Cela consiste à analyser les données retournées et trouver les termes qui apparaissent à une fréquence *anormalement* supérieure
Anormalement signifie : par rapport à la fréquence pour l'ensemble des documents
=> Ces anomalies statistiques révèlent en général des choses intéressantes

Fonctionnement

- *significant_terms* part d'un résultats d'une recherche et effectue une autre recherche agrégé
- Il part ensuite de l'ensemble des documents et effectue la même recherche agrégé
- Il compare ensuite les résultats de la première recherche qui sont anormalement pertinent par rapport à la recherche globale
- Avec ce type de fonctionnement, on peut :
 - Les personnes qui ont aimé ... ont également aimé ...
 - Les clients qui ont eu des transactions CB douteuses sont tous allés chez tel commerçant
 - Tous les jeudi soirs, la page untelle est beaucoup plus consultée
 - ...

Example

```
{  
  "query" : {  
    "terms" : {"force" : [ "British Transport Police" ]}  
  },  
  "aggs" : {  
    "significantCrimeTypes" : {  
      "significant_terms" : { "field" : "crime_type" }  
    }  
  }  
}
```


Réponse

```
"aggregations" : {  
  "significantCrimeTypes" : {  
    "doc_count": 47347, // Total résultat de requête  
    "buckets" : [  
      {  
        "key": "Bicycle theft",  
        "doc_count": 3640, // Nbr docs le résultat de  
        "score": 0.371235374214817,  
        "bg_count": 66799 // Nbr pour le total de l'index  
      },  
      ...  
    ]  
  }  
}
```

=> Le taux de vols de vélos est anormalement élevé pour « British Transport Police »

Types de métriques

- ELS propose de nombreux métriques :
 - *avg, min, max, sum*
 - *value_count, cardinality* : Comptage de valeur distinctes
 - *top_hit* : Les meilleurs documents
 - *extended_stats* : Fournit plein de métriques (count, sum, variance, ...)
 - *percentiles* : percentiles

Recherche avec DSL

Syntaxe DSL

Principaux opérateurs

Agrégations

Géolocalisation

Introduction

- ELS permet de combiner la géo-localisation avec les recherches full-text, structurées et les agrégations
- ELS a 2 modèles pour représenter des données de géolocalisation
 - Le type ***geo_point*** qui représente un couple latitude-longitude. Cela permet principalement le calcul de distance
 - Le type ***geo_shape*** qui définit une zone via le format GeoJSON. Cela permet de savoir si 2 zones ont une intersection

Geo-point

- Les Geo-points ne peuvent pas être détectés automatiquement par le *dynamic mapping*. Ils doivent être explicitement spécifiés dans le mapping:

```
PUT /attractions
```

```
{ "mappings": { "restaurant": {  
  "properties": {  
    "name": { "type": "string" },  
    "location": { "type": "geo_point" }  
  }  
} } }
```

```
----
```

```
PUT /attractions/restaurant/1
```

```
{ "name": "Chipotle Mexican Grill", "location": "40.715, -74.011" }
```

```
PUT /attractions/restaurant/2
```

```
{ "name": "Pala Pizza", "location": { "lat": 40.722, "lon": -73.989 } }
```

```
PUT /attractions/restaurant/3
```

```
{ "name": "Mini Munchies Pizza", "location": [ -73.983, 40.719 ] }
```

Filtres

- 4 filtres peuvent être utilisés pour inclure ou exclure des documents vis à vis de leur *geo-point* :
 - ***geo_bounding_box*** : Les geo-points inclus dans le rectangle fourni
 - ***geo_distance*** : Distance d'un point central inférieur à une limite. Le tri et le score peuvent être relatif à la distance
 - ***geo_distance_range*** : Distance dans un intervalle
 - ***geo_polygon*** : Les geo-points incluent dans un polygone

Example

GET /attractions/restaurant/_search

```
{
  "query": {
    "bool": {
      "filter": {
        "geo_bounding_box": {
          "location": { "top_left": { "lat": 40.8, "lon": -74.0 },
                        "bottom_right": { "lat": 40.7, "lon": -73.0 }
          }
        }
      }
    }
  }
}
```

Agrégation

- 3 types d'agrégation sur les geo-points sont possibles
 - ***geo_distance*** (*bucket*): Groupe les documents dans des ronds concentriques autour d'un point central
 - ***geohash_grid*** (*bucket*): Groupe les documents par cellules (geohash_cell, les carrés de google maps) pour affichage sur une map
 - ***geo_bounds*** (*metrics*): retourne les coordonnées d'une zone rectangle qui engloberait tous les geo-points. Utile pour choisir le bon niveau de zoom

Example

```
GET /attractions/restaurant/_search
{
  "query": { "bool": { "must": {
    "match": { "name": "pizza" }
  },
  "filter": { "geo_bounding_box": {
    "location": { "top_left": { "lat": 40.8, "lon": -74.1 },
                  "bottom_right": { "lat": 40.4, "lon": -73.7 }
    }
  } } } },
  "aggs": {
    "per_ring": {
      "geo_distance": {
        "field": "location",
        "unit": "km",
        "origin": {
          "lat": 40.712,
          "lon": -73.988
        },
        "ranges": [
          { "from": 0, "to": 1 },
          { "from": 1, "to": 2 }
        ]
      }
    }
  }
}
```

Geo-shape

- Comme les champs de type *geo_point* , les **geo-shape** doivent être mappés explicitement :

```
PUT /attractions
```

```
{  
  "mappings": { "landmark": {  
    "properties": {  
      "name": { "type": "string" },  
      "location": { "type": "geo_shape" }  
    } } } }  
}
```

```
---
```

```
PUT /attractions/landmark/dam_square
```

```
{  
  "name" : "Dam Square, Amsterdam",  
  "location" : {  
    "type" : "polygon",  
    "coordinates" : [[ [ 4.89218, 52.37356 ], [ 4.89205, 52.37276 ], [ 4.89301,  
52.37274 ],[ 4.89392, 52.37250 ], [ 4.89218, 52.37356 ] ]  
  } }  
}
```

Example

```
GET /attractions/landmark/_search
{
  "query": {
    "geo_shape": {
      "location": {
        "shape": {
          "type": "circle",
          "radius": "1km"
          "coordinates": [ 4.89994, 52.37815]
        }
      }
    }
  }
}
```

Concepts Kibana

Présentation

Data Views et Discover

KQL

ES|QL

Introduction

- Kibana est une plateforme d'analyse et de visualisation fonctionnant avec ElasticStack
- Il est capable de rechercher et agréger les données stockées dans les index d'ElasticSearch
- Il propose une interface web permettant de créer des tableaux de bord dynamiques affichant le résultat des requêtes en temps réel
- Il s'adresse également aux administrateurs de la pile ElasticStack qui peuvent alors administrer les index et surveiller l'infrastructure ElasticStack

Types de données

- Kibana travaille avec différents types de données :
 - Structurées
 - Non-structurées (full text)
 - Données numériques
 - Séries temporelles
 - Données de géo-localisation
 - Logs
 - Métriques
 - Evènements de sécurité
 - ...

Application prêt à l'emploi

- Kibana fournit des solutions pour des cas d'utilisation typique de la pile ELK :
 - Enterprise Search permet d'ajouter des fonctionnalités d'un moteur de recherche une application d'entreprise ou un site Web.
 - Elastic Observability permet de surveiller et d'appliquer des analyses en temps réel aux événements qui se produisent dans une infrastructure.
Analyser les événements de log, surveiller les mesures de performances, vérifier la disponibilité des services.
 - Elastic Security fournit un aperçu des événements sécurité et des alertes de votre environnement et aide à se défendre contre les menaces.

Analyse

- Le menu Analytics permet de rechercher rapidement dans de grandes quantités de données, explorer des champs et des valeurs, créer rapidement des graphiques, des tableaux, des mesures et de les rassembler dans des tableaux de bord.
- 1) Le menu explore permet d'effectuer et d'explorer vos données
 - 2) Visualize permet de créer des visualisations et de les disposer dans un tableau de bord, d'effectuer des présentations ou d'identifier les relations entre vos données
 - 3) Dans la version payante, des modèles de Machine Learning permettent de détecter des anomalies ou de prévoir les évolutions futures de vos métriques. Des alertes déclenchant des actions lorsque une situation arrive peuvent être créées
 - 4) Des fonctionnalités de partage permettent de diffuser les visualisations

Espaces

- Kibana permet d'organiser son contenu via des espaces
- Les espaces permettent de regrouper les visualisations, les tableaux de bords et les index ES dans des dossiers indépendants
- Un système de tags permet également d'organiser le contenu



Select your space

You can change your space at anytime



Default

This is your Delightful Default Space!



Engineering

This is the Elegant Engineering Space!



Marketing

This is the Magical Marketing Space!



Security operations

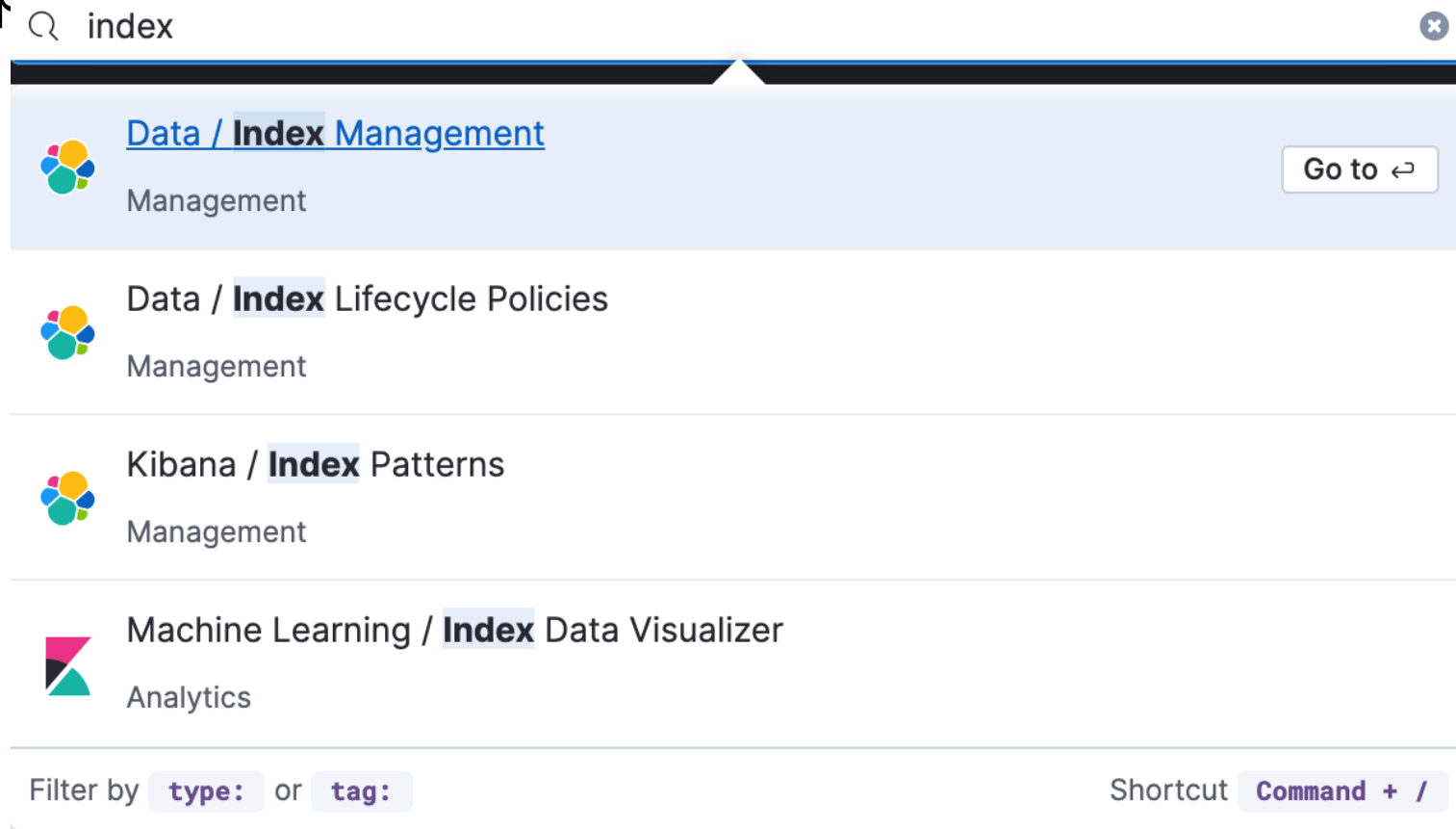
This is the Super Security operations Space!

Sécurité

- Kibana permet via son modèle de sécurité de définir les accès des utilisateurs
- Des rôles et des privilèges peuvent être définis et associés aux utilisateurs
- L'authentification peut se faire sur l'annuaire embarqué ou sur d'autres fournisseurs d'identité externes à la stack (OpenID par exemple)
- Des données d'audit sont disponibles pour voir qui a fait quoi

Recherche

- Un moteur de recherche est disponible pour retrouver facilement les objets Kibana via leur type, leur nom ou leur tag



Gestion de la stack

- Kibana permet également de gérer la stack Elastic :
 - Gestion des pipelines
 - Gestion des index, politique d'archivage et de suppression, sauvegarde, réplication sur des clusters distants
 - Gestion des alertes
 - Des jobs de Machine Learning

Administrer ElasticStack

- Gérer les données Elasticsearch.
Stack Management > Data
- Mettre en place des règles.
Stack Management > Rules and Connectors
- Organiser l'espace de travail et les utilisateurs.
Stack Management > Spaces
- Définir les rôles et permissions.
Stack Management > Users
- Personnaliser Kibana.
Stack Management > Advanced Settings

Atelier 5.1 : Première analyse avec Kibana

Concepts Kibana

Présentation
Data Views et Discover
KQL
ES|QL

Data Views

- Kibana nécessite une **Data View** pour indiquer à quelles données Elasticsearch on souhaite accéder et si les données sont basées sur le temps.
- Une data view référence un ou plusieurs data stream ou index contenant généralement un champ @timestamp qui horodate le document
- Elles sont généralement créées par un administrateur dans Stack Management.
- Kibana montre la liste des champs correspondant à une data view.
- Il est possible de personnaliser le nom d'affichage et le format de chaque champ.

Création

- La création de DataView s'effectue dans la partie StackManagement

Create data view

Name

Index pattern

Timestamp field

Select a timestamp field for use with the global time filter.

[Show advanced settings](#)

× CloseUse without savingSave data view to Kibana

✓ Your index pattern matches 3 sources.
















All sources	Matching sources
kibana_sample_data_ecommerce	Index
kibana_sample_data_flights	Index
kibana_sample_data_logs	Data stream

Rows per page: 10 ▾

Edition de DataView

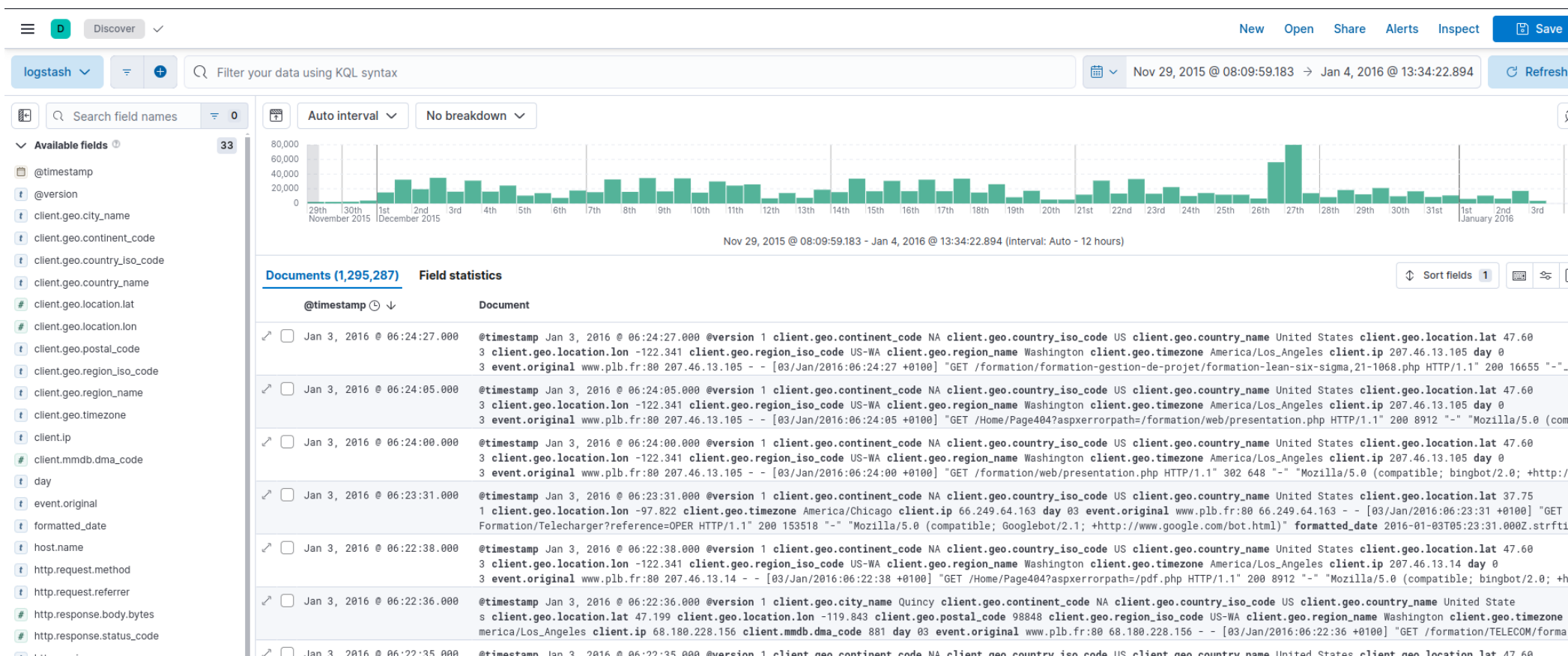
- La vue détaillé d'une data view permet :
 - De voir les caractéristiques des champs, en particulier les colonnes Searchable et Agregatable
 - D'éditer les champs en particulier leur format d'affichage par défaut
 - De rechercher un champ particulier
 - D'ajouter des champs scriptés (champs calculés à partir d'autre champs)
 - D'exclure des champs présents dans l'index
 - De visualiser les objets sauvegardés liés à la Data View : Requêtes, Visualisations, Dashboard, Alertes, ...

Vue détaillée

logstash						 Delete	 Edit
Index pattern: logstash* Time field: @timestamp ★ Default							
Fields (64) Scripted fields (0) Field filters (0) Relationships (0)							
<input type="text" value="Search"/>						Field type 8 ▾	Schema type 2 ▾
						 Refresh	 Add field
Name ↑	Type	Format	Searchable	Aggregatable	Excluded		
@timestamp 	date		•	•			
@version	text		•				
@version.keyword	keyword		•	•			
_id	_id		•				
_index	_index		•	•			
_score							
_source	_source						
client.geo.city_name	text		•				
client.geo.city_name.keyword	keyword		•	•			
client.geo.continent_code	text		•				
Rows per page: 10 ▾						< 1 2 3 4 5 6 7 >	





Discover

- Une fois le Data View mis en place on peut explorer ses données via le menu Discover



Recherche

- Kibana propose plusieurs méthodes pour créer des requêtes de recherche.
 - un filtre temporel
 - Des filtres sur les données structurées
 - Des recherches via les syntaxes KQL, ES/QL et Lucene
- Les recherches peuvent être sauvegardées

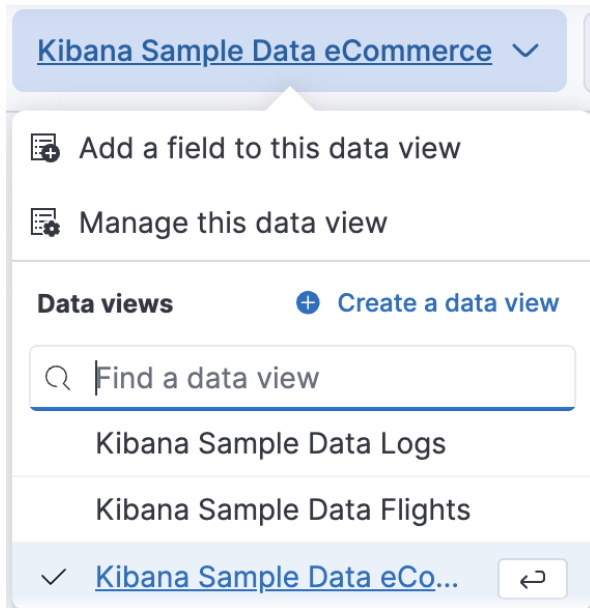
Data view	Save query & add filter	Semi-structured search	Time filter
kibana_sample_data_ecommerce ▾	 	 Filter your data using KQL syntax	 ▾ Last 7 days
geoip.country_iso_code: US × NOT day_of_week: Wednesday ×			

Extra filters with AND

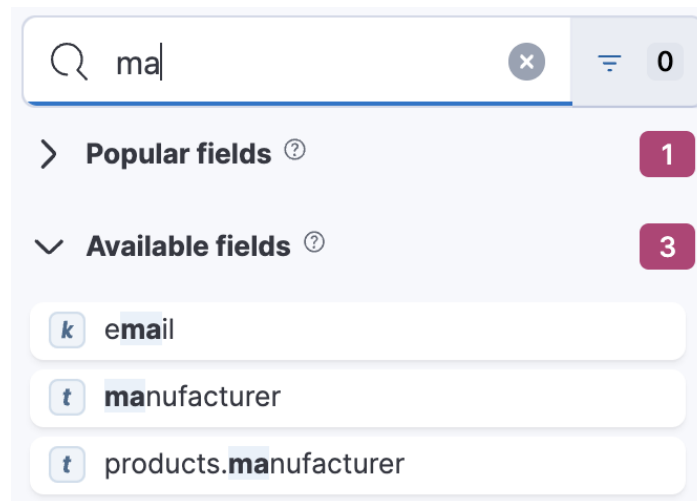
Filtre temporel

- Le filtre temporel permet de limiter la plage de temps que l'on visualise
- Il est présent dans la plupart des pages Kibana
- Plusieurs moyens sont disponibles pour spécifier la plage
 - Quick Select. Les dernières ou les prochaines 15 minutes par exemple
 - Les plages fréquentes
 - Les plages de dates récemment utilisées.
- Il est possible d'activer le rafraîchissement automatique.

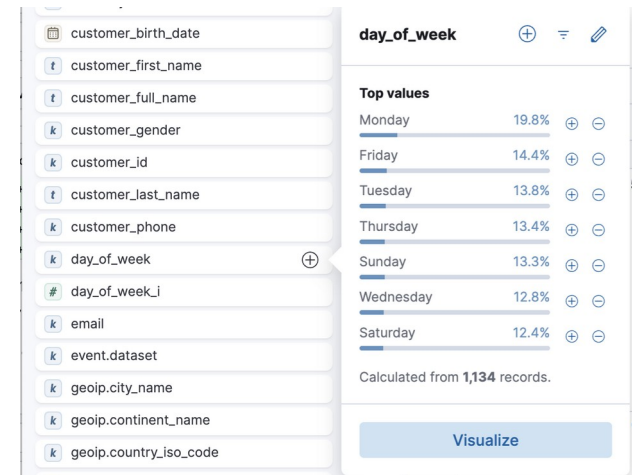
Sélecteur de DataView



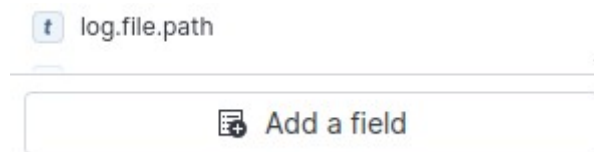
Exploration des champs



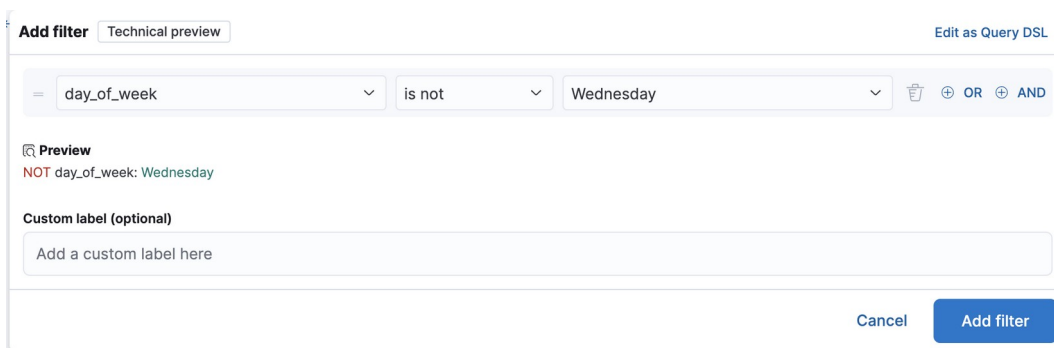
Agrégation



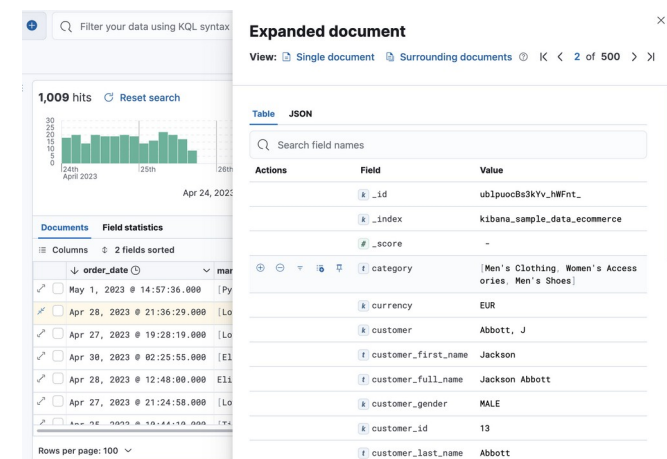
Ajout de champ calculé



Ajout de filtres



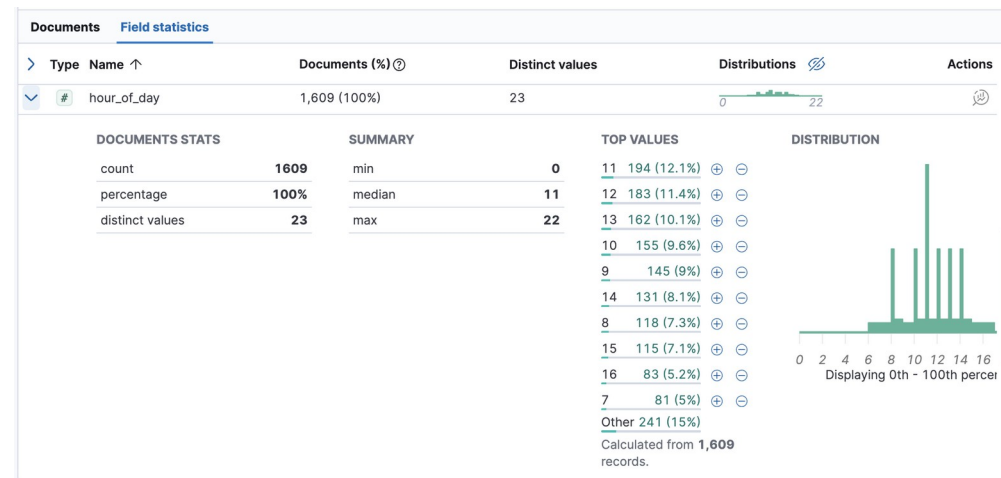
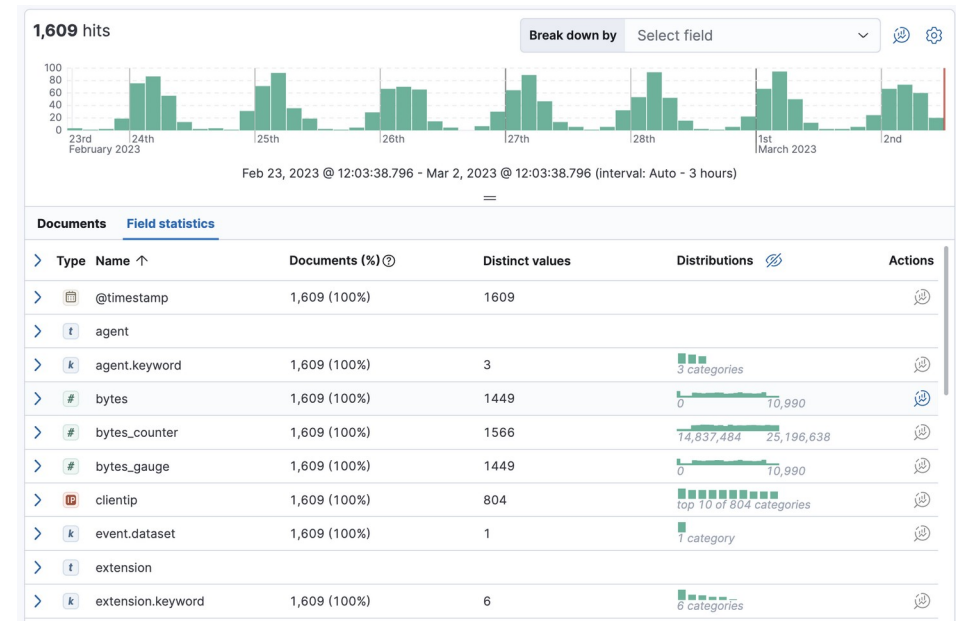
Vue détaillée du document



Modifier la table de résultats :

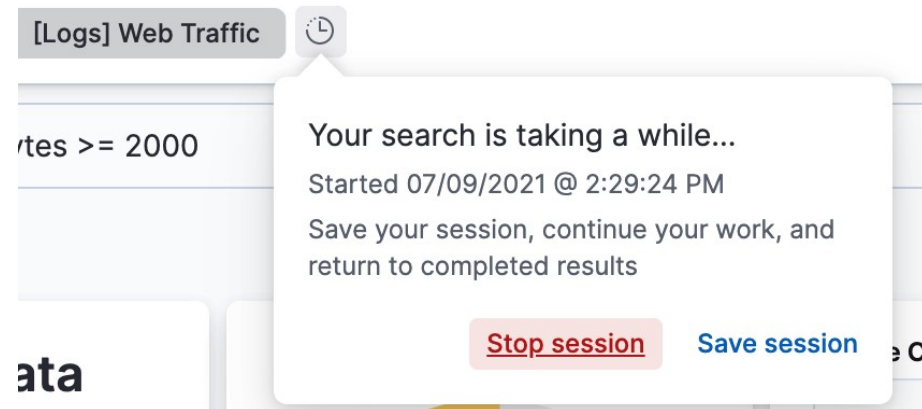
- Ordonner et dimensionner les colonnes
- Ajuster la hauteur de ligne, le nombre de lignes d'une page
- Ordonner
- Editer le format de visualisation d'un champ
- N'afficher que les documents sélectionnés

Statistiques sur les Champs



Requêtes en tâche de fond

- Quelquefois, il est nécessaire de rechercher sur de très gros volumes de données
- Si la requête est très longue, on peut l'exécuter en tâche de fond



- La requête est alors stocké et visible dans la partie *StackManagement*

Search Sessions

Manage your saved search sessions.

Search... Status App Refresh

App	Name	# Searches	Status	Created ↓	Expiration	
[Logs] Web Traffic -	07/09/2021 @ 2:24 PM	11	Complete	9 Jul, 2021, 14:24:06	16 Jul, 2021, 14:24:06	...
Flights out of Warsaw or Venice -	07/09/2021 @ 2:22 PM	1	Complete	9 Jul, 2021, 14:22:36	16 Jul, 2021, 14:22:36	Inspect Edit name Extend Delete

Rows per page: 10

Concepts Kibana

Présentation
Data Views et Discover
KQL
ES|QL

Recherche semi-structurée KQL

- Kibernetate Query Language sert à filter les données. (Il n'effectue pas de tri ni d'aggrégations)
- Il permet de combiner la recherche en full-text avec la recherche par champ via (KQL).
- Lors d'une recherche full-text les résultats sont triables par pertinence.

Exemples

- Filtrer les documents pour lesquels un champ existe :
http.request.method: *
- Filtrer les documents correspondant à une valeur
 - Documents dont le champ est égale une valeur
http.request.method: GET
 - Documents dont le champ est égale une valeur ou une autre valeur
http.request.method: GET POST
 - Documents dont un des champs est égal à une valeur
GET
- Valeur exacte si le champ n'est pas un texte, sinon recherche full-text.
 - Le champ texte contient les mots null et pointer
http.request.body.content: null pointer
 - Le champ texte contient la phrase « null pointer »
http.request.body.content: "null pointer"
- Recherche d'une phrase :
http.response.body.content.text:"quick brown fox"

Intervalles et wildcard

- La recherche par intervalle est utilisée principalement pour les chiffres ou les dates
 - ***http.response.bytes < 10000***
 - ***http.response.bytes > 10000 and http.response.bytes <= 20000***
 - ***@timestamp < now-2w***
- La recherche par wildcard permet de filtrer des documents par le préfixe d'une valeur :
 - ***http.response.status_code: 4****
- Les wildcard peuvent être utilisés pour indiquer plusieurs champs :
 - ***datastream.*: logs***

Opérateurs booléen

- La négation s'effectue avec NOT
 - ***NOT http.request.method: GET***
- On peut combiner plusieurs requêtes avec AND , OR et les parenthèses
 - ***http.request.method: GET OR http.response.status_code: 400***
 - ***http.request.method: GET AND http.response.status_code: 400***
 - ***(http.request.method: GET AND http.response.status_code: 200)
OR
(http.request.method: POST AND http.response.status_code: 400)***

Champs imbriqués

Exemple de document :

```
{
  "user" : [
    {
      "first" : "John",
      "last" : "Smith"
    },
    {
      "first" : "Alice",
      "last" : "White"
    }
  ]
}
```

- Pour recherche Alice White :
user:{ first: "Alice" and last: "White" }

Champs imbriqués (2)

- Si il y a plusieurs niveaux d'imbrication, il faut donner le chemin complet du champ.

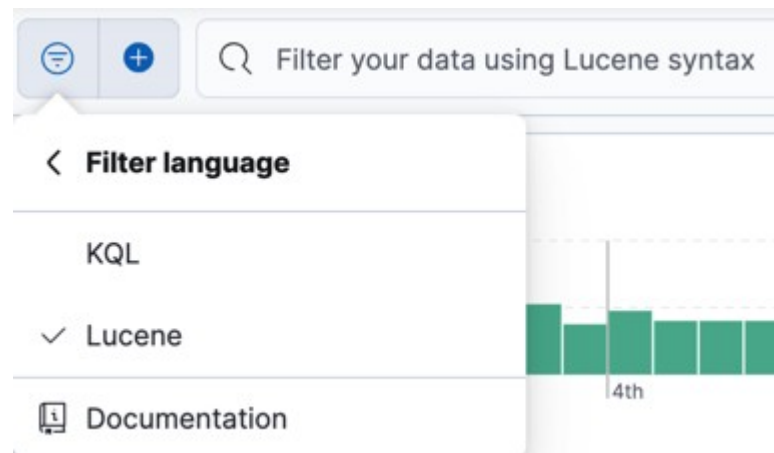
```
{
  "user": [
    {
      "names": [
        {
          "first": "John",
          "last": "Smith"
        },
        {
          "first": "Alice",
          "last": "White"
        }
      ]
    }
  ]
}
```

La requête pour Alice devient :

user.names:{ first: "Alice" and last: "White" }

Syntaxe Lucene

- Lucene permet des fonctionnalités avancées, telles que les expressions régulières ou les fuzzy-search (mot approchant).
- Cependant, la syntaxe Lucene ne permet pas de rechercher des objets imbriqués ou des champs scriptés.



Concepts Kibana

Présentation
Data Views et Discover
KQL
ES|QL

ES|QL

- **Elasticsearch Query Language, ES|QL** peut accélérer l'exploration des données
- ES|QL permet d'enchaîner plusieurs commandes
 - Avec une seule requête, on peut rechercher, agréger, calculer et effectuer des transformations de données.
- ES|QL est disponible dans *Discover* ou dans *DevTools* avec l'API ES|QL. (`_eq/`)
- De l'aide est apportée via :
 - Un assistant pour la complétion
 - Un éditeur multiligne pour les requêtes longues
 - Un historique des requêtes

Elements de syntaxe

- Chaque requête démarre avec une commande indiquant la source des données :
from kibana_sample_data_logs
- Une commande source peut être complétée par des commandes de traitement :
from kibana_sample_data_logs | limit 10
- Le résultat de la requête est un tableau de résultat.
 - Il est possible de sélectionner les colonnes que l'on veut garder :
*FROM kibana_sample_data_logs
| KEEP @timestamp, bytes, geo.dest*
 - De trier :
*FROM kibana_sample_data_logs
| KEEP @timestamp, bytes, geo.dest
| SORT bytes DESC*
 - De filtrer :
*FROM kibana_sample_data_logs
| WHERE bytes > 10000*

Analyse et visualisation

- La commande **STATS** permet des calculs statistiques comme SUM, AVG, MAX, MIN, COUNT, etc. Elle s'utilise avec le mot-clé BY pour regrouper les données.
- La visualisation de Discover¹ s'adapte à la query ES|QL
- La visualisation est éditable : Changement des couleurs, des axes, etc.
- Et peut être sauvegardée pour utilisation dans un Tableau de bord

1. Par défaut un histogramme comptant le nombre de document groupé via le temps



```
1 FROM kibana_sample_data_logs
2 | STATS total_bytes = SUM(bytes) BY geo.dest
3 | SORT total_bytes DESC
4 | LIMIT 3
```

4 lines @timestamp found

[Submit feedback](#) [Show recent queries](#)



Search field

0



Selected fields

2

total_bytes

k geo.dest

Available fields

2

k geo.dest

total_bytes

total_...



geo.dest

Apr 16, 2024 @ 00:00:00.000 - May 16, 2024 @ 11:41:42.252

3 results

Columns 2

Sort fields



total_bytes k geo.dest



2996101 CN




2428634 IN



1125692 US

Edition et sauvegarde

- L'histogramme peut être édité et sauvegardé via la barre de bouton The image shows a small rectangular button with a light gray border. It is divided into two equal square sections. The left section contains a dark gray pencil icon, representing an edit function. The right section contains a dark gray floppy disk icon, representing a save function.
- L'édition permet de modifier le type de visualisation, de couleur,
- La sauvegarde permet d'ajouter la visualisation à un dashboard
- A l'intérieur du dashboard, on peut ré-éditer la requête

Enrichissement

- La commande **ENRICH** permet d'enrichir le jeu de données avec des champs provenant d'un autre jeu de données.
- Il est nécessaire d'avoir préalablement mis au point le jeu de données et la jointure dans une **ENRICH POLICY**

Enrich Policy

Create enrich policy

Specify how to retrieve and enrich your incoming data.

✓ Configuration

Policy name

countries

Policy type

Match

Source indices

[Upload a file](#)

countries ×

Query (optional)

```
{  
  
}
```

Defaults to: [match_all](#) query.

Next >

2

Field selection

Match field

geo.dest

Enrich fields

country_name ×

< Back

Next >

Utilisation de ENRICH

```
FROM kibana_sample_data_logs  
| STATS total_bytes = SUM(bytes) BY geo.dest  
| SORT total_bytes DESC  
| LIMIT 3  
| ENRICH countries
```

Sans ENRICH Policies on peut également écrire la requête comme suit :

```
FROM kibana_sample_data_logs  
| ENRICH WITH country_index  
  ON kibana_sample_data_logs.geo.dest = country_index.country_name  
FIELDS country_code, country_area
```

Alertes

- ES|QL pour servir à créer des alertes.
Discover → Alerts → Create search threshold rule
- La requête ES|QL est associée à une fenêtre temporelle.
- Si la requête retourne des documents l'alerte est déclenchée.

Create rule

Name

High traffic

Tags (optional)

Elasticsearch query

Alert when matches are found during the latest query run. [Learn more](#)

Define your query using ES|QL

```
1 FROM kibana_sample_data_logs
2 | STATS total_bytes = SUM(bytes) BY geo.dest
3 | SORT total_bytes DESC
4 | WHERE total_bytes>10000000
```

4 lines @timestamp detected

Select a time field

@timestamp

Set the time window

5

minutes

▶ Test query

Query matched 2 documents in the last 5m.

total_bytes	geo.dest
14888980	CN
12974807	IN

Alerts generated query matched

Cancel

✓ Save

Atelier 5.4 : ES|QL

Visualisation Kibana

Les tableaux de bord

Lens

TSVB

Vega

Agregations

Timelion

Tableaux de bords et panels

- Les tableaux de bord agrègent en une page un ensemble de panels qui clarifient les données et qui permettent de se concentrer uniquement sur les données qui sont importantes.
- Les panels affichent les données sous forme de graphiques, de tableaux, de cartes, etc..
- Un tableau de bord peut être partagé sous forme de lien.
- Il s'actualise automatiquement en fonction de son time filter

Types de panels

- Kibana propose différents types de panels :
 - **Éditeurs** : Visualisations des données.
 - **Cartes** : Affichages de données géographiques.
 - **Swinlane des anomalies** : Adapté aux jobs de ML détectant les anomalies.
 - **Graphique d'anomalies** : Graphique d'anomalies à partir de l'explorateur d'anomalies. (ML)
 - **Flux de journaux** : Tableau de journaux en direct.
 - **Outils** : Filtres interactifs avec les panneaux de contrôle.
 - **Texte** : Ajout de simple texte.
 - **Image** : Ajout d'image personnalisée.

Visualisations

- Les visualisations sont basées sur des recherches ES
- En utilisant des agrégations, on peut créer des graphiques qui montrent des tendances, des pics, ...
- Différents éditeurs pour les visualisations :
 - Lens : Drag and Drop
 - Maps : Géolocalisation
 - TSVB : Adaptées aux séries temporelles
 - Visualisations personnalisées : Vega
 - Visualisations basées sur des agrégations
 - Timelion : Séries temporelles avec un langage d'expression

Mise en place de Dashborad

- ***Dashboard → Create Dashboard***

Lors de la création, On est tout de suite en mode édition

- Les panneaux sont créés à l'aide des éditeurs, accessible
 - via la barre d'outils du tableau de bord
 - Ou via la bibliothèque de visualisation (Visualize Library)
- Ils peuvent être ajoutés à un dashboard via :
 - La bibliothèque de visualisation
 - Directement à partir des résultats de recherche de Discover
- A la sauvegarde d'un panels, on a le choix de le sauvegarder :
 - Dans la bibliothèque (pour réutilisation)
 - Directement dans le tableau de bord
- Lors que l'on modifie un panel provenant de la bibliothèque, on a le choix entre :
 - Modifier le panel de la librairie
 - Dissocier le panel de la librairie

Disposition des panels

- Les panels peuvent ensuite être déplacés, redimensionnés
- Chaque panel contient un titre, une description et un intervalle de temps
- Des panels de texte ou d'image permettent de contextualiser les informations
- Sur chaque panneau, il est possible via le menu inspect de voir la requêtes correspondantes et les données présentées au format CSV

Finalisation du dashboard

- Lorsque les panels du tableau de bord sont mis au point, on peut renseigner les **Settings** du tableau de bord :
 - Titre et description
 - Les tags
 - D'autres options :
 - Sauvegarder le filtre temporel avec le dashboard
 - Jouer sur les marges entre les panneaux
 - Afficher ou pas les titres de panneaux
 - Synchroniser les panneaux entre eux :
 - même palette de couleur
 - même forme de curseur
 - Même tooltip
- Il ne reste plus alors que partager le tableau de bord
- On peut également l'exporter au format JSON pour le réimporter sur une autre instance Kibana

Options de partage

- Les tableaux de bord, les requêtes sauvegardées, les librairies et les Canvas peuvent être partagés.
- Différents formats de partage sont possibles :
 - Rapports PDF ou PNG : Version payante
 - Rapports CSV pour les recherches enregistrées et de visualisations Lens.
 - Sous forme de lien : recherches enregistrées, des tableaux de bord et visualisations.
 - Sous forme d'iframe : le tableau de bord peut être intégré dans une autre page Web.

Atelier 6.1 : Création du 1^{er} tableau de bord

Visualisation Kibana

Les tableaux de bord

Lens

TSVB

Vega

Agregations

Timelion

Introduction

- **Lens** permet de travailler via Glisser/Déposer des champs.
- Il propose alors les meilleures pratiques de visualisation pour appliquer les champs et créer une visualisation qui affiche au mieux les données.
- Il émet également des suggestions permettant de changer de type de visualisation
- Avec Lens, il est possible de :
 - Créer des graphiques en aires, en courbes et à barres avec des calques pour afficher plusieurs indices et types de graphiques.
 - Modifier la fonction d'agrégation pour modifier les données dans la visualisation.
 - Créer des tables personnalisées.
 - Effectuer des calculs sur les agrégations à l'aide de formules.
 - Utiliser des décalages temporels pour comparer les données dans deux intervalles de temps, par exemple d'un mois à l'autre.
 - Ajouter des annotations et des lignes de référence.

Etapes de création

- Si on sait quel type de visualisation on veut utiliser :
 - Avant de glisser les champs, sélectionner le type de Visualization.
 - Regarder quand même les suggestions proposées
 - Glisser les champs voulus, Lens choisit automatiquement le type d'agrégation en fonction du type du champ
 - Adapter ensuite si nécessaire l'agrégation voulue
 - Il est possible de visualiser différentes data view via le bouton Add layer

Les tables

- Les tableaux sont hautement personnalisables et offrent des options d'alignement de texte, de formatage de valeur et de coloration
 - Pour trier ou masquer les colonnes, cliquez sur la flèche à côté de l'en-tête de la colonne, puis sélectionnez une option.
 - Pour modifier les options d'affichage, cliquez sur un champ de mesures et configurer :
 - Le nom, la possibilité de regrouper les mesures ayant la même valeur, le formatage, l'alignement,

The image shows a configuration panel for a table measure, divided into two sections: 'Appearance' and 'Summary'.

Appearance

- Name:** Maximum of http.response.body.byte
- Value format:** Default (dropdown menu)
- Text alignment:** Left, Center, Right (radio buttons, with 'Right' selected)
- Color by value:** None, Cell, Text (radio buttons, with 'Cell' selected)
- Color:** A horizontal color gradient bar with a small icon to its right.
- Hide column:** A toggle switch that is currently turned off.

Summary

- Summary Row:** Sum (dropdown menu)

Split Metric By

- Nombre de ventes par jour
- Split Metric By : Continent

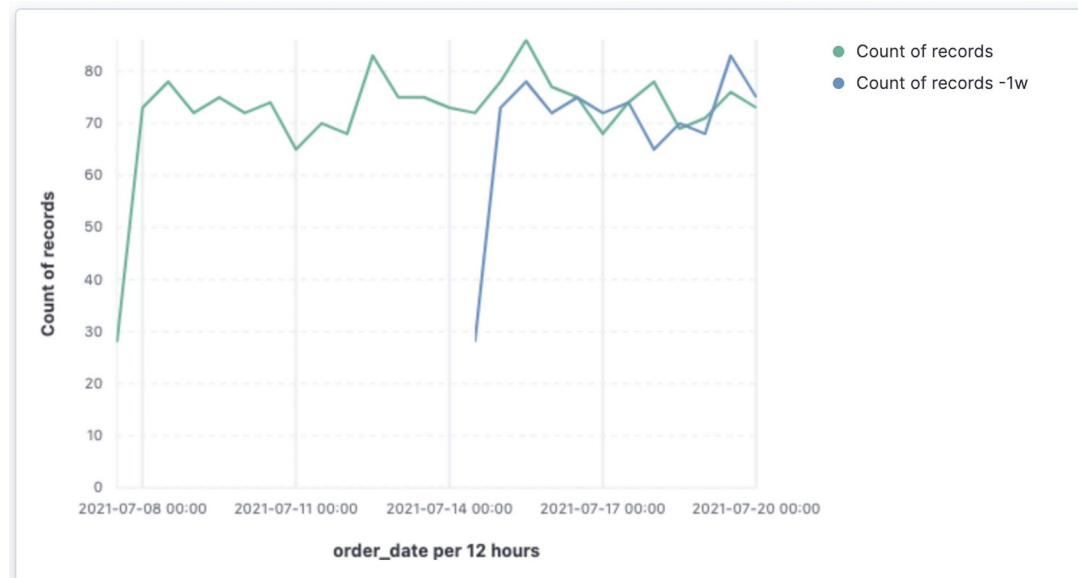
Sale per day	Asia › Customer: ›	Europe › Custom ›	North America ›	Other › Custome ›
2021-07-07	4	8	3	4
2021-07-08	11	12	9	11
2021-07-09	13	12	8	10
2021-07-10	13	12	10	10
2021-07-11	12	12	10	9
2021-07-12	13	12	9	10
2021-07-13	11	11	9	11
2021-07-14	12	12	10	10
2021-07-15	12	9	9	10
2021-07-16	12	12	9	10

Comparer les différences entre 2 périodes

- Possibilité de comparer les données avec les données décalées dans le temps.

Ex : Comparer des résultats avec l'année dernière

- Cliquez sur le champ que vous souhaitez décaler.
- Cliquez sur Avancé.
- Dans le champ Time shift, saisissez l'incrément de décalage horaire.



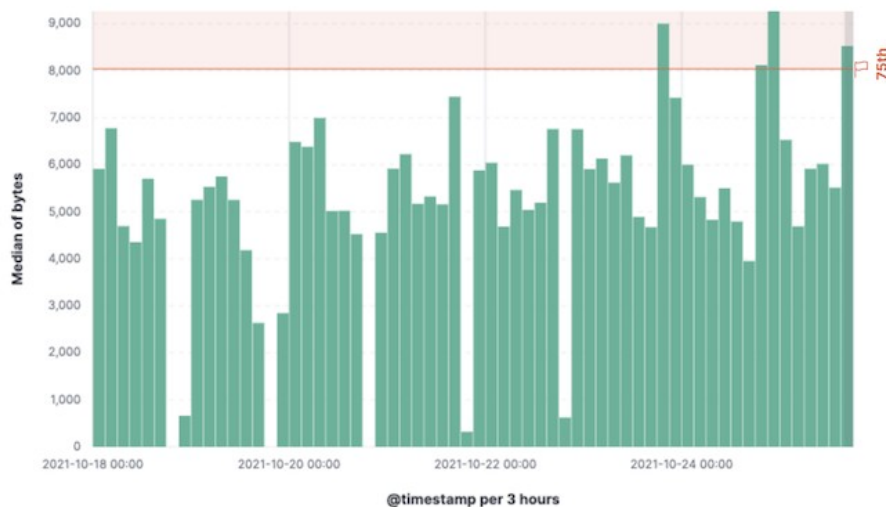
Formules

- Les formules vous permettent d'effectuer des opérations mathématiques sur des données agrégées.
 - Les formules les plus courantes divisent deux valeurs pour produire un pourcentage.
Choisir Percent en Value Format dans ce cas
- Une assistance dans l'éditeur permet d'élaborer sa formule
- Exemples :
 - `count(kql='response.status_code > 400') / count()`
 - `percentile(system.network.in.bytes, percentile=99) /`
`percentile(system.network.in.bytes, percentile=99, shift='1w')`
 - `sum(products.base_price) / overall_sum(sum(products.base_price))`

Ligne de référence

- Avec les lignes de référence, vous pouvez identifier des valeurs spécifiques dans vos visualisations à l'aide d'icônes, de couleurs et d'autres options d'affichage.
- Vous pouvez ajouter des lignes de référence à tout type de visualisation qui affiche des axes.

Add layer > Reference lines



Les filtres

- Avec la fonction Filter, vous pouvez appliquer plusieurs filtres KQL et appliquer un filtre KQL à une seule couche afin de pouvoir visualiser les données filtrées et non filtrées en même temps.

Champ > Filters

- Avec l'option Filter by (Advanced), vous pouvez attribuer une couleur à chaque groupe de filtres dans les visualisations à barres, à lignes et à zones, et créer des tableaux complexes.

Champ > Add advanced > Filter by.

- Appliquez des filtres directement à partir des valeurs de la légende.

Configuration des composants de visualisation

- Chaque type de visualisation a un ensemble de composants qui sont configurables à partir de la barre d'outils de l'éditeur.



- Options visuelles : spécifie comment afficher les options de graphique à aires, à courbes et à barres. Par exemple, vous pouvez spécifier comment afficher les étiquettes dans les graphiques à barres.
- Labels : spécifie comment afficher les étiquettes .
- Légende : afficher la légende à l'intérieur de la visualisation et tronquer les valeurs de légende.
- Axes : ajoutez des étiquettes d'axe et modifiez l'orientation et les limites.

Liens de haut de fenêtre

Explore in Discover

Inspect

- **Explore in Discover** permet d'explorer les données de la visualisation dans la page Discover
- **Inspect** permet de voir
 - les données de la visualisation. Elles sont téléchargeables au format CSV
 - la requête Elasticsearch et l'ouvrir directement dans la DevConsole.

Visualisation Kibana

Les tableaux de bord

Lens

TSVB

Agregations

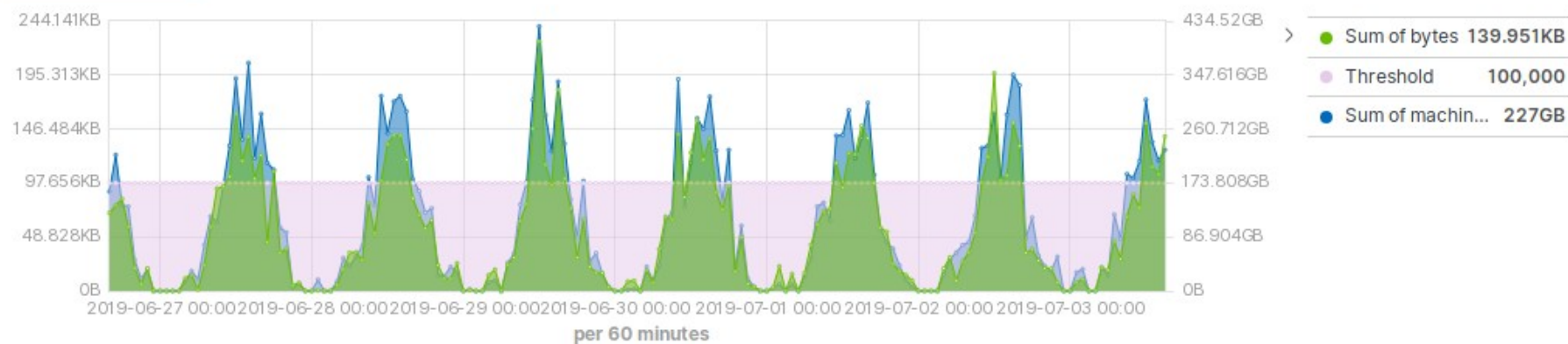
Timelion

TSVB

- Time Series Visual Builder (TSVB) est un outil plus spécialisé, conçu spécifiquement pour la création de visualisations de séries temporelles complexes.
- Outre les graphiques de lignes classiques, TSVB supporte des types de visualisations comme les "gauge" (jauges), "markdown" (texte enrichi), et des barres d'état, ce qui le rend idéal pour les tableaux de bord de monitoring et d'observation.
- TSVB offre des options avancées pour l'agrégation et le filtrage des données

Séries multiples : TSVB

- Les fonctionnalités de TSVB sont :
 - Combiner un nombre infini d'agrégations pour afficher vos données.
 - Annoter les données de séries chronologiques avec des événements horodatés à partir d'un index Elasticsearch.
 - Afficher les données dans plusieurs types de visualisations, notamment des graphiques, des tableaux de données et des panneaux de démarques.
 - Afficher plusieurs vues de données dans chaque visualisation.
 - Utiliser des fonctions personnalisées et des calculs mathématiques sur les agrégations.
 - Personnaliser les données avec des étiquettes et des couleurs.



☒ Auto apply

The changes will be automatically applied.

...

Data Panel options Annotations

Label =

Metrics Options

Aggregation Field

Sum bytes

Group by

Everything

> Threshold =

> Label =

Configuration des séries

- Chaque série peut être considérée comme une agrégation Elasticsearch distincte.
- Chaque série a des options d'affichage distinctes.
- Une série indique sa fonction d'agrégations et éventuellement un regroupement

Autres graphiques

- Après avoir configuré ces séries, d'autres visualisations sont générées dans les onglets de TSVB
 - Metric : Simple valeur
 - Top N
 - Gauge : Visualisation Simple valeur
 - Markdown : Texte riche pouvant utiliser des variables fournies par les séries
 - Table : Nécessite de définir un autre critère de regroupement

Visualisation Kibana

Les tableaux de bord

Lens

Vega

Agregations

Timelion

Vega

- Vega et Vega-Lite sont deux grammaires permettant de créer des visualisations personnalisées.
- Il est nécessaire d'écrire des requêtes Elasticsearch manuellement.
- Vega et Vega-Lite utilise JSON

Cas d'usage

- Utilisez Vega ou Vega-Lite lorsque vous souhaitez créer des visualisations avec :
 - Des agrégations qui utilisent un mapping imbriqué ou parent/enfant
 - Des agrégations sans dataview
 - Des requêtes qui utilisent des filtres temporels personnalisés
 - Des calculs complexes
 - Des données extraites de la source du document (`_source`)
 - Des graphiques en nuage de points, des graphiques Sankey et des cartes personnalisées
 - Un thème visuel non pris en charge
- Les tables ne sont pas supportées

HJSON

- Vega et Vega-Lite utilisent tous deux JSON, mais Kibana a simplifié la saisie en intégrant HJSON.
 - Guillemets facultatifs
 - Guillemets doubles ou guillemets simples
 - Virgules facultatives
 - Commentaires utilisant la syntaxe // ou /*
 - Chaînes multilignes

Visualisation Kibana

Les tableaux de bord

Lens

TSVB

Vega

Agregations

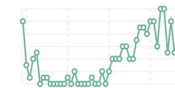
Timelion

Introduction

- Avec les visualisations basées sur l'agrégation, vous pouvez :
 - Diviser les graphiques jusqu'à trois niveaux d'agrégation, ce qui est plus que Lens et TSVB
 - Créer une visualisation avec des données non chronologiques
 - Utiliser une recherche enregistrée comme entrée
 - Trier les tableaux de données et utiliser les fonctionnalités de ligne de résumé et de colonne de pourcentage
 - Attribuer des couleurs aux séries de données
 - Étendre les fonctionnalités avec des plugins
- Elles ont les limitations suivantes :
 - Options de style limitées
 - Math n'est pas prise en charge
 - Les indices multiples ne sont pas pris en charge

Agrégations

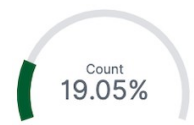
- **Area** : Visualise les contributions totales de plusieurs séries
- **Table de données** : Affichage en tableau des données agrégées
- **Line** : Compare différentes séries
- **Metric** : Affichage d'une seule métrique
- **Gauge/Goal** : Une unique valeur avec des intervalles de bonnes/mauvaises valeurs ou par rapport à un objectif
- **Pie** : Camembert .
- **Heat map** : Tableau coloré
- **Nuage de tags** : Les tags les plus importants ont la plus grande police
- **Horizontal / Vertical bar** : Histogramme permettant la comparaison de séries



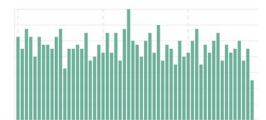
\$446.23
Avg. Ticket Price



Top values of response keyword	
keyword	value
21:00	200
21:30	200
22:00	200
22:30	200
23:00	200
23:30	200



Heavy Fog Clear
Cloudy
Rain Sunny
Thunder & Lightning



Étapes de création

- Les étapes de création de l'éditeur classique consistent donc à :
 1. Choisir un type de graphique
 2. Spécifier les critères de recherche ou utiliser une recherche sauvegardée
 3. Choisir le calcul d'agrégation pour l'axe des Y (*count, average, sum, min, ...*)
 4. Choisir le critère de regroupement des données (bucket expression)
(Histogramme de date, Intervalle, Termes, Filtres, Significant terms, ...)
 5. Définir éventuellement des sous-agrégations
 6. Choisir les couleurs des séries
- Lens permet de créer sa visualisation plus intuitivement via Drag & Drop

Bucket expressions

- **Histogramme Date** : Un histogramme de date est construit sur un champ entier pour lequel on a précisé une fenêtre temporel
- **Intervalle** : Entier, date ou IPV4
- **Terme** : Permet de présente les n meilleurs (ou plus basse) pertinence ordonné par la valeur agrégée
- **Filtres** : Il est possible d'ajouter des filtres de données à ce niveau
- **Significant Terms** : Agrégation *significant terms*
- **Geohash** : Agrégation sur les coordonnées géographiques (Data Table et carte)

Options

- Lors de plusieurs agrégations en Y, il est possible de spécifier leur mode d'affichage
 - **Stacked** : Empile les agrégations les unes sur les autres.
 - **Overlap** : Superposition avec transparence
 - **Wiggle** : Ombrage
 - **Percentage** : Chaque agrégation comme une portion du total
 - **Silhouette** : Chaque agrégation comme variance d'une ligne centrale
 - **Grouped** : Groupe les résultats horizontalement par les sous-agrégations (Grapique barre) .

Atelier : Agrégations

Visualisation Kibana

Les tableaux de bord

Lens

TSVB

Vega

Agregations

Timelion

Timelion

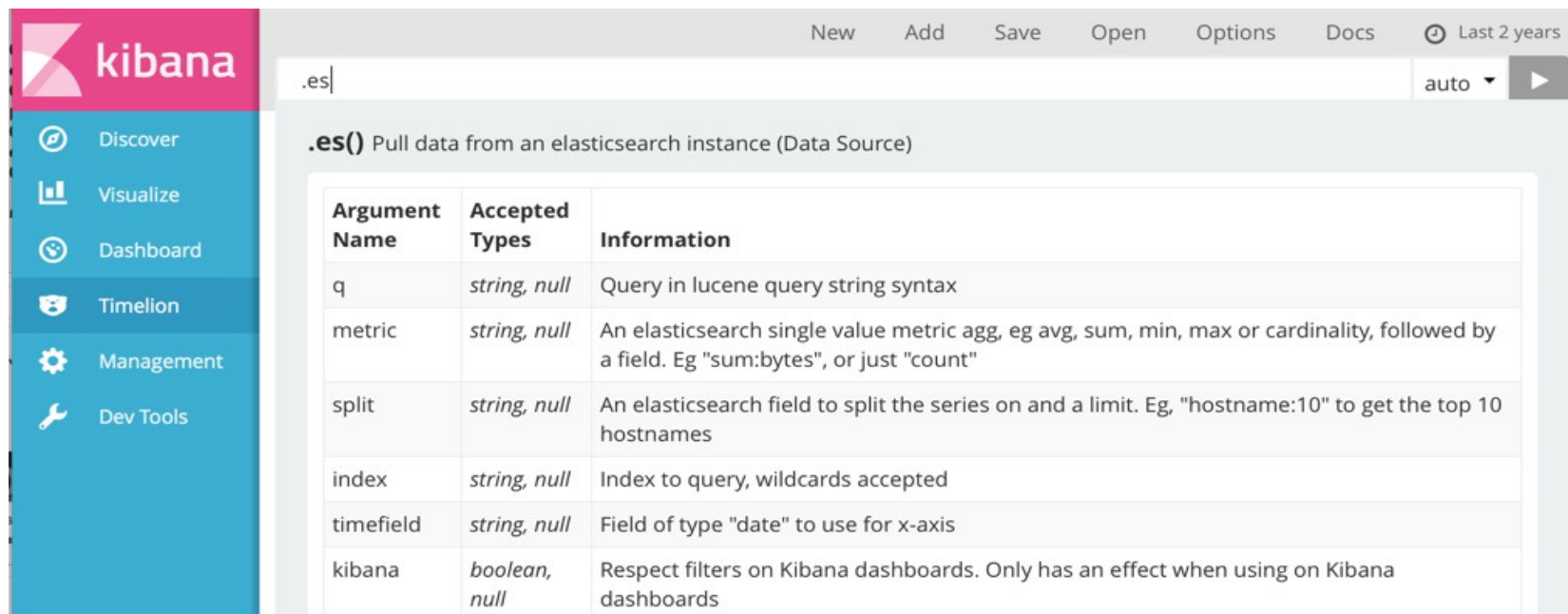
- Timelion est un visualiseur dédié aux données temporelles qui permet de combiner des sources de données complètement indépendantes dans le même graphique.
- Il est piloté par un langage d'expression simple permettant de récupérer les données et d'effectuer des calculs.
- Avec timelion, il est possible de répondre à ce type de question :
Quel est le pourcentage de la population japonaise qui a visité mon site ?

Éléments de syntaxe

- L'interface de timelion consiste en :
 - Une zone de saisie d'expression
 - Une zone de visualisation
- Chaque expression timelion démarre avec une fonction identifiant une source de données. Pour *ElasticSearch* :
`es(*)`
- 2 sources de données peuvent être dessinées côte à côte en utilisant la virgule (,) ;
`.es(*) , .es(metric=cardinality:user)`
- Elles peuvent être combinées via des fonctions :
`.es(*).divide(es(metric=cardinality:user))`
- D'autres sources de données peuvent être utilisées (Worldbank's Data API par exemple)
`.wbi(MZ) // Population du Mozambique`

Aide en ligne et documentation

- La documentation fait partie de *timelion*
- Elle est disponible par le lien Docs
- De plus, la complétion est très efficace



The screenshot shows the Kibana user interface. On the left is a sidebar with the Kibana logo and navigation links: Discover, Visualize, Dashboard, Timelion (highlighted), Management, and Dev Tools. The main area displays the documentation for the `.es()` function, titled ".es() Pull data from an elasticsearch instance (Data Source)". At the top of the main area, there is a search bar with ".es|" and a dropdown menu set to "auto". Below the title is a table with three columns: Argument Name, Accepted Types, and Information.

Argument Name	Accepted Types	Information
q	string, null	Query in lucene query string syntax
metric	string, null	An elasticsearch single value metric agg, eg avg, sum, min, max or cardinality, followed by a field. Eg "sum:bytes", or just "count"
split	string, null	An elasticsearch field to split the series on and a limit. Eg, "hostname:10" to get the top 10 hostnames
index	string, null	Index to query, wildcards accepted
timefield	string, null	Field of type "date" to use for x-axis
kibana	boolean, null	Respect filters on Kibana dashboards. Only has an effect when using on Kibana dashboards

Exemples

```
# Affichage en temps réel de la moyenne du
# pourcentage d'utilisation CPU user à partir
# d'un index metricbeats d'elastic search
.es(index=metricbeat-*, timefield='@timestamp',
metric='avg:system.cpu.user.pct')
```

```
# La même chose en ajoutant une série affichant
# les données pour un offset d'1h
.es(index=metricbeat-*, timefield='@timestamp',
metric='avg:system.cpu.user.pct'), .es(offset=-
1h,index=metricbeat-*, timefield='@timestamp',
metric='avg:system.cpu.user.pct')
```

Visualisation Kibana

Les tableaux de bord

Lens

TSVB

Vega

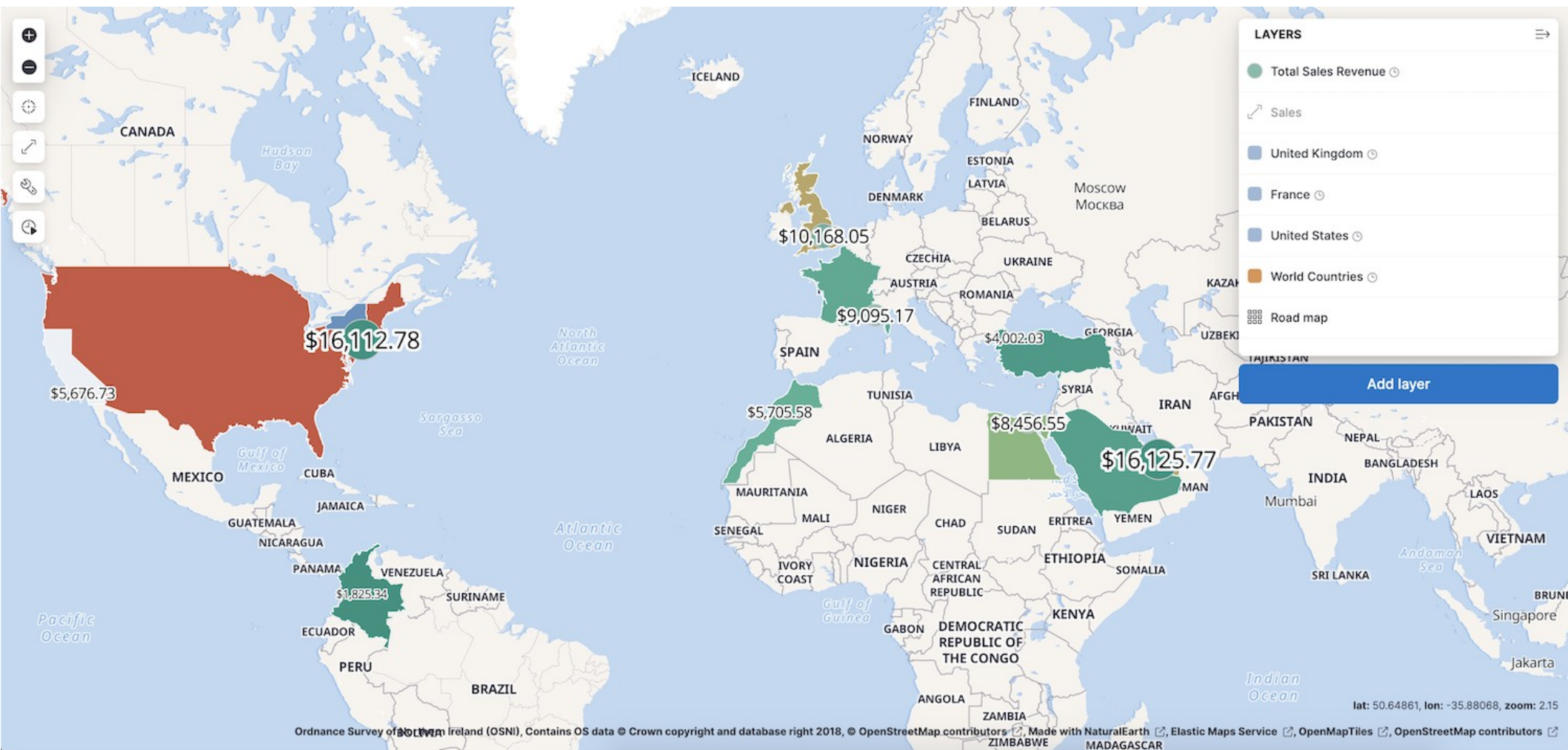
Agregations

Timelion

Maps

Maps

- L'éditeur Maps permet de :
 - Créez des cartes avec plusieurs couches et index.
 - Animez des données spatio-temporelles.
 - Téléchargez des fichiers GeoJSON et des calques de forme.
 - Intégrez votre carte dans des tableaux de bord.
 - Utiliser des symboles en fonction des valeurs d'une donnée.



TP : Atelier Map

Interactions

Interactions
Canvas

Interactions

- Les tableaux de bord peuvent avoir des fonctionnalités interactives :
 - Ajouter des contrôles permettant le filtrage des données
 - Proposer des liens vers Discover
 - Personnaliser des liens pour faire du Drill-down

Contrôles

- Les contrôles sont des panneaux interactifs ajouté aux tableaux de bord pour filtrer les données
- 3 types de contrôle sont disponibles :
 - Liste déroulante
 - Slider d'intervalle
 - Slider de temps

Liens vers Discover

- Les panneaux, utilisant une seule dataview, peuvent ajouter des liens vers Discover permettant d'ouvrir la page avec les données de la visualisation.
- 3 types d'interactions vers Discover sont possibles :
 - Interactions du **panneau** : Ouvre les données du panneau dans Discover, y compris les filtres au niveau du tableau de bord, mais pas les filtres au niveau du panneau.
 - Interactions de données de **série** — Ouvre les données de série dans Discover.
 - Interactions de **recherche enregistrées** — Ouvre les données de recherche enregistrées dans Discover

Drilldowns

- 3 types d'interactions drilldowns sont possible :
 - **Tableau de bord** : Permet de naviguer d'un tableau de bord à un autre.
Par exemple, créez une vue détaillée pour un panneau Lens qui vous permet de naviguer d'un tableau de bord récapitulatif à un tableau de bord avec un filtre pour un nom d'hôte spécifique.
 - **URL externe** : Naviguer vers un site externe en passant des paramètres
 - **Discover** : Ouvrir les données dans Discover pour accéder au détail

Interactions

Interactions
Canvas

Canvas

- Kibana Canvas, permet de créer des rapports complètement personnalisés
- Dans *Canvas*, les projets sont appelés "présentations", elles sont analogues aux présentations habituelles Powerpoint et peuvent comporter plusieurs pages ...
Et en plus les données sont dynamiques !

Workspace

- Canvas propose un workspace où il est possible de placer des éléments directement connectés aux données EL
- Les éléments peuvent être personnalisés en travaillant directement sur le CSS

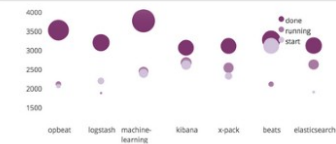
Éléments

🔍 Filter elements



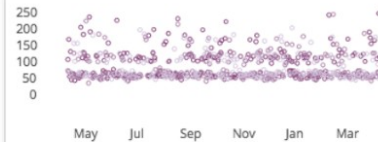
Area chart

A line chart with a filled body



Bubble chart

A customizable bubble chart



Coordinate plot

Mixed line, bar or dot charts

cost #	username #	state #	cost #	price #
22.99	acollinsd9	done	22.99	51
23.43	kphillipsmv	done	23.43	54
21.84	jfloresn0	running	21.84	71
23.12	jcarpenter5c	done	23.12	53
21.93	sbutlerb1	running	21.93	67
23.13	agardnerd0	done	23.13	60

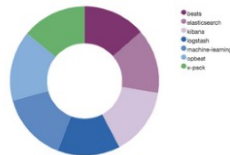
Data table

A scrollable grid for displaying data in a tabular format

```
"time": 1460444400000,
"username": "swhitejp",
},
{
  "age": 74,
  "cost": 22.69,
  "country": "CN",
  "price": 79,
```

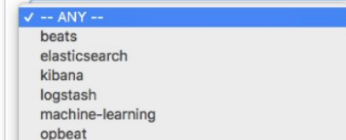
Debug

Just dumps the configuration of the element



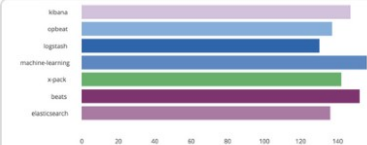
Donut chart

A customizable donut chart



Dropdown filter

A dropdown from which you can select values for an "exactly" filter



Horizontal bar chart

A customizable horizontal bar chart

Dismiss

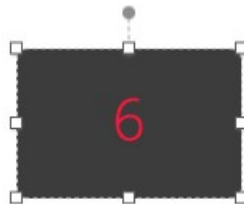
Canvas expression

- Derrière chaque élément configuré dans l'interface, il y a une **expression Canvas** éditable via l'*Expression editor* qui définit comment ce composant est construit
- Une expression est composée de plusieurs fonctions chaînées par |.
- Les fonctions disponibles permettent
 - De définir des jeux de données : démo, query, expression timeline
 - De filtrer, transformer les données
 - d'exécuter des fonctions mathématiques complexes
 - De la logique
 - Définir les axes de visualisation, le style CSS, formater les dates, ...

Example

- Exemple Markdown

```
filters
| essql
query="SELECT timestamp, anomaly_score FROM \".ml-anomalies-*\" WHERE
result_type = 'bucket' AND anomaly_score > 10 AND job_id = 'nginx-traffic'"
| markdown "
#
#
# {{rows.length}}"
| render css="h1 {
text-align: center;
color: #ff1744;
}
"
containerStyle={containerStyle backgroundColor="#444444" border="5px none
#FFFFFF" borderRadius="7px" padding="px"}
```

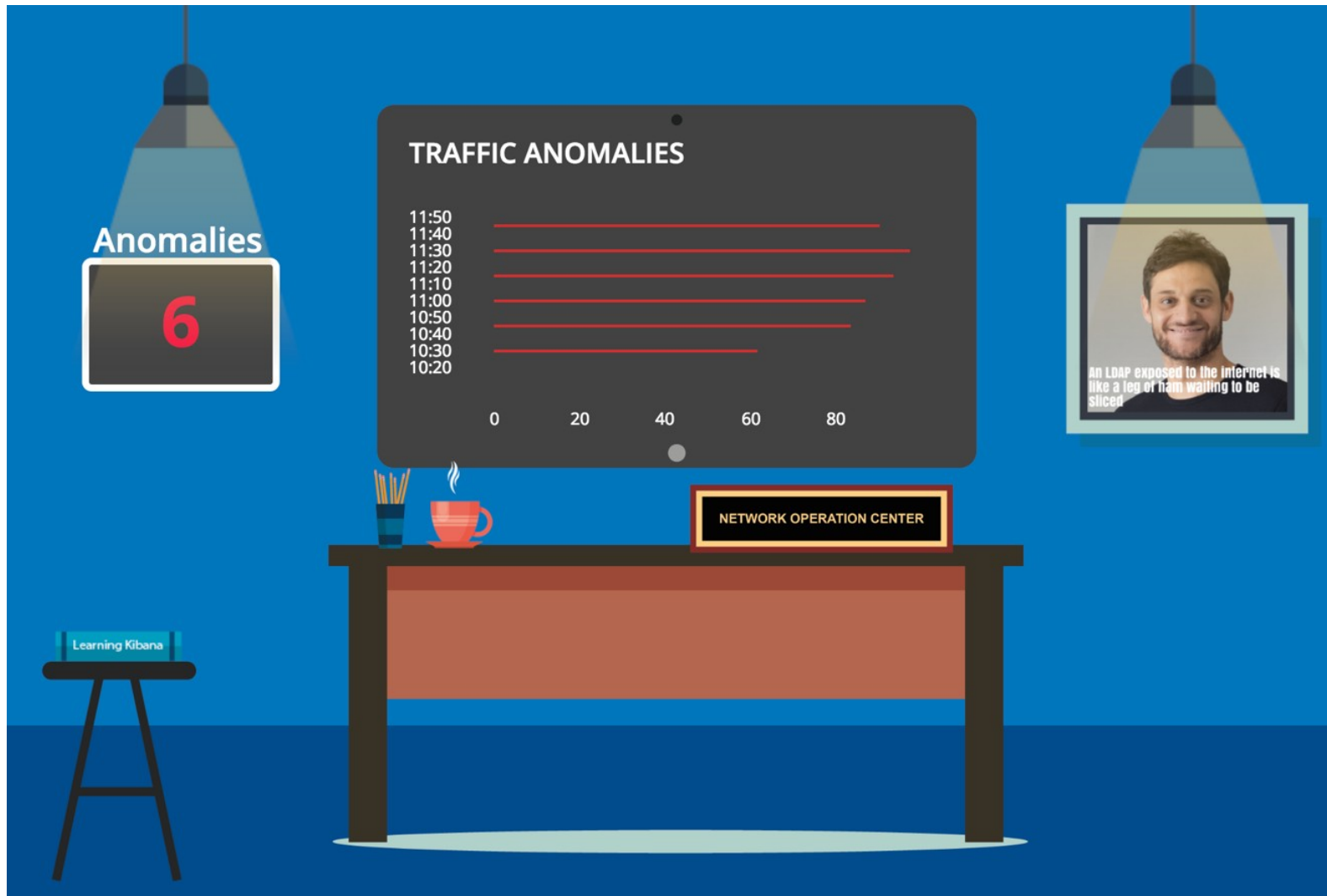


Example

- Exemple personnalisation d'un graphique (bar)

```
filters
| essql
query="SELECT timestamp, anomaly_score FROM \".ml-anomalies-shared\"
WHERE result_type = 'bucket' AND anomaly_score > 10 AND job_id =
'nginx-
traffic'"
| pointseries x="anomaly_score" y="timestamp"
| plot
defaultStyle={seriesStyle lines=0 bars="2" points=0 horizontalBars=true
color="#d32f2f"} legend=false xaxis=true yaxis=true
font={font family="'Open Sans', Helvetica, Arial, sans-serif" size=12
align="left" color="#FFFFFF" weight="normal" underline=false
italic=false}
| render containerStyle={containerStyle backgroundColor="#444444"}
```

Exemple Canvas



Annexes

Machine Learning

Annexes

FileBeat
Pipelines ElasticSearch
Analyseurs
Machine Learning

Introduction Machine Learning

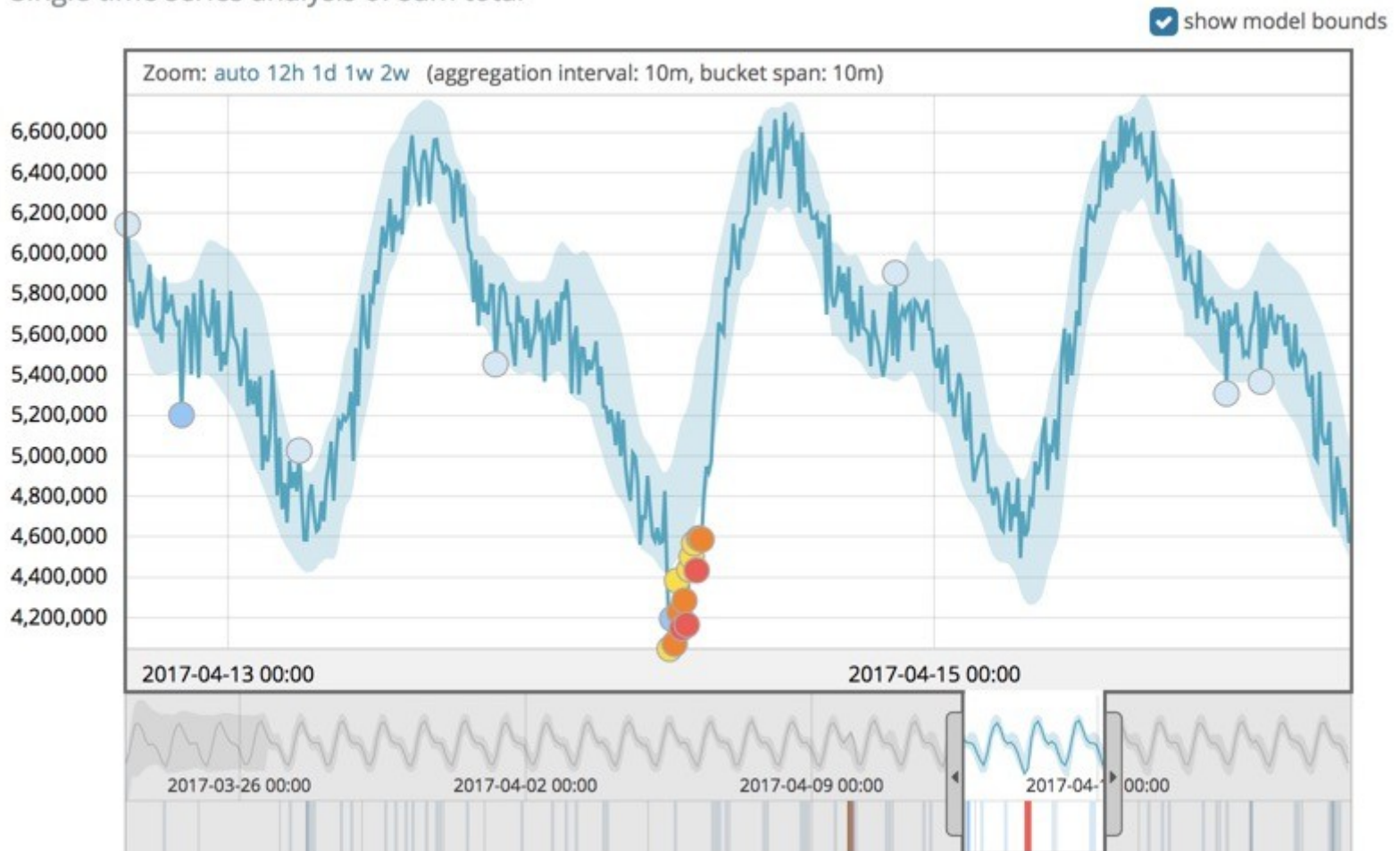
- Rachat de la société Prevert
- Introduit dans la Version 5.4, fonctionnalités X-Pack
- Permet de se poser les questions :
 - Certains de mes services ont-ils changé de comportement ? »
 - « Y a-t-il des processus inhabituels qui s'exécutent sur mes machines ?
- Basé sur des modèles comportementaux, permet la la détection d'anomalies dans des données temporelles

Principe

- Les données relatives au temps sont extraites d'ElasticSearch pour analyse :
 - soit de façon continue
 - soit périodiquement
- Les résultats anormaux sont affichés dans Kibana.
- Différentes situations sont alors remontées :
 - Anomalies liées à des écarts temporels dans les valeurs, les décomptes ou les fréquences
 - Des raretés statistiques
 - Des comportements inhabituels pour un membre d'une population

Valeurs actuelles, limites normales et anomalies

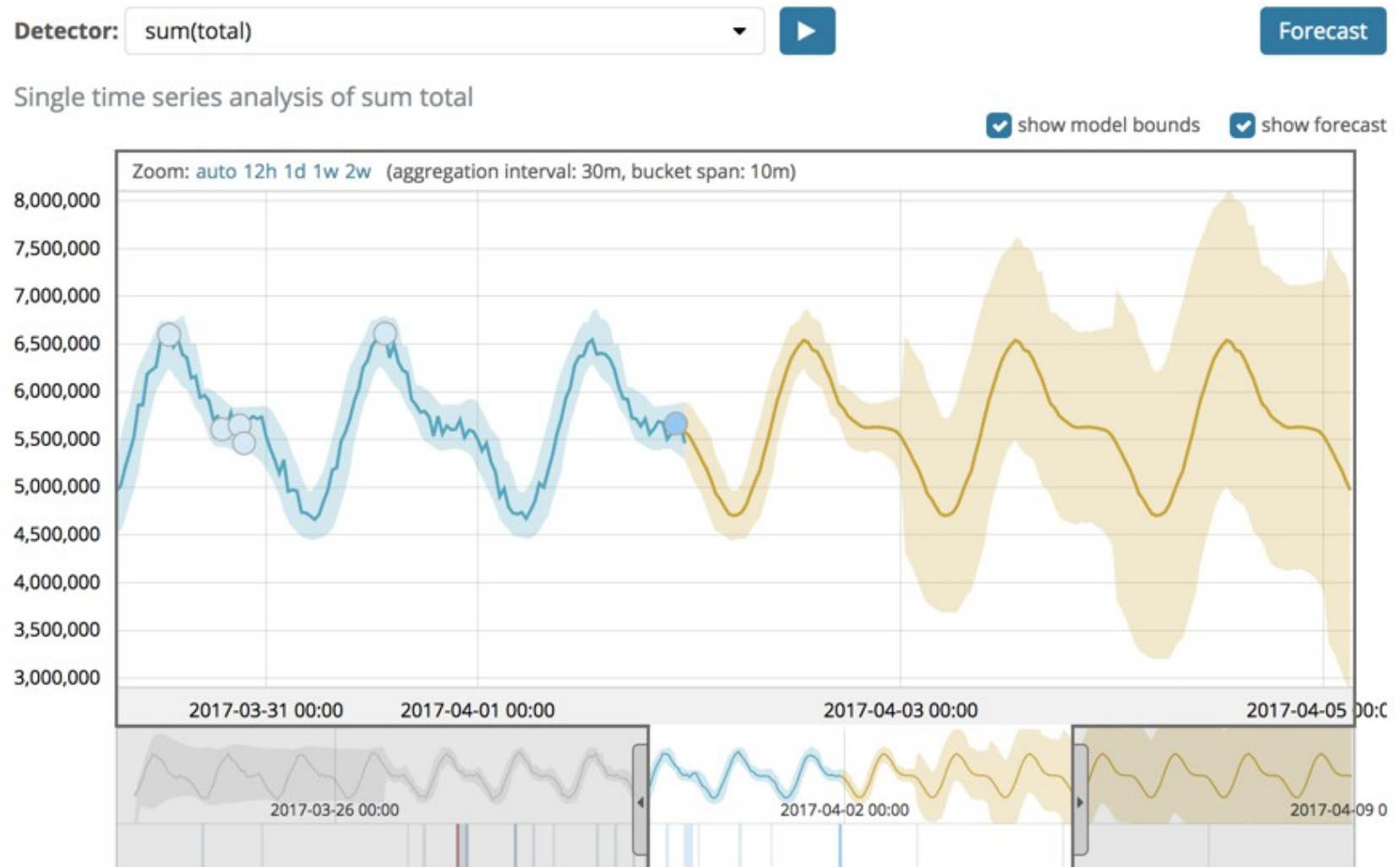
Single time series analysis of sum total



Anticipation

- Les profils extraits du passé peuvent être utiliser pour anticiper le futur.
- Par exemple, prévoir :
 - Le nombre de visites d'un site web dans 1 mois
 - Quand mon disque atteindra 100 % d'utilisation
- Ces prévisions peuvent être visualisées dans Kibana

Intervalle de prévisions en jaune



Tâches d'analyse

- Les **Machine Learning jobs** contiennent les informations de configuration nécessaire à une tâche d'analyse.
 - Chaque job a un ou plusieurs détecteurs correspondant à une fonction analytique sur certains champs de donnée
 - Il contient également des propriétés indiquant quels événements doivent être analysés par rapport à des comportements précédents ou une population
- Kibana propose des assistants permettant de créer ce type de job
- Les jobs peuvent également être groupés afin de visualiser ensemble leurs résultats
- Les jobs sont exécutés sur des nœuds du cluster qui ont les propriétés de configuration **xpack.ml.enabled** et **node.ml** positionnées à *true*

Fonctions analytiques

- Fonction de :
 - **Comptage** : Détecte des anomalies lorsque le nombre d'événements dans un groupement est anormal
 - **Géographiques** : Détecte les anomalies dans l'emplacement géographique d'une donnée
 - Sur le **contenu** : Détecte les anomalies dues au volume d'une donnée String
 - **Métriques** : Anomalies sur des moyennes, des min, des max.
 - Détection de **rareté** : Anomalies qui arrivent rarement sur le temps ou rarement dans une population
 - **Somme** : Anomalies lorsque la somme d'un champ dans un groupement est anormal
 - **Temporelle** : Détecte des événements qui arrivent à des moments inhabituels. Exemple une week-end

Flux de données

- Les flux de données d'entrée d'un job d'analyse sont créés
 - Soit à partir d'un index pattern ElasticSearch (Typiquement, lorsque l'on utilise Kibana)
 - Soit programmatiquement via une API
- Un job n'est associé qu'à un seul flux de données d'entrée
- Dans la cas d'ELS, les flux de données doivent être démarrés (et arrêtés) ; cela est fait via Kibana

Buckets

- Les ***buckets*** sont utilisés pour diviser les séries temporelles en traitement par lots
- Ils font partie de la configuration d'un job et déterminent l'intervalle de temps utilisé pour agréger les données.
En particulier, calculer un score d'anomalie

Événements hors norme

- Il est possible de planifier des périodes où l'activité d'événements sera inhabituelle.
(Par exemple : black fridays ou autre)
- Dans ce cas, les tâches d'analyse ne détectent pas d'anomalies