



# Machine Learning avec Elastic Stack

David THIBAU – 2021

david.thibau@gmail.com

# Agenda

- **Introduction**

- Offre Elastic Stack
- Clustering et Big Data
- Machine Learning pour l'IT

- **Offre Elastic ML**

- Concepts Elastic
- Interface Kibana

- **Détection d'anomalies**

- Détection de changement
- API
- Analyse de cause et jobs multiples
- Prévisions

- **Analyse de données**

- Introduction
- Détection de valeurs anormales
- Régression
- Classification
- Inférence

- **Visualisations Kibana**

- URLs personnalisés
- Les différentes visualisations pour le ML

# Introduction

**L'offre Elastic Stack**  
Clustering  
Machine Learning pour l'IT

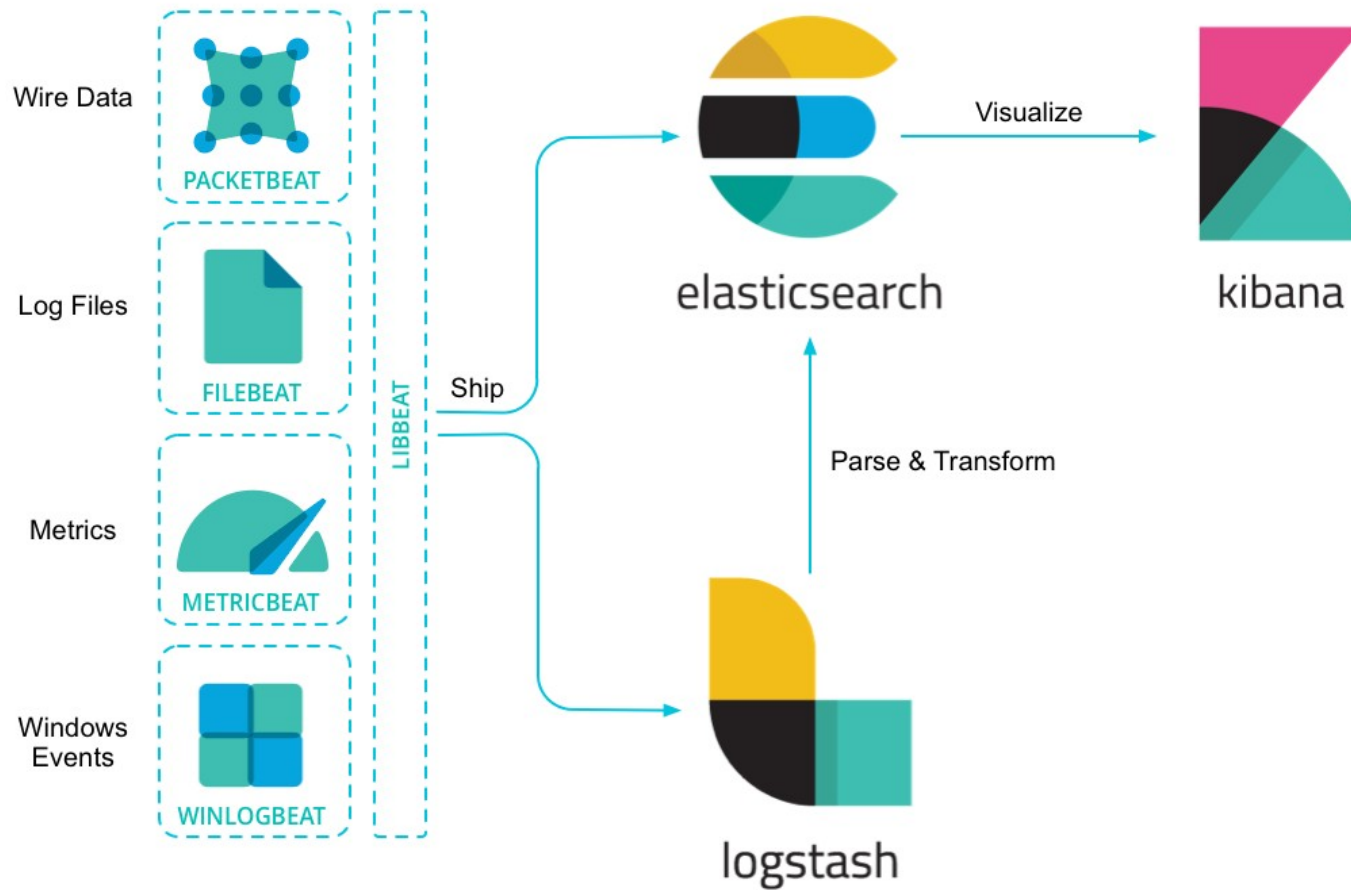
# Introduction

- **Elastic Stack** : Suite d'outils facilitant l'analyse et la visualisation temps-réel de données volumineuses
- Offre OpenSource et Commerciale de la société *Elastic*

# La pile

- La suite est composé des outils suivants :
  - **Elasticsearch** : Base documentaire NoSQL basée sur le moteur de recherche Lucene
  - **Logstash** : Outil de traitement d'événements permettant la mise en place de pipeline de transformations, et qui exporte les données vers des destinations diverses dont les index ElasticSearch
  - **Beats** : Agents spécialisés permettant de fournir régulièrement des données à logstash ou directement ElasticSearch
  - **Kibana** : UI End-user, permettant la data visualisation, l'exploitation de la suite et proposant des IDEs pour le dév.

# Architecture



# Autres outils

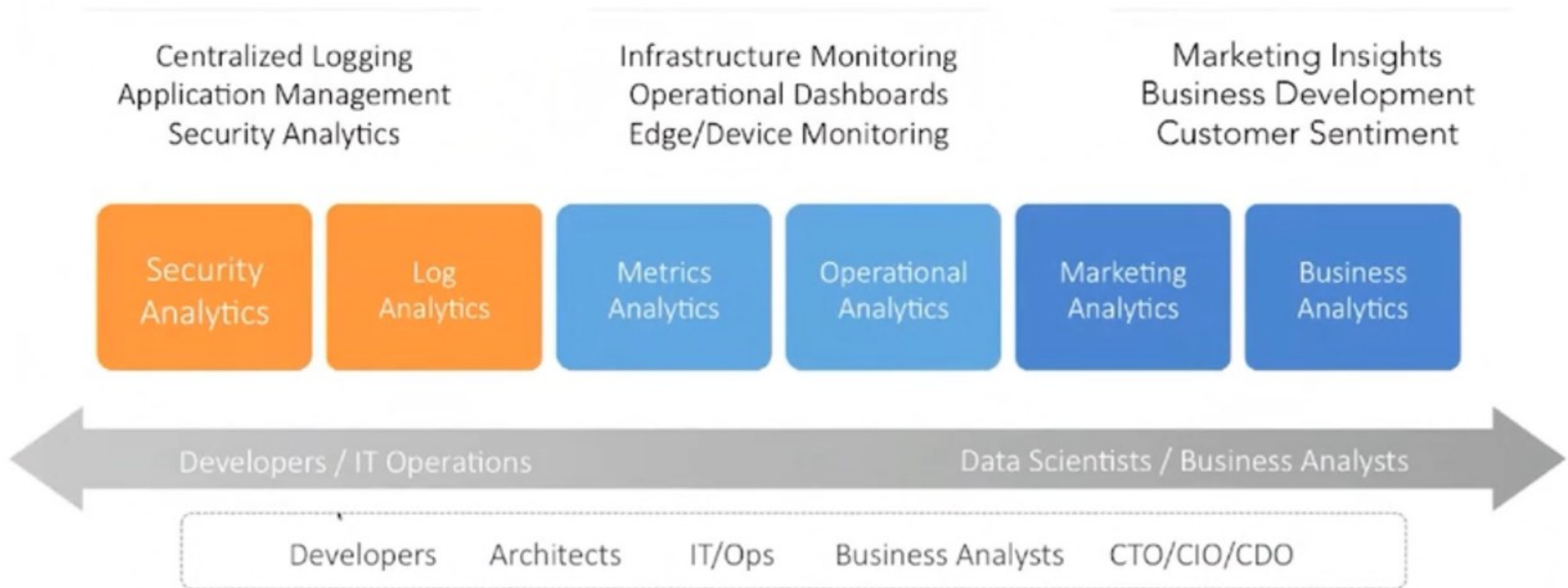
- D'autres outils connexes peuvent être ajoutés :
  - ***X-Pack*** : Inclus dans la distribution, certaines fonctionnalités doivent être activées avec une licence commerciale ou d'évaluation
    - Sécurité (Libre à partir de 6.3+)
    - Monitoring, Alerte, Reporting, Machine Learning
  - ***ES-Hadoop*** : Stockage BigData



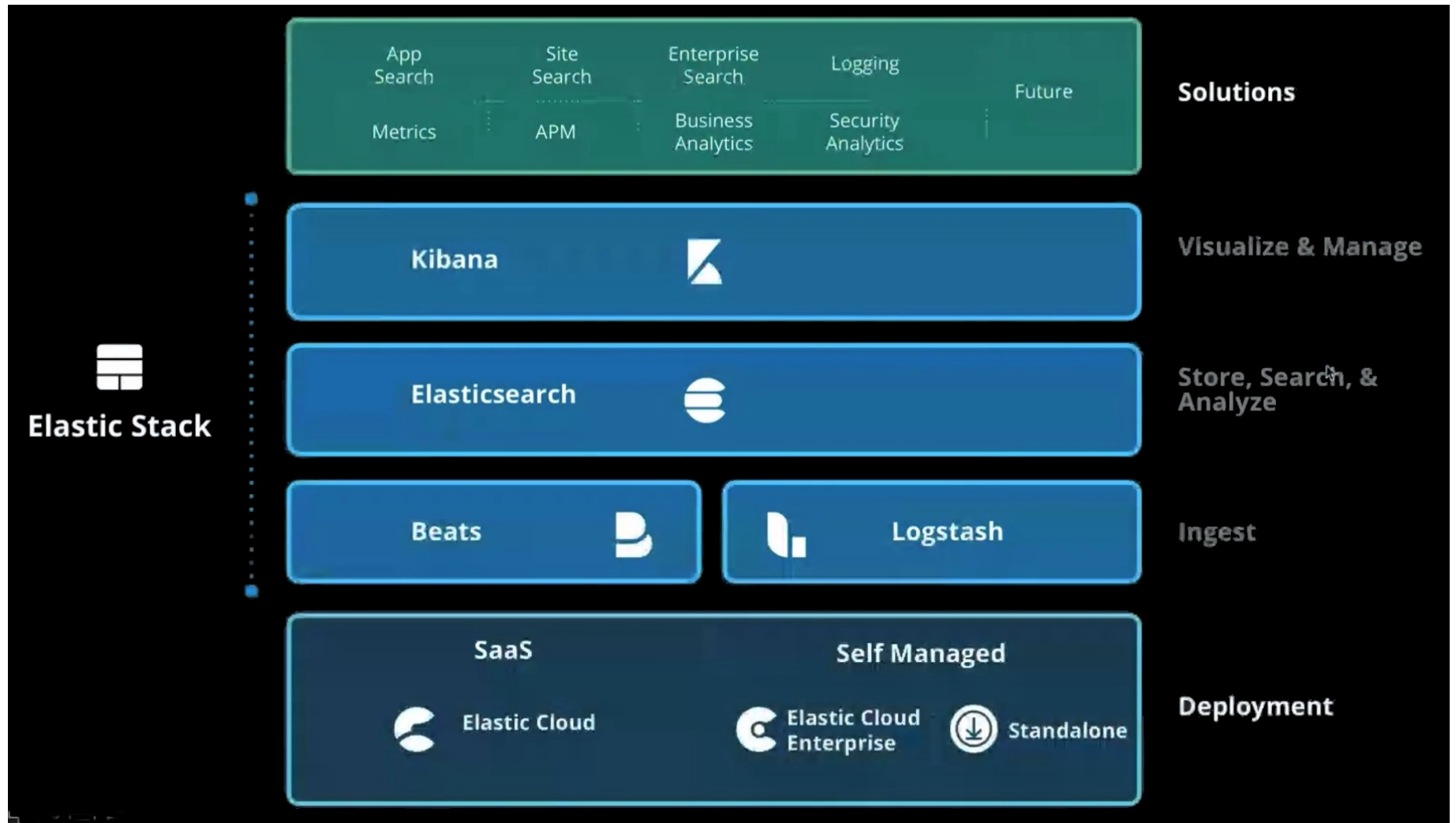
# Offre Commerciale

- ***Elastic Cloud / Enterprise*** : ELK As A Service,
- Distributions sur des cas d'usage :
  - ***Elastic Enterprise Search*** : Moteur de recherche, Site Web,
  - ***Elastic Observability*** : Stack pour surveiller un SI : Logs, APM, ...
  - ***Elastic Security*** : Protéger un SI contre les menaces

# Use cases de la suite



# En résumé



# Introduction

L'offre Elastic Stack  
**Clustering**  
Machine Learning pour l'IT

# Cluster ElasticSearch

- 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 n'être constitué que d'un seul nœud
- => Un nœud ne peut pas appartenir à 2 clusters distincts

# Nœud

- Un **nœud** est un simple processus faisant partie d'un cluster.
- Un nœud stocke des données, et participe aux fonctionnalités du cluster : indexation, recherche, ingestion, ...
  - 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
- Dans une première approche, tous les nœuds sont interchangeables.

# 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



# 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 transparente pour l'utilisateur

# Réplica

- 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)
  - De **scaler** le volume des requêtes car les recherches peuvent être exécutées sur toutes les répliques en parallèle .

# Spécialisation des nœuds

- Tous les nœuds d'un cluster se connaissent mutuellement et peuvent rediriger des requêtes HTTP vers le nœud approprié. Il est possible de spécialiser les nœuds afin de répartir la puissance entre la charge de gestion des données et la charge d'ingestion.
- Les différents types de nœuds sont :
  - Nœuds pouvant être **maître**
  - Nœuds de **données** : Détient les données et effectue les tâches d'indexation et de recherche
  - Nœuds **d'ingestion** : Exécute les pipelines d'ingestion
  - Nœuds de **coordination** : Nœuds acceptant les requêtes et redirigeant vers les nœuds appropriés
  - Nœuds **cross-cluster** : Nœuds pouvant effectuer des recherches vers plusieurs cluster.
  - Nœuds **ML** : Dédiés à l'exécution des jobs de Machine Learning

# Nœuds maître

- Le nœud maître est responsable d'opérations légères :
  - Création ou suppression d'index
  - Surveillance des nœuds du cluster
  - Allocations des shards
- Dans un environnement de production, il est important de s'assurer de la stabilité du nœud maître.
- Il est conseillé pour de gros cluster de ne pas charger les nœuds maîtres avec des travaux d'ingestion, d'indexation ou de recherche.

```
node.roles: [ master ]
```

# Configurations des nœuds

- Configuration d'un nœud de **données**

`node.roles: [ data ]`

Ou spécialisation des données récentes

`node.roles: [ data_hot, data_warm ]`

- Configuration d'un nœud **d'ingestion**

`node.roles: [ ingest ]`

- Configuration d'un nœud de **coordination**

`node.roles: [ ]`

# Introduction

L'offre Elastic Stack  
Clustering  
**Machine Learning pour l'IT**

# Introduction

- L'offre de ML d'Elastic même si elle peut être appliquée à de nombreux domaines se concentre sur la surveillance IT :
  - Surveillance : Détection de panne, de dysfonctionnement
  - Sécurité : Détection d'intrusion, d'exfiltration
- Dans la surveillance IT, le constat est :  
*On génère beaucoup de données mais on a pas le temps de les regarder et analyser*  
=> Automatisation de ces analyses par des systèmes pouvant apprendre seul



# Anomalies

- La majorité des exigences de surveillance sont des variations sur le thème :  
*Trouver quelque chose qui est différent de l'ordinaire*
- Les processus d'analyse tente alors de détecter des anomalies par rapport à l'observation de l'historique  
=> Cela nécessite que l'historique soit assez volumineux afin que le ML puisse affiner son modèle

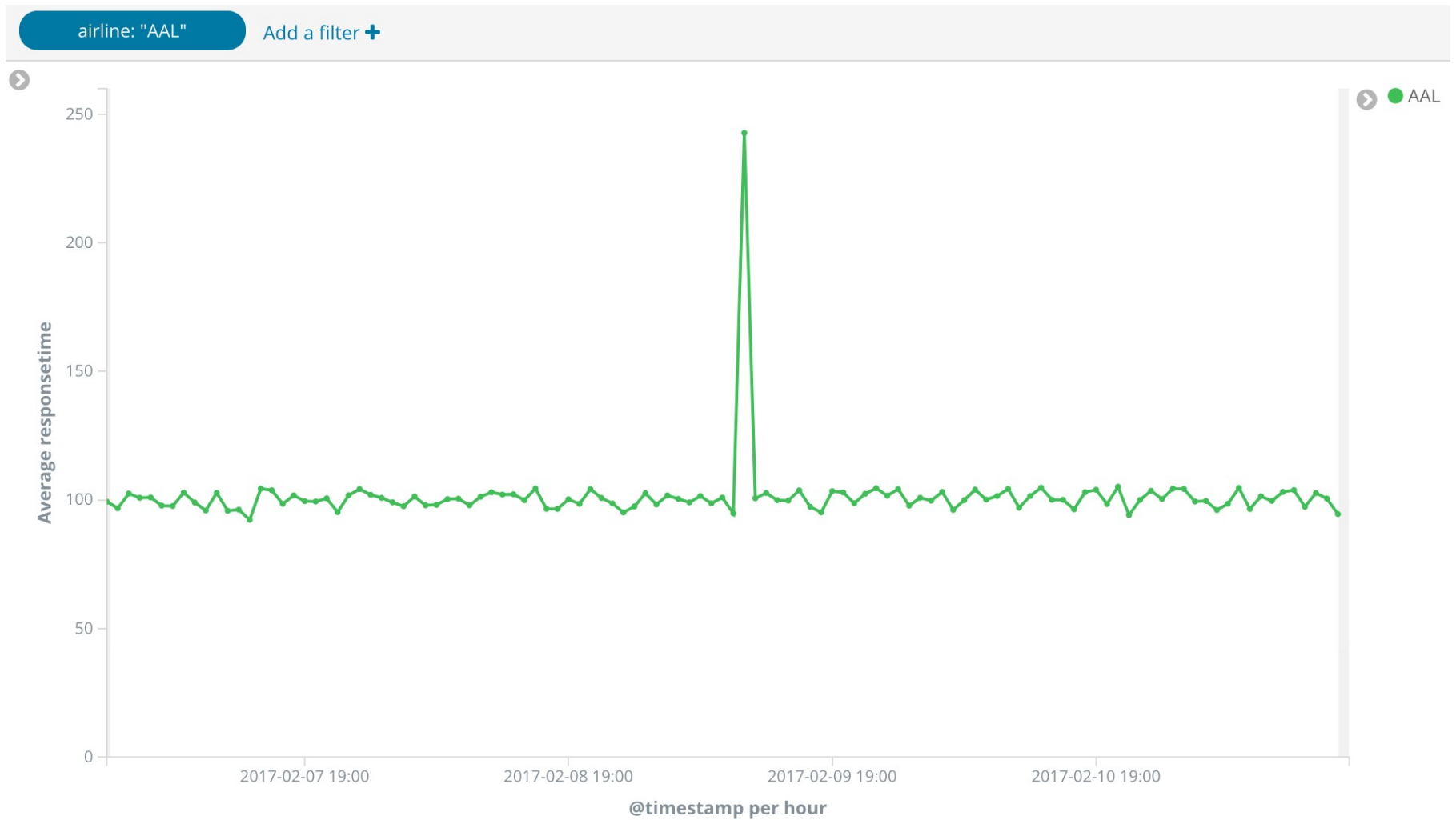
# Objectifs des solutions ML

- **Rapidité**: Notification d'une panne, d'une brèche ou de toute autre anomalie significative pro-activement et le plus rapidement possible afin de l'atténuer.
- **Scalability**: Les algorithmes doivent pouvoir scaler linéairement avec les données.
- **Performance** : Utiliser du matériel modeste plutôt que des super-ordinateurs
- **Applicabilité**: Prise en compte de la diversité des données dans les environnements informatiques.
- **Adaptabilité**: Les environnements informatiques en constante évolution peuvent rapidement rendre fragile un algorithme statique.
- **Précision**: Ne pas générer de fausses alarmes

# Qu'est-ce qu'une anomalie ?

- 2 définitions d'une anomalie :
  - Quelque chose est inhabituel si son comportement a dévié de manière significative d'un modèle établi basé sur son histoire passée
  - Quelque chose est inhabituel si certaines de ses caractéristique sont significativement différentes des mêmes caractéristiques des autres membres d'un ensemble ou d'une population

# Ex : Histoire passée



# Ex : Population



# Apprentissage non supervisé

- En ML, l'apprentissage non supervisé ne nécessite aucune intervention humaine pour la mise en place et l'affinement du modèle.
  - => Le modèle se construit, s'affine, se modifie en fonction des données qui lui sont présentées
  - => Plus il y a de données, plus le modèle se précise
- Elastic Stack propose 2 types d'analyse **non supervisé**
  - La détection d'anomalie : Basée sur des données temporelles
  - Détection de valeurs anormales : Basée sur des densité de valeurs

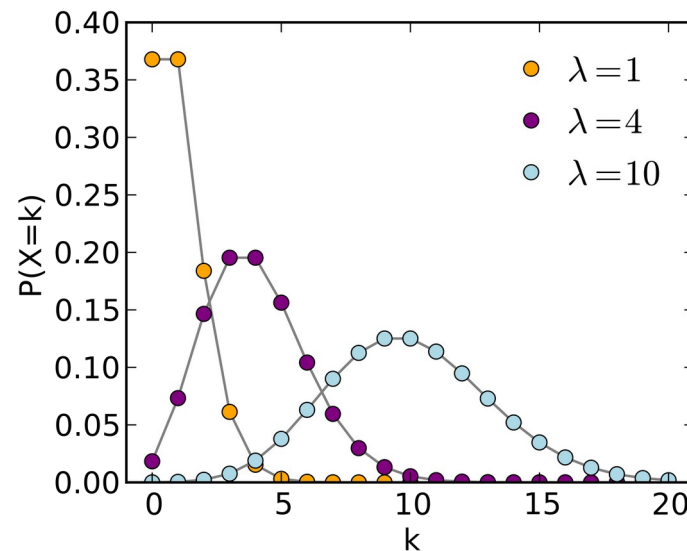
# Apprentissage supervisé

- Les apprentissages supervisés nécessitent des données d'entraînement.
- ELK-ML propose de types d'analyse supervisés :
  - Classification : Classification d'évènements  
Ex : Requêtes normales ou malicieuses
  - Régression : Prédire des valeurs numériques  
Ex : réponse pour une requête Web.

# Ex : Distribution de poisson

- Des modèles statistiques sont utilisés pour modéliser le comportement d'un système.
- Par exemple, la distribution de poisson permet de modéliser des événements tels que
  - Le nombre de météorites de plus d'un mètre de diamètre qui frappent la Terre chaque année
  - Le nombre de patients arrivant dans une salle d'urgence entre 10h00 et 23h00
  - Le nombre de photons frappant un détecteur dans un intervalle de temps particulier

$$p(k) = \mathbb{P}(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$



Si un certain type d'événements se produit en moyenne 4 fois par minute, pour étudier le nombre d'événements se produisant dans un laps de temps de 10 minutes, on choisit comme modèle une loi de Poisson de paramètre  $\lambda = 10 \times 4 = 40$



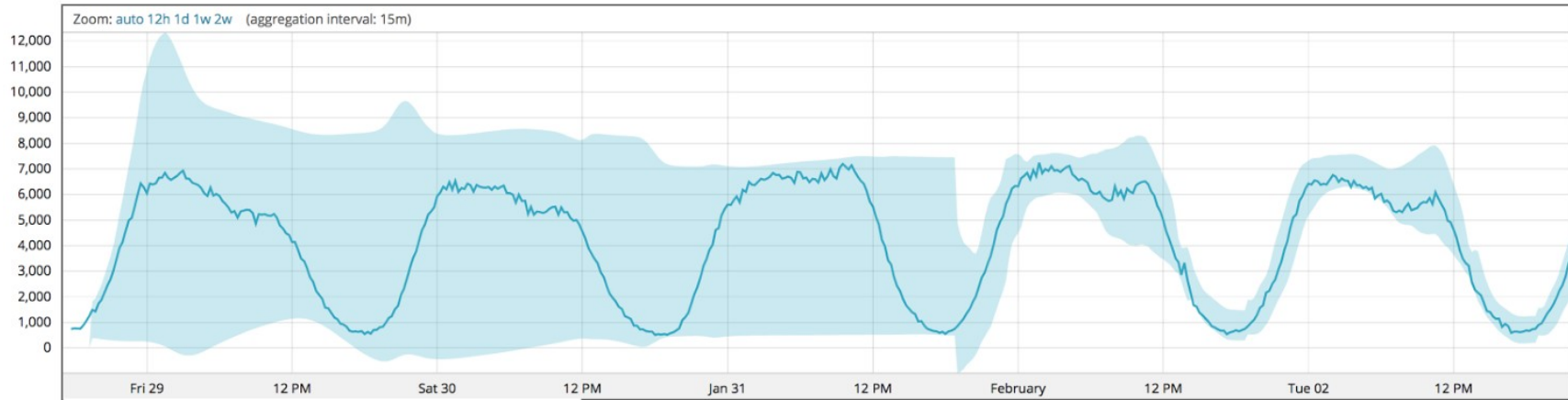
# Processus de modélisation

- Les processus de machine learning sélectionne à partir de ses observations le modèle de probabilité approprié (poisson, gaussien, log-normal, etc.) et ses coefficients
- Des techniques bayésiennes sont utilisées pour évaluer les probabilités des valeurs du modèle compte tenu de l'ensemble de données observées et permettent de tempérer les résultats en fonction du nombre d'informations visualisées.
- La modélisation effectuée est continue, de sorte que de nouvelles informations sont considérées avec les anciennes, avec une pondération exponentielle de l'information plus fraîche
- La modélisation peut être arrêtée puis redémarrée plus tard, d'où la nécessité de persister le modèle.

# Cycles

- Un autre aspect important de la modélisation de données réelles est de prendre en compte les tendances harmoniques importantes qui se produisent naturellement.
- ML recherche automatiquement les tendances marquantes dans les données (croissance linéaire, harmoniques cycliques, etc.) et les factorise

# Exemple



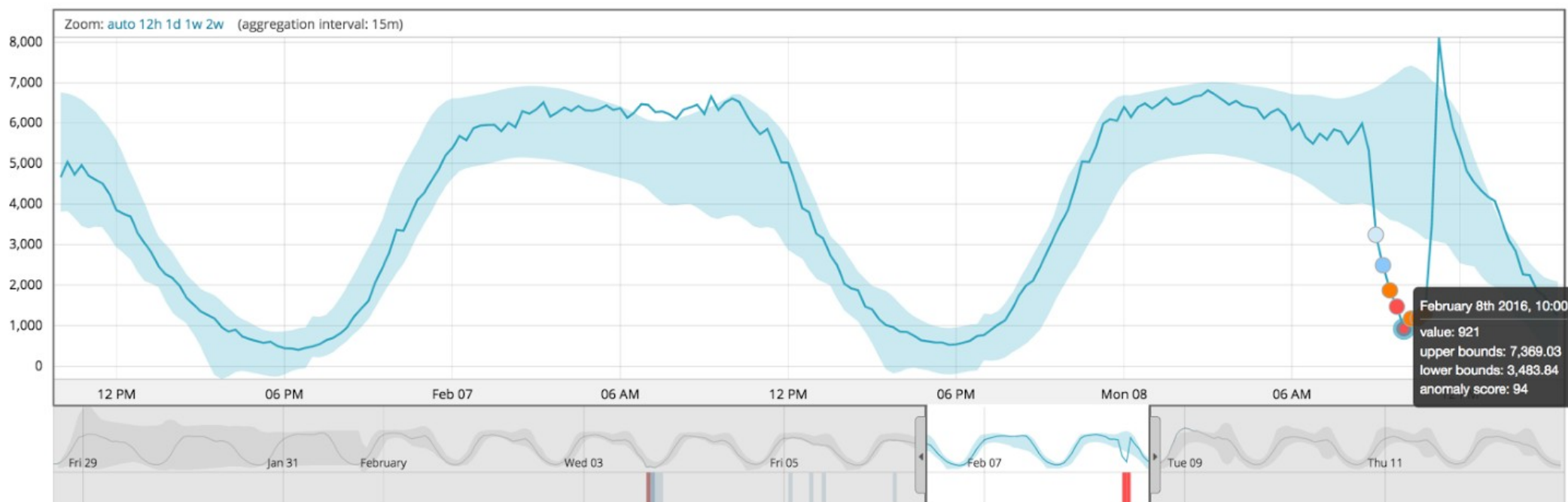
- Le cycle quotidien périodique est appris, puis factorisé.  
La prédiction du modèle s'ajuste après la détection automatique de trois itérations successives de ce cycle.

# Score d'anomalie

- Les modèles de probabilité permettent de calculer le taux de probabilité qu'une mesure prenne une certaine valeur.
- Les taux de probabilité oscillant entre 0 et 1 sont normalisés en **scores d'anomalie** qui oscille entre 0 et 100
- Des niveaux de sévérité, des alertes sont associés à des seuils des scores d'anomalie

# Exemple

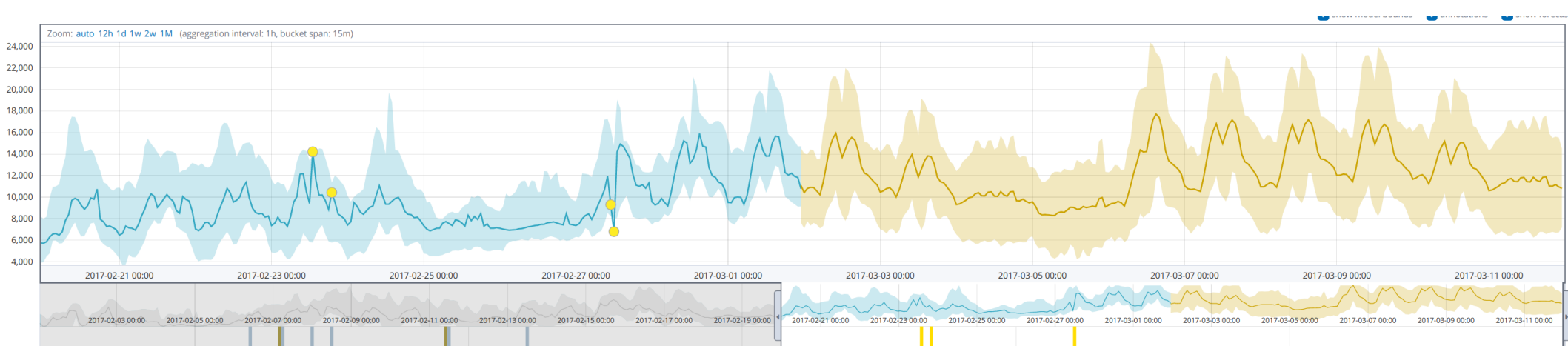
- La probabilité de la valeur 921 était de  $6.3634e-7$
- Cela donne un score d'anomalie de 94
- Ce qui donne une anomalie critique ( $> 75$ )



# Prévisions

- Une fois que ML a créé son modèle comportemental, il l'utilise pour extrapoler le comportement futur :
  - Estimer une valeur à une date future
  - Probabilité qu'une valeur atteigne un seuil
- Chaque prévision a un *id* unique, une durée, un délai d'expiration
- Les prévisions ne peuvent pas s'effectuer sur tous les types de valeurs
- Elles sont limitées à 3 pour le même job
- Elles nécessitent pas mal de mémoire

# Example



# L'offre Elastic ML

**Concepts Elastic**  
Interface Kibana



# Job

- Avec ELK-ML, un **job** est l'unité de travail. Il est constitué de :
  - Un **id** (nom)
  - Le **bucket span** : fenêtre temporelle d'analyse, i.e. la granularité de l'analyse
  - Le **datafeed** : La définition des données à analyser (une requête)
  - Le (ou les) **détecteurs** : la configuration de la détection d'anomalie
- Un job s'exécute sur un nœud ML
- Il peut s'exécuter en continu ou à la demande

# Nœuds ML

- La ML peut être activée sur tout ou partie des nœuds, mais il est recommandé dans un système de production d'avoir dédié des nœuds au ML  
Propriétés de configuration ***xpack.ml.enabled*** et ***node.roles = [ml]***
- Contrairement aux nœuds de données effectuant bcp d'I/O, les nœuds ML nécessitent davantage de CPU et de mémoire
- Les algorithmes ML ne s'exécutent pas dans la JVM. Ce sont des exécutables C++ qui utilisent la RAM laissée par la JVM

# Bucketization

- Le ***bucket-span*** correspond à la fenêtre de temps pour laquelle les données sont agrégées à des fins d'analyse
  - Plus la durée de *bucket\_span* est courte, plus l'analyse est fine, mais aussi plus l'influence du bruit dans les données est élevée
- Pour la configurer prendre en compte :
  - La granularité de l'analyse
  - La fréquence des données d'entrée
  - La durée typique d'une anomalie
  - La rapidité d'alerte voulue
- Typiquement entre 5mn et 1h

# Influence du bucket



# Datafeed

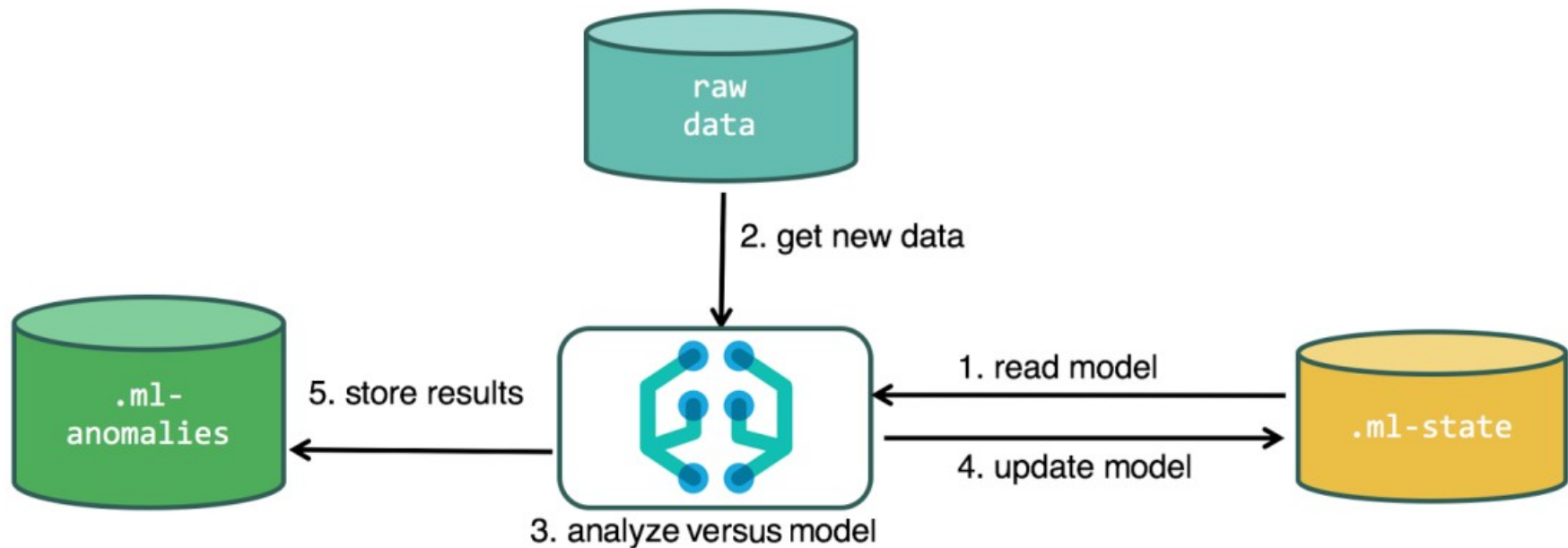
- Le ***datafeed*** est le mécanisme par lequel les données sont régulièrement récupérées (recherchées dans les index) et présentées aux algorithmes ML
- Un datafeed peut être démarré (il s'exécute toutes les x temps) et arrêté (Désactivation)
- La fréquence et le volume de données de chaque requête dépendent de :
  - ***bucket\_span*** : Date range queries
  - ***frequency*** : Égal au *bucket* sauf si celui-ci est supérieur à 20 mn, dans ce cas la fréquence est + courte
  - ***query\_delay*** : La latence, afin d'être sûr que toutes les données du *bucket\_span* ont bien été ingérées
  - ***scroll\_size*** : Taille de la pagination lors de la requête elastic search.

# Index ML

- Elastic ML utilise 3 index :
  - ***.ml-state*** : Informations internes sur le modèle statistiques appris
  - ***.ml-notifications*** : Stocke les messages d'audit qui apparaissent dans l'UI : *Job* → *Messages*
  - ***.ml-anomalies-\**** : contient les résultats détaillés des jobs.  
L'index *.ml-anomalies-shared* contient les informations de plusieurs jobs.  
Il est possible de dédier un index à un job particulier

# Workflow

- Séquence pour un bucket\_span



# Types de job

- Le job « **Single Metric** » permet d'analyser une unique métrique. (Typiquement un KPI)
- Un job « **Multi-metric** » permet de corréler différentes analyses portant sur des métriques différents.  
Ce type de job est adapté à l'analyse de cause
- Le job **Population** permet de comparer des entités entre-elles
- Le job « **Advanced** » permet d'avoir plus de contrôle sur la configuration de l'analyse.  
Les autres types de job ne sont que des simplifications des job « *Advanced* »



# Détecteurs

- Lors de la création de job, on spécifie un ou plusieurs **détecteurs**, qui définissent les champs de données du job et le type d'analyse à effectuer :
- Un détecteur est défini par les propriétés suivantes :
  - ***field\_name*** : Le champ utilisé
  - ***function*** : La fonction d'analyse utilisée
  - ...

# Autres concepts

- Influenceurs :
  - les influenceurs sont des champs que l'on soupçonne influencer ou contribuer aux anomalies
- Calendrier et événements planifiés
  - EL – ML permet d'indiquer des plages de temps où le comportement du système est inhabituel (« black friday »).  
=> Les jobs peuvent être configurés pour exclure ces plages lors de l'analyse
- Règles personnalisées :
  - Permettent de personnaliser la détection d'anomalies

# ML APIs

- L'UI de Kibana offre une simplification de l'utilisation de l'API Rest mais celle-ci reste la plus puissante et la plus flexible.
- Avec l'API, il est possible de :
  - Gérer les jobs : créer/éditer/exécuter, déclencher des prévisions
  - Gérer les datafeeds : Créer/démarrer/arrêter/surveiller
  - Accéder au résultats des analyses : Bucket/Anomalies/Influenceurs
  - ....

# Exemple : Création de Job

```
PUT _ml/anomaly_detectors/total-requests
{
  "description" : "Total sum of requests",
  "analysis_config" : {
    "bucket_span": "10m",
    "detectors": [
      {
        "detector_description": "Sum of total",
        "function": "sum",
        "field_name": "total"
      }
    ]
  },
  "data_description" : {
    "time_field": "timestamp",
    "time_format": "epoch_ms"
  }
}
```

# L'offre Elastic ML

Concepts Elastic  
**Interface Kibana**

# Introduction

- L'interface Kibana propose 4 menus principaux :
  - **Détection d'anomalies** : ML non supervisé, permet de définir les jobs de détection d'anomalie et des visualisations
  - **Analyse de trame de données** : ML supervisé pour la classification et les régressions
  - **Data Visualizer** : Visualiser les données d'entrées
  - **Settings** : Configuration des calendriers, ...

# Data Visualizer

- ***Data Visualizer*** est un outil pratique pour explorer le contenu d'un jeu de données
- Il identifie automatiquement les champs de l'index qui sont non vides et propose un graphe de distribution des valeurs de chaque champ  
=> Data Analyser vous aide à choisir les champs à analyser avec Elastic ML

# Informations sur les champs

- Les champs dans les index sont répertoriés dans 2 sections :
  - Les champs **numériques** ("métriques").  
Pour chaque champ, le Visualiseur indique le nombre de documents contenant le champ dans la période sélectionnée, les valeurs minimales, médianes et maximales, le nombre de valeurs distinctes et leur distribution.
  - **keyword** :  
Nombre de valeurs distinctes, Top values, % de documents contenant le champ
  - Les **autres** champs (text, date, boolean)  
*date*:  
Plus ancienne, plus récente, % de documents contenant le champ



# Tableau de bord des jobs

- Les jobs sont séparés dans les menu « Détection d'anomalie » et « Analyse de données »
- Les tableaux présentent :
  - L'id du job, sa description, Le nombre de documents analysés
  - Son statut mémoire, i.e. la mémoire utilisée par le modèle :
    - ok,
    - soft\_limit : les vieux modèles vont être nettoyés
    - hard\_limit : toutes les données n'ont pas pu être traitées
  - Le statut du job :
    - opened : Prêt à recevoir et traiter des données
    - closed : S'est arrêté correctement, ses données persistées
    - closing : En train de se fermer
    - failed : En échec, peut nécessiter une suppression et recréation
  - Le statut du datafeed
    - started :
    - stopped :
  - Dernière exécution
  - Action : Démarrage du datafeed, Edition/Suppression du job

# Job Management

Machine Learning / Job Management

30 seconds

[Job Management](#) [Anomaly Explorer](#) [Single Metric Viewer](#) [Data Visualizer](#) [Settings](#)

Active ML Nodes: 0 Total jobs: 1 Open jobs: 0 Closed jobs: 1 Active datafeeds: 0

Refresh

+ Create new job

Search...

Opened Closed Failed Started Stopped Group ▾

<input type="checkbox"/>	ID ↑	Description	Processed records	Memory status	Job state	Datafeed state	Latest timestamp	Actions
<input type="checkbox"/>	> total-requests	Total sum of requests	14,040	ok	closed	stopped	2017-04-01 23:59:00	  

Rows per page: 10 ▾

# Résultats

- Pour visualiser les résultats :
  - **Anomaly Explorer** : Couloirs indiquant le score d'anomalie maximal au fil du temps. Il y a des couloirs pour
    - le résultat global
    - chaque influenceur.  
Chaque bloc représente un bucket-span et est sélectionnable
  - **Single Metric Viewer** : Graphique représentant les valeurs réelles et attendues dans le temps.  
Uniquement disponible pour les jobs Single Metric.  
Les points de données anormaux sont affichés en différentes couleurs en fonction de leur score.

# *Single Metric Viewer*

- Les valeurs réelles sont indiquées par une ligne bleue. Une zone bleue ombrée représente les limites des valeurs attendues.
- Si une valeur est en dehors de la zone ombragée, elle est anormale.
  - Un score d'anomalie est calculé pour le bucket et donne une couleur au point
  - Les détails des anomalies est visible dans la partie inférieure
- Il est possible de faire glisser le sélecteur de temps

# Single Metric Viewer

Machine Learning / Single Metric Viewer

Auto-refresh ◀ ⌚ March 23rd 2017, 06:00:00.000 to April 22nd 2017, 04:59:00.000 ▶

Job Management Anomaly Explorer Single Metric Viewer Data Visualizer Settings

Job total-requests

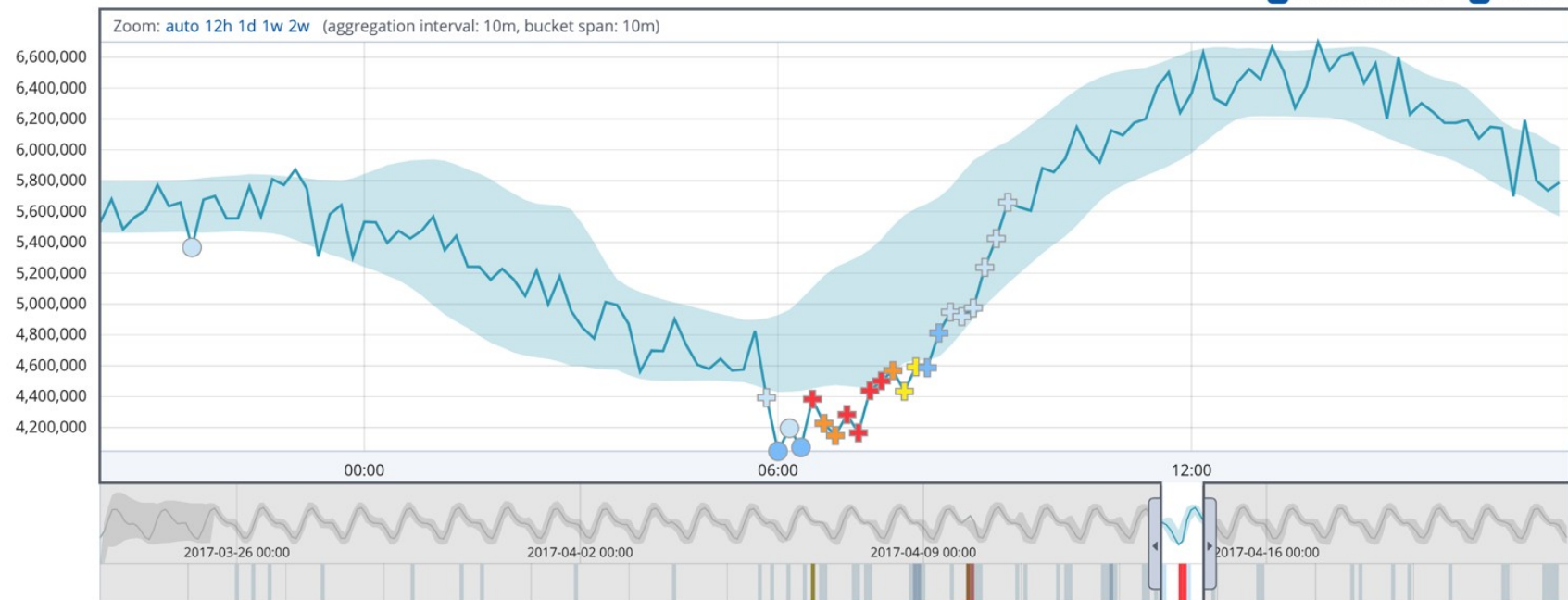
Detector: sum(total)



Forecast

Single time series analysis of sum total

☒ show model bounds ☒ annotations



## Anomalies

Severity threshold

warning

Interval

Auto

time	max severity ↓	detector	actual	typical	description	job ID	actions
> April 14th 2017, 07:00	● 96	sum(total)	4,283,309	4,869,925.771	↓ 1.1x lower	total-requests	⚙
> April 14th 2017, 06:00	● 88	sum(total)	4,382,911	4,779,628.322	↓ 1.1x lower	total-requests	⚙

# Multi-bucket impact

- Il se peut que des événements anormaux n'existent pas au sein d'une seule plage mais se produisent sur une gamme de plages contigus.
- Pour prendre en compte cette situation, EL-ML effectue une analyse multi-bucket en prenant en compte les buckets contigus
- Lorsque l'on visualise les résultats, une propriété ***multi\_bucket\_impact*** indique la force avec laquelle le score final d'une anomalie est influencé par l'analyse multi-buckets
- Dans Kibana, les anomalies avec des impacts multi\_bucket élevé sont indiquées avec des croix plutôt que des points

# Anomaly Explorer

- Cette vue est adaptée à tout type de job. Elle permet d'explorer la chronologie globale des anomalies.
  - Pour chaque section de la période spécifiée, le score d'anomalie maximal est indiqué.
  - La section ne correspondent pas nécessairement au bucket span. Elles s'adaptent à la période sélectionnée. La taille la plus petite cependant est le bucket span.
  - La section est clickable et permet d'accéder aux détails des anomalies (graphique et tableau)
- Elle met en lumière les influenceurs, i.e. facteurs contributifs aux anomalies globales
  - À gauche, la liste des principaux influenceurs avec leur score maximal d'anomalies.
  - Un couloir par influenceur affichant la chronologie des anomalies

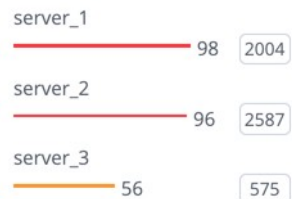
# Example

## Top Influencers

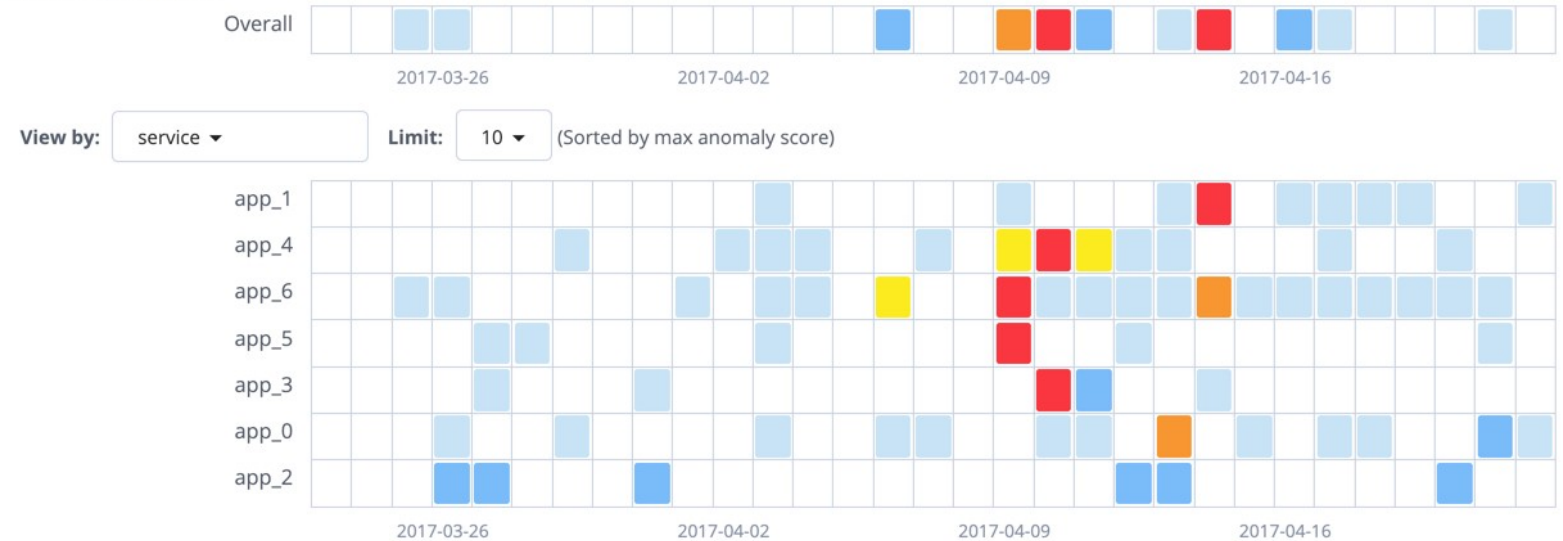
### service



### host



## Anomaly timeline



## Anomalies

Severity threshold

warning

Interval

Auto

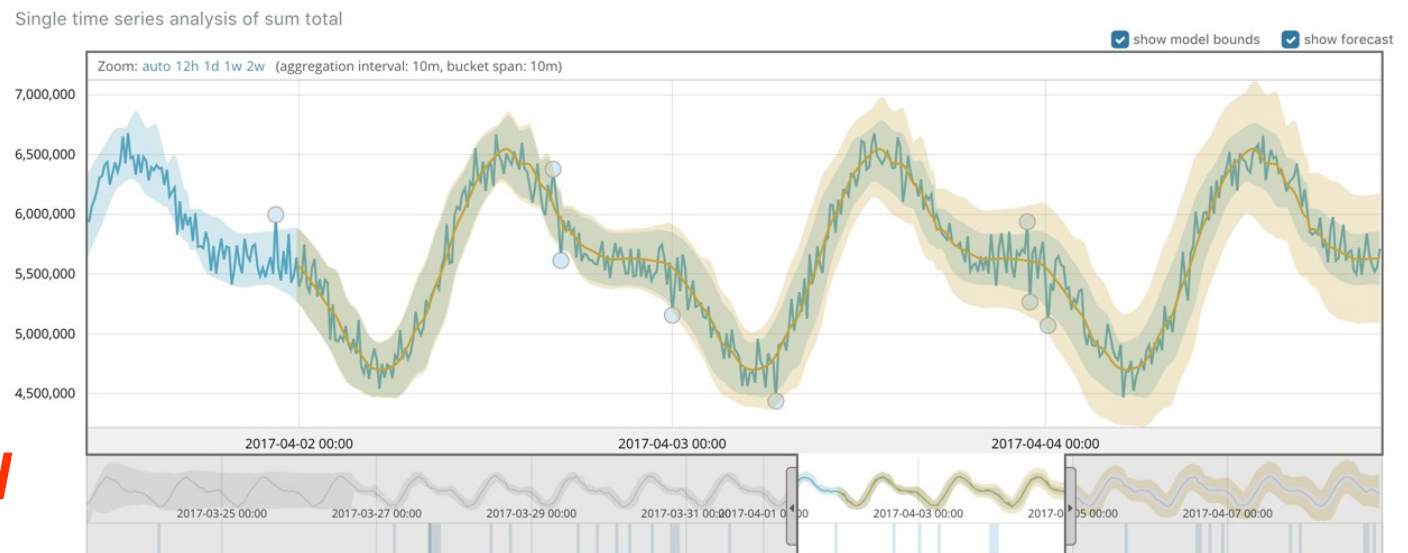
time	max seve...	detector	found for	influenced by	actual	typical	description	job ID	actions
> April 14th 2017	99	sum(total)	app_1	service: app_1	707,990	1,301,631.575	↓ 2x lower	response_requ ests_by_app	⚙
> April 9th 2017	99	high_mean(res ponse)	app_6	host: server_2 service: app 6	2.762	2.45	↑ 1.1x higher	response_requ ests by app	⚙



# Prévisions

- Les prévisions sont visibles dans le Single Metric Viewer
- La ligne jaune dans le graphique représente les valeurs de données prédites.
- La zone jaune ombrée représente les limites des valeurs prédites, ce qui donne également une indication de la confiance des prédictions.  
Les limites augmentent généralement avec le temps (i.e. les niveaux de confiance diminuent).
- Si les niveaux de confiance sont trop bas, la prévision s'arrête

# Prévisions



# Jobs de détection d'anomalies

## **Détection de changement**

API, Détecteurs et fonctions

Analyse de cause et jobs multiples

Alertes

Prévisions

# Introduction

- La surveillance IT consiste à détecter les changements anormaux dans les métriques surveillés.
- Cependant, de nombreux cas d'utilisation importants tournent autour de l'idée de détection de changement d'événement :
  - Des messages d'erreur qui apparaissent soudainement dans un fichier journal
  - Une baisse soudaine du nombre de commandes traitées par un système en ligne
  - Un nombre excessif soudain de tentatives d'accès à quelque chose (authentification par brute force)
  - ...

# Taux d'occurrence

- Ces changements sont caractérisés par un taux d'occurrence anormale
- EL-ML offre plusieurs moyens pour détecter ses changements :
  - Fonctions de comptage d'événements
  - Comptage en analyse de population
  - Détecter des choses qui se produisent « rarement »
  - Grouper des messages de traces similaires et compter les catégories trouvées

# Fonctions de comptage

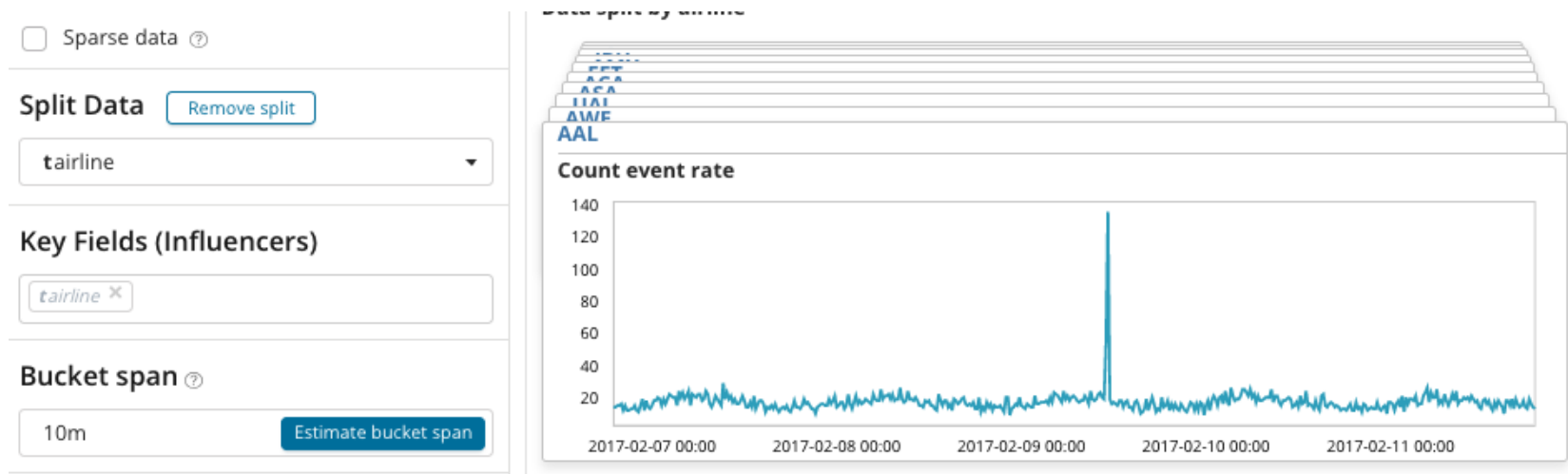
- EL-ML propose 3 fonctions de comptage de documents :
  - ***count*** : compte le nombre de documents dans le bucket-span résultant d'une interrogation de l'index de données brutes.
  - ***High count*** : Identique à *count*, mais ne signalera une anomalie que si le nombre est plus élevé que prévu
  - ***Low count*** : Identique à *count*, mais ne signalera une anomalie que si le nombre est inférieur à celui attendu
- Il est quelquefois plus avantageux d'utiliser 2 détecteurs *low*, *high* plutôt que le *count*. Surtout si le nombre d'anomalies high et low sont très différents.

# Comptage résumé

- Si un des champs de l'index contient déjà le comptage d'une occurrence, il est possible via un *advanced job* de choisir une directive ***summary\_count\_field\_name***
- Dans ce cas EL-ML ne compte pas les documents mais utilise ce champ pour trouver le nombre d'occurrence et appliquer les fonctions *count*, *low\_count* ou *high\_count*

# Catégorisation

- Les fonctions de comptage peuvent être associées à un champ de catégorisation et ainsi les comptages sont effectués pour chaque valeur du champ de catégorie





# Non-zero

- Les fonctions de comptage ***non\_zero*** (*non\_zero\_count*, *low\_non\_zero\_count* et *high\_non\_zero\_count*) permettent de prendre en compte les cas où les données sont clairsemées et que l'absence de données ne doit pas être prise en compte
- 4,3,0,0,2,0,5,3,2,0,2,0,0,1,0,4  
=>  
4,3,2,5,3,2,2,1,4

# Comptage distinct

- Les fonctions de **comptage distincts** (*distinct\_count*, *low\_distinct\_count* et *high\_distinct\_count*) mesure l'unicité (cardinalité) des valeurs d'un champ particulier.
- Utilisation possible : dans un contexte d'analyse de population, découvrir des individus qui ont un ensemble trop diversifié de valeurs de champ
  - Nombre d'adresse IP impliquées lors d'un balayage des ports par un malware

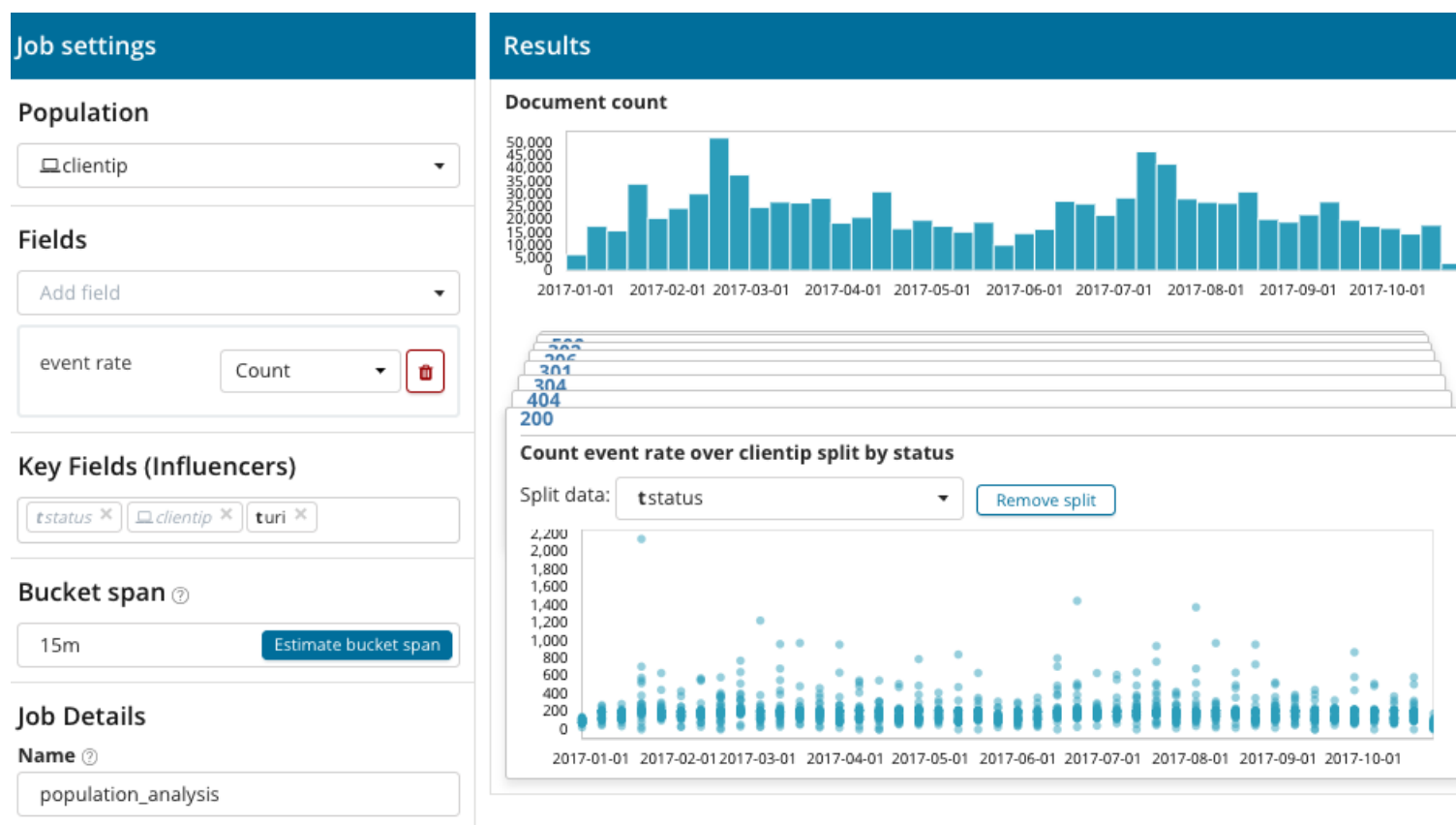
# Analyse de population

- Des entités ou des événements peuvent être considérés comme anormaux lorsque :
  - Leur comportement change au fil du temps, par rapport à leur propre comportement antérieur, ou
  - Leur comportement est différent de celui des autres entités d'une population spécifiée.C'est l'analyse de population
- Kibana propose un assistant dédié à l'analyse de population

# Comptage et analyse de population

- Comptabiliser les événements au travers d'une population a de nombreux cas d'utilisation :
  - Trouver des serveurs qui génèrent beaucoup plus de traces que leurs pairs
  - Un utilisateur qui effectue beaucoup plus de requêtes que les autres
  - ....

# Exemple Configuration



# API

```
PUT _ml/anomaly_detectors/population
{
  "description" : "Population analysis",
  "analysis_config" : {
    "bucket_span": "15m",
    "influencers": [
      "clientip"
    ],
    "detectors": [
      {
        "function": "mean",
        "field_name": "bytes",
        "over_field_name": "clientip"
      }
    ]
  },
  "data_description" : {
    "time_field": "timestamp",
    "time_format": "epoch_ms"
  }
}
```

# Fonction rare

- La notion de rareté tient compte du contexte :
  - S'il y a beaucoup de choses uniques, alors rien n'est rare.
  - S'il y a beaucoup de choses identiques et quelques choses uniques, alors elles sont rares.
- La fonction **rare** s'applique sur un champ et permet de détecter les valeurs rares d'un champ. Par exemple :
  - Un message de log rare
  - Un process s'exécutant rarement
  - Des destinations de connexion rares

# Fonctions rare

- Les fonctions *rare* détectent des valeurs qui arrivent rarement dans le temps ou par rapport à une population
- 2 fonctions existent :
  - ***rare*** : détecte les anomalies selon le nombre de valeurs rares distinctes
  - ***freq\_rare*** : détecte les anomalies en fonction du nombre de fois (fréquence) qu'apparaissent de valeurs rares



# Kibana

- *Kibana* propose désormais un assistant pour créer ce type de job

## Create job: Rare

Using index pattern logstash-apache\*



Time range

2

Pick fields

3

Job details

### Pick fields

#### Rare detector

##### Rare

Find rare values over time.

✓ Selected

##### Rare in population

Find members of a population that have rare values over time.

Select

##### Frequently rare in population

Find members of a population that frequently have rare values.

Select

#### Rare field

Select a field in which to detect rare values.

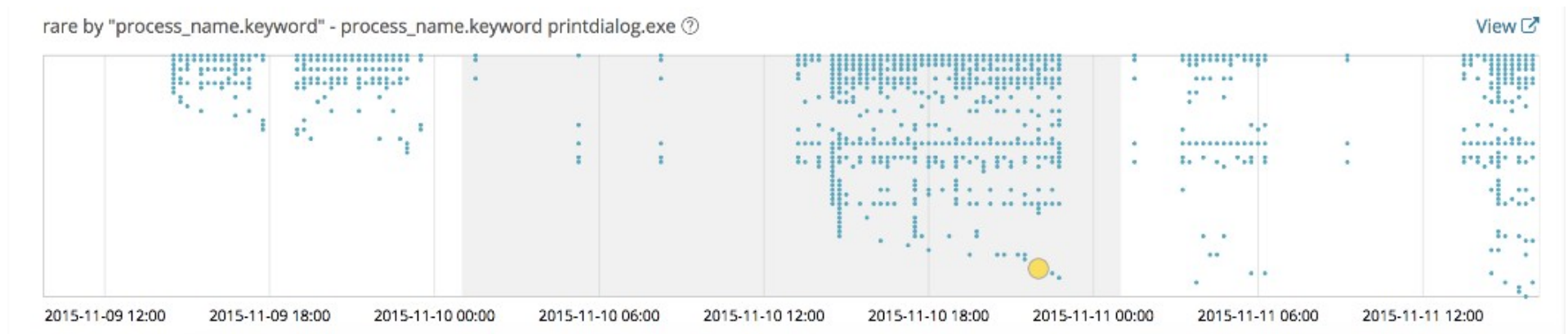
Rare field

< Previous

Next >

# Vue « Rareté »

- Les points bleus indiquent les taux d'occurrence des valeurs de champ dans le temps.
- Ceux qui se retrouvent près du bas sont les plus rares et l'anomalie sélectionnée est affichée sous forme d'un point agrandi



# Catégorisation de messages

- EL-ML permet de retrouver des messages de logs similaires et ainsi de les catégoriser.
  - Les messages de log doivent être générés par une machine (Pas de texte libre saisi par un humain)
- EL-ML utilise un algorithme de similarité de chaîne :
  - Se concentre sur les mots du dictionnaire (anglais) et non pas les chaînes variables (serveur, adresse, etc..)
  - Utilise un algorithme proche de celui de Levenshtein pour calculer une distance entre 2 messages
    - Si la distance est petite, positionne les 2 messages dans la même catégorie
    - Sinon, crée une nouvelle catégorie

# Exemple

Error writing file "foo" on host "acme6"

Error writing file "bar" on host "acme5"

Opening database on host "acme7"

- Dans ce cas EL-ML trouve 2 catégories, il compte alors le nombre d'occurrence pour chaque catégorie qu'il nomme *mlcategory* *N* :

mlcategory 1: 2

mlcategory 2: 1

# Usage

- Le modèle d'apprentissage automatique apprend quel volume et quel motif est normal pour chaque catégorie avec le temps.  
=> On peut donc détecter des anomalies et des événements rares, des types de messages inhabituels en utilisant des fonctions de comptage ou rare.

# Kibana

- Kibana propose également un assistant

## Pick fields

### Categorization detector

#### Count

Look for anomalies in the event rate of a particular category.

✓ Selected

#### Rare

Look for categories that occur rarely in time.

Select

### Categorization field

Specifies which field will be categorized. Using text data types is recommended. Categorization works best on machine written log messages, typically logging written by a developer for the purpose of system troubleshooting.

### Categorization field

# Jobs ML

Détection de changement

**API**

Analyse de cause et jobs multiples

Alertes

Prévisions

# API

- ELK-ML propose une API avec la base ***\_ml***
  - ***/anomaly\_detectors/***: Création et gestion des MLmachine jobs
  - ***/calendars/***: Création et gestion des calendriers
  - ***/datafeeds/***: Selection des données à analyser
  - ***/filters/***: Création et gestion des filtres utilisés par les règles personnalisées
  - ***/results/***: Accès aux résultats des jobs
  - ***/model\_snapshots/***: Gestion des instantanés de mémoire



# Objets détecteur

- La configuration d'un détecteur spécifie les champs de données analysés et les fonctions analytiques utilisées.
- Plusieurs détecteurs peuvent être configurés pour un job. Chaque détecteur a les propriétés suivantes :
  - ***detector\_description*** : Une description
  - ***detector\_index*** : Un indice pour le détecteur
  - ***field\_name*** : Le champ utilisé par la fonction. Pas applicable pour les calculs d'occurrence (count ou rare)
  - ***function*** : La fonction analytique
  - ***by\_field\_name*** : Le champ utilisé pour séparer les données
  - ***over\_field\_name*** : Pour une analyse de population
  - ***partition\_field\_name*** : Champ pour segmenter l'analyse. Les analyses sont alors indépendantes
  - ***use\_null*** : Définit si une série est créée pour les valeurs nulles de *by\_field\_name* ou *partition\_field\_name*
  - ***exclude\_frequent*** : *all*, *none*, *by*, ou *over*. Si défini, les entités fréquentes (en temps ou par population) sont exclues pour le calcul d'anomalie. On peut distinguer les champs *by* ou *over*
  - ***custom\_rules*** : Règles personnalisant le fonctionnement du détecteur. Permet par exemple d'indiquer de ne pas prendre en compte certains résultats

# *by\_field\_name* versus *partition\_field\_name*

- La configuration de ses 2 champs divise les données et sépare les affichages
- Ils peuvent être utilisés séparément ou ensemble dans un détecteur
  - Si on veut séparer complètement les analyses, utiliser “*partition\_field\_name*”  
=> Le calcul du score des anomalies est **indépendant**
  - Pour une séparation « douce », utiliser “*by\_field\_name*”  
=> Le calcul du score prend en compte les autres valeurs du champ

# Fonctions

- La plupart des fonctions détectent des anomalies dans les valeurs basses et hautes. Dans la terminologie statistique, ils appliquent un test *bilatéral*.  
Certaines fonctions ne font un test que d'un côté (*high\_count*, *low\_count*)
- Si les données sont éparées, les buckets vides peuvent être ignorés.

# Fonctions de comptage

- ***count, high\_count, low\_count*** :  
supportent *by\_field\_name* , *over\_field\_name* et *partition\_field\_name*
- ***non\_zero\_count, high\_non\_zero\_count, low\_non\_zero\_count*** :  
supportent *by\_field\_name* et *partition\_field\_name*
- ***distinct\_count, high\_distinct\_count, low\_distinct\_count*** :  
nécessitent *field\_name*  
et supportent *by\_field\_name*, *over\_field\_name*, et *partition\_field\_name*

# Exemple API

```
PUT _ml/anomaly_detectors/example2
{
  "analysis_config": {
    "detectors": [{
      "function" : "high_count",
      "by_field_name" : "error_code",
      "over_field_name": "user"
    }]
  },
  "data_description": {
    "time_field": "timestamp",
    "time_format": "epoch_ms"
  }
}
```

# Fonctions géographiques

- Une seule fonction ***lat\_long*** qui détecte des anomalies sur la position géographique des données  
Nécessite : *field\_name*  
Supporte : *by\_field\_name*, *over\_field\_name* et *partition\_field\_name*
- *field\_name* : chaîne contenant 2 nombres séparés par des virgules (latitude et longitude).  
=> il faut alors utiliser un script pour adapter les données au format *geo\_point*

# Example API

```
PUT _ml/datafeeds/datafeed-test2
{
  "job_id": "farequote",
  "indices": ["farequote"],
  "query": {
    "match_all": {
      "boost": 1
    }
  },
  "script_fields": {
    "lat-lon": {
      "script": {
        "source": "doc['coords'].lat + ',' + doc['coords'].lon",
        "lang": "painless"
      }
    }
  }
}
```

# Fonction sur le contenu

- ***info\_content, high\_info\_content, low\_info\_content***  
détecter les anomalies dans la quantité d'informations  
contenues dans les string  
Nécessitent : *field\_name*  
Supportent : *by\_field\_name, over\_field\_name* et  
*partition\_field\_name*
- Peuvent être utilisées pour identifier les exfiltrations de données :
  - Détecter des interrogations DNS suspectes
  - Détecter des changements de volume dans les messages de traces



# Exemple API

## Activité DNS suspecte

```
PUT _ml/anomaly_detectors/example_infocontent
{
  "analysis_config": {
    "detectors": [{
      "function" : "high_info_content",
      "field_name" : "query",
      "over_field_name": "src_ip"
    }]
  },
  "data_description": {
    "time_field": "timestamp",
    "time_format": "epoch_ms"
  }
}
```

# Fonctions métriques

- Toutes ces fonctions nécessitent : *field\_name* et supportent : *by\_field\_name*, *over\_field\_name* et *partition\_field\_name*
- ***min, max*** : Anomalie dans les valeurs min ou max calculées pour chaque bucket
- ***median, high\_median, low\_median*** : Anomalie dans les valeurs médianes calculées pour chaque bucket
- ***mean, high\_mean, low\_mean*** : Anomalie dans les valeurs moyennes calculées pour chaque bucket
- ***metric*** : Combine min, max et mean
- ***varp, high\_varp, low\_varp*** : Anomalies dans la variance d'une valeur (mesure de la variabilité et de la dispersion des données).

# Exemple API

Min, max et moyenne des temps de réponse des applications

PUT \_ml/anomaly\_detectors/example\_infocontent

```
{
  "analysis_config": {
    "detectors": [{
      "function" : "metric",
      "field_name" : "responsetime",
      "by_field_name" : "application"
    }]
  },
  "data_description": {
    "time_field": "timestamp",
    "time_format": "epoch_ms"
  }
}
```

# Fonctions somme

- Ces fonctions détectent les anomalies lorsque la somme d'un champ dans un bucket est anormale.
- ***sum, high\_sum, low\_sum*** :  
nécessitent *field\_name*  
supportent *by\_field\_name*, *over\_field\_name* et *partition\_field\_name*
- ***non\_null\_sum, high\_non\_null\_sum, low\_non\_null\_sum*** :  
nécessitent *field\_name*  
supportent *by\_field\_name* et *partition\_field\_name*

# Exemple API

Pour chaque centre de coût, employé suspect ayant effectué beaucoup de dépense

```
PUT _ml/anomaly_detectors/example_infocontent
{
  "analysis_config": {
    "detectors": [{
      "function" : "sum",
      "field_name" : "expenses",
      "by_field_name" : "costcenter",
      "over_field_name" : "employee"    }]
  },
  "data_description": {
    "time_field": "timestamp",
    "time_format": "epoch_ms"
  }
}
```

# Fonction rare

- Ces fonctions 2 fonctions pour détecter une rareté :
  - **rare** : Détecte les valeurs qui surviennent rarement dans le temps ou rarement pour une population. Anomalies détectées en fonction du nombre de valeurs rares distinctes.  
=> Les individus suspects sont les individus qui ont de nombreuses valeurs rares  
nécessite *by\_field\_name*  
supporte *over\_field\_name* et *partition\_field\_name*
  - **freq\_rare** : Pour une population, elle détecte les anomalies en fonction du nombre de fois (fréquence) où des valeurs rares se produisent.  
nécessite *by\_field\_name* et *over\_field\_name*  
supporte *partition\_field\_name*

# Exemple API

- Status code qui est arrivé rarement par le passé

```
{  
  "function" : "rare",  
  "by_field_name" : "status"  
}
```

- IPs qui ont de nombreux différents codes statut rares

```
{  
  "function" : "rare",  
  "by_field_name" : "status",  
  "over_field_name" : "clientip"  
}
```

- IPs qui ont beaucoup d'URI rares par rapport aux autres

```
{  
  "function" : "freq_rare",  
  "by_field_name" : "uri",  
  "over_field_name" : "clientip"  
}
```

# Fonctions horaires

- Les fonctions horaires détectent les événements qui se produisent à des heures inhabituelles, du jour ou de la semaine.
- 2 fonctions sont proposées :
  - ***time\_of\_day*** : Cette fonction s'attend que le comportement journalier soit similaire. Détecte lorsqu'un événement se produit à un moment inhabituel de la journée
  - ***time\_of\_week*** : Détecte lorsqu'un événement se produit à un moment inhabituel de la semaine



# Règles personnalisées

- Les règles personnalisées décrivent quand un détecteur doit effectuer une certaine action plutôt que de suivre son comportement par défaut.
- Elles sont configurées via des **scopes** (portée) qui ont chacun des **conditions**
- Elles peuvent s'appuyer sur des objets **filter** stockés au préalable dans EL, qui liste des valeurs à inclure ou exclure

# Structure d'une règle personnalisée

- **scope** (optionnel) : Par défaut, la portée inclut toutes les séries. La portée peut être spécifiée vis à vis des champs *by\_field\_name*, *over\_field\_name* ou *partition\_field\_name*. Le champ est indiqué comme clé d'un possédant les propriétés suivantes :
  - **filter\_id** : L'id du filtre à utiliser
  - **filter\_type** : include ou exclude
- **actions** [ ] : L'ensemble des actions à déclencher lorsque la règle s'applique :
  - **skip\_result** : Le résultat ne sera pas créé. Ceci est la valeur par défaut. Le modèle sera mis à jour.
  - **skip\_model\_update** : La valeur de cette série ne sera pas utilisée pour mettre à jour le modèle. Les résultats seront créés comme d'habitude.
- **conditions** ([ ] optionnel) : Plusieurs conditions sont combinées avec un ET logique.
  - **applies\_to** : Spécifie la propriété de résultat à laquelle la condition s'applique. Les options disponibles sont *actual*, *typical*, *diff\_from\_typical*, *time*.
  - **operator** : Spécifie l'opérateur de condition. Les options disponibles sont *gt*, *gte*, *lt* et *lte*
  - **value** : La valeur comparée au champ *apply\_to* à l'aide de l'opérateur

# Exemple

- Création d'un filtre

```
PUT _ml/filters/safe_domains
```

```
{  
  "description": "Our list of safe domains",  
  "items": ["safe.com", "trusted.com"]  
}
```

# Exemple

- Création d'un détecteur avec une règle personnalisée utilisant un filtre

PUT \_ml/anomaly\_detectors/dns\_exfiltration\_with\_rule

```
{
  "analysis_config" : {
    "bucket_span": "5m",
    "detectors" : [{
      "function": "high_info_content",
      "field_name": "subdomain",
      "over_field_name": "highest_registered_domain",
      "custom_rules": [{
        "actions": ["skip_result"],
        "scope": {
          "highest_registered_domain": {
            "filter_id": "safe_domains",
            "filter_type": "include"
          }
        }
      }]
    }]
  },
  "data_description" : {
    "time_field": "timestamp"
  }
}
```

# Exemple avec conditions

```
PUT _ml/anomaly_detectors/rule_with_range
{
  "analysis_config" : {
    "bucket_span": "5m",
    "detectors" : [{
      "function": "count",
      "custom_rules": [{
        "actions": ["skip_result"],
        "conditions": [
          {
            "applies_to": "actual",
            "operator": "gt",
            "value": 30
          },
          {
            "applies_to": "actual",
            "operator": "lt",
            "value": 50
          }
        ]
      }]
    }]
  },
  "data_description" : {
    "time_field": "timestamp"
  }
}
```

# Jobs ML

Détection de changement

API, Détecteurs et fonctions

**Analyse de cause et jobs multiples**

Alertes

Prévisions

# Introduction

- Pour mieux comprendre toute la portée d'un problème, il est souvent utile d'examiner de nombreux aspects d'un système. Cela nécessite une analyse intelligente de plusieurs types de jeux de données liés.
  - Compréhension des rôles et de l'importance des indicateurs de performance clés (KPI). Détecter les anomalies
  - Méthodes d'apprentissage pour organiser, filtrer et enrichir les données afin de fournir un contexte à l'anomalie
  - Exploiter cette information contextuelle via des jobs ML utilisant le fractionnement de données et des influenceurs statistiques
  - Combiner les anomalies créées par ML dans une vue commune permettant une meilleure résolution

# KPI

- Il est naturel de se concentrer sur les KPI pour exploiter le volume des données collectées.
- Ces KPI peuvent être aussi diverses que :
  - Axé utilisateur : Temps de réponse, Nombre d'erreurs
  - Opérations : Disponibilité, délai moyen de réparation
  - Métier : Transactions à la minute, CA/Bénéfices , Nombre d'utilisateurs actifs
- Ces KPIs sont souvent présentés dans des tableaux de bord (Kibana)



# Limitations des Tableaux de bord de KPI

- L'inspection manuelle de ces tableaux de bord a quand même des limitations :
  - Interprétation: difficulté à comprendre la différence entre fonctionnement normal et anormal, à moins que cette différence ne soit déjà comprise intrinsèquement.
  - Défis d'échelle: Le nombre de KPIs peut être nombreux résultant dans des tableaux de bord surchargés
  - Manque de proactivité: Les tableaux de bord n'ont pas leurs métriques liées aux alertes, ce qui nécessite une surveillance constante.

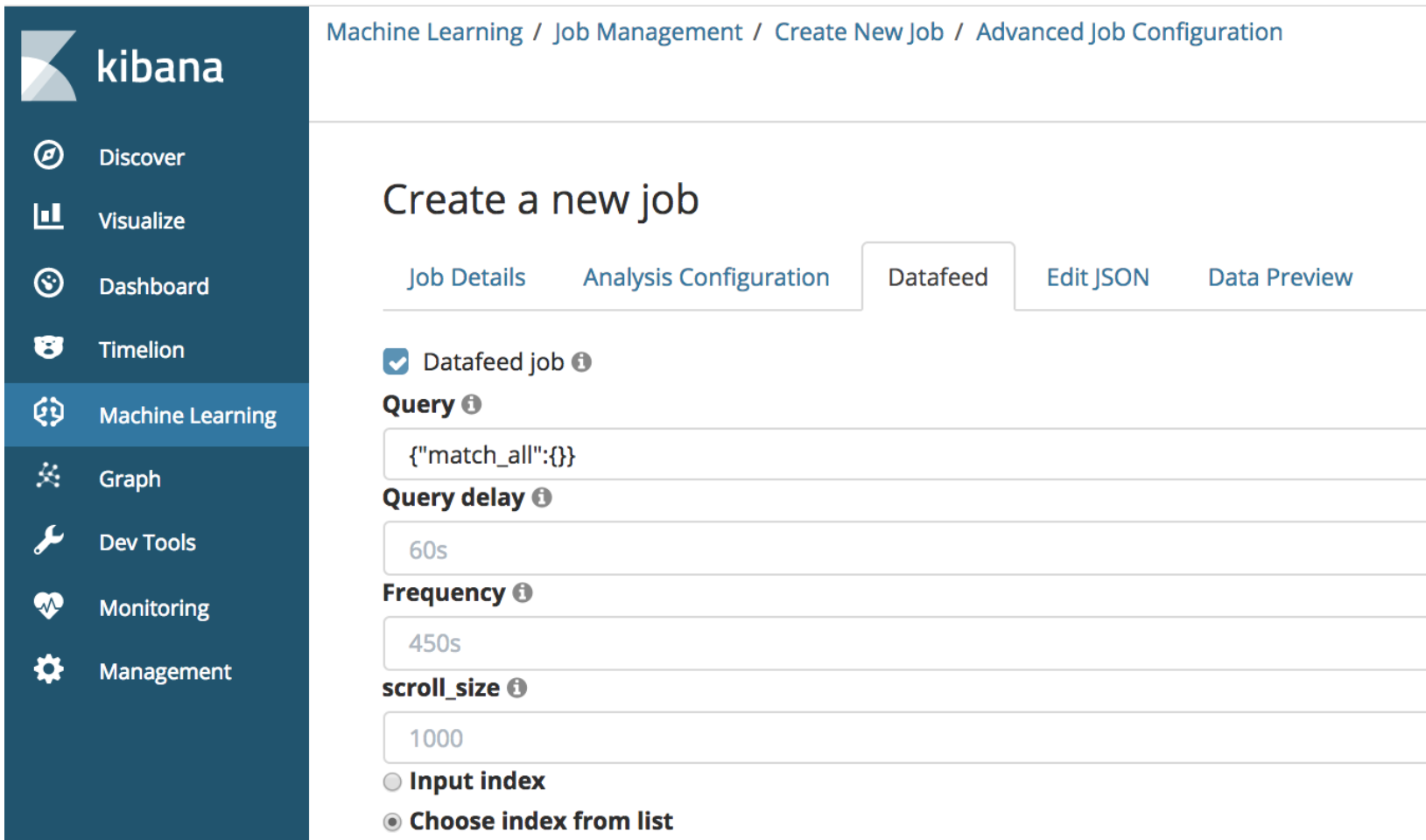
# KPI et Détection de changement

- Naturellement, les KPIs sont de bon candidats pour les métriques à suivre par EL-ML
- Toutefois, pour avoir une vision plus globale de ce qui peut contribuer à un problème opérationnel, l'analyse doit être élargie aux systèmes et aux technologies sous-jacents qui supportent l'application.  
=> Beats, APM
- Avant de pouvoir traiter efficacement toutes ces données, celles ci doivent être segmentées intelligemment (queries) et éventuellement enrichies (logstash).

# Segmentation des données

- La segmentation peut s'effectuer de 2 façons :
  - Le paramètre **query** de l'onglet *datafeed* permet de filtrer les données de l'index source  
Il supporte toute requête DSL de Elastic Search
  - Les requêtes enregistrées via Kibana peuvent également être utilisées pour filtrer les données de l'index

# Paramètre query



The screenshot shows the Kibana interface for creating a new machine learning job. The left sidebar contains navigation links: Discover, Visualize, Dashboard, Timelion, Machine Learning (selected), Graph, Dev Tools, Monitoring, and Management. The main content area has a breadcrumb trail: Machine Learning / Job Management / Create New Job / Advanced Job Configuration. Below this is the title 'Create a new job' and a tabbed interface with four tabs: Job Details, Analysis Configuration, Datafeed (selected), Edit JSON, and Data Preview. The 'Datafeed job' checkbox is checked. The configuration fields are: Query (set to {"match\_all":{}}), Query delay (60s), Frequency (450s), scroll\_size (1000), and Index selection (radio buttons for 'Input index' and 'Choose index from list', with the latter selected).

kibana

- Discover
- Visualize
- Dashboard
- Timelion
- Machine Learning**
- Graph
- Dev Tools
- Monitoring
- Management

Machine Learning / Job Management / Create New Job / Advanced Job Configuration

## Create a new job

Job Details Analysis Configuration **Datafeed** Edit JSON Data Preview

☒ Datafeed job ⓘ

**Query** ⓘ

`{"match_all":{}}`

**Query delay** ⓘ

60s

**Frequency** ⓘ

450s

**scroll\_size** ⓘ

1000

☐ Input index

☒ Choose index from list

# Filtres Kibana

- Utiliser une recherche enregistrée dans Kibana

The screenshot shows the Kibana interface with the 'Add filter' dialog box open. The dialog has a title bar 'Add filter' with a close button. Inside, there's a 'Filter' section with a dropdown menu showing 'beat.name', a relationship dropdown showing 'is one of', and a list of filters to choose from. The filter 'site-search-es-3' is selected and highlighted in blue. Other filters in the list include 'site-search-es-9', 'site-search-es-7', 'site-search-es-8', 'site-search-es-4', 'site-search-es-11', 'site-search-es-10', and 'site-search-es-6'. There are also two existing filters at the top: 'site-search-es-1' and 'site-search-es-2'. A link 'Edit Query DSL' is visible in the top right of the dialog. The background shows the Kibana sidebar with options like Discover, Visualize, Dashboard, Timelion, Machine Learning, Graph, Dev Tools, Monitoring, and Management. The top of the interface shows '72,223,884 hits' and a search bar with the text 'Search... (e.g. status:200 AND extension:PHP)'.

**kibana**

72,223,884 hits

Search... (e.g. status:200 AND extension:PHP)

Add a filter +

Add filter

Filter

beat.name is one of

site-search-es-1 site-search-es-2

Label

Optional

Edit Query DSL

site-search-es-9

**site-search-es-3**

site-search-es-7

site-search-es-8

site-search-es-4

site-search-es-11

site-search-es-10

site-search-es-6

# \_score

t \_type

# Enrichissement des données

- Typiquement via *logstash* et ses filtres, on ajoute des données qui pourront être utilisées :
  - Comme filtre
  - Comme catégorisation
  - Comme influenceur

# Tirer parti des informations contextuelles

- Une fois les données efficacement organisées et enrichies, deux méthodes principales sont disponibles pour en tirer parti :
  - Diviser l'analyse (split) en séparant les données pour identifier des modèles comportementaux distincts.  
Ex : Une région, un type de serveur, etc ...
  - Et les influenceurs.

# Influenceur clé

- Lors de la configuration, il est possible de définir des champs comme **influenceur** (ou champ clé)
  - Un influenceur est la valeur d'un champ que ML identifie comme responsable de l'existence de l'anomalie, ou du moins qui a eu une contribution significative.
- La recherche d'influenceurs potentiels se produit après que ML ait découvert l'anomalie :
  - Une fois l'anomalie trouvée, ML examine systématiquement toutes les valeurs de chaque champ d'influenceur candidat et supprime la contribution de ces valeurs aux données de cette période.
  - Si, une fois supprimées, les données restantes ne sont plus anormales, alors la contribution de cette valeur est marquée comme influente.



# Score de l'influenceur

- Un **score d'influence** est attribué à chaque valeur du champ influenceur  
Plus le score est élevé, plus cette entité aura contribué ou sera responsable des anomalies.
- Les scores sont calculés pour chaque détecteurs du job et sont rassemblés dans la même vue

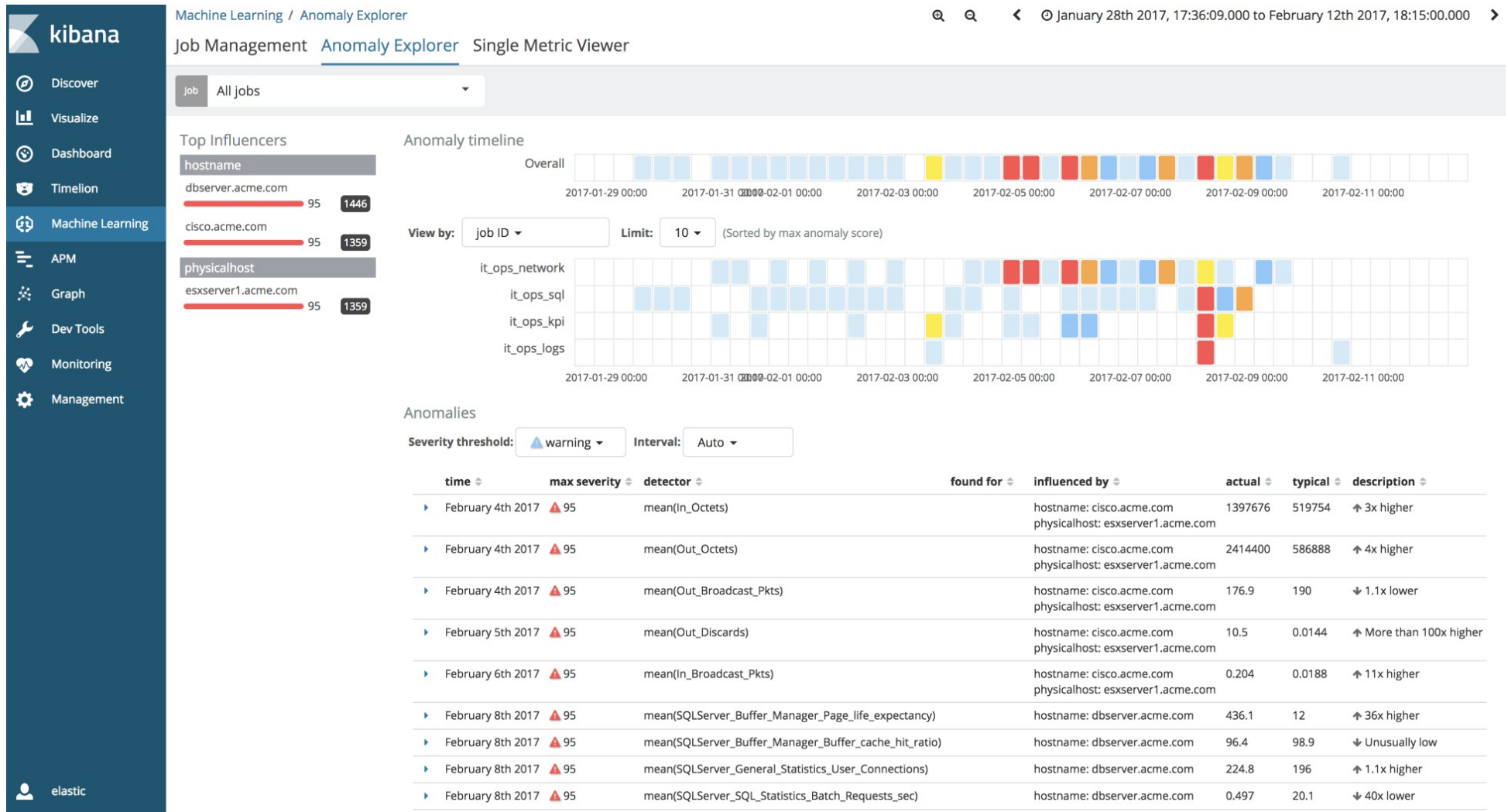
# Corrélation de données

- Pour analyser la cause d'une anomalie dans un KPI, il est souvent nécessaire de configurer d'autres jobs surveillant d'autres métriques
- Par exemple, pour un magasin on-line :
  - Comptage résumé (1 minute) du volume de transactions (le KPI)
  - Logs applicatifs du moteur de transaction (filebeat/logstash/Categorisation de message)
  - Mesures de performances SQL du Serveur de base de données associé au moteur de transactions (metricbeats avec module bas de données)
  - Mesures de performance d'utilisation du réseau (packetbeat)

# Corrélation de données

- L'explorateur d'anomalies permet de superposer différents jobs sur la même période de temps
  - Avec les couloirs temporels, il est facile d'identifier des corrélations entre les anomalies des jobs
- Les tops influenceurs sont également présentés, les chiffres correspondent à :
  - Le score max de l'influenceur dans un bucket
  - Le score total sur tous les buckets de la sélection temporelle
- Les influenceurs qui sont communs à tous les jobs ont leur score additionné  
=> ce qui les font monter dans la top-list

# Corrélation



# Jobs ML

Détection de changement  
API, Détecteurs et fonctions  
Analyse de cause et jobs multiples  
Alertes  
**Prévisions**

# Introduction

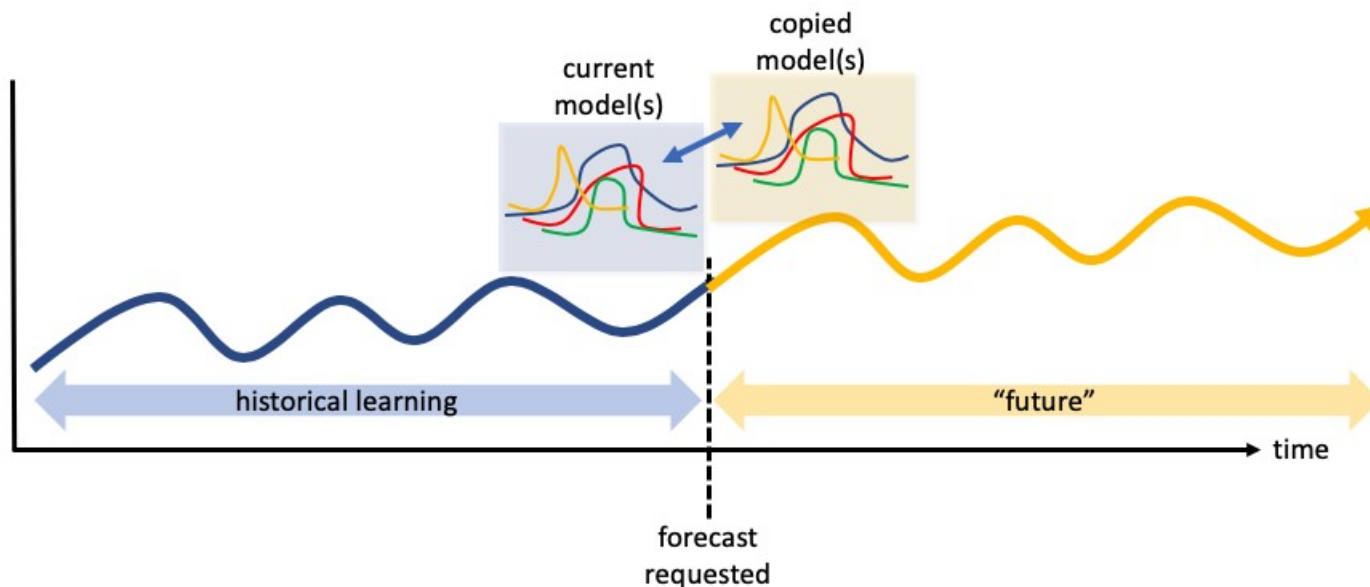
- La prévision est une extension naturelle de la modélisation comportementale d'ELK- ML. Il est donc possible de projeter ces informations dans le temps et de prédire le comportement futur.
- Mais attention, il n'est pas toujours possible de prédire une tendance si un facteur externe inconnu est en jeu (En IT, configuration manuelle incorrecte, matériel défaillant, etc.).
- On peut donc utiliser une analyse probabiliste pour donner la meilleure estimation de l'avenir, mis à part les facteurs externes possibles.

# Cas d'utilisation

- Avec EL-ML, il y a 2 cas d'utilisation des prévisions :
  - **Axé sur la valeur** : Extrapoler une série chronologique dans le futur pour comprendre une valeur future probable.  
Ex : "combien de widgets vais-je vendre par jour dans deux mois?"
  - **Axé sur le temps** : Obtenir une probabilité qu'une valeur atteigne un seuil à un temps donné  
Ex : "Est-ce que je compte atteindre une utilisation de 80% dans la semaine prochaine?"

# Processus séparés

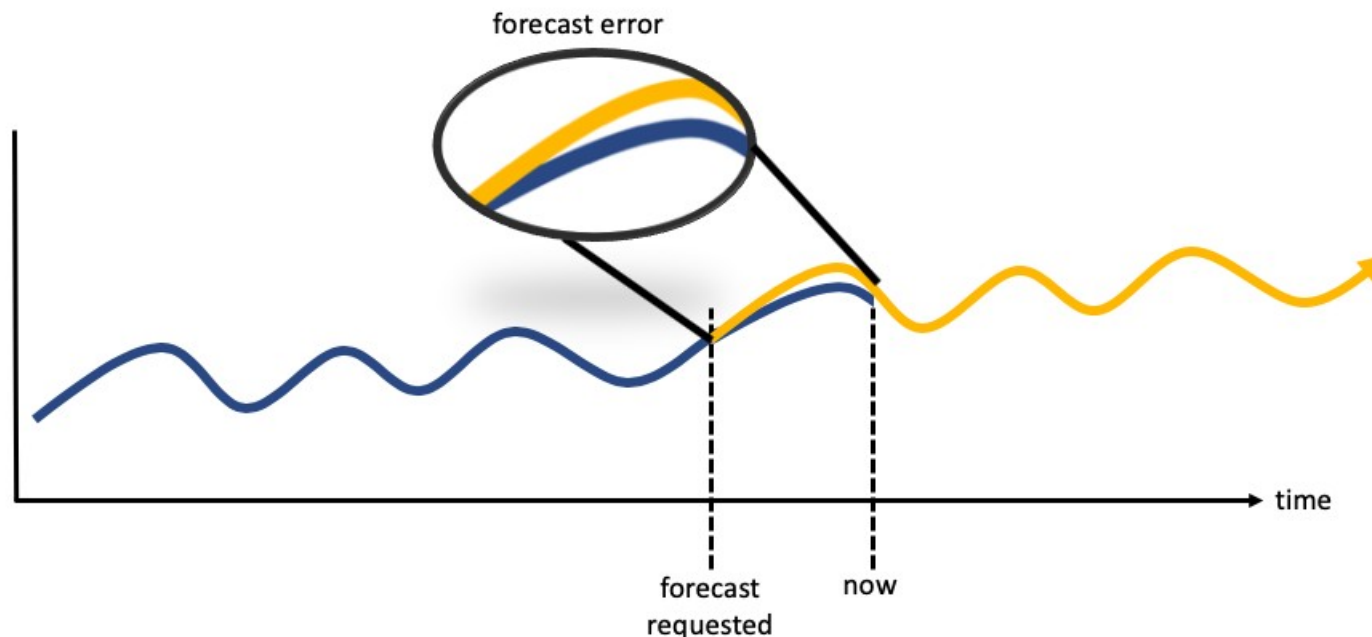
- Lorsqu'une prévision est demandée, une copie des modèles du job est créée et un processus séparé est utilisé pour extraire ces modèles et les extrapoler dans le "futur".
- Ce processus est exécuté en parallèle pour ne pas interférer avec les modèles originaux et leur évolution.





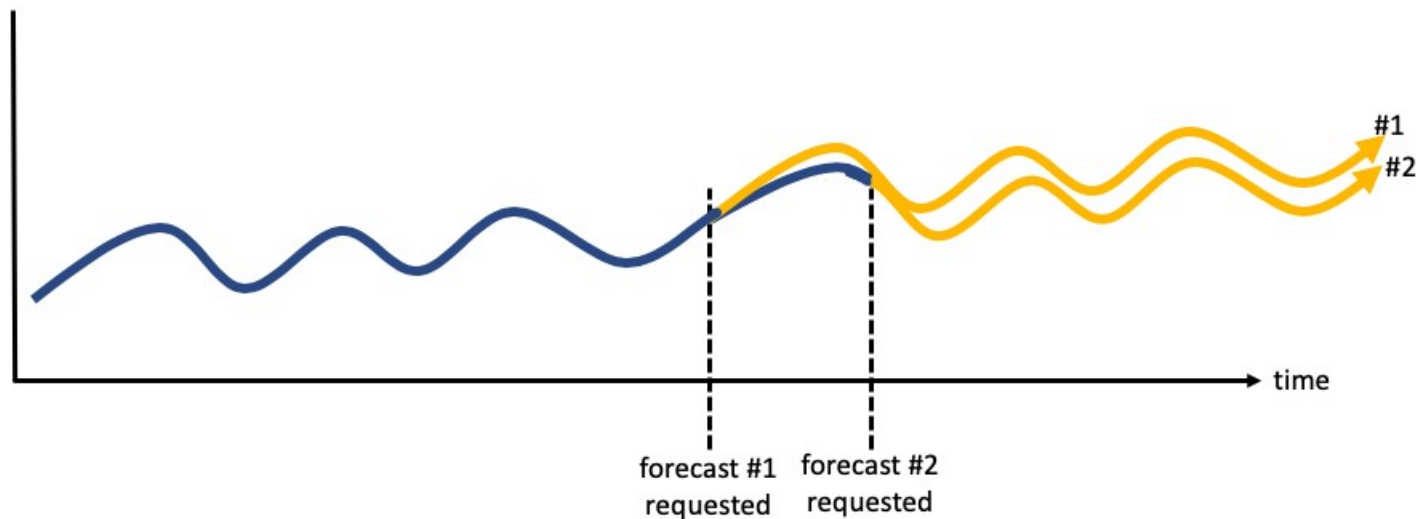
# Comparaison avec le réel

- Le job ML analysant les données réelles se poursuit (s'il fonctionne en temps réel) et qu'après un certain laps de temps, une différence peut apparaître entre la valeur prédite et la valeur réelle



# Plusieurs prévisions

- Plusieurs prévisions peuvent être demandées par l'utilisateur. Chaque prévision est stockée séparément.



# Exécution

- Indiquer une durée :
  - limité à 8 semaines pour le moment
  - Ne pas indiquer une durée plus grande que l'historique
  - L'historique doit avoir un minimum de 3 cycles de pattern périodiques
- Si le job est configuré pour afficher plusieurs séries temporelles, la prévision s'exécute pour tous les détecteurs et les partitions de données
- Les résultats ont une durée de vie par défaut de 14 jours. Après cela, les résultats sont définitivement supprimés.  
Il est possible d'indiquer un autre délai d'expiration via la ressource REST *\_forecast*,

# Single Metric

Machine Learning / Single Metric Viewer

Untitled Workpad - Kibana

Auto-refresh

January 31st 2017, 19:00:00.000 to March 1st 2017, 00:00:00.000

Job Management Anomaly Explorer **Single Metric Viewer** Data Visualizer Settings

Job a\_forecast\_example

Detector: sum(amount)

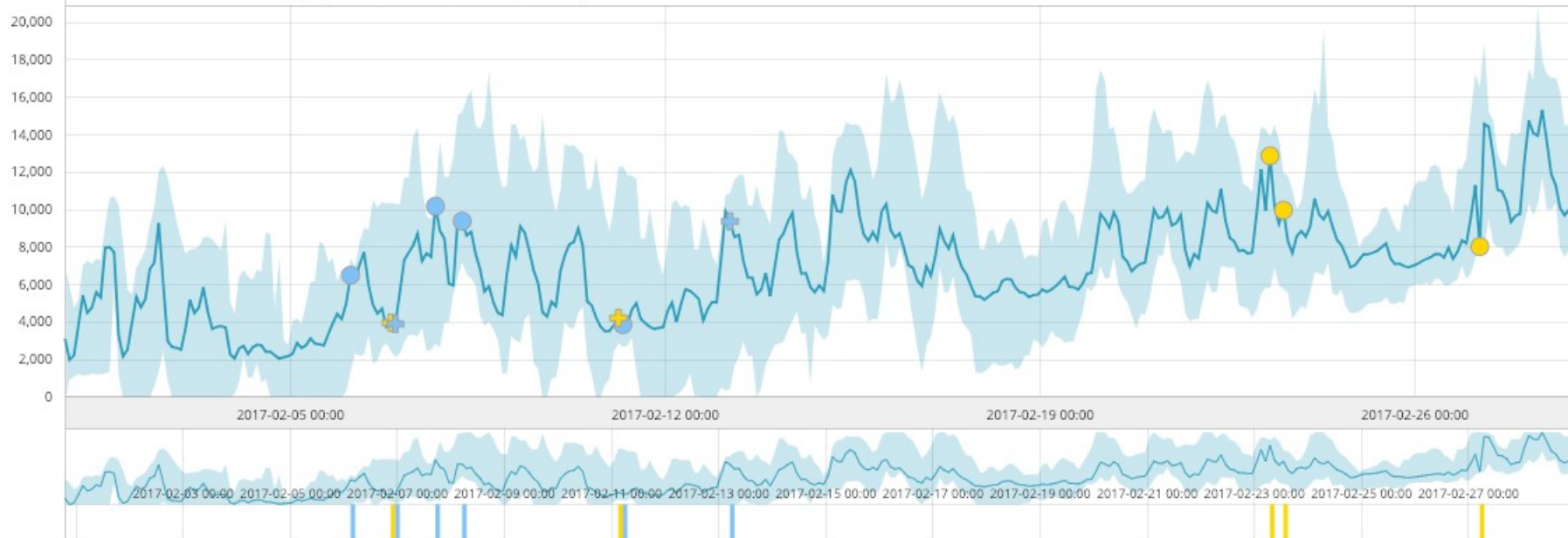


Forecast

Single time series analysis of sum amount

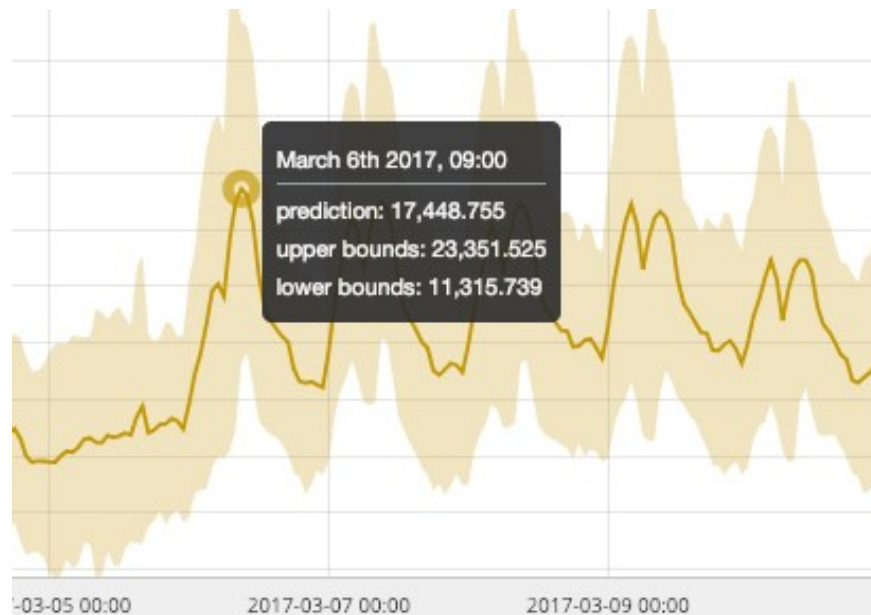
☒ show model bounds

Zoom: auto 12h 1d 1w 2w (aggregation interval: 2h, bucket span: 15m)



# Résultats de prévisions

- Une fenêtre contextuelle, disponible pour chaque point de données des résultats, affiche la valeur de prédiction, la limite supérieure et la valeur de limite inférieure. (intervalle de confiance du 95 e centile.)



# API

- Les résultats sont également accessible via l'API search

GET .ml-anomalies-\*/\_search

```
{
  "query": {
    "bool": {
      "filter": [
        {"term": {"timestamp": "1488808800000"}},
        {"term": {"result_type": "model_forecast"}},
        {"term": {"job_id": "a_forecast_example"}}
      ]
    }
  }
}
```

# Réponse

```
{
  ... "hits" : [
    {
      "_index" : ".ml-anomalies-shared",
      "_type" : "doc",
      "_id" :
        "a_forecast_example_model_forecast_i2DxbGgBITRq2rXM21p4_1488808800000_900_0_961_0",
      "_score" : 0.0,
      "_source" : {
        "job_id" : "a_forecast_example",
        "forecast_id" : "i2DxbGgBITRq2rXM21p4",
        "result_type" : "model_forecast",
        "bucket_span" : 900,
        "detector_index" : 0,
        "timestamp" : 1488808800000,
        "model_feature" : "'bucket sum by person'",
        "forecast_lower" : 11315.739312779506,
        "forecast_upper" : 23080.83486433322,
        "forecast_prediction" : 17198.287088556364
      }
    }
  ]
}
```

# Prévision axée sur le temps

- Pour une prévision axée sur le temps, il faut exécuter une query
- Par exemple : Query avec elastic-sql

```
POST /_xpack/sql?format=txt
```

```
{  
  "query": "SELECT timestamp FROM \".ml-anomalies-*\" WHERE  
    job_id='a_forecast_example' AND result_type='model_forecast' AND  
    forecast_prediction>'17700' ORDER BY timestamp DESC"  
}
```

- Réponse

```
timestamp
```

```
-----
```

```
2017-03-06T14:45:00.000Z
```



# Plusieurs séries temporelles

- Une prévision peut également être démarrée via l'API
- Par exemple :

POST

```
_xpack/ml/anomaly_detectors/web_traffic_per_country/_forecast  
{  
  "duration": "7d"  
}
```

Réponse :

```
{  
  "acknowledged" : true,  
  "forecast_id" : "DGT6bWgBITRq2rXMb1Rr"  
}
```

# Analyse de données

## **Introduction**

Détection de valeurs anormales

Régression

Classification

Inférence

# Introduction

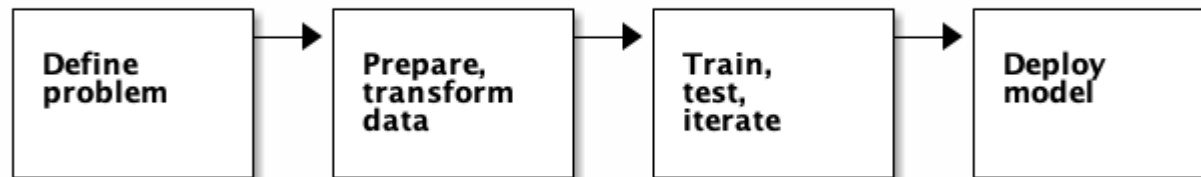
- Pour ce type d'analyse les données sources doivent être structurées sous la forme d'un tableau à 2 dimensions
- Un module de transformation est disponible dans la suite pour restructurer les données provenant d'index
- Permet différentes analyses et l'annotation des données avec les résultats

# Analyses

- Les différentes analyses disponibles :
  - Détection de valeurs anormales (non supervisés)
  - Prévisions sur des données
  - Classification des données
  - Inférence permet de confronter les modèles ML entraînés vis à vis des données réelles

# Apprentissage supervisé

- permet de créer un modèle ML en fournissant des exemples d'entraînement
- Le modèle peut être alors utilisé pour des prédictions
- Le workflow est alors :



# Définition du problème

- Quels types de modèles souhaitez-vous découvrir dans vos données ?
- Quel type de valeur voulez-vous prédire : une catégorie ou une valeur numérique ?
- ELK-ML propose 2 types d'analyse :
  - **regression**: prédit des valeurs **numériques continues**  
Ex : le temps de réponse d'une requête Web.
  - **classification** : prédit des valeurs **discrètes**  
Ex : Requête DNS malveillant ou normale.

# Préparation des données

- Pour les analyses supervisées, il faut fournir un jeu de données **étiqueté, important**
  - Par exemple, pour entraîner un modèle de classification qui décide si un e-mail est un spam ou non, on a besoin d'un ensemble de données étiqueté contenant d'exemple de bons emails et de spams.
- Les données doivent être structurées sous un format tabulaire

# Entraînement du modèle

- L'entraînement est un processus itératif : chaque itération est suivie d'une évaluation pour évaluer l'efficacité du modèle.
  - La première étape consiste à définir les champs pertinents dans l'ensemble de données
  - Ensuite, séparer les données en un ensemble pour entraîner et un ensemble pour évaluer le modèle
- Les données d'apprentissage sont transmises à l'algorithme d'apprentissage.

Le modèle prédit la valeur et la compare à la vérité terrain  
Puis le modèle est affiné pour rendre les prédictions plus précises.



# Déploiement du modèle

- Après avoir obtenu une bonne performance, le modèle est déployé et utilise les nouvelles données
- L'inférence permet de faire des prédictions sur les nouvelles données en utilisant :
  - un processeur dans une pipeline d'ingestion
  - Dans un processus de transformation continue (Roll-up d'index)
  - Une agrégation lors d'une recherche

# Jobs d'analyse

- Les jobs d'analyse sont constitués de 4 ou 5 phases :
  - **Réindexation** (API Reindex) : Copie des documents de l'index source vers un autre index (possibilité de changer le settings et le mapping)
  - **Chargement des données** : Chargement des données et conversions dans la structure demandée par l'analyse
  - **Analyse** : Une seule phase pour la détection de valeurs anormales, plusieurs phases pour la régression et la classification :
    - Identification des champs significatifs
    - Affinement de paramètres (les hyperparamètres)
    - Entraînement du modèle
  - **Écriture des résultats** : Les résultats issus de l'analyse sont réécrits dans l'index
  - **Inférence** : Pour la régression et la classification, validation du modèle sur les données d'évaluation

# Recommandations

- Démarrer petit et itérer rapidement
- Fournir un petit pourcentage de données pour l'apprentissage
- Désactiver le calcul de l'importance des champs (Feature importance) pour réduire le temps d'exécution de l'analyse
- Optimiser le nombre de champs inclus dans l'analyse
- Augmenter le nombre de threads (par défaut 1)
- Optimiser la taille de l'index source (par des filtres par exemple)
- Configurer manuellement les *hyperparamètres*

# Evaluation des résultats

- ELK fournit une API pour évaluer les résultats d'analyse `POST _ml/data_frame/_evaluate`
- Cela permet de comprendre les distributions d'erreurs et identifier les points où le modèle d'analyse de trame de données fonctionne bien ou de manière moins fiable
- Les métriques fournis sont :
  - La matrice de confusion
  - La précision
  - Le recall
  - La courbe caractéristique de fonctionnement du récepteur (ROC).

# Matrice de confusion

- Une matrice de confusion fournit quatre mesures :
  - Vrais positifs (TP) : membres de la classe que l'analyse a identifiés comme étant des membres de la classe.
  - Vrais négatifs (VN) : Les membres du groupe que l'analyse a correctement identifiés comme n'étant pas membres du groupe.
  - Faux positifs (FP) : Les membres que l'analyse a identifiés à tort comme des membres du groupe.
  - Faux négatifs (FN) : Les membres du groupe que l'analyse a identifiés à tort comme n'étant pas membres du groupe.
- Les résultats de l'analyse ne sont pas des valeurs exactes mais des probabilités. Il faut alors donner des seuils où le considère la valeur comme exacte
- L'API fournit par défaut des matrices de confusion prenant comme seuils 0,25 ; 0,5 et 0,75

# Précision et recall

- Les valeurs de **précision** et de **recall** résument les performances de l'algorithme sous la forme d'un nombre unique qui facilite la comparaison des résultats de l'évaluation
  - La précision indique combien de points que l'algorithme a identifiés comme membres de la classe étaient en fait des membres de la classe. C'est le nombre de vrais positifs divisé par la somme des vrais positifs et des faux positifs ( $TP/(TP+FP)$ ).
  - Recall répond à une question légèrement différente. Cette valeur indique combien de points de données qui sont des membres réels de la classe ont été correctement identifiés en tant que membres de la classe. C'est le nombre de vrais positifs divisé par la somme des vrais positifs et des faux négatifs ( $TP/(TP+FN)$ ).
- Les seuils s'appliquent également pour ces 2 mesures

# ROC

- ROC est un tracé qui représente les performances du processus de classification binaire à différents seuils.
- Il compare le taux de vrais positifs au taux de faux positifs aux différents niveaux de seuil pour créer la courbe.
- À partir du tracé, on peut calculer la valeur de l'aire sous la courbe (AUC), qui est un nombre compris entre 0 et 1.  
Plus la valeur est proche de 1, meilleures sont les performances de l'algorithme.

# Analyse de trame de données

Introduction

**Détection de valeurs anormales**

Régression

Classification

Inférence



# Introduction

- La détection des valeurs aberrantes est une analyse permettant d'identifier les points de données (valeurs aberrantes) dont les valeurs de caractéristiques sont différentes de celles des points de données normaux.
- Les valeurs aberrantes peuvent indiquer des erreurs ou un comportement inhabituel.

# Algorithme

- L'algorithme calcule 4 valeurs pour déterminer si une valeur est aberrante :
  - La distance au Kème voisin
  - La distance des K plus proches voisins
  - Lof : Calcule sur la densité, (les voisins proche ont ils également des voisins)
  - Ldof : Ratio, également sur la densité
- Ces 4 valeurs détecte des valeurs aberrantes qui peuvent être différentes. ELK-ML, agrège et normalise ces données et fournit alors une probabilité (entre 0 et 1) qu'une valeur soit aberrante.

# Feature Influence

- Une autre valeur est calculée durant l'importance : le facteur d'influence d'une feature
- Pour les champs de l'index, ELK-ML évalue l'influence du champ sur le résultat

# Analyse de trame de données

Introduction

Détection de valeurs anormales

**Régression**

Classification

Inférence

# Introduction

- But : Estimer les relations entre les différents champs de vos données, puis faire d'autres prédictions basées sur ces relations
  - Par exemple, prédire la valeur d'achat d'un appartement à partir de sa superficie, sa localisation, son étage, etc.
-

# Feature variables

- La première étape consiste donc à identifier les champs de notre index qui serviront au modèle pour prédire la valeur d'un autre champ
- Les types supportés sont :
  - Numérique
  - Catégorie : Ensemble fixe de valeurs
  - Booléen

# Apprentissage et évaluation du modèle

- Il faut fournir des ensemble de données étiquetés qui contiennent les feature variables et la variable dépendante
- La régression consiste à identifier une fonction mathématique qui permet de calculer la variable dépendante en fonction des variables de feature  
Algorithme : *eXtreme Gradient boost*
- Une fonction de perte mesure ensuite l'efficacité du modèle. ELK-ML propose différents algorithmes :
  - *mse* : Mean Squarred Error
  - *msle* : Mean Squarred Logarithmique
  - *hub* : Essaie d'écarter les valeurs aberrantes
- Le modèle créé est stocké dans des index internes d'Elasticsearch
- Des résultats d'impacts des champs (Feature importance) sont également évalués

# Mesure de la performance du modèle

- On peut mesurer l'efficacité du modèle via l'API `POST _ml/data_frame/_evaluate` {  
    `"evaluation" : "regression"`  
}
- On peut obtenir le `mse`
- L'autre mesure cruciale, c'est de voir comment le modèle sur les données de test. La métrique remontée de l'analyse s'appelle ***model generalization error***



# Mesure de la performance du modèle (2)

- 2 cas où le modèle n'est pas très efficace
  - Le **sous-ajustement** se produit lorsque le modèle ne peut pas capturer la complexité de l'ensemble de données.
  - Le **surapprentissage** se produit lorsque le modèle est trop spécifique à l'ensemble de données d'apprentissage et capture des détails qui ne se généralisent pas aux nouvelles données.  
Typiquement, une valeur MSE faible sur l'ensemble de données d'entraînement et une valeur MSE élevée sur l'ensemble de données de test

# Analyse de trame de données

Introduction

Détection de valeurs anormales

Régression

**Classification**

Inférence

# Introduction

- Application typique : Accord de prêt bancaire  
En fonction du montant du prêt et du profil de l'emprunteur, accorder ou pas le prêt
- Lors de la création d'un job de classification, il faut spécifier :
  - La variable dépendante : Max de 30 valeurs discrètes
  - Les feature variables : Variables qui influent

# Apprentissage

- Comme pour la régression, il faut fournir un ensemble de données étiquetées divisées en :
  - Un ensemble pour l'apprentissage
  - Un ensemble pour l'évaluation
- La division s'effectue en fournissant un pourcentage
- L'algorithme utilisé est appelé *boosted tree regression model*.

Il utilise des arbres de décision pour prédire la probabilité d'une valeur discrète
- Le modèle est stocké dans un index interne à Elasticsearch

# Résultats

- Le résultat est fourni via 2 indicateurs :
  - ***class\_probability*** : valeur comprise entre 0 et 1, qui indique la probabilité qu'un point donné appartienne à une certaine classe
  - ***class\_score*** : Fonction de *class\_probability* qui a une valeur  $\geq 0$ .  
Il prend en considération l'objectif (défini via la configuration de job `class_assignment_objective`) :
    - Précision : Score pondéré avec les bonnes classifications
    - Recall : Prend en compte également les mauvaises classifications

# Mesure de la performance du modèle

- On peut mesurer l'efficacité du modèle via l'API `POST _ml/data_frame/_evaluate` {  
    `"evaluation" : "classification"`  
}
- Le retour est une matrice de confusion

# Analyse de trame de données

Introduction

Détection de valeurs anormales

Régression

Classification

**Inférence**

# Introduction

- L'inférence permet d'utiliser les analyses régression ou classification de manière continue.
- Cela veut dire appliquer les modèles prédictifs sur les nouvelles données
- Plusieurs alternatives
  - Utiliser une pipeline d'ingestion
  - Utiliser des agrégations



# Inference processor

- La configuration d'un processeur contient les champs :
  - **"*model\_id*"** : Le modèle
  - **"*target\_field*"** : Le champ ajouté au document contenant la prédiction
  - **"*inference\_config*"** : Le type d'analyse *regression* ou *classification* avec un bloc dépendant du type

# Examples

```
"inference":{  
  "model_id":"my_model_id"  
  "inference_config": {  
    "regression": {  
      "results_field": "my_regression"  
    }  
  }  
}
```

```
"inference":{  
  "model_id":"my_model_id"  
  "inference_config": {  
    "classification": {  
      "num_top_classes": 2,  
      "results_field": "prediction",  
      "top_classes_results_field": "probabilities"  
    }  
  }  
}
```

# Agrégation

- L'inférence peut également être utilisée comme agrégation de pipeline. Le modèle est référencé dans l'agrégation pour inféré sur le champ de l'agrégation parente
-

# Example

```
{  
  "inference": {  
    "model_id": "a_model_for_inference",  
    "inference_config": {  
      "regression_config": {  
        "num_top_feature_importance_values": 2  
      }  
    },  
    "buckets_path": {  
      "avg_cost": "avg_agg",  
      "max_cost": "max_agg"  
    }  
  }  
}
```

# Visualisations Kibana

**URLs personnalisés**  
Différents types de visualisation

# URLs personnalisées

- Via des *Advanced jobs*, il est possible d'ajouter des hyperliens dans les vue *Anomaly Explorer* ou *Single Metric Viewer*. Ces liens peuvent être dirigés vers :
  - Un tableau de bord Kibana
  - La page Discovery de Kibana
  - Une URL externe

# Configuration

- Pour chaque URL, on indique :
  - Un label
  - Optionnellement un intervalle de temps.

Edit response\_requests\_by\_app

Job Details

Detectors

Datafeed

Custom URLs

## Create new custom URL

Label ⓘ

My link 1

Link to ⓘ

☐ Kibana dashboard

☒ Discover

☐ Other URL

Index pattern ⓘ

server-metrics\*

Query entities ⓘ

service ✕

Time range ⓘ Interval

interval ▼

2h

Add

Update

Cancel

# Expression dans l'URL

- L'URL peut contenir des expressions entourées par des **\$** qui sont substituées à l'exécution :
  - Soit par les champs disponibles dans le document anomalie
  - Soit par des valeurs prédéfinies :
    - **\$earliest\$ \$latest\$** : le début et la fin de la période de l'anomalie sélectionnée
    - **\$mlcategoryregex\$ \$mlcategoryterms\$** : Utile lors de la catégorisation de messages
      - **\$mlcategoryregex\$** expression régulière de l'anomalie sélectionnée (champ mlcategory)
      - **\$mlcategoryterms\$** valeur des termes de la catégorie de l'anomalie sélectionnée

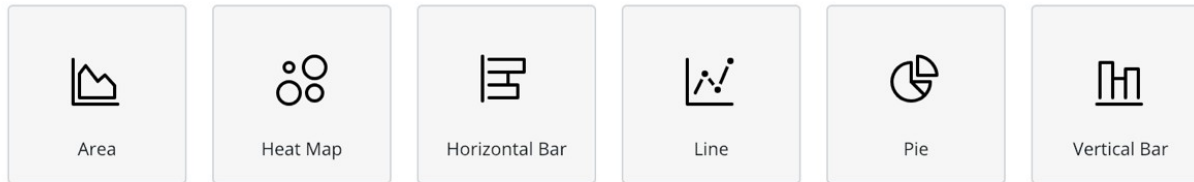


# Visualisations Kibana

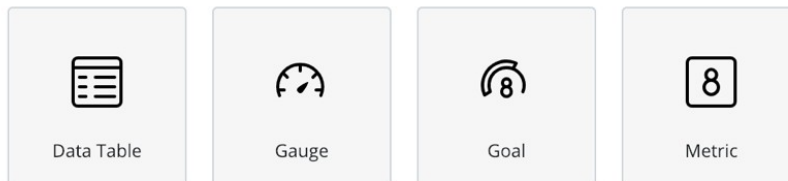
URLs personnalisés  
**Différents types de visualisation**

# Visualisations de Kibana

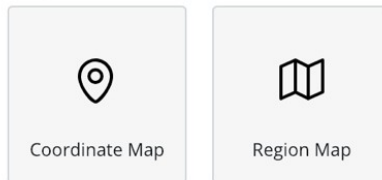
## Basic Charts



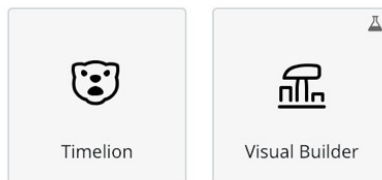
## Data



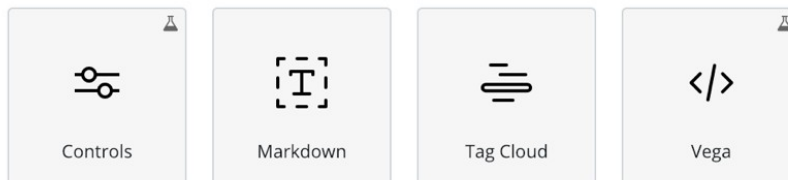
## Maps



## Time Series



## Other



# Visualisations utiles pour ML

- Data Table : Liste de données
- Heat map :
  - Sévérité d'une anomalie
  - Corrélation de plusieurs
- Timelion :
  - Basée sur une expression
  - Permet de combiner plusieurs sources de données
- Time series visual builder
  - Permet les agrégations et les agrégations en pipeline
  - Propose différents types de graphiques
  - Permet les annotations

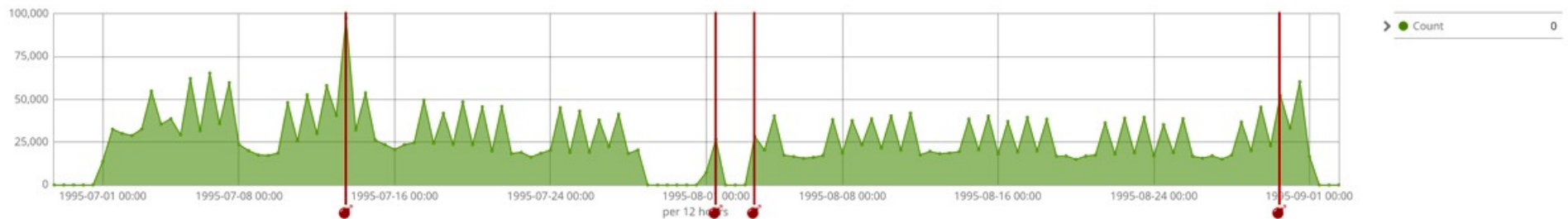
# Exemple TSVB

- La configuration d'un TSVB contient 3 onglets :
  - **Data** : Configuration de l'agrégation et de la fonction à appliquer
  - **Panel** : Source de données
  - **Annotations** : Possibilité d'utiliser une autre source de données, par exemple le job ML

# Affichage des anomalies via les annotations

⚠ This visualization is marked as experimental. Have feedback? Please create an issue in [GitHub](#).

Time Series Metric Top N Gauge Markdown Table



Auto Apply ☒ Apply Changes The changes will be automatically applied.

Data Panel Options Annotations

Data Sources

☒ Index Pattern (required)  Time Field (required)

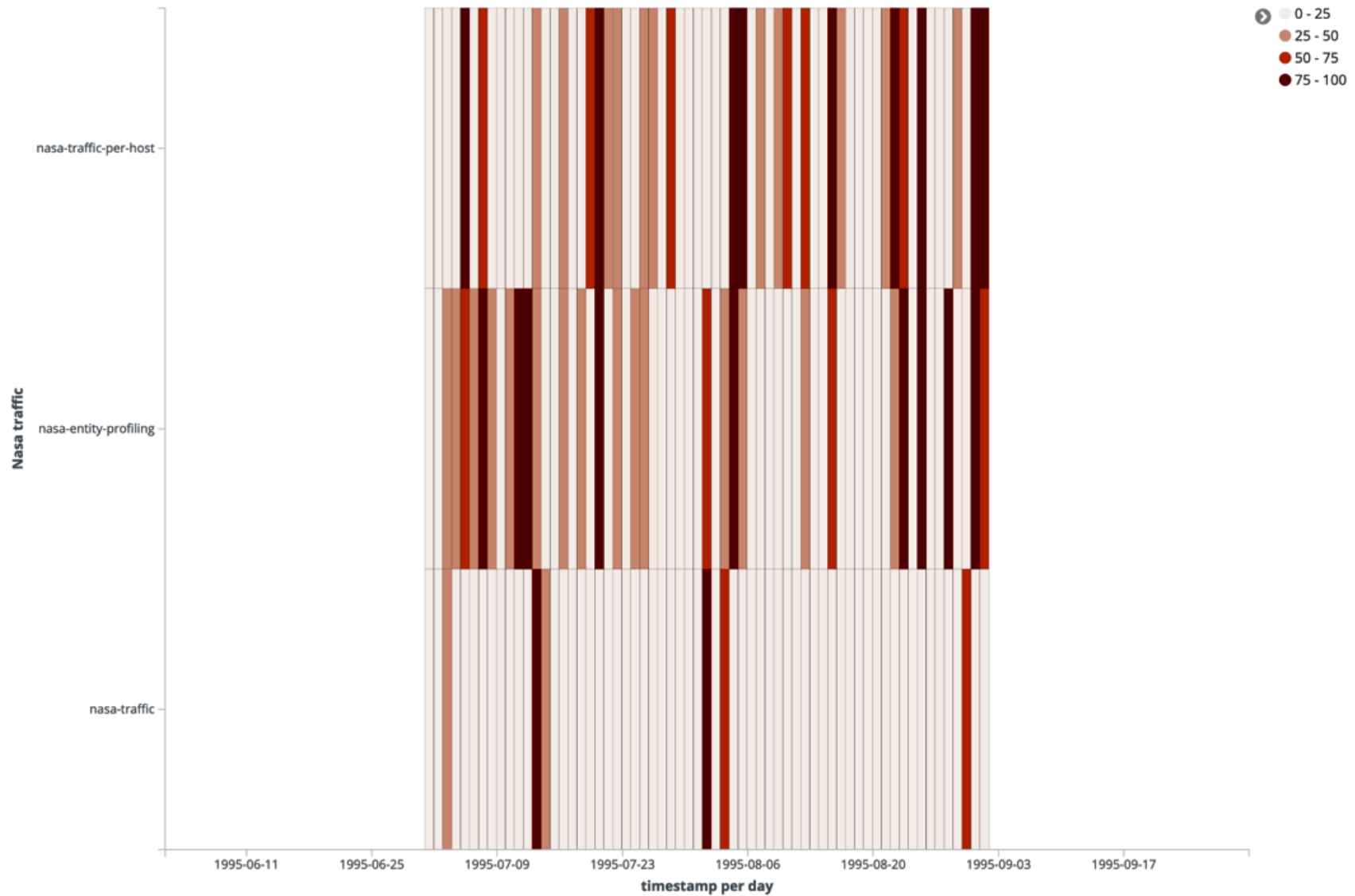
Query String  Ignore Global Filters ☒ Yes ☐ No Ignore Panel Filters ☒ Yes ☐ No

Icon (required)  Fields (required - comma separated paths)  Row Template (required - eg. {{field}})

# Exemple heat map

- Un heat map peut être utilisé pour visualiser toutes les anomalies de tous les jobs ML :
  - Choisir l'index pattern ***.ml-anomalies\****
  - 1 bucket de type Date Histogram
  - 1 sub-bucket de type term sur le ***job id***
  - Metrics : Max ***anomaly\_score***
  - Color schema : Rouge

# Exemple heat map



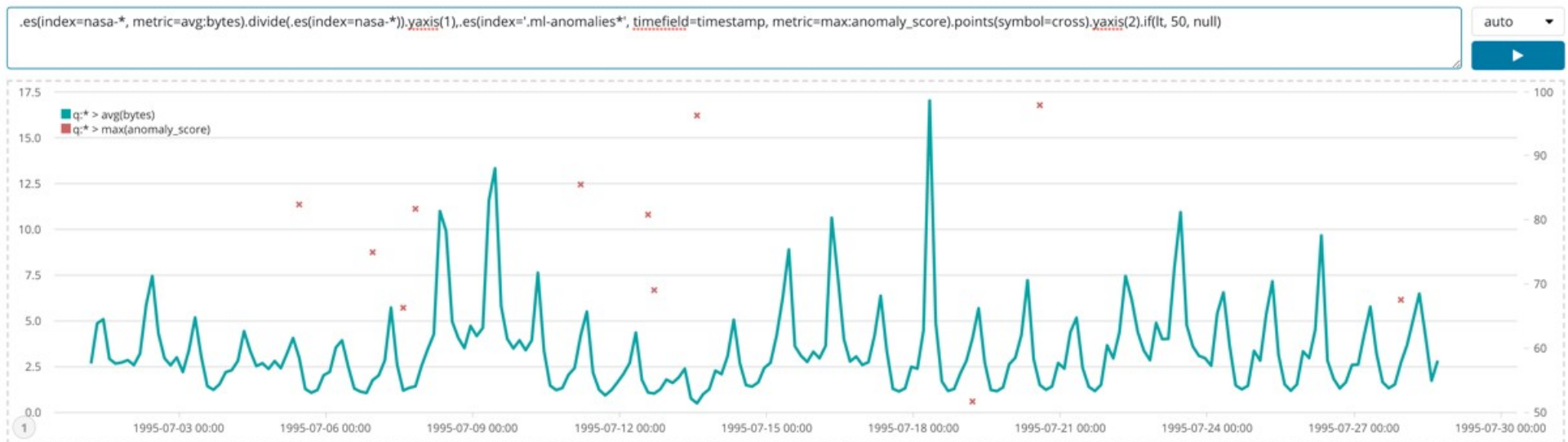
# Timelion

- *Timelion* peut également être utilisé pour combiner dans la même vue les données et les anomalies détectées
- Par exemple afficher la taille moyenne des requêtes d'un site web en même temps que les anomalies de trafic détectées
- L'expression pourrait être :

```
.es(index=nasa-*, metric=avg:bytes)
  .divide(.es(index=nasa-*)).yaxis(1),
.es(index='.ml-anomalies*', timefield=timestamp,
metric=max:anomaly_score)
  .points(symbol=cross).yaxis(2).if(lt, 50, null)
```



# Exemple timelion



# 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 !
- Canvas peut s'appliquer sur les index d'anomalies

# 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

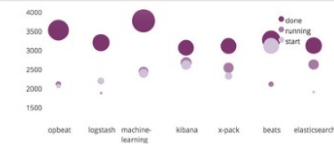
# Éléments

🔍 Filter elements



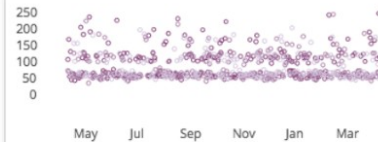
## Area chart

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

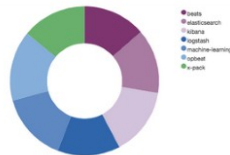
## Data table

A scrollable grid for displaying data in a tabular format

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

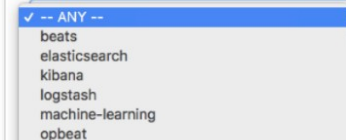
## Debug

Just dumps the configuration of the element



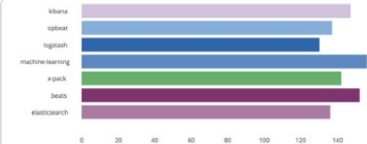
## Donut chart

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

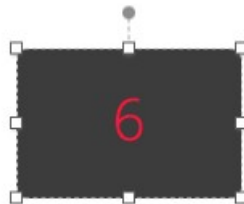
# Canvas expression

- 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"}
```

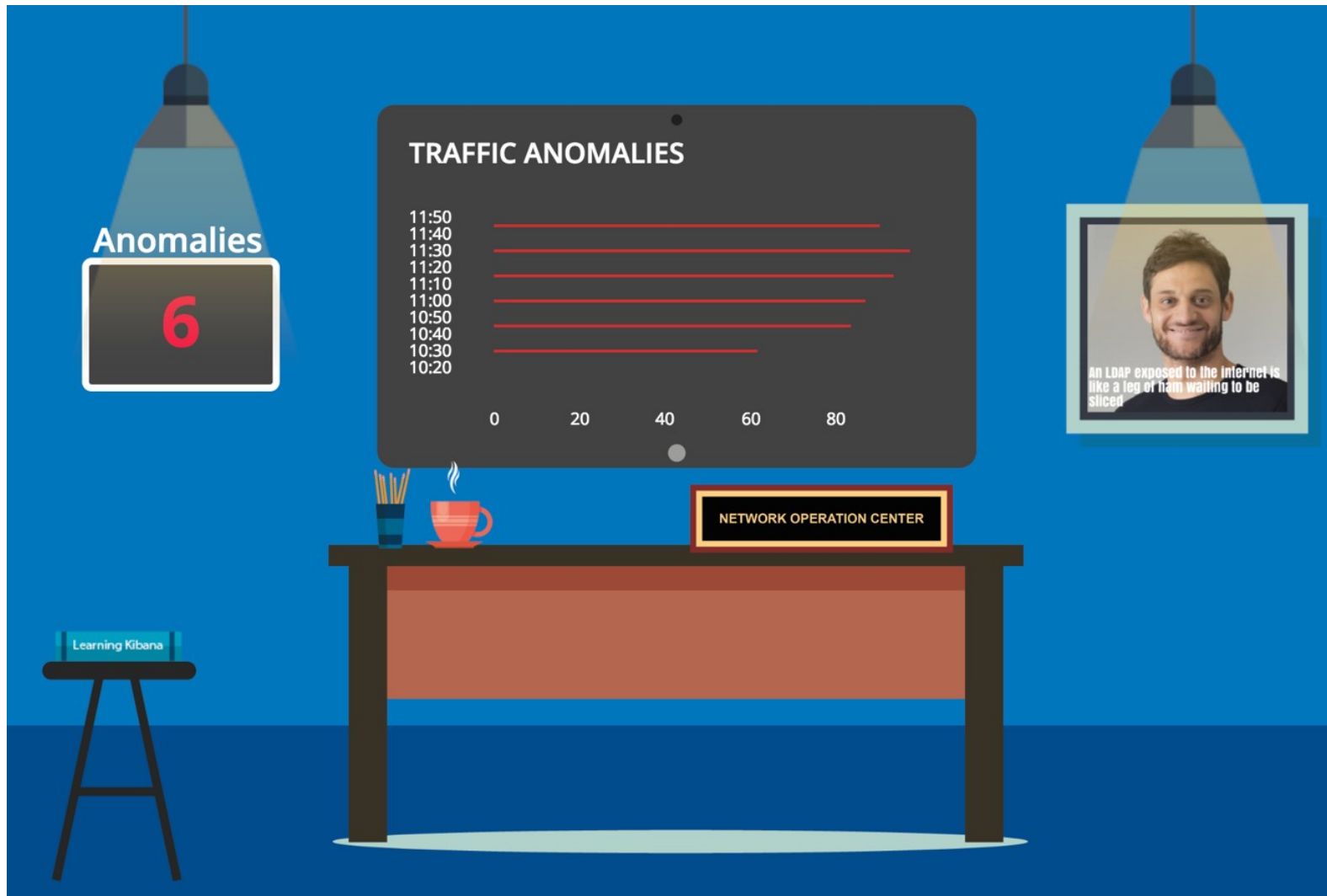


# Example

- 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





MERCI !!

Pour votre attention

# Référence

- « *Machine Learning with Elastic Stack* » :  
Rich Collier et Bahaaldine Azarmi
- Web Site :

## Overview

<https://www.elastic.co/guide/en/elastic-stack-overview/current/xpack-ml.html>

## Data structure

<https://www.elastic.co/guide/en/elasticsearch/reference/7.0/api-definitions.html>

# Annexe

## **Alertes**

# Résultats des jobs

- Les résultats des jobs ML sont présentés à trois différents niveaux d'abstraction
  - **Bucket** : Représente comment ce bucket est inhabituel vis à vis des détecteurs du job
  - **L'enregistrement** : Ce sont les informations les plus détaillées sur chaque occurrence anormale ou entité anormale dans un intervalle de temps
  - **Influenceur** :
- Pour accéder aux résultats :
  - Utiliser l'API ML */results*
  - Rechercher dans les index créés par ML. Méthode la plus flexible

# L'index résultat

- ML analyse les données et stocke les résultats dans un index nommé ***.ml-anomalies-shared*** par défaut ou ***.ml-anomalies-custom-myname*** si on l'a indiqué dans la configuration du job le champ *results\_index\_name*
- De plus, un alias d'index est également créé sous la forme ***.ml-anomalies-jobname*** :

```
GET ml-anomalies-*/_alias
".ml-anomalies-farequote": {
  "filter": {
    "term": {
      "job_id": {
        "value": "farequote",
        "boost": 1
      }
    }
  }
}
```

# Types de document

- Dans l'index des résultats, il existe une variété de documents différents, chacun ayant sa propre utilité en ce qui concerne Alerting :
  - **result\_type:bucket**: pour donner des résultats au niveau du bucket.  
1 document par bucket, timestamp égal au démarrage du bucket
  - **result\_type:record**: Pour donner des résultats au niveau de l'enregistrement.  
1 document pour chaque anomalie trouvé dans le bucket span.  
timestamp égal au démarrage du bucket
  - **result\_type:influencer**: Pour donner des résultats au niveau des influenceurs.  
1 document pour chaque influenceur d'une anomalie, timestamp égal au démarrage du bucket

# Documents bucket

- Les champs d'un document bucket :
  - ***timestamp***
  - ***anomaly\_score*** : Le score normalisé. La valeur peut fluctuer à mesure que de nouvelles données arrivent
  - ***initial\_anomaly\_score*** : Le score normalisé lors de la première analyse
  - ***event\_count*** : Le nombre de documents analysés pendant le bucket span
  - ***is\_interim*** : Un flag indiquant si ML attend encore des données pour ce bucket
  - ***bucket\_influencers*** : Un tableau des influenceurs identifiés pour ce bucket. (Il y a toujours l'influenceur par défaut influencer\_field\_name:bucket\_time)

# Détails des influenceurs

```
"bucket_influencers": [  
  {  
    "job_id": "farequote",  
    "result_type": "bucket_influencer",  
    "influencer_field_name": "airline",  
    "initial_anomaly_score": 85.06429298617539,  
    "anomaly_score": 99.7634,  
    "raw_anomaly_score": 15.040566947916583,  
    "probability": 6.5926436244031685e-18,  
    "timestamp": 1486656000000,  
    "bucket_span": 900,  
    "is_interim": false  
  },  
  {  
    "job_id": "farequote",  
    "result_type": "bucket_influencer",  
    "influencer_field_name": "bucket_time",  
    "initial_anomaly_score": 85.06429298617539,  
    "anomaly_score": 99.76353,  
    "raw_anomaly_score": 15.040566947916583,  
    "probability": 6.5926436244031685e-18,  
    "timestamp": 1486656000000,  
    "bucket_span": 900,  
    "is_interim": false  
  }  
],
```



# Document enregistrement

- Les champs d'un document enregistrement :
  - ***timestamp***
  - ***record\_score*** : Le score normalisé pouvant fluctuer
  - ***initial\_record\_score*** : Le score normalisé de la première analyse
  - ***detector\_index*** : Indique le détecteur ayant provoqué l'anomalie
  - ***function*** : La fonction utilisé par le détecteur
  - ***is\_interim***
  - ***actual*** : La valeur réelle observée pour ce bucket.
  - ***typical*** : Une représentation de la valeur attendue par le modèle
- Si le job a été catégorisé (par *by\_field\_name*, *partition\_field\_name* ou des influenceurs), des données additionnelles sur les valeurs de la catégorie ayant provoqué l'anomalie sont disponibles

# Analyse de population

- Si le job effectue une analyse de population via le champ *over\_field\_name*, les résultats sont présentés différemment
  - Un premier bloc identifie l'individu anormal
  - Un tableau causes liste toutes les anomalies provoquée par cet individu

# Example

```
{ ...
  "_source": {
    "job_id": "gallery",
    "result_type": "record",
    "..."
  },
  "over_field_name": "clientip",
  "over_field_value": "173.203.78.60",
  "causes": [
    {
      "probability": 4.593248987780688e-31,
      "by_field_name": "status",
      "by_field_value": "404",
      "function": "count",
      "typical": [ 1.1177332137173952 ],
      "actual": [ 1215 ],
      "over_field_name": "clientip",
      "over_field_value": "173.203.78.60" } ],
    "influencers": [
      { "influencer_field_name": "uri",
        "influencer_field_values": [ "/wp-login.php" ] },
      { "influencer_field_name": "status",
        "influencer_field_values": [ "404" ] },
      { "influencer_field_name": "clientip",
        "influencer_field_values": [ "173.203.78.60" ] }
    ],
    "clientip": [ "173.203.78.60" ],
    "uri": [ "/wp-login.php" ],
    "status": [ "404" ]
  }
```

# Document influenceur

- Les documents influenceurs ont les champs suivants :
  - ***timestamp***
  - ***influencer\_score*** : Le score normalisé pouvant fluctuer
  - ***initial\_influencer\_score*** : Le score normalisé de la première analyse
  - ***influencer\_field\_name*** : Le nom de l'influenceur
  - ***influencer\_field\_value*** : La valeur de l'influenceur
  - ***is\_interim***

# Définition d'une alerte via l'UI

- La première méthode pour définir une alerte liée à un job ML consiste à utiliser l'assistant de l'UI, lorsque l'option *Continue job in real time*, est cochée

☒ Continue job in real-time

☒ Create watch for real-time job

**Time range**  
Now - 30m

**Severity threshold**  
critical ▼

☒ Send email

admin@elastic.co

Apply

# Paramètres de l'alerte

- Les paramètres demandés par l'UI sont :
  - **Time range** : Par défaut : now-2xbucket span. La valeur minimum étant : now -(bucket span + query delay)
  - **Seuil de sévérité** : Minimum score du bucket. Par exemple critical = 75
  - Adresse mail à alerter. L'alerte écrit toujours un message de log
- Après la création, l'alerte est visible, éditabile et peut être testée via l'UI Watcher

# Implications

- La condition principale pour l'alerte est un score d'anomalie du bucket. L'alerte n'est pas déclenchée pour des anomalies qui ne provoqueraient pas le dépassement du seuil au niveau du bucket.
- Par défaut, seul un maximum des trois scores les plus élevés dans le bucket est indiqué dans la sortie, et uniquement si l'action d'envoi de mail est choisie.
- L'alerte existe toujours même si le job est supprimé.
- Les seules actions possibles de l'alerte sont la journalisation et l'envoi de mail.

# Alertes manuelles

- Des exemples beaucoup plu complexes peuvent être effectué manuellement.
- Voir :  
[https://github.com/PacktPublishing/Machine-Learning-with-the-Elastic-Stack/blob/master/Chapter06/custom\\_ML\\_watch.json](https://github.com/PacktPublishing/Machine-Learning-with-the-Elastic-Stack/blob/master/Chapter06/custom_ML_watch.json)
- Cet exemple
  - produit une alerte si 2 jobs différents détecte une anomalie pour le même bucket
  - Consolide les informations des 2 jobs dans le résultat