

## ~ Objectifs ~

- Comprendre les bases d'Elasticsearch, notamment son architecture et son modèle de données.
- Démontrer le processus d'installation, de configuration et de chargement des données dans Elasticsearch.
- Se familiariser avec l'outil Discover et les capacités de recherche et d'agrégation de données.
- Présenter les capacités de visualisation de Kibana à travers des exemples significatifs.

## I. Introduction à Elasticsearch

### 1. Définition et aperçu d'Elasticsearch

Elasticsearch est un moteur de recherche et d'analyse open-source puissant, conçu pour nous aider à rechercher, analyser et visualiser de grandes quantités de données rapidement et en temps quasi réel. Grâce à son architecture évolutive et flexible, Elasticsearch peut gérer un large éventail de cas d'utilisation, de la simple recherche plein texte à l'analyse et à la visualisation de données complexes.



### 2. Principales caractéristiques et avantages

- Évolutivité : il peut évoluer horizontalement, ce qui lui permet de gérer de grandes quantités de données et de trafic.
- Vitesse : il est conçu pour fournir des résultats de recherche rapides, même pour les grands ensembles de données.
- Personnalisation : il permet aux utilisateurs d'adapter le moteur de recherche à leurs besoins spécifiques grâce à son API et à ses plugins.
- API conviviale : il dispose d'une API RESTful, ce qui facilite son intégration à d'autres applications et systèmes.
- Analyse intégrée : il fournit un support intégré pour l'agrégation et l'analyse, ce qui permet d'effectuer des analyses de données complexes sans outils supplémentaires.

### 3. Cas d'utilisation et applications

#### BY TOPIC

Power of Elastic

Improving digital customer experiences

Evolving the DevOps lifecycle

Security without limits

[View all topics](#)

#### BY INDUSTRY

Public Sector

Financial Services

Telecommunications

Healthcare

Technology

Retail and Ecommerce

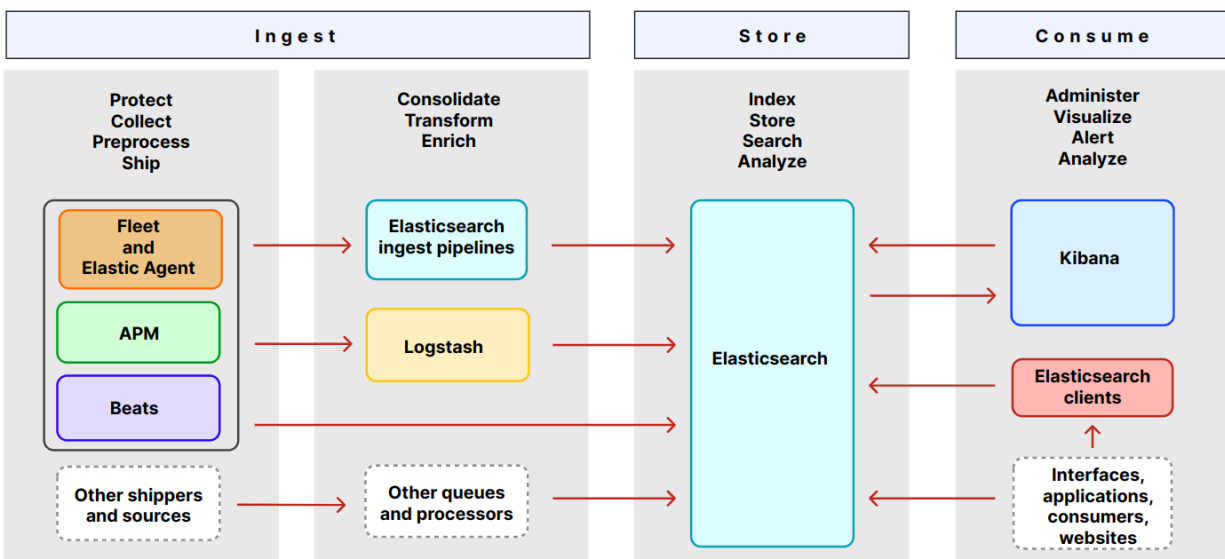
Media and Entertainment

Manufacturing and Automotive

[View all industries](#)

Source : <https://www.elastic.co/>

### 4. Elastic Stack



Source : <https://www.elastic.co/guide/index.html>

## II. Architecture et modèle de données

### 1. Architecture distribuée

Elasticsearch est un moteur de recherche distribué, c'est-à-dire qu'il fonctionne sur un cluster de nœuds pour assurer l'évolutivité, la fiabilité et la haute disponibilité. Chaque nœud du cluster effectue les mêmes tâches et est capable de répondre aux demandes de recherche. Cela permet au cluster de gérer une charge de travail accrue, car les nœuds peuvent partager la charge des demandes entrantes.

### 2. Node, Cluster, et Index:

Elasticsearch utilise un cluster de nœuds pour gérer ses données. Chaque nœud appartient à un seul cluster et chaque cluster contient un ou plusieurs index. Un index est une collection de documents qui partagent des caractéristiques similaires, comme les types de données et la structure. Chaque document d'un index est identifié par un ID unique et contient un ou plusieurs champs, qui sont des paires clé-valeur contenant les données réelles.

Elasticsearch	SGBD relationnel
Cluster	Base de données
Index	Table
Field	Column (colonne)
Document	Ligne (row)

### 3. Modèle de données et Mapping

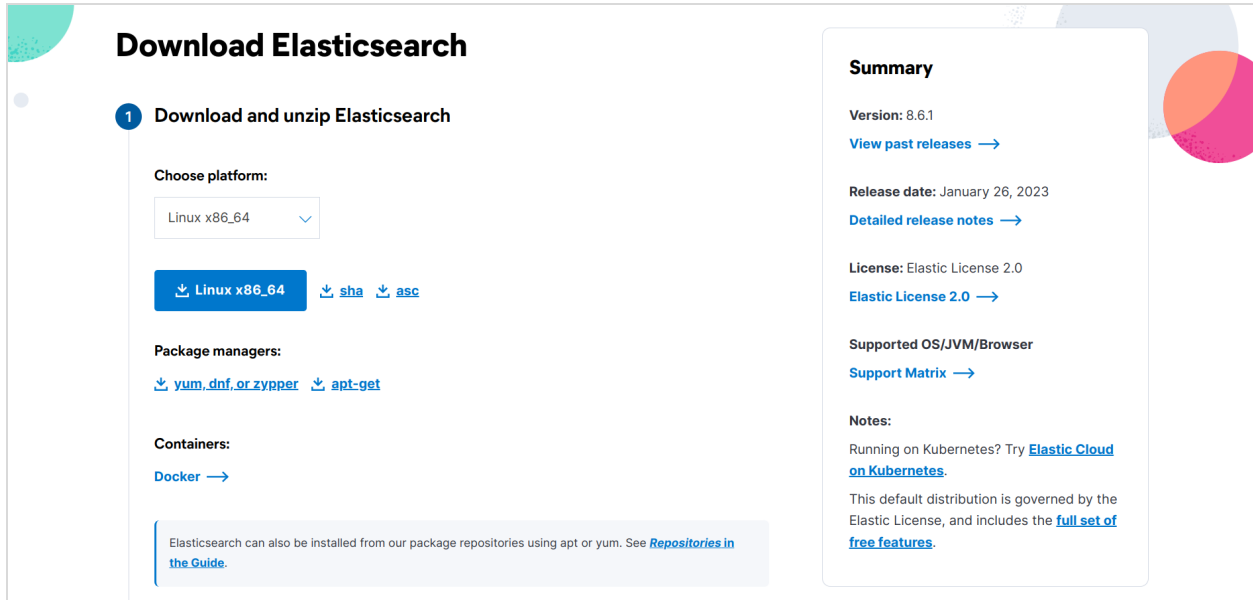
Le modèle de données d'Elasticsearch est Document-Based, ce qui signifie que chaque document représente une instance d'un type de données particulier. Le mapping est utilisé pour définir le type de données et la structure de chaque champ du document, y compris son type de données, ses options d'indexation et son analyseur de recherche.

### III. Utilisation sur la machine locale

#### 1. Installer ELasticsearch sur votre machine

Elasticsearch peut être téléchargé du lien : [Download Elasticsearch](#), de 3 façons :

1. Télécharger un dossier archive compatible avec l'OS de votre machine.
2. Utiliser un package manager comme yum ou apt
3. Utiliser un conteneur Docker.



#### 2. Lancer Elasticsearch

Aller dans le répertoire d'origine d'Elasticsearch et dans le dossier bin exécuter :

```
> ./elasticsearch
```

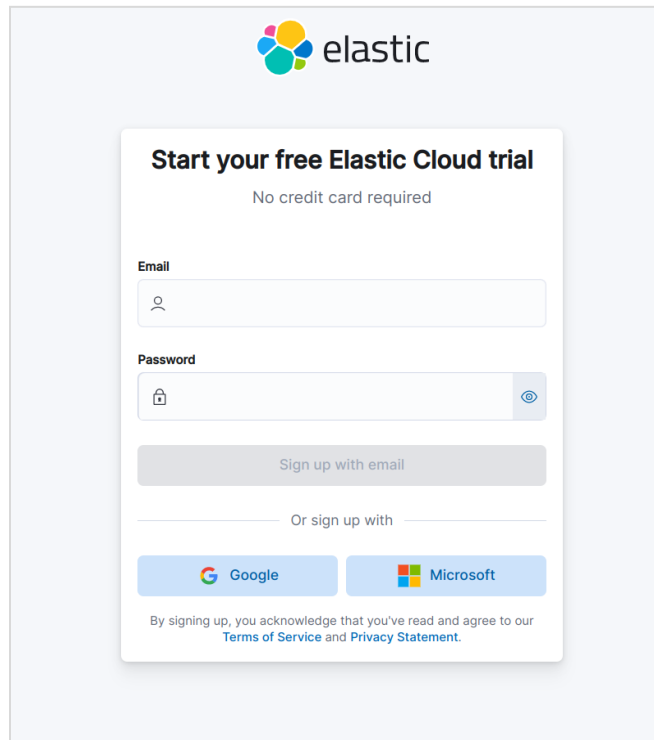
Le port par défaut pour l'interface web d'Elasticsearch est **9200**. On peut le changer en modifiant *http.port* dans le fichier *elasticsearch.yml* présent dans le répertoire bin.

Pour vérifier si le serveur est lancé, aller sur *http://localhost:9200*. Il retournera un objet JSON, qui contient les informations sur l'Elasticsearch installé.

Pour le reste de ce tutoriel, on va travailler sur la version Cloud d'Elastic. Les mêmes fonctionnalités seront disponibles en installant Kibana sur [Download Kibana](#).

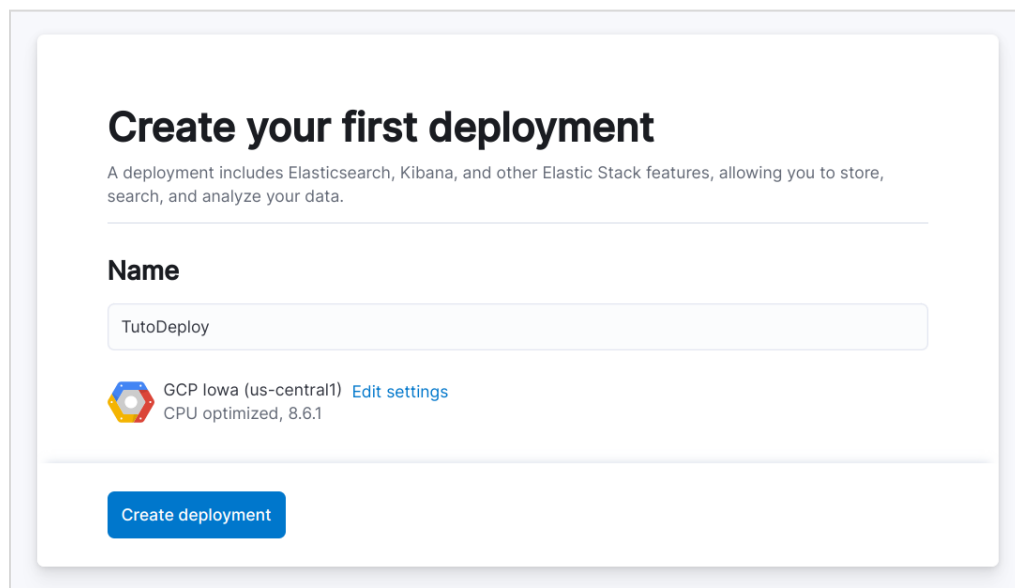
## IV. Création d'un déploiement Elastic Cloud

- Créer un compte : [Elastic Cloud Trial](#)



The image shows the Elastic Cloud trial sign-up page. At the top is the Elastic logo. Below it, the heading "Start your free Elastic Cloud trial" is followed by the subtext "No credit card required". There are two input fields: "Email" with a person icon and "Password" with a lock icon and a toggle for visibility. Below these is a "Sign up with email" button. Underneath is a section "Or sign up with" with buttons for "Google" and "Microsoft". At the bottom, a small disclaimer states: "By signing up, you acknowledge that you've read and agree to our [Terms of Service](#) and [Privacy Statement](#)."

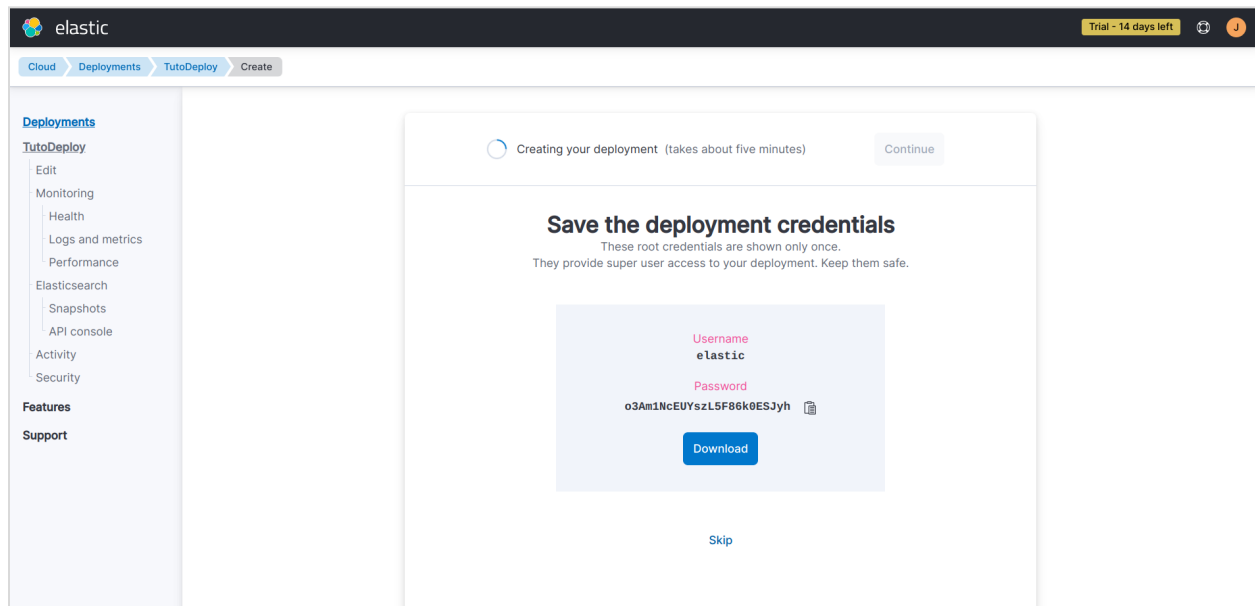
- Créer un déploiement en lui donnant un nom



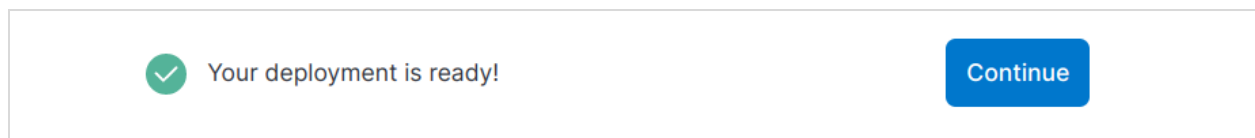
The image shows the "Create your first deployment" page. The heading is "Create your first deployment". Below it, a description says: "A deployment includes Elasticsearch, Kibana, and other Elastic Stack features, allowing you to store, search, and analyze your data." There is a "Name" label followed by an input field containing the text "TutoDeploy". Below the input field is a section for the deployment configuration, showing the Google Cloud logo, "GCP Iowa (us-central1)", and "CPU optimized, 8.6.1", with an "Edit settings" link. At the bottom is a blue "Create deployment" button.

## Tutoriel : Elasticsearch

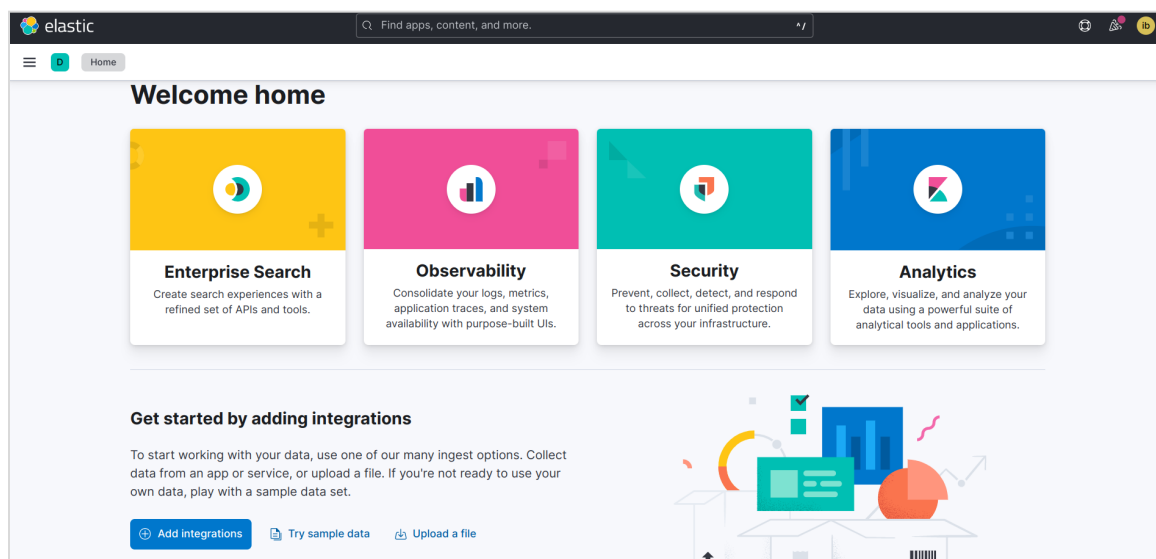
- Enregistrer les informations d'identification de votre déploiement, en attendant qu'il soit créé :



Et voilà, c'est créé :)) !

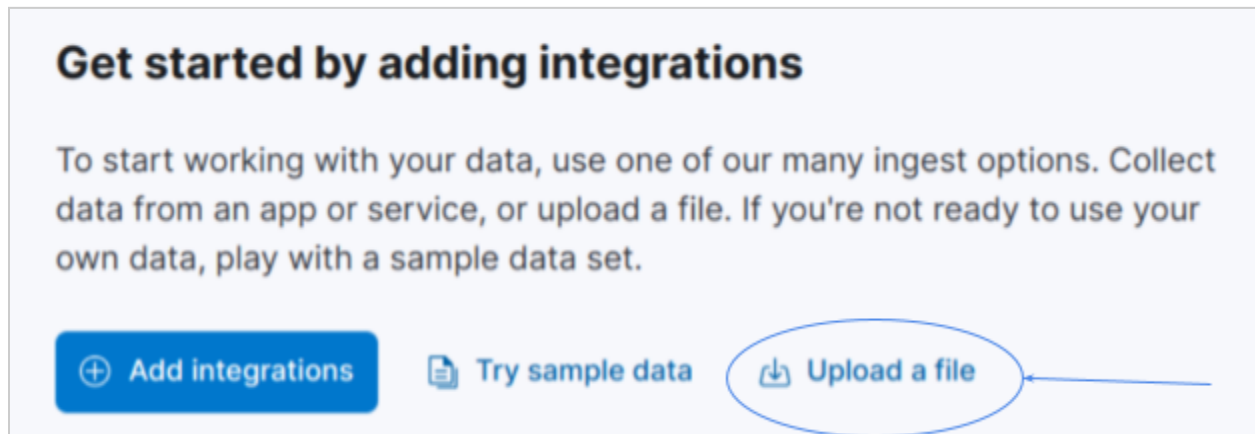


A ce stade, vous accédez à Kibana :



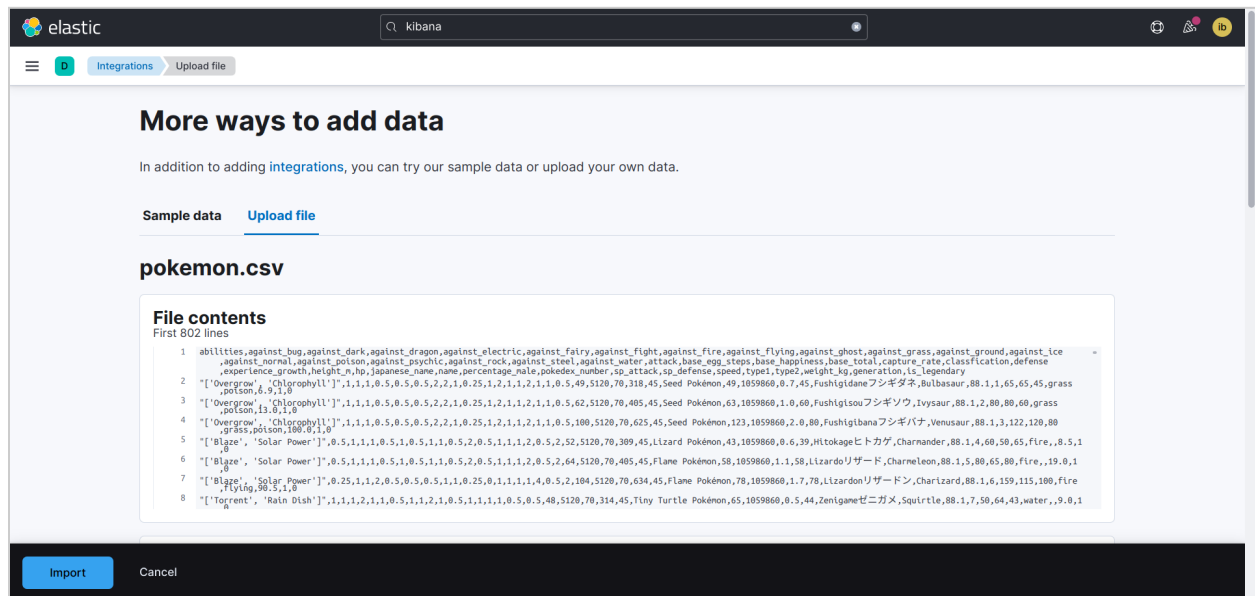
## V. Importation des données

- Sur la page d'accueil de kibana, cliquer sur "Upload a file"



- Importer un fichier : CSV, TSV, JSON ..

Ici, on a importé le fichier pokemon.csv, extraite de : [Kaggle](#)



## Tutoriel : ElasticSearch

- Donner un nom au Index (qui est l'équivalent d'une table dans les SGBD relationnels) :

elastic kibana

Integrations Upload file

### More ways to add data

In addition to adding [integrations](#), you can try our sample data or upload your own data.

Sample data [Upload file](#)

**pokemon.csv**

#### Import data

[Simple](#) [Advanced](#)

Index name

pokemon

☒ Create data view

Import

Back Cancel

- Attendre que le processus de création d'index, son remplissage et la création d'une Data View :

File processed Index created Ingest pipeline created Data uploaded Data view created

✓ Import complete

Index	pokemon
Data view	pokemon
Ingest pipeline	pokemon-pipeline
Documents ingested	801

View index in Discover

Index Management

Data View Management

Create Filebeat configuration

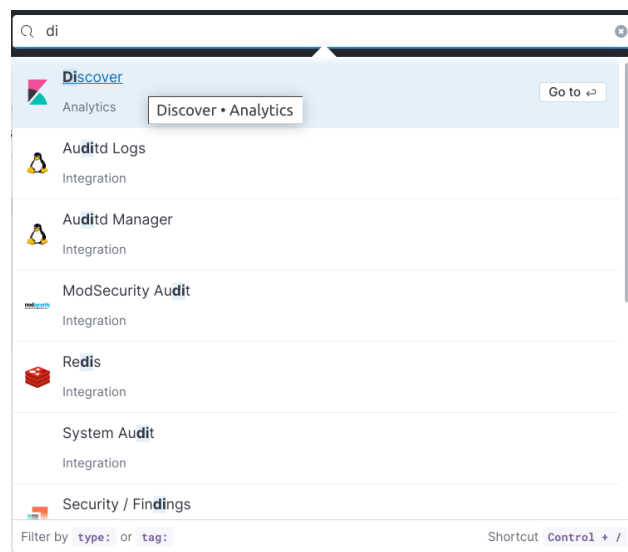
C'est terminé :))



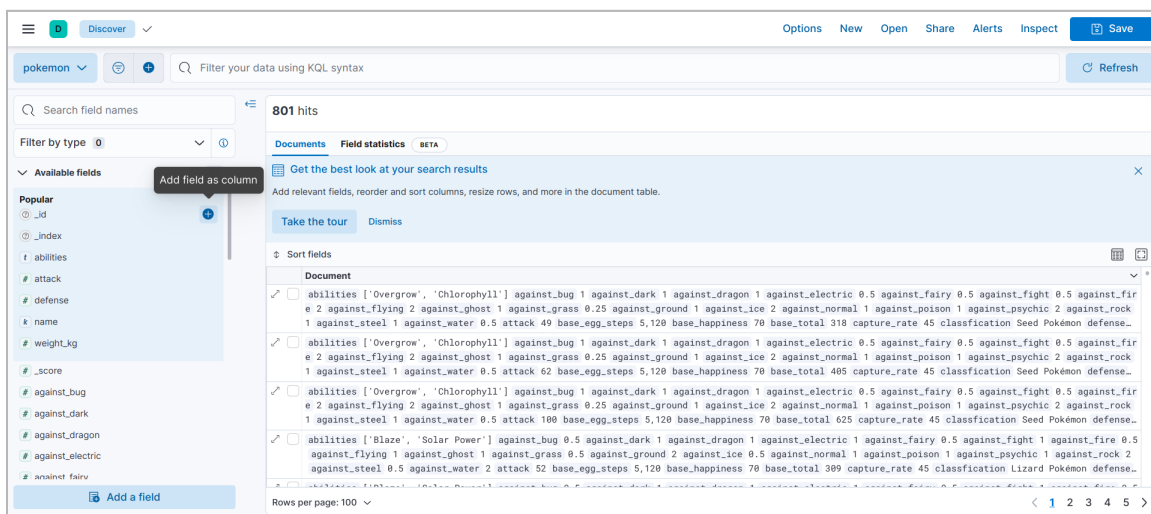
## VI. L'outil Discover

Avec Discover, on peut rapidement rechercher et filtrer nos données, obtenir des informations sur la structure des champs et afficher nos résultats dans une visualisation. On peut également personnaliser et enregistrer nos recherches et les placer sur un tableau de bord.

- Pour y accéder, chercher Discover sur la barre de recherche de Kibana :



- Voici un aperçu sur les données, si on clique sur les + à côté des *fields*, on sélectionne les *fields* qui nous intéressent :



## Tutoriel : ElasticSearch

- Ici on s'intéresse à l'id, le nom du pokémon, ses abilities, et son poids :

The screenshot shows the Kibana interface with the 'Discover' tab selected. The search bar contains 'pokemon'. On the left, the 'Selected fields' list includes '\_id', 'name', 'abilities', and 'weight\_kg'. The 'Available fields' list includes '\_index', 'attack', 'defense', '\_score', 'against\_bug', 'against\_dark', and 'against\_dragon'. The main table displays 801 hits. The first few rows are:

_id	name	abilities	weight_kg
0r4eQIYB0VxH-6S97HaH	Bulbasaur	['Overgrow', 'Chlorophyll']	6.9
074eQIYB0VxH-6S97HaH	Ivysaur	['Overgrow', 'Chlorophyll']	13
PL4eQIYB0VxH-6S97HaH	Venusaur	['Overgrow', 'Chlorophyll']	100
Pb4eQIYB0VxH-6S97HaH	Charmander	['Blaze', 'Solar Power']	8.5

- On clique sur “Sort fields” pour trier selon, par exemple, le poids en ordre décroissant:

The screenshot shows the 'Sort fields' dropdown menu. It displays 'Currently no fields are sorted' and a 'Pick fields to sort by' dropdown. The dropdown menu is open, showing 'name' and 'weight\_kg' as options. The 'weight\_kg' option is selected, and the sort order is set to 'High-Low'.


- Voici le résultat du tri :


The screenshot shows the Kibana interface with the 'Discover' tab selected. The search bar contains 'pokemon'. On the left, the 'Selected fields' list includes '\_id', 'name', 'abilities', and 'weight\_kg'. The main table displays 801 hits, sorted by 'weight\_kg' in descending order. The first few rows are:

_id	name	abilities	weight_kg
T74eQIYB0VxH-6S97HmI	Cosmoem	['Sturdy']	999.9
Vr4eQIYB0VxH-6S97HmI	Celesteela	['Beast Boost']	999.9
uL4eQIYB0VxH-6S97HeH	Groudon	['Drought']	950
J74eQIYB0VxH-6S97HmI	Mudsdale	['Own Tempo', 'Stamina', 'Inner Focus']	920
WL4eQIYB0VxH-6S97HmI	Guzzlord	['Beast Boost']	888
IL4eQIYB0VxH-6S97HiH	Giratina	['Pressure', 'Telepathy', 'Levitate']	750
HL4eQIYB0VxH-6S97HiH	Dialga	['Pressure', 'Telepathy']	683
sb4eQIYB0VxH-6S97HeH	Metagross	['Clear Body', 'Light Metal']	550
Ar4eQIYB0VxH-6S97HmI	Avalugg	['Own Tempo', 'Ice Body', 'Sturdy']	505
yL4eQIYB0VxH-6S97HaH	Snorlax	['Immunity', 'Thick Fat', 'Gluttony']	460
Hr4eQIYB0VxH-6S97HiH	Heatran	['Flash Fire', 'Flame Body']	430
H74eQIYB0VxH-6S97HiH	Regigigas	['Slow Start']	420
Cb4eQIYB0VxH-6S97HeH	Steelix	['Rock Head', 'Sturdy', 'Sheer Force']	400
er4eQIYB0VxH-6S97HeH	Wailord	['Water Veil', 'Oblivious', 'Pressure']	398

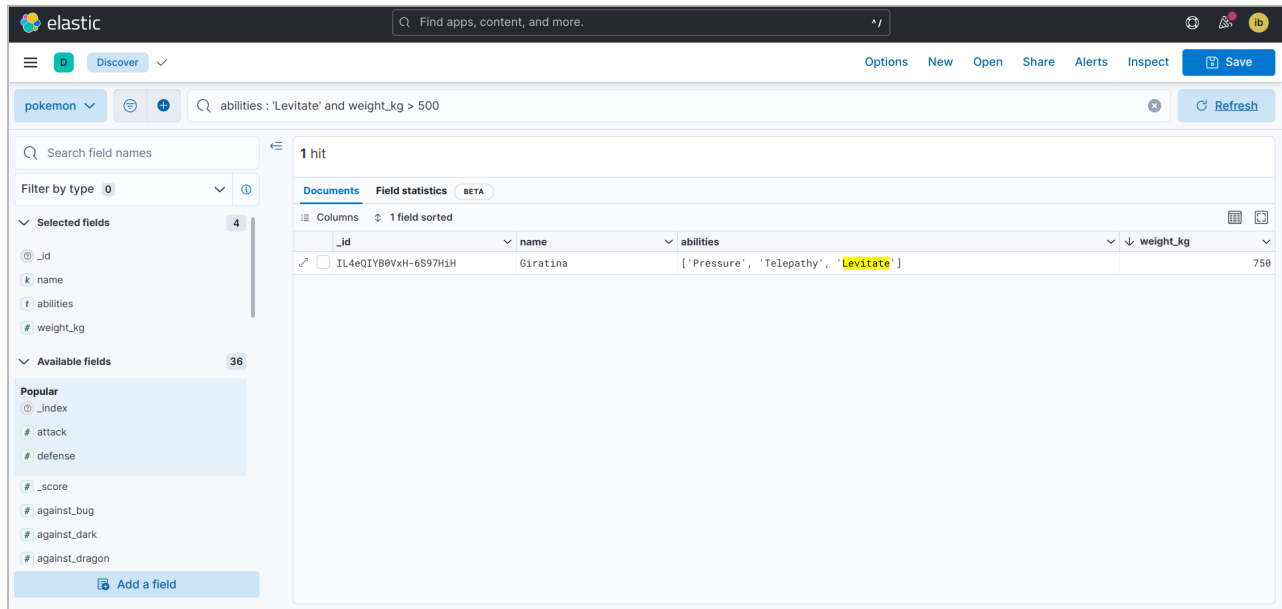
## Tutoriel : Elasticsearch

- On peut filtrer nos données en utilisant cette barre de requêtes :





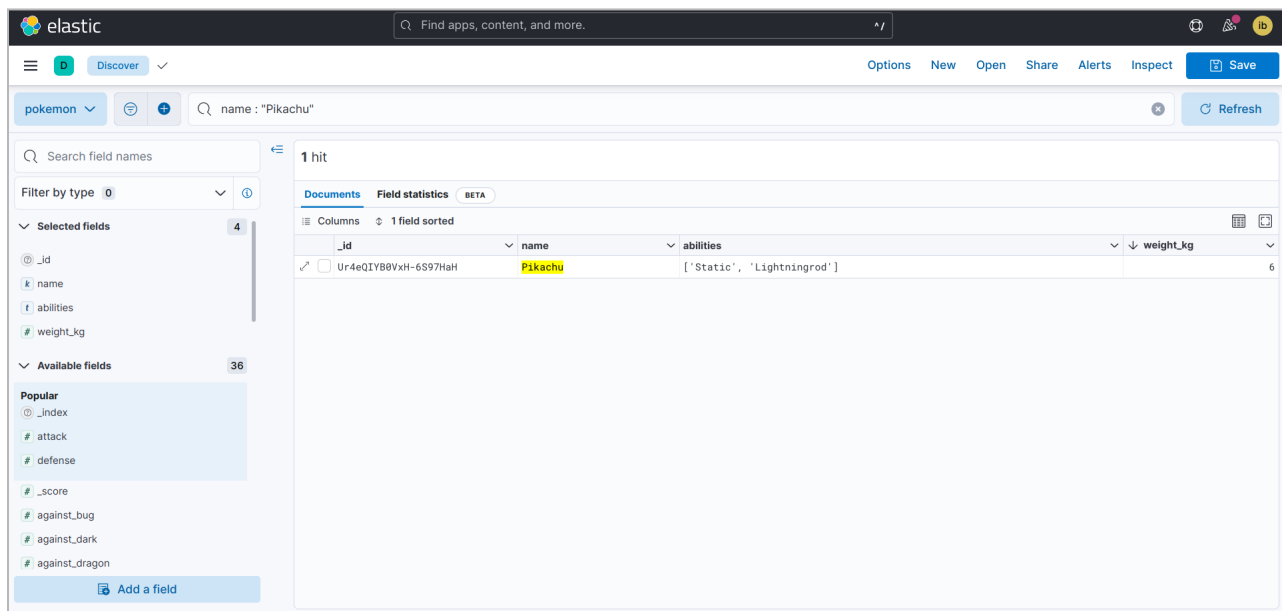
→ Par exemple, ici on cherche les pokémons ayant “levitate” comme ability et pesant plus de 500 kg :



The screenshot shows the Elastic UI interface. The search bar contains the query: `abilities : 'levitate' and weight_kg > 500`. The results table shows one hit for Giratina with weight 750 kg.

_id	name	abilities	weight_kg
IL4eQIYB0VxH-6S97H3H	Giratina	['Pressure', 'Telepathy', 'Levitate']	750

→ Ici, on s'intéresse à afficher les informations concernant votre pokémon préféré, Pikachu :

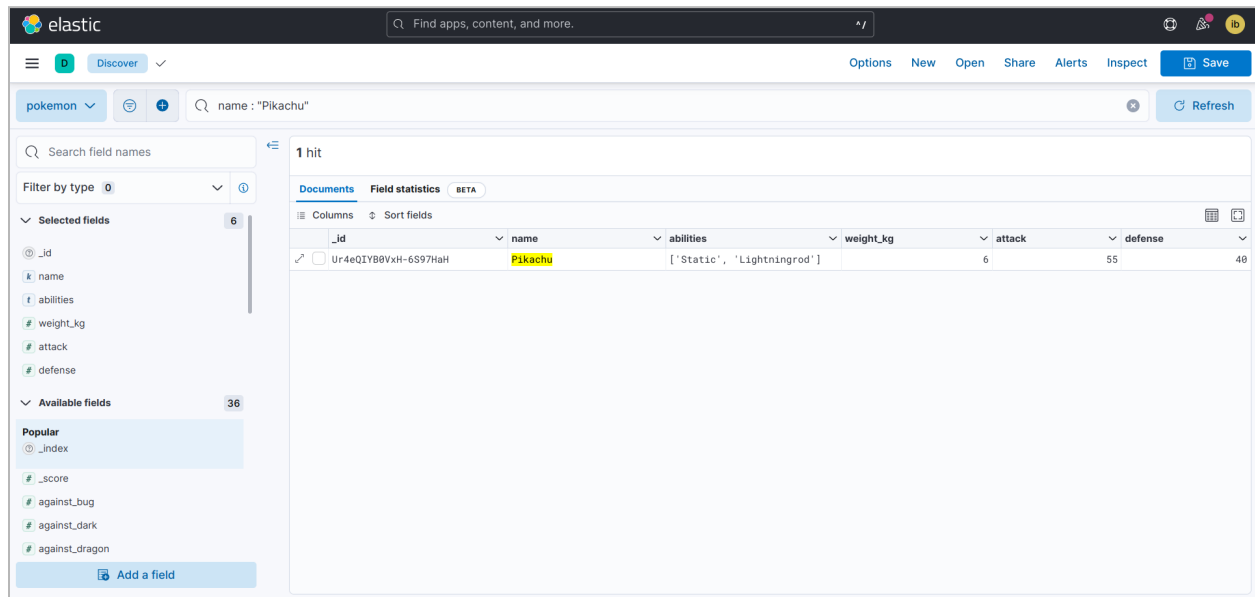


The screenshot shows the Elastic UI interface. The search bar contains the query: `name : 'Pikachu'`. The results table shows one hit for Pikachu with weight 6 kg.

_id	name	abilities	weight_kg
Ur4eQIYB0VxH-6S97HaH	Pikachu	['Static', 'Lightningrod']	6

## Tutoriel : ElasticSearch

- On voulait plus d'informations, donc on ajoute des fields supplémentaires à notre affichage (attaque et défense) :



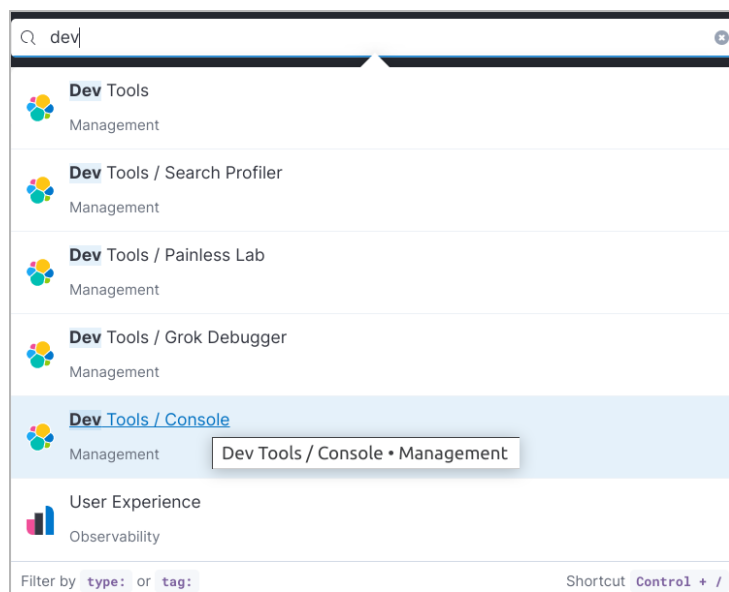
Cet outil est très utile pour répondre aux questions qu'on se pose sur nos données.

*Par exemple :*

- Quelles pages de notre site Web contiennent un mot ou une expression spécifique ?
- Quels événements ont été enregistrés le plus récemment ?
- Quels processus prennent plus de 500 millisecondes pour répondre ?

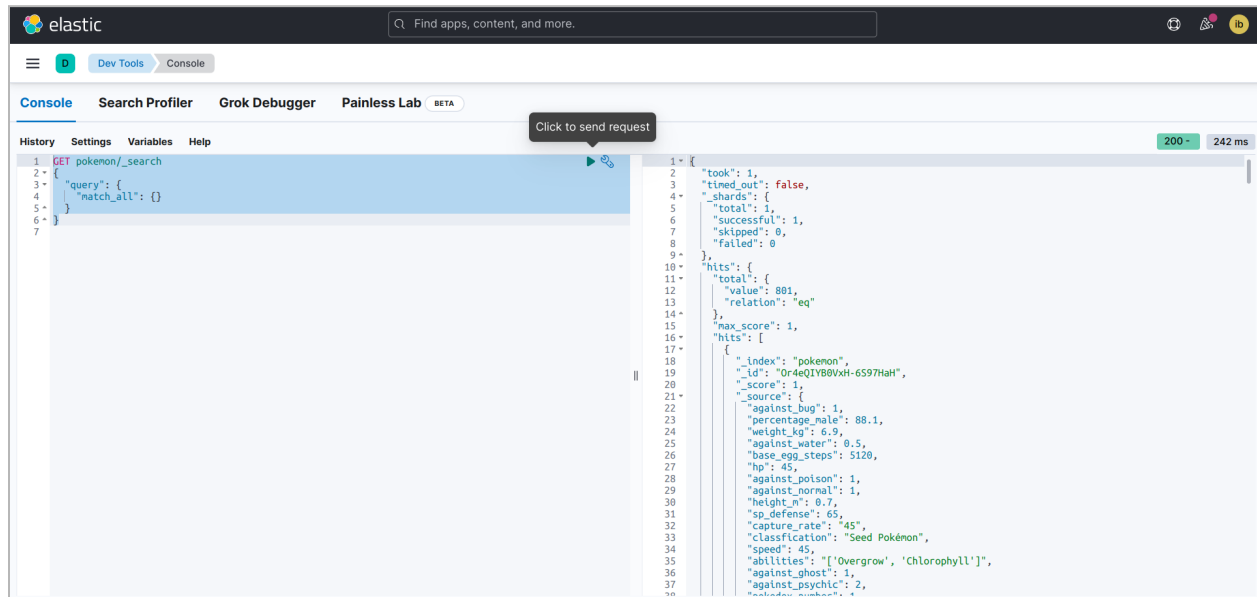
## VII. Recherche de données & Aggregations

- Sur la barre de recherche sur Kibana, on cherche : Dev tools > console :



# Tutoriel : ElasticSearch

- Comme première requête, on commence par faire l'équivalente de la fameuse : `SELECT * FROM Pokemon :`



- On peut aussi insérer un nouveau document :



## 1. Langage d'interrogation de Elasticsearch

Elasticsearch n'utilise pas SQL pour interroger les données, mais il dispose d'un langage de requête spécifique appelé *Elasticsearch Query DSL*, c'est un *Domain Specific Language* utilisé pour rechercher et récupérer les données stockées dans les index. Ce langage utilise le format JSON et est optimisé pour l'interrogation de grandes quantités de données d'une manière évolutive et efficace.

Documentation : [Query DSL](#)

## 2. Requetes

- Trouver tous les Pokémons dont le numéro de Pokédex est compris entre 100 et 150, et les trier dans l'ordre croissant.

```
GET pokemon/_search
{
  "query": {
    "range": {
      "pokedex_number": {
        "gte": 100,
        "lte": 150
      }
    }
  },
  "sort": [
    {
      "pokedex_number": {
        "order": "asc"
      }
    }
  ]
}
```

- Trouver tous les Pokémons légendaires de type "Feu" ou "Volant", triés par numéro de pokédex dans l'ordre décroissant.

```
GET pokemon/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "is_legendary": true
          }
        }
      ],
      "should": [
        {
          "term": {
            "type1": "Fire"
          }
        },
        {
          "term": {
            "type2": "Flying"
          }
        }
      ]
    }
  },
  "sort": [
    {
      "pokedex_number": {
        "order": "desc"
      }
    }
  ]
}
```

```
    }
  }
]
},
"sort": [
  {
    "pokedex_number": {
      "order": "desc"
    }
  }
]
```

- Trouver tous les Pokémons dont la taille est supérieure à 2.0 mètres, triés en ordre décroissant :

```
GET pokemon/_search
{
  "query": {
    "range": {
      "height_m": {
        "gt": 2.0
      }
    }
  },
  "sort": [
    {
      "height_m": {
        "order": "desc"
      }
    }
  ]
}
```

- Trouver tous les pokémons ayant un sexe équiprobable, triés en ordre décroissant :

```
GET pokemon/_search
{
  "query": {
    "range": {
      "percentage_male": {
        "gt": 50
      }
    }
  }
}
```

```
    },  
    "sort": [  
      {  
        "percentage_male": {  
          "order": "desc"  
        }  
      }  
    ]  
  }  
}
```

- Trouver tous les Pokémons qui ont un type primaire "Feu" **et** un type secondaire "Volant", triés par numéro de pokédex dans l'ordre croissant :

```
GET pokemon/_search  
{  
  "query": {  
    "bool": {  
      "must": [  
        {  
          "term": {  
            "type1": "Fire"  
          }  
        },  
        {  
          "term": {  
            "type2": "Flying"  
          }  
        }  
      ]  
    }  
  },  
  "sort": [  
    {  
      "pokedex_number": {  
        "order": "asc"  
      }  
    }  
  ]  
}
```



- Trouver tous les pokémons qui sont de type "Eau", ont une vitesse de base supérieure à 80, et ont une vitesse de capture inférieure à 25 :

```
GET pokemon/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "bool": {
            "should": [
              {
                "term": {
                  "type1": "Water"
                }
              },
              {
                "term": {
                  "type2": "Water"
                }
              }
            ]
          }
        }
      ],
      {
        "range": {
          "speed": {
            "gt": 80
          }
        }
      },
      {
        "range": {
          "capture_rate": {
            "lt": 25
          }
        }
      }
    ]
  }
}
```

- Trouver tous les Pokémon qui ne sont pas légendaires, ont un bonheur de base supérieur à 150, et ont un poids inférieur à 30, triés par poids en ordre croissant :

```
GET pokemon/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "is_legendary": false
          }
        },
        {
          "range": {
            "base_happiness": {
              "gt": 150
            }
          }
        },
        {
          "range": {
            "weight_kg": {
              "lt": 30
            }
          }
        }
      ]
    }
  },
  "sort": [
    {
      "weight_kg": {
        "order": "asc"
      }
    }
  ]
}
```

- Recherche de tous les Pokémon qui sont de type primaire "Combat", ont une vitesse de base supérieure à 80, une défense de base supérieure à 60, et une attaque de base supérieure à 70, triés par la moyenne de la vitesse de base, de la défense de base et de l'attaque de base par ordre décroissant :

```
GET pokemon/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "bool": {
            "should": [
              {
                "term": {
                  "type1": "Fighting"
                }
              },
              {
                "term": {
                  "type2": "Fighting"
                }
              }
            ]
          }
        }
      ],
      {
        "range": {
          "speed": {
            "gt": 80
          }
        }
      },
      {
        "range": {
          "defense": {
            "gt": 60
          }
        }
      }
    ],
  },
}
```

```
{
  "range": {
    "attack": {
      "gt": 70
    }
  }
},
"sort": [
  {
    "_script": {
      "type": "number",
      "script": {
        "lang": "painless",
        "source": "(params.speed + params.defense + params.attack) / 3",
        "params": {
          "speed": {
            "field": "speed"
          },
          "defense": {
            "field": "defense"
          },
          "attack": {
            "field": "attack"
          }
        }
      }
    },
    "order": "desc"
  }
]
}
```

→ Cette requête utilise un script de tri pour calculer la moyenne des champs *base\_speed*, *base\_defense* et *base\_attack* et trier les résultats par ordre décroissant en fonction de cette moyenne.

- Trouver le poids moyen de tous les Pokémons légendaires, et retourner seulement les Pokémons qui ont un poids dans un écart-type de la moyenne :

```
GET pokemon/_search
{
  "size": 0,
  "query": {
    "term": {
      "is_legendary": true
    }
  },
  "aggs": {
    "legendary_weights": {
      "terms": {
        "field": "name.keyword"
      },
      "aggs": {
        "weight_kg": {
          "avg": {
            "field": "weight_kg"
          }
        }
      }
    },
    "weight_kg_stats": {
      "stats": {
        "field": "weight_kg"
      }
    },
    "weight_kg_std_deviation": {
      "bucket_script": {
        "buckets_path": {
          "weight_kg_sum": "weight_kg_stats.sum",
          "weight_kg_count": "weight_kg_stats.count",
          "weight_kg_avg": "weight_kg_stats.avg",
          "weight_kg_std_deviation": "weight_kg_stats.std_deviation"
        },
        "script": "weight_kg_std_deviation",
        "gap_policy": "skip"
      }
    }
  }
}
```

→ Cette requête calcule le “*poids\_kg*” moyen de tous les Pokémon légendaires, puis calcule l'écart-type de toutes les valeurs de “*poids\_kg*” légendaires. L'agrégation “*bucket\_script*” est utilisée pour retourner uniquement les Pokémon dont les valeurs de “*poids\_kg*” sont inférieures à un écart-type de la moyenne.

- Quel est le pokémon le plus fort ?

```
GET pokemon/_search
{
  "size": 1,
  "sort": [ { "sum_of_stats": { "order": "desc" } } ],
  "script_fields": {
    "sum_of_stats": {
      "script": {
        "source": "doc['hp'].value + doc['attack'].value +
doc['defense'].value + doc['sp_attack'].value + doc['sp_defense'].value +
doc['speed'].value"
      }
    }
  }
}
```

→ Cette requête trie les documents Pokémon par ordre décroissant en fonction de la somme de leurs valeurs de “*hp*”, “*attaque*”, “*défense*”, “*sp\_attaque*”, “*sp\_défense*”, et “*vitesse*”, et renvoie le résultat le plus élevé. Le champ “*sum\_of\_stats*” est calculé à l'aide d'un champ “*script*”, qui additionne les valeurs des champs spécifiés dans chaque document.

- Trouver le *damage multiplier* moyen de chaque type contre tous les Pokémons :

PS: Un *damage multiplier*, dans le contexte des combats Pokémon, est un facteur utilisé pour déterminer la quantité de dégâts infligés par un coup. Il est déterminé par les types de Pokémon attaquant et défendant, ainsi que par d'autres facteurs tels que la puissance et la précision du coup utilisé. Le *damage multiplier* est utilisé pour calculer les dégâts finaux, en prenant en compte les statistiques du Pokémon et l'efficacité du coup utilisé. Par exemple, si un coup à un *damage multiplier* de 2x, il fera deux fois plus de dégâts qu'un coup avec un *multiplier* de 1x.

```
GET pokemon/_search
{
  "size": 0,
  "aggs": {
    "damage_multiplier_against_all_types": {
      "terms": {
        "field": "type1.keyword"
      },
      "aggs": {
        "against_all_types": {
          "avg_bucket": {
            "buckets_path": "against_.*"
          }
        }
      }
    }
  }
}
```

→ Cette requête agrège tous les champs "*against\_?*" en utilisant l'agrégation "*avg\_bucket*". Les *damage multipliers* moyens résultants sont regroupés par *type1*, qui est le type primaire du Pokémon. La syntaxe *.\** dans le "*buckets\_path*" spécifie que tous les champs "*against\_?*" doivent être inclus dans le calcul.

## Tutoriel : Elasticsearch

- Trouver les 10 meilleurs Pokémon avec le damage multiplicier moyen le plus élevé contre le type "Feu" :

```
GET pokemon/_search
{
  "size": 10,
  "sort": [
    {
      "damage_multiplier": {
        "order": "desc"
      }
    }
  ],
  "script_fields": {
    "damage_multiplier": {
      "script": {
        "source": "doc['against_Fire'].value"
      }
    }
  }
}
```

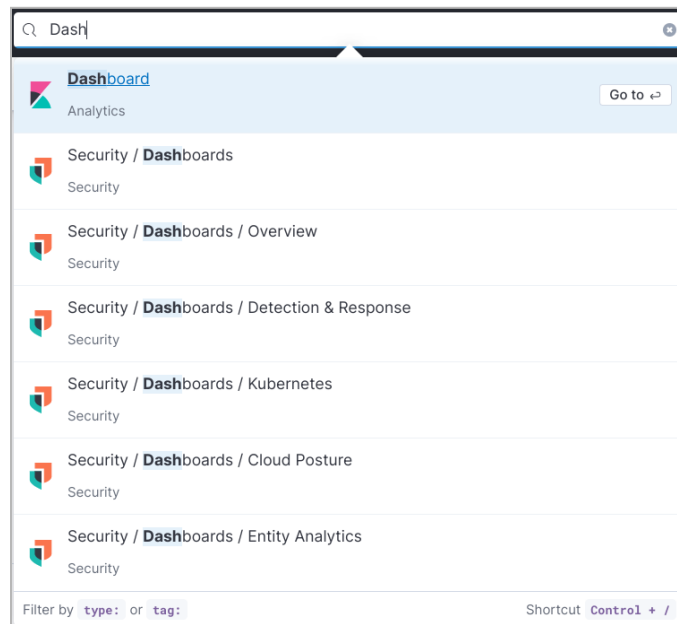
→ Cette requête trie les documents Pokémon par ordre décroissant en fonction de leur multiplicateur de dommages moyen contre le type "Feu", qui est calculé en utilisant un champ de script. La liste des 10 meilleurs Pokémon est retournée en fonction du multiplicateur de dommages moyen le plus élevé contre le type "Feu".



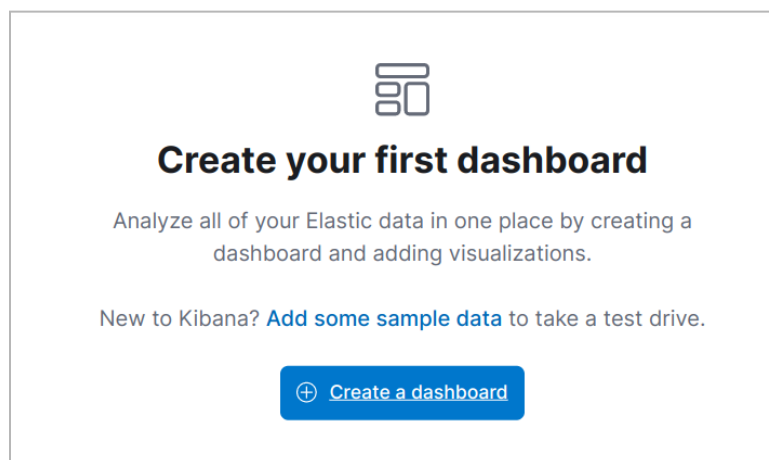
## VII. Visualisation

La meilleure façon de comprendre nos données est de les visualiser. Grâce aux tableaux de bord, nous pouvons transformer nos données d'une ou plusieurs vues de données en une collection de panneaux qui apportent de la clarté à nos données, racontent une histoire à leur sujet et nous permettent de nous concentrer uniquement sur les données qui sont importantes pour nous.

- Trouver Dashboard dans kibana :

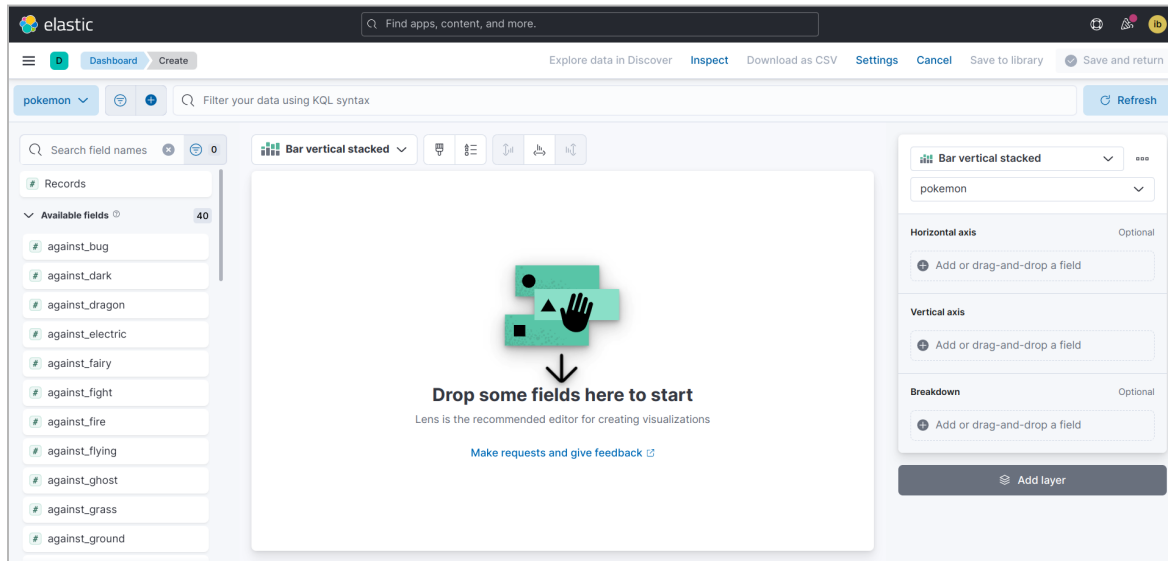


- Créer un Tableau de bord :



## Tutoriel : ElasticSearch

- Créer une visualisation :

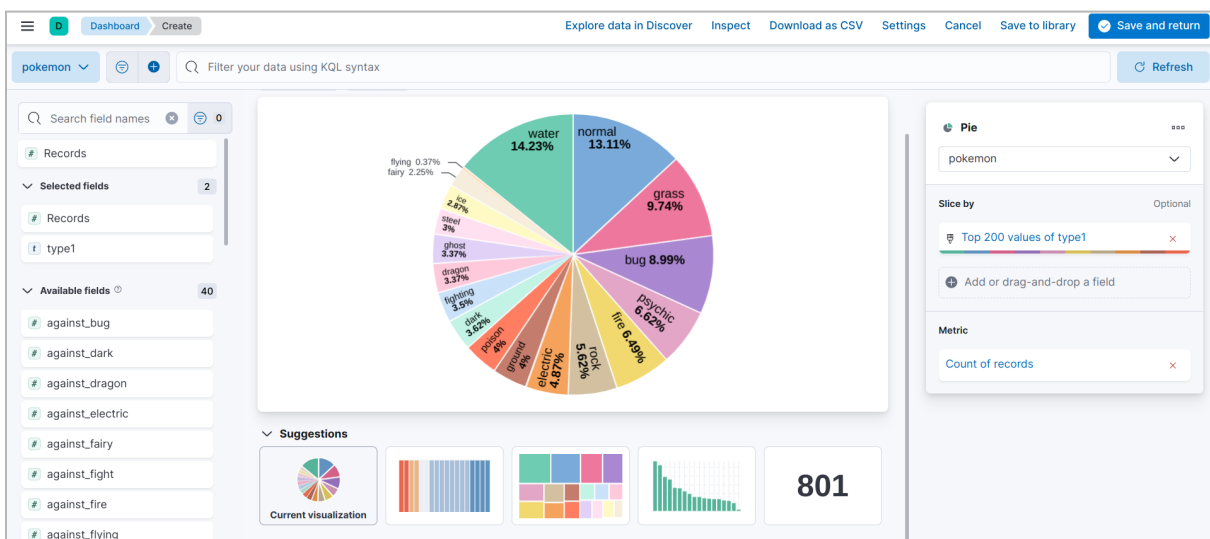


### Quelques visualisations pour notre tableau de bord

- Distribution par type :

Un graphique circulaire qui montre la distribution des Pokémon par type. Cela peut nous donner une idée des types de Pokémon les plus courants, et nous aider à identifier tout déséquilibre dans les données.

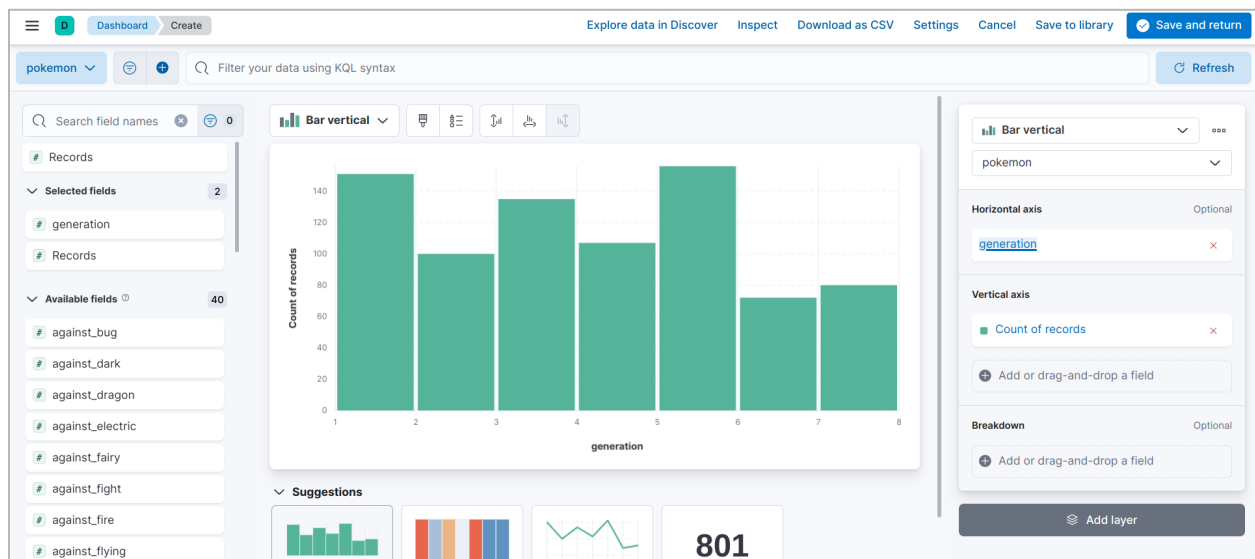
Sélectionner la visualisation "Pie Chart" > Sélectionner l'index Pokémon comme source de données > Dans la section "Split Slices", sélectionner le champ "type1".



- **Distribution des generations :**

Un graphique à barres qui montre la distribution des Pokémon par génération. Cela peut nous aider à comprendre l'évolution des Pokémon au fil du temps et à voir quelles générations ont le plus de Pokémon.

Sélectionner la visualisation "Bar Chart" > Sélectionner l'index Pokémon comme source de données > Dans la section "X-axis", sélectionner le champ "generation" > Dans la section "axe des Y", sélectionner l'agrégation "Count".

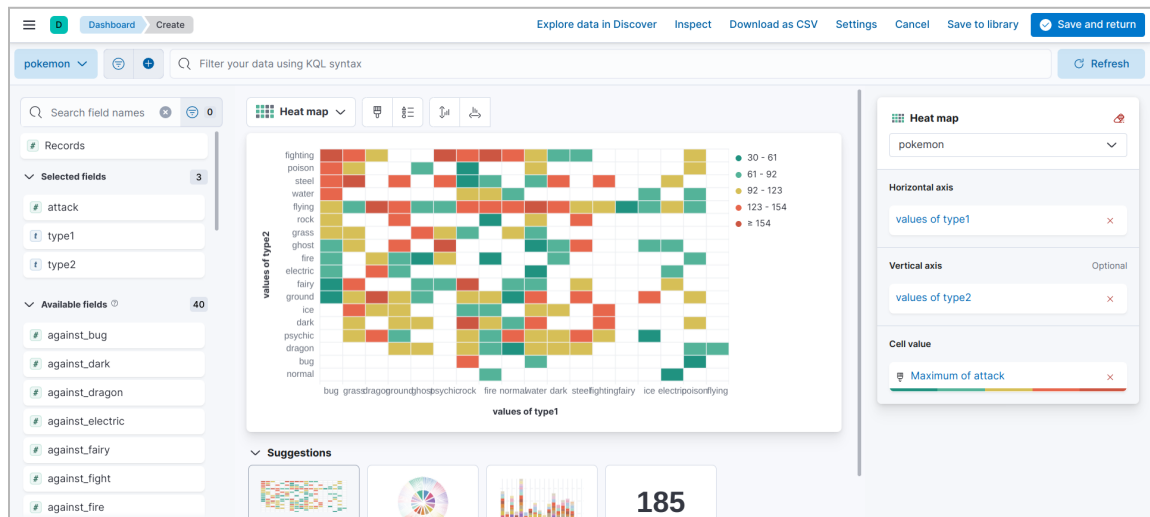


- **Heatmap d'attaques :**

On affiche la valeur d'attaque maximale de chaque Pokémon en fonction de son type primaire et secondaire. Le type primaire est affiché sur l'axe des X et le type secondaire est affiché sur l'axe des Y. L'intensité de la couleur de chaque cellule de la carte thermique représente la valeur d'attaque maximale du Pokémon avec cette combinaison de types particulière. Cette visualisation peut nous aider à comprendre quelles combinaisons de types ont des valeurs d'attaque maximale plus élevées, et à comparer l'attaque maximale de différentes combinaisons de types.

Sélectionner la visualisation "Heatmap" > Choisir l'index Pokémon comme source de données pour la visualisation > Dans la section "X-axis", sélectionner l'agrégation "Terms" pour le champ "type1" > Dans la section "Axe des Y", sélectionner l'agrégation "Termes" pour le champ "type2" > Dans la section "Metrics", sélectionner l'agrégation "Max" pour le champ "attack".

## Tutoriel : Elasticsearch



Félicitations pour avoir terminé votre voyage à travers le monde d'Elasticsearch! Vous avez appris à capturer, entraîner et faire des batailles avec vos données, et vous êtes devenu un véritable maître de la recherche et de l'analyse. Et en reconnaissance de votre dur labeur et de votre dévouement, vous avez gagné une récompense spéciale : un Pokémon rien que pour vous!



Ce Pokémon n'est pas un être ordinaire. C'est un maître puissant et féroce d'Elasticsearch, capable de rechercher rapidement et facilement à travers de vastes quantités de données avec aisance. Avec ses compétences, vous serez en mesure de relever même les défis les plus difficiles et vous sortirez toujours vainqueur :)

Alors, continuez à affiner vos compétences dans l'art d'Elasticsearch. Et n'oubliez pas, il y a toujours de la place pour l'amélioration, alors continuez à explorer et à apprendre. Le monde des données vous attend!

<https://www.elastic.co/>  
<https://www.tutorialspoint.com/elasticsearch/index.htm>