FORMATION ELK AVEC THE ELASTIC STACK V5

Présentation par

Daniel Lavoie
Talent Development Manager
+33 (0)6 45 97 73 33
daniel.lavoie@invivoo.com

Déroulement

Jour 1

Introduction
Elasticsearch
Logstash
Kibana
Beats

Jour 2

Administration

Jour 3

Bonnes pratiques
Mise en pratique d'une architecture avancé

Plan de cours

Introduction

L'écosystème d'Elasticsearch

Le rôle d'Elasticsearch, Logstash, Kibana et Beats

Simplifier la gestion des versions avec The Elastic Stack V5

Principes et fonctionnement

Exemples d'architectures

Cas d'utilisations

Plan de cours

Elasticsearch

Introduction à Elasticsearch
Indexation et recherche
Analyse de données
Mappings et configuration de l'analyse
Requêtage avec Elasticsearch
Système de plugins & Configuration
Queries et Filters
Agrégations
Réplication et partitionnement
Serveur ElasticSearch
Mettre en place un cluster
Les rôles des noeuds

Plan de cours

LOGSTASH

Concepts: Input, Output, Filter (filtre), Codecs…
Les Inputs: File, Redis, RabbitMQ…
Les Filters: Grok, Date, Mutate…
Les Outputs: File, Elasticsearch, Redis…
Threading et haute-disponibilité

KIBANA

Installation et configuration
Découverte des données et construction des requêtes / Queries
Agrégations et construction de Visualizations
Panels
Création des vues
Mise en place d'un tableau de bord

Plan de cours

BEATS

Introduction aux Data Shippers et au monitoring temps réel
Monitorez votre réseau grâce à PacketBeat
Monitorez vos fichiers grâce à FileBeat
Monitorez vos Windows event logs grâce à WinlogBeat
Récupérer les métriques importantes de vos serveurs grâce à Metricbeat

ADMINISTRATION

X-Pack
Sauvegardes : Snapshots et Restore
Monotiring du cluster
Alerting
Tuning et architectures avancés
Templates et indices

Plan de cours

Bonnes pratiques

Configuration pour la production
Architectures avancé
Consignes sur le hardware
Gestion de données timeseries

L'écosystème d'Elasticsearch

elasticsearch

logstash

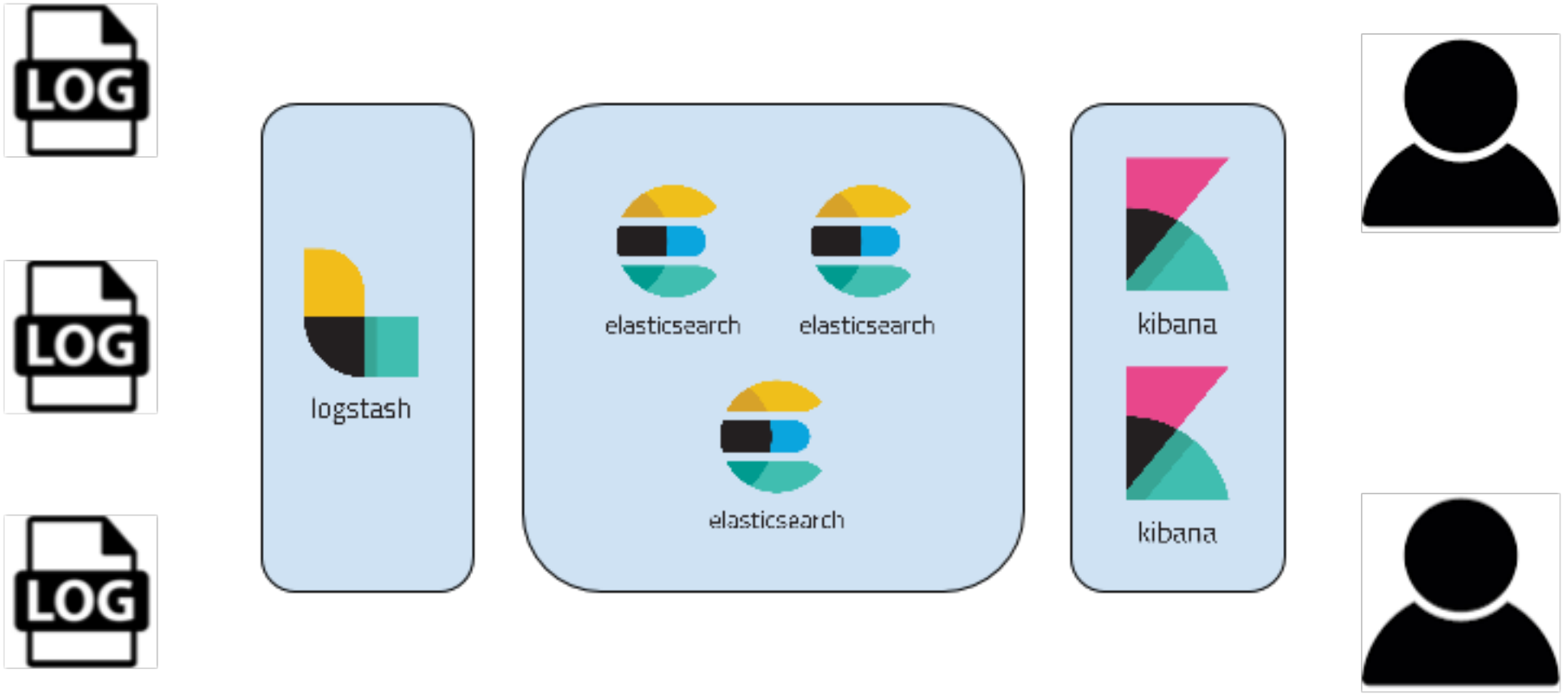
beats

kibana

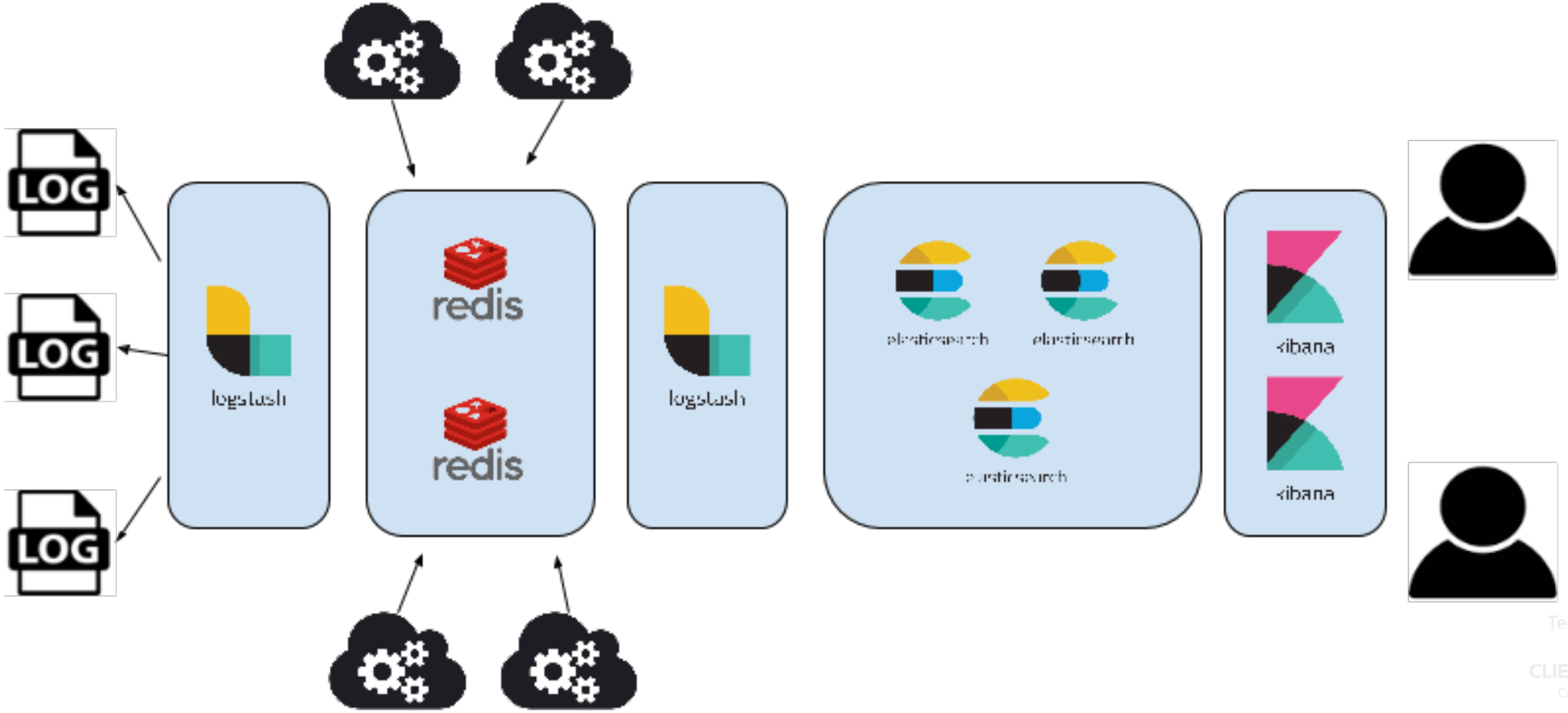
Le rôle d'Elasticsearch, Logstash, Kibana et Beats

- **Elasticsearch**
  - Base de données distribuée orientée document
  - Moteur de recherche Full Distrib
- **Logstash**
  - Pipeline d'intégration de données
- **Kibana**
  - Plateforme de visualisation pour Elasticsearch
- **Beats**
  - Famille d'agent collecteur de données

Exemples d'architectures - Architecture Simple



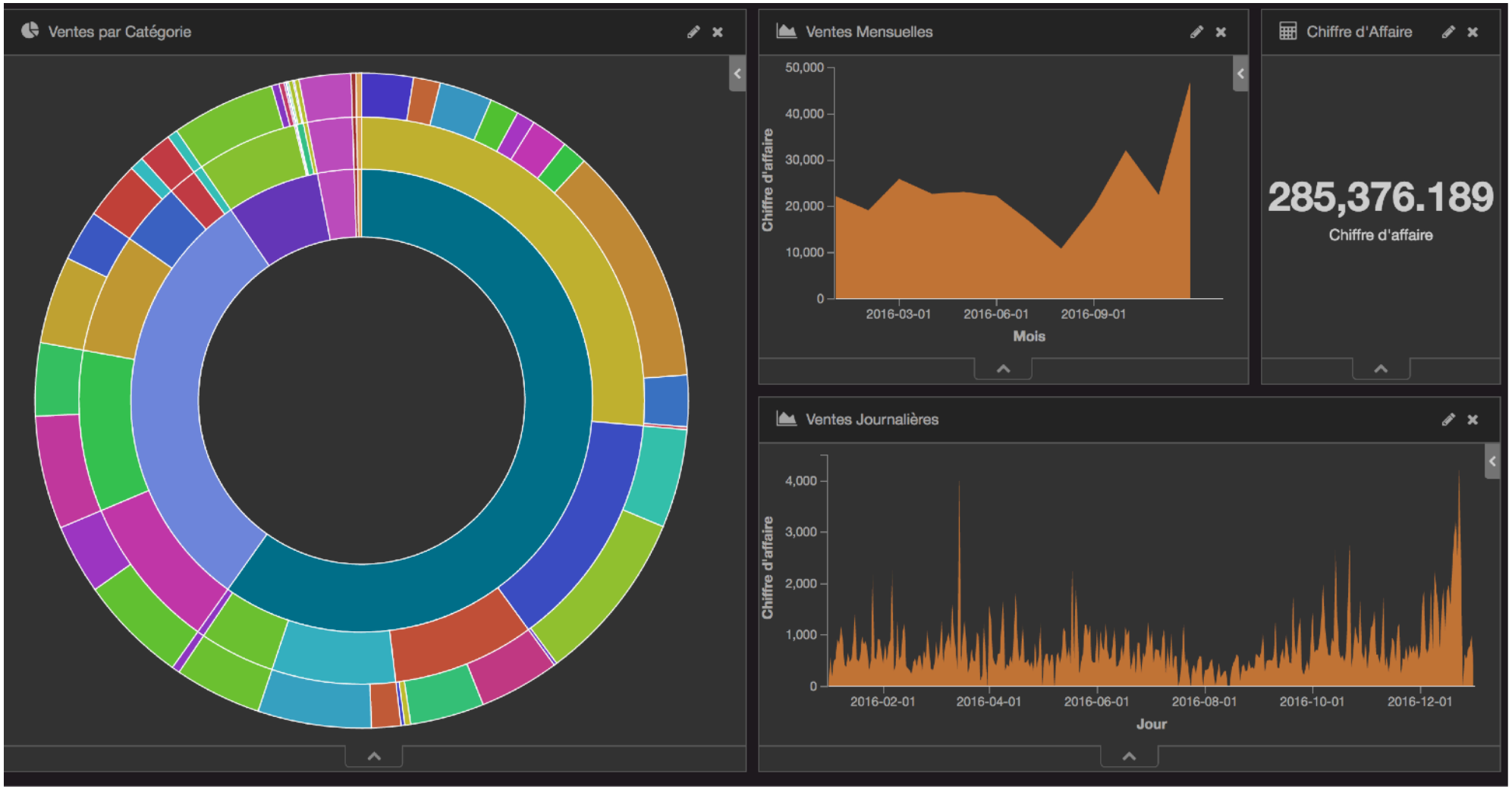
Exemples d'architectures - Architecture Découplé



Cas d'utilisations - Recherche de logs

Cas d'utilisations - Monitoring en temps réel

Cas d'utilisations - Business Intelligence



Simplifier la gestion des versions avec The Elastic Stack V5

- **Mise à niveau de la nomenclature de tous les composants de la stock**
- **Publication simultané de tous les composants de la stock**

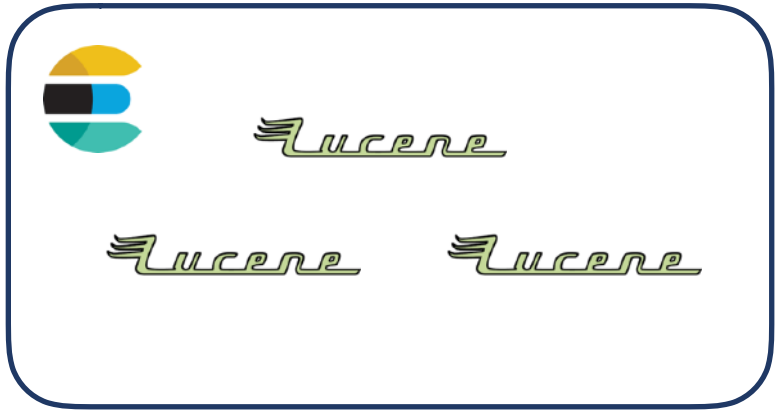
elasticsearch

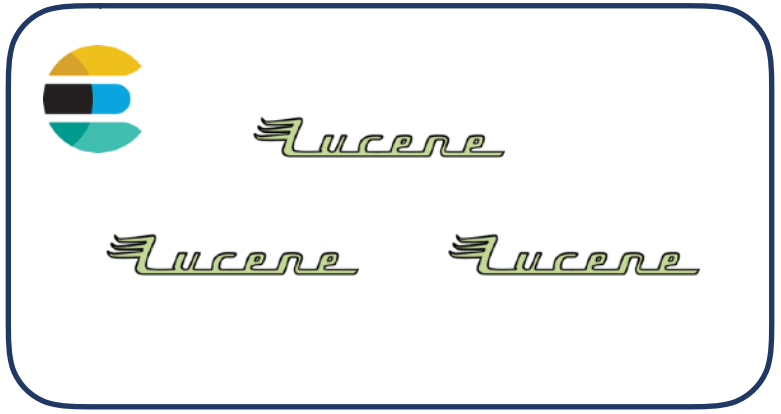
Introduction à Elasticsearch

- **Elasticsearch est une base de donnée orienté document**
  - voir : https://en.wikipedia.org/wiki/Document-oriented_database
- **Les données sont inséré et extraites au format JSON.**
- **Par défaut, tous les champs sont indexé, trié et disponible à la recherche full text.**
- **Toute donnée stocké est immuable.**

Coeur d'Elasticsearch

- **Basé sur Apache Lucene**
  - Pour plus d'infos : http://lucene.apache.org/core/
- **Cache la complexité de Lucene pour offrir un système distribué simple et efficace**





API

- **Native : Client Java avec 2 modes de communication**
  - Node Client : Client se joignant au cluster
  - Transport Client : Client se reposant sur 1 noeud pour toute communication
- **HTTP : Api polyglot RESTfull**

  •Utilisé par les clients non Java. Les outils d'administrations se repose généralement sur l'API HTTP.

Node Client vs Transport Client

Node Client

- Membre à part entière du cluster
- Connait la topologie du cluster
- Coût en ressource plus élevé
- Requêtes plus performantes

Transport Client

- Délègue toute requête à un noeud du cluster
- Moins couteux en ressource
- Requêtes moins réactives

Composition d'un cluster - Node

Noeud (Node)

Un noeud correspond à une instance d'Elasticsearch.

Dans un cluster, chaque noeud est égal.

Aux yeux d'un client, tout noeud peut répondre à tout type de demande.

L'ajout ou la suppression de noeud est transparent. Il suffit de démarrer ou d'arrêter les noeuds en questions.

Composition d'un cluster - Shard

Shard

Représente un bloc des données d'un index.

Deux type de shard : *Primary* et *Réplica*

Techniquement, un shard représente une instance de Lucene.

Par défaut, un shard primary se doit toujours d'avoir un réplica présent sur un autre noeud.

Les Réplicas peuvent répondre aux requêtes de lecture afin d'alléger les IO sur le shard primaire.

Démonstration - Création d'un cluster

Exercice - Installation d'Elasticsearch

Gestion de la configuration

Les configurations d'Elasticsearch sont applicable de deux manières :

Arguments au lancement

Toutes les configurations prévus pour le système peuvent être passé par paramètre de JVM.

Ex : elasticsearch -Ecluster.name=superb-cluster

Fichier de configuration

Le fichier **conf/elasticsearch.yml** permet de centraliser toutes les configurations possible sur le noeud du cluster.

Note importante

Il est impératif de prévoir un outil de provisioning type Chef, Puppet ou Ansible pour le maintient des configurations. Une gestion manuelle peut vite devenir ingérable avec un Cluster contenant beaucoup de noeud.

Principales configurations

cluster.name

Nom de cluster recherché par le noeud à son démarrage. Tous les noeuds tentant de former un cluster doivent utiliser le même nom de cluster.

node.name

Nom du noeud. Un nom aléatoire sera généré si non spécifié.

path.data

Par défaut, Elasticsearch persiste ses données parmi le répertoire **data** de la distribution. Il faut donc changer cette configuration pour éviter de perdre les données sur une monté de version d'Elasticsearch.

path.logs

Répertoire dans lequel les logs d'Elasticsearch seront écrit.

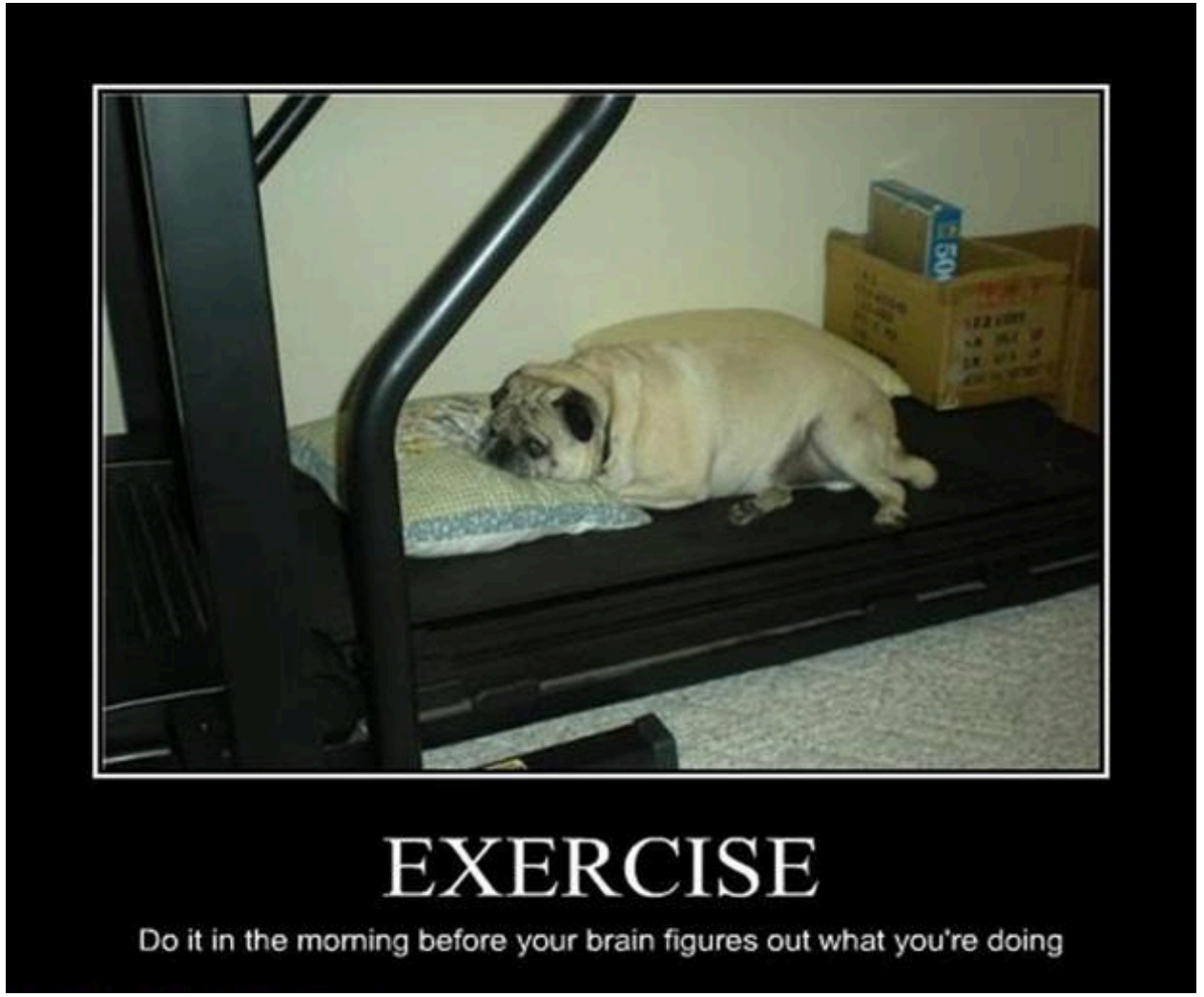
path.plugins

Tout comme les données, les plugins sont persisté parmi les binaires de la distribution. Il est donc utile de configurer ce paramètre pour pouvoir garder ses plugins après une monté de version.

discovery.zen.ping.unicast.hosts

Liste des noeuds appartenant au cluster. C'est avec cette liste que sera construit le Cluster.

Exercice - Création d'un cluster



Terminologie

- **Index :** Représente une collection de document de la même famille (l'équivalent d'une table SQL)
- **Type :** Détermine les champs disponible dans l'index (l'équivalent d'une structure de table SQL).
- **Document :** Représente une donnée stocké dans un index (équivalent d'une ligne de table SQL).

L'action d'insérer un nouveau document dans un index se nomme indexer.

Requêtages sur Elasticsearch

- **GET = Recherche**
- **PUT / POST = Création / Mise à jour**
- **DELETE = Suppression**

Structure

VERBE http://***serveur***:***port-http***/***index***/***type***/***identifiant***

Exemple

GET http://localhost:9200/users/v1/123123412

Types de recherches

- Recherche par identifiant
- Recherche Lite
- Recherche avec Query DSL

Recherche par identifiant

GET */index/type/id*

Exemple

GET /users/v1/1829183210

Recherche Lite

GET /***index***/***type***/_search?q=**query**

Exemples

GET /address/v1/_search?q=nom_comm:seine

GET /address/v1/_search?q=+nom_comm:seine%2B+numero:23

Très puissant mais difficile à lire de par l'encodage HTTP.

GET /person/_search?q=%2Bname%3A(mary+john)+%2Bdate%3A%3E2014-09-10

Pour toute les options de recherche :

https://www.elastic.co/guide/en/elasticsearch/guide/current/search-lite.html

Recherche avec QueryDSL

- **Le QueryDSL est un language de recherche expressif utilisé par Elasticsearch pour exposer toutes les fonctionnalités de recherche offerte par Lucene à travers une interface JSON.**

Exemple

```
GET /_search
{
    "query": {
        "match": {
            "tweet": "elasticsearch"
        }
    }
}
```

Recherche avec QueryDSL - Contexte

Une requête QueryDSL propose deux contextes applicable à nos paramètres de recherche :

- **Contexte de recherche :**
  - Tout critères spécifiés dans un contexte de recherche contribuera à augmenter ou diminuer le score de pertinence des documents.
- **Contexte de filtre :**
  - Tout critères spécifié dans un contexte de filtre aura comme impact d'exclure ou d'inclure des documents dans la recherche.

Recherche avec QueryDSL - Exemple de Query

```
GET index/type/_search
{
   "query": {
      fonction: {
         argument1: valeur1,
         argument2: valeur2
      }
   }
}
```

Exemple

```
GET /_search
{
   "query": {
      "match": {
         "nom_comm": "seine"
      }
   }
}
```

Recherche avec QueryDSL - Exemple de Filtre

```
GET index/type/_search
{
   "query": {
      "bool": {
         "filter": {
            fonction: {
               argument1: valeur1,
               argument2: valeur2
            }
         }
      }
   }
}
```

Exemple

```
GET index/type/_search
{
   "query": {
      "bool": {
         "filter": {
            fonction: {
               argument1: valeur1,
               argument2: valeur2
            }
         }
      }
   }
}
```

Recherche avec QueryDSL - Fonctions

- **match :** Applique une recherche Full-Text sur les champs spécifié
- **multi_match :** Permet d'appliquer la fonction match à plusieurs champs simultanément.
- **range :** Propose plusieurs opérateurs gt, gte, lt et lte permettent de déclarer l'intervalle de recherche.
- **bool :** Permets la construction de conditions mutuellement inclusives ou exclusives

Pour une liste exhaustive des fonctions :

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

Exercices - Recherche

Recherche par identifiant

Recherche lite

Recherche QueryDSL

Indexations

Insertion avec identifiant

```
PUT /{index}/{type}/{id}
{
  "field": "value",
  ...
}
```

Insertion avec Identifiant Auto-Généré

```
POST /{index}{type}
{
  "field": "value",
  ...
}
```

Modifications en lots

Insertion bulk

{ *action*: { *metadata* }}\n
{ *request body* }\n
{ *action*: { *metadata* }}\n
{ "message" : "Test d'insertion builk 1" }
{ "create": { "_index": "test-bulk", "_type": « test-bulk",}}
{ "message" : "Test d'insertion builk 2" }
…

Actions supporté

create : Créer un document seulement s'il n'existe pas.
index : Créer ou remplace un document.
update : Mise à jour partiel d'un document
delete : Suppression du document.

metadata doit contenir l'index, le type et l'id du document à insérer.

Mapping et Analyse

Toutes les données stockées dans Elasticsearch sont systématiquement indexées. Les critères de recherche seront comparés au contenu de l'index.

Les informations qui sont stockés dans cet index sont donc cruciaux.

Elasticsearch permet d'influencer comment nos données seront représenté dans l'index. Ceci permet donc d'agir sur le comportement attendu en fonction des mots clefs saisis pour la recherche.

Analyse

Action de prendre une chaine de recherche et de la décomposer en élément de comparaison pour l'index.

L'analyse se produit lors de l'indexation d'une donnée dans l'index et lors de la comparaison d'un mot clef de recherche avec le contenu d'un index.

L'action d'analyse permet de mettre les documents stocké et ceux recherché sur le même pied de comparaison.

Mapping

Le mapping correspond aux instructions d'analyse pour chaque champs à comparer lors de la recherche.

Le mapping détermine comment un champs sera stocké dans l'index.

Il est impossible de modifier un mapping sans devoir ré indexer toutes les données de l'index impacté.

Agrégations - Concepts

Bucket

Résultat d'un groupe agrégé.

Metrics

Valeur résultant d'un calcul d'agrégation. Applicable sur une requête globale ou sur un bucket.

Agrégations - Exemple

```
GET /address/address/_search
{ "aggs": {
    "adresse_par_ville": {
      "terms": {
        "field": "nom_comm"
      }
    }
  }
}
```

Les agrégations sont sensible aux définitions du mapping. Pour obtenir une agrégation sur un champs sans qu'il soit découpé, il faut prévoir désactiver l'analyse sur le champ à agréger.

Ne pas hésiter à se référer à la documentation de référence Elasticsearch pour identifier toutes les fonctions d'agrégations.

https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html

En route pour la Prod

Mémoire

- Très important, suffisamment de mémoire signifie moins d'IO. Ultimement, tout finit sur disque, mais plus il y a de mémoire, moins il y a d'aller retour avec les disques.
- 64Gb représente un juste milieu pour une instance.
- 8Gb est un minimum. Créer un cluster basé sur des machines plus petite se traduit en perte d'efficacité.

CPU et Réseau

CPU

Les processeurs modernes suffisent amplement à répondre aux besoin d'Elasticsearch. Il est très peu gourmand en temps CPU. Il est préférable d'opter sur un maximum de coeur possible car la capacité de traitement concurrentiel est préférable à la puissance de calcul brute.

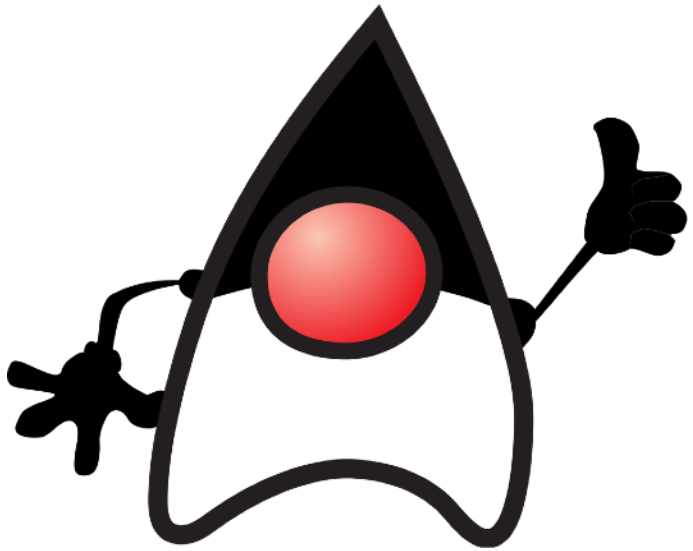
Réseau

Un cluster construit sur un réseau 1Gb ou 10Gb suffit amplement. Il est a noter qu'Elasticsearch ne gère pas nativement la localisation des données par Datacenter. Des techniques avancé mais plus complexes permettent de dupliquer les données sur différents cluster élasticsearch déployé dans différent Datacenter.

Et la JVM dans tout ça ?

Il est recommandé de toujours utilisé la dernière version du JDK (Oracle ou OpenJDK). Lucene est une librairie très complexe qui a dévoilé énormément de bug sur les dernières versions du JDK.

Il est recommandé que les clients Java qui utilisent les API native (Transport Client ou Node Client) tournent avec une JVM identique à celle des noeuds du cluster. Oracle a tendance à impacter les mécanismes de sérialisation même sur des versions mineures.



Et si je veux ABSOLUMENT ajuster la JVM ?

Les paramètres de lancement de la JVM sont spécialement optimisé pour les besoins d'Elasticsearch. Les paramètres utilisé par Elasticsearch viennent de cas d'études sur des cluster gigantesques.

Un problème de performance sur Elasticsearch se résume très souvent à ajouter des noeuds ou à revoir la structure des données indexés.

On touche pas à la JVM, sinon…

Gestion de la mémoire par Elasticsearch

L'utilisation de la mémoire vive par Elasticsearch se gère en deux temps. L'instance applicative d'Elasticsearch travaille avec la heap tandis que les instances Lucene qu'elle utilisent, elles, travaillent avec la non heap.

Par conséquent, il est normalement conseillé de configurer une heap qui permet de laisser libre autant de mémoire pour la non heap.

La heap peut être configuré par la variable d'environment ES_HEAP_SIZE.

Que reste t'il pour optimiser le cluster ?

- **SSD ! SSD ! SSD !!!!**
- **Insertion en lots (attention à la taille des requêtes).**
- **Pas de RAID mirroring**
- **Augmenter le Refresh Interval lorsque l'on peut se permettre un décalage entre les insertions et les écritures**

logstash

Logstash

Concepts

Lancement de logstash

Composition d'un fichier de configuration

Input plugins

Filter plugins

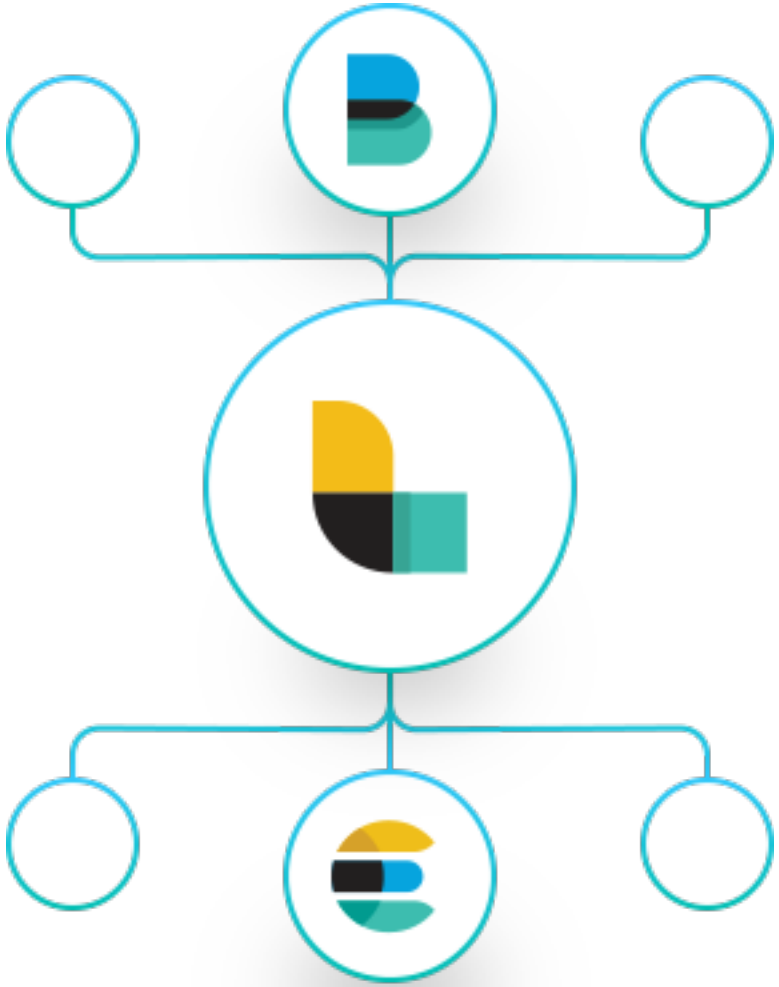
Output plugins

Concepts

Logstash a originalement été conçu pour récolter des logs systèmes. Cependant, son système de plugins en fond un pipeline de traitement de données très flexible.

Le système modulaire composé de plugins d'entrés, de sorties et de filtres permet d'ingérer des données de n'importe quel source, de les transformer puis de les envoyé vers n'importe quel système en sortie.

La déclaration des plugins prend en considération un méta langage, propre à Logstash, qui permet de définir des conditions sur les différents plugins déclaré dans le pipe line.



Lancement de Logstash

Logstash s'exploite sous forme d'outil en ligne de commande. Il suffit de le lancer en spécifiant le fichier contenant la définition et les configurations du pipeline d'intégration de donnée que l'on désire construire.

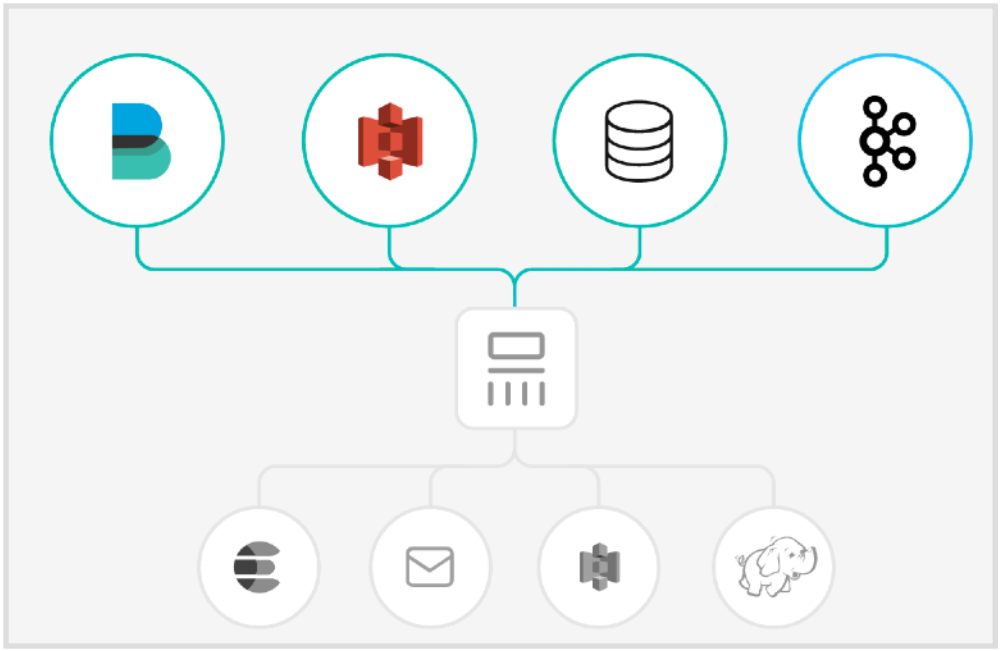
Exemple : bin/logstash -f logstash-logparser.conf

Input Plugins

Un input filter s'occupe d'écouter une source de donnée. Voici quelques exemple de source :

Liste complète et documentation de Référence pour tous les plugins :

https://www.elastic.co/guide/en/logstash/current/input-plugins.html



Exemple - File Input Plugin

```
input {
  file {
    path => "/tmp/access_log"
    start_position => "beginning"
  }
}
```

Exemple - JDBC Input Plugin

```
input {
  jdbc {
    jdbc_driver_library => "mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://localhost:3306/mydb"
    jdbc_user => "mysql"
    parameters => { "favorite_artist" => "Beethoven" }
    schedule => "* * * * *"
    statement => "SELECT * from songs where artist = :favorite_artist"
  }
}
```

D'autres Input Plugins

elasticsearch

exec

eventlog

file

github

http

irc

imap

jdbc

jmx

kafka

rss

rabbitmq

redis

s3

syslog

tcp

twitter

unix

udp

websocket

zeromq

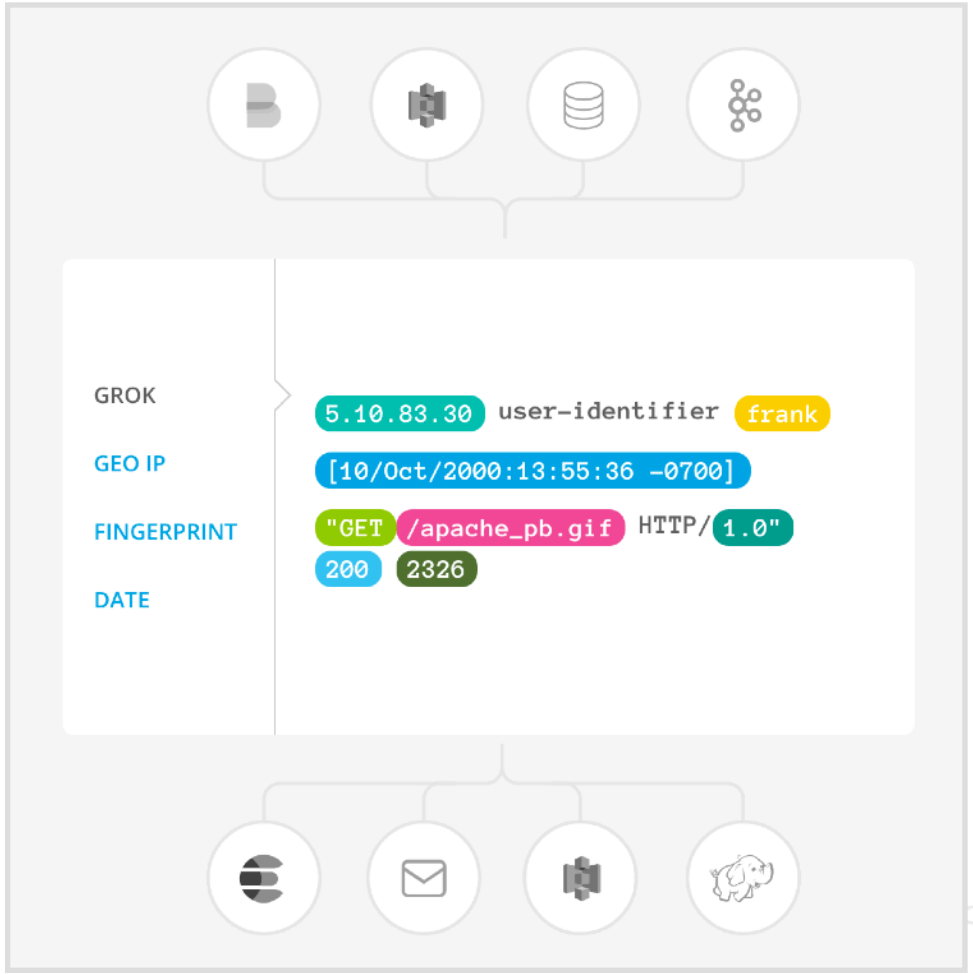
Filter plugin

Les plugins de type Filter recevront toutes les données provenant des différents Input Plugin.

Les filtres ont sont généralement utilisé pour formater certaines données avant leur insertions

Références technique sur tous les filtres disponible :

https://www.elastic.co/guide/en/logstash/current/filter-plugins.html



Les filtres classiques

grok

csv

geoip

throttle

mutate

anonymize

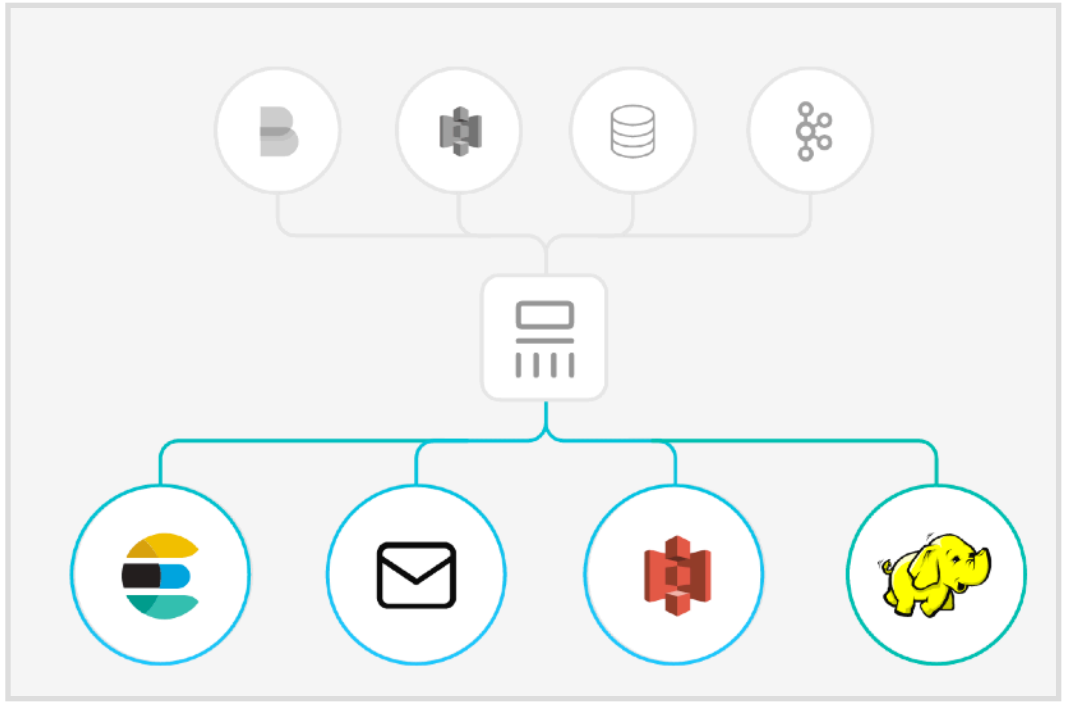
xml

Output Plugin

Correspond à la dernière partie du pipeline d'intégration de données. Les plugins output prennent les sorties des filtres et les exporteront vers les destinations configurés dans cette section.

Références technique sur tous les Output Plugin :

https://www.elastic.co/guide/en/logstash/current/output-plugins.html



Nouveauté sur la V5 de Logstash

Enfin des outils de monitoring !
https://www.elastic.co/fr/blog/logstash-5-0-0-released

Doc de référence :

https://www.elastic.co/guide/en/logstash/current/monitoring.html

Beats

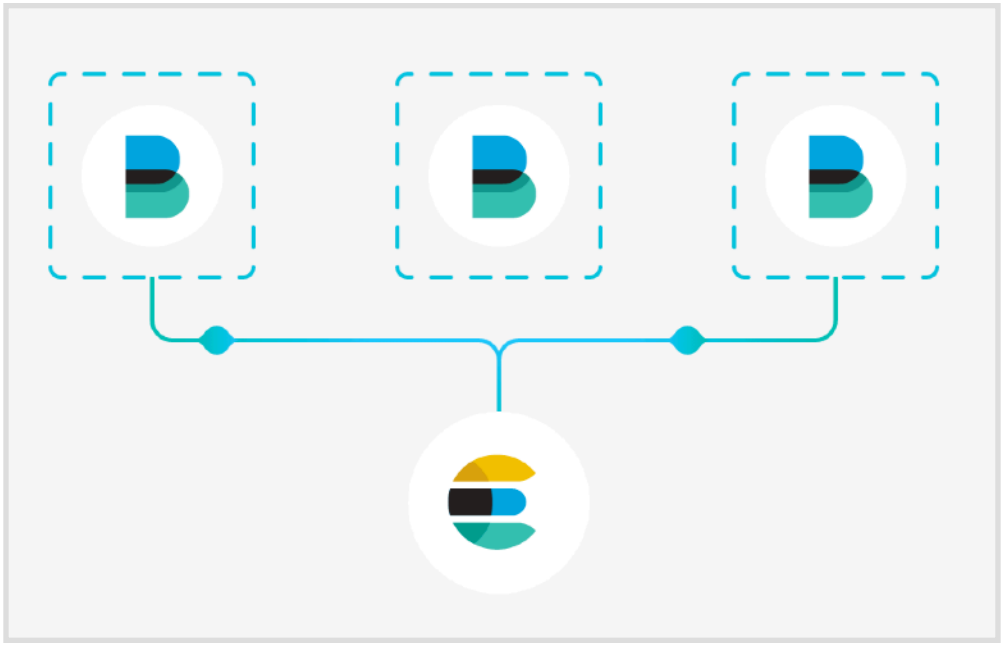
beats

Beats - Concepts

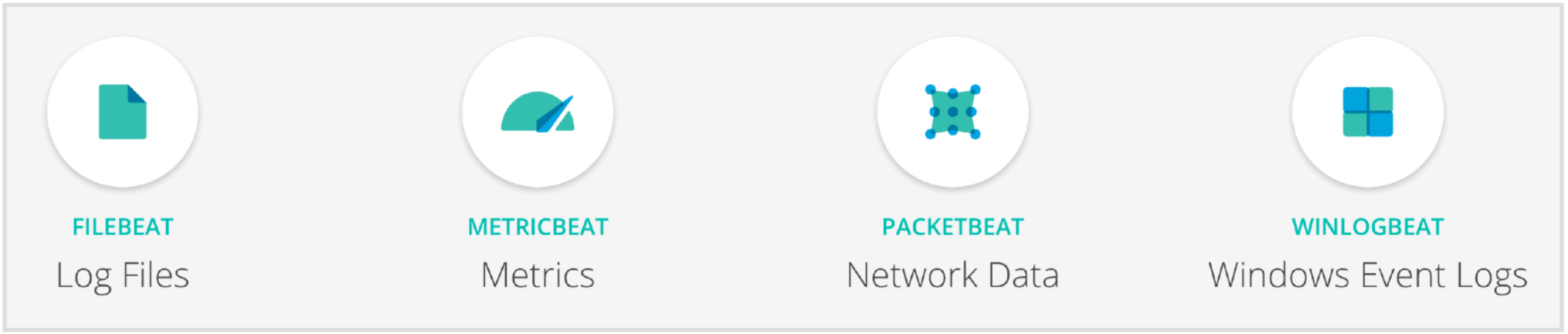
Beats est un agent léger permettant d'expédier des données brutes vers Logstash ou directement vers une instance Elasticsearch. Il est beaucoup plus performant et efficace que Logstash.

Beats est un framework.

Il faut installer un binaire pour chaque type de données que nous voulons injecter dans Elasticsearch



Variantes de Beats



Des modules pour tous les besoins