Une invitation à

Elasticsearch Logstash Kibana

Auteur : Pascal SALAUN

En date du 06/08/2017

Table des matières

Préambule

Je ne suis pas un expert ELK, que ce soit dit.

Je ne vais donc pas vous expliquer les subtilités dans la configuration d'une plateforme Elasticsearch gérant les méta-données d'un DataCenter.

Ce document n'a pour seul objectif que de vous initier à ELK.

Enfin, n'étant pas Prof, je ne suis pas contraint au style académique. Il y a les RFC pour ça.

C'est bon, on peut y aller ?

Et bien, on attaque !

Les pré-requis

Avant d’attaquer les installations, il faut s’assurer que :

- **java** soit **bien installé**. La commande ‘java -version’ doit vous retourner des informations
- les **locales** « **en_us** » soit bien **installées**

1 – ELK, c’est quoi ?

Avant d’attaquer l’installation et la configuration des briques, il est souhaitable de faire un petit rappel de ce qu’est ELK.

La triplette « Elasticsearch Logstash Kibana » constitue une plateforme de gestion d’informations « légères ». Elle est principalement utilisée pour stocker et gérer des logs, des métriques (cpu, loadaverage, network trafic…), des informations textuelles.

Donc pas de streaming audio, video...

Si vous vous avez adoptée la solution Bluemind, vous utilisez Elasticsearch, ne serait-ce pour la recherche textuelle dans vos messages.

Mais quel composant fait quoi ?

1.1 - Elasticsearch

C’est un moteur d’indexation. Pour faire simple, c’est une base de données dont le but est l’accès rapide aux informations. Et comme les SGDB dignes de ce nom, Elasticsearch est configurable en cluster multi-noeuds, permet la répartition géographique des data...

Pour mémoire, un nœud est une instance, un service qui tourne et écoute sur un port particulier. Vous pouvez, si vous êtes riches, avoir des machines qui hébergent plusieurs instances.

Mais, en principe, un nœud correspond à une machine.

Les données sont consultables via des requêtes REST, donc au format JSON.

1.2 - Logstash

C’est un concentrateur d’appels, le terme usité est « pipeline »,

Le but de Logstash est d’intercepter toutes les requêtes JSON à destination d’Elasticsearch pour les transmettre ensuite au(x) serveur(s) Elasticsearch.

Il a l’avantage de fournir des outils de traitement sur ces données qui lui sont envoyées. On peut, ainsi, modifier un message JSON en lui ajoutant, par exemple un autre champ « nécessaire » aux traitements finaux.

Il n’est pas nécessaire, Elasticsearch et Kibana suffisent dans la plupart des cas.

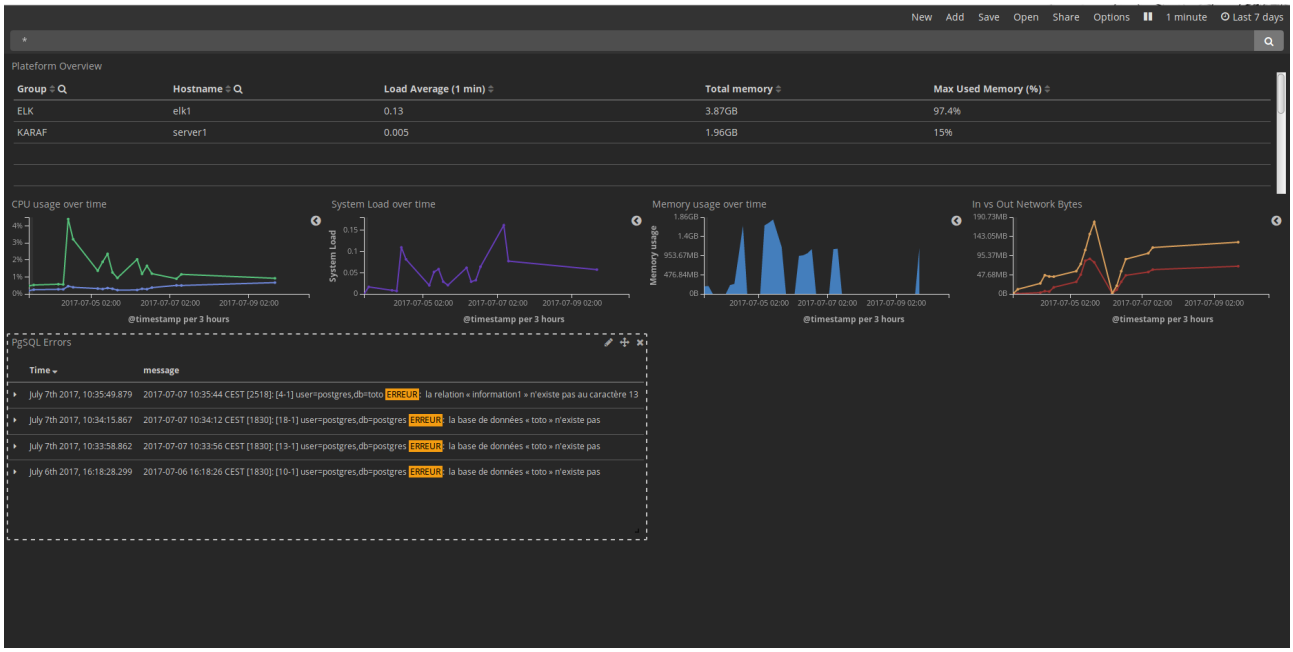
1.3 - Kibana

C’est une interface graphique permettant l’accès aux données stockées dans Elasticsearch .

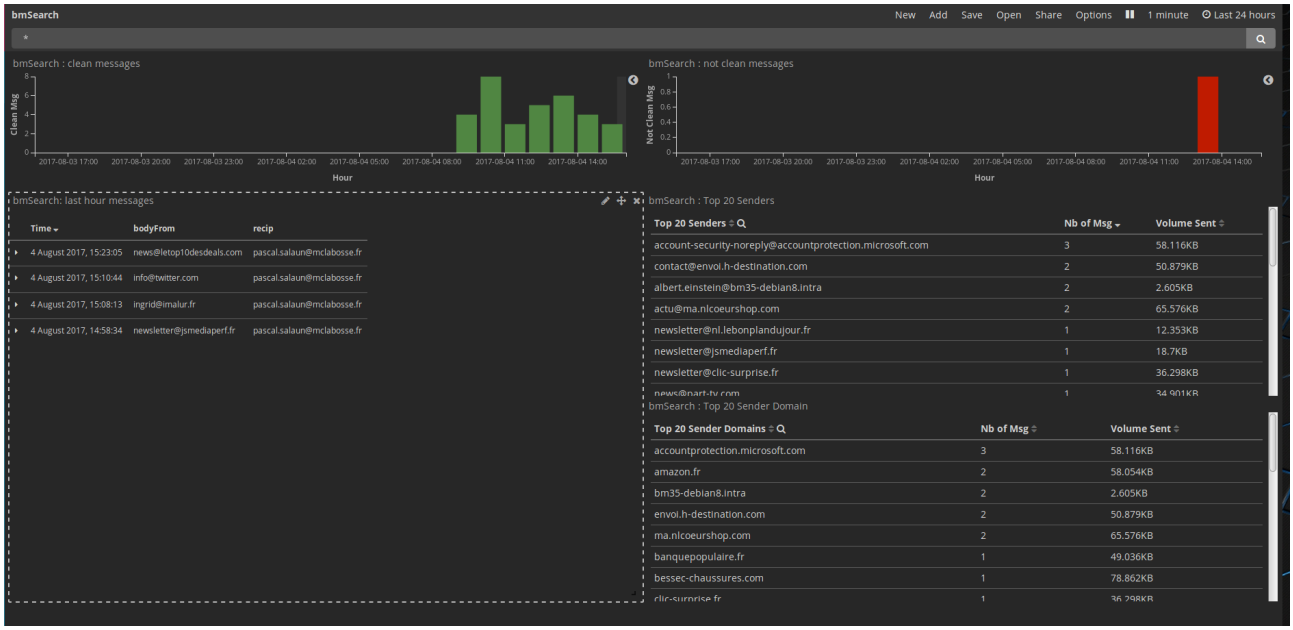
Cette interface fournit , à minima :

- un accès rapide aux données,
- des outils de requêtes déjà programmés,
- des outils de création de requêtes,
- des « dashboards» pouvant afficher graphiques, listes d’évènements

En gros quelque chose comme ceci :



ou



C’est cette vue que je vous propose avec bmSearch !

1.4 - Les « beat »

Ils s’agit de composants annexes, des plugins, que l’on installe et configure sur les serveurs pourvoyeurs d’informations. Elles sont ensuite poussées soit vers Elasticsearch, soit vers Logstash.

Nous nous contenterons de :

- **metricbeat**, qui récupère les infos générales d’une machine (CPU, RAM…)
- **filebeat**, qui scrute des fichiers de logs

Mais il en existe d’autres :

- **packetbeat**, orienté « réseau »
- **winlogbeat**, orienté windows
- **heartbeat**

2 – Installation des beats

Vous devrez installer ces composants sur toutes les machines de vos différentes infrastructures.

Vous devez être cohérent quant à la version des beats. Elle doit être identique à celle du cluster ELK.

2.1 - Metricbeat

Vous aurez pris soin de télécharger la version qui vous convient sur le site « elastic.co ».

Pour installer le paquet, utilisez soit « rpm », soit « yum » sur Redhat/Centos ou « dpkg » sur Debian/Ubuntu.

L'installation faite, les principales informations sont stockées à 2 endroits.

2.1.1 - La configuration

Les éléments sont situés dans **« /etc/metricbeat/ »**.

Ce dossier contient :

- un fichier de configuration standard/passe-partout (metricbeat.yml)
- un fichier complet de configuration (metricbeat.full.yml)
- deux fichiers json contenant la structure du « schéma » à intégrer dans Elasticsearch

Le fichier « metricbeat.yml » doit être configuré comme indiqué dans l' « **ANNEXE 1 ».**

Les principaux éléments à configurer sont :

« ***name:*** » , c'est à dire le nom facilement répérable dans Elasticsearch. Dans le cas présenté, ce champ a pour valeur «*ELK.KIBANA*», soit le schéma « <plate-forme>.<rôle|application> »

« ***output.***», c'est à dire vers quoi vous voulez envoyer les données, Logstash ou Elasticsearch. Soyons fous et envoyons-les vers un Logstash

```
#---------------------------- Logstash output ----------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.122.195:5044"]
  timeout: 300s
```

2.1.2 - Les traces

Par principe/défaut, un beat trace son activité dans « /varl/log/ <beatname>/ », ce qui donne le fichier « **/var/log/metricbeat/metricbeat** »

2.1.3 - Pousser le schéma « metricbeat.yml »

Lors d'une primo installation, il faut déclarer le schéma, comme sur un SGDB classique.

Soit la commande magique, depuis le dossier « /etc/metricbeat » :

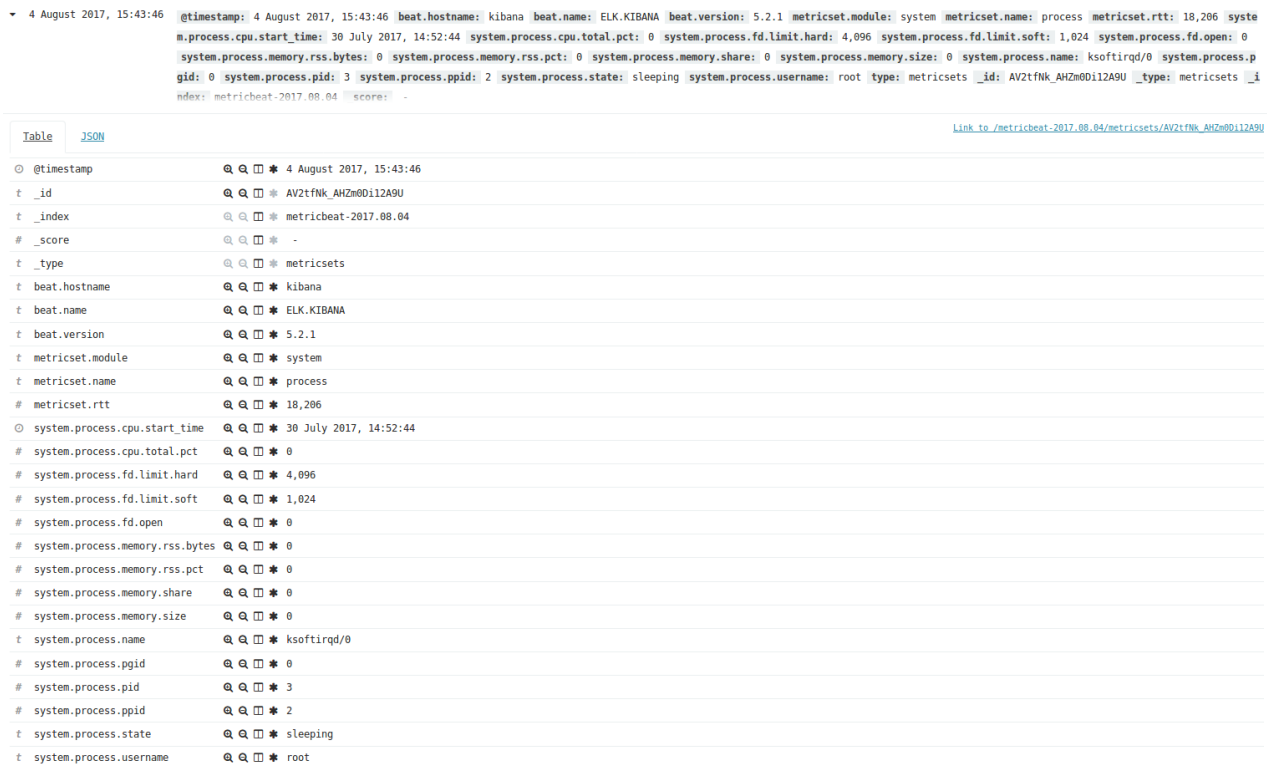
curl -XPUT 'http://<serverElastic>:<portElastic>/_template/metricbeat?pretty' -d@metricbeat.template.json

Et vous aurez une belle réponse :

```
{
  "acknowledged" : true
}
```

2.1.4 - Le résultat dans Elasticsearch

Voici comment est affiché dans Kibana une donnée venant d'un metricbeat



2.2 - Filebeat

Si installer « metricbeat » est simple, il en est de même pour « filebeat ».

Après avoir téléchargé le bon paquet dans la bonne version, et l'avoir installé avec votre packager préféré, on va donc pouvoir le configurer.

Vous savez quoi, les informations principales sont stockées à 2 endroits.

2.2.1 – La configuration

Toujours dans la même logique, les fichiers de conf sont dans « **/etc/filebeat/** »

Ce dossier contient :

- un fichier de configuration standard/passe-partout (filebeat.yml)
- un fichier complet de configuration (filebeat.full.yml)
- quelques fichiers annexes de configuration de la JVM Logstash

L' « **ANNEXE 2 »** contient une configuration pour se plugger sur un fichier de type « log4j »,

donc des traces qui peuvent s'étendre sur plusieurs lignes.

On va y retrouver, à la fin, les mêmes champs à renseigner, le « name », l' « output. ».

Mais comme ce beat est spécialisé dans la lecture de fichier de log, il faut déclarer nos inputs.

C'est à déclarer dans le bloc « **filebeat.prospectors:** »

Une entrée (un prospector) peut lire plusieurs fichiers à la fois.

Si l'on prend le cas d'une application BusinessWorks étagée dans plusieurs espace (jvm), on aura envie de regrouper le tout. Ce qui peut donner par exemple :

```
#---- IPS
- input_type: log
  paths:
    - /var/log/bm-ips/ips.log

  fields:
    Source: BlueMind
    Soft: IPS

  multiline.pattern: '^[0-9]{4}-[0-9]{2}-[0-9]{2}'
  multiline.negate: true
  multiline.match: after
  scan_frequency: 30s
```

Ce qui est important ici, c'est :

- « **paths** » : on y inscrit tous les fichiers liés à l'application
- « **fields** » : on y ajoute des champs supplémentaires qui pourront s'avérer utiles pour nos requêtes futures
- « **multiline** » : on y définit les règles concernant la façon dont on gère les lignes du fichiers de logs. Dans l'exemple supra, on gère les lignes de logs JAVA qui commencent toujours par une date, et qui s'inscrivent sur plusieurs lignes.

2.2.2 - Les traces

Par principe/défaut, un beat trace son activité dans « /varl/log/ <beatname>/ », ce qui donne le fichier « **/var/log/filebeat/filebeat** »

2.2.3 - Pousser le schéma « filebeat.yml »

Lors d'une primo installation, il faut déclarer le schéma, comme sur un SGDB classique.

Soit la commande magique, depuis le dossier « /etc/filebeat/ » :

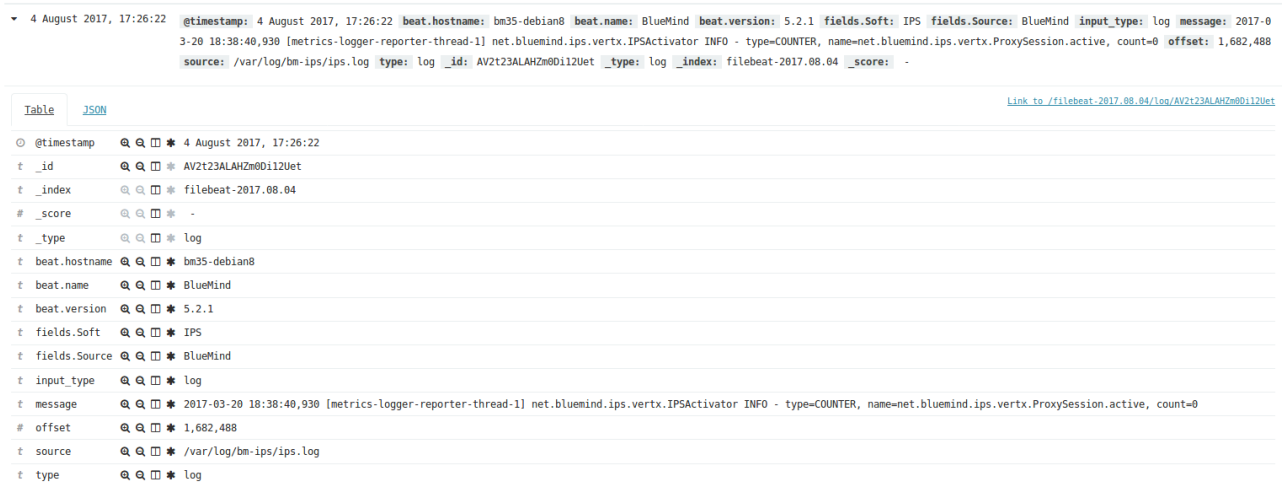
curl -XPUT 'http://<serverElastic>:<portElastic>/_template/filebeat?pretty' -d@filebeat.template.json

Et vous aurez une belle réponse :

```
{
  "acknowledged" : true
}
```

2.2.4 - Le résultat final dans Elasticsearch

Voici comment est affiché dans Kibana une donnée venant d'un filebeat



On y retrouve bien nos champs additionnels (fields.Soft, fields.Source).

L'info, la log, est stockée dans le champ « message »

2.3 Le démarrage

On dit merci à systemd :

- service metricbeat start|stop|status
- service filebeat start|stop|status

3 – Installation de Logstash

Normalement, l'installation et la configuration de « metricbeat » et « filebeat » sont suffisantes pour travailler dans Kibana.

Mais, on peut aussi avoir besoin d'agrémenter la data json de quelques champs supplémentaires dont l'info est stockée dans le message. Et ça, ils ne savent pas faire.

On va donc passer par Logstash qui en a la capacité.

3.1 La configuration

Après avoir récupéré le bon paquet dans la bonne version, et l'avoir installé, il nous reste à configurer le tout.

Il y a 2 niveaux de configuration :

- logstash lui-même
- ce qu'on lui demande de faire

3.1.1 - Logstash

Sans surprise, il s'agit du fichier « **/etc/logstash/logstash.yml** ».

Et avec le contenu suivant, il va tourner à minima :

```
node.name: ELK.LOGSTASH
path.data: /var/lib/logstash
path.config: /etc/logstash/conf.d
path.logs: /var/log/logstash
```

Pour faire simple, on indique :

- un nom
- un lieu de stockage temporaire des données en transit
- un lieu où stocker les fichiers de « traitement »
- un lieu où stocker les logs

3.1.2 - Les fichiers de « traitement »

Par « traitement » , j'englobe :

- la déclaration des entrées (comment et où on reçoit les data)
- la déclaration des filtres (dans quel cas je dois faire quoi)
- la déclaration des sorties (vers qui j'envoie la data)

Et tout ce petit monde est déclaré dans le dossier « **/etc/logstash/conf.d** », enfin celui que vous avez configuré dans « /etc/logstash/logstash.yml »

Comme tous les fichiers sont chargés en mémoire, vous avez le droit de les classer, les « input » d'abord, puis les « filter », et enfin les « output ».

3.1.2.1 – « Input » venant d'un beat

On va appeler ce fichier « 01-beats-input.conf », contenant :

```
input {
        beats {
                port => 5044
                #ssl => false
                #ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
                #ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
        }
}
```

En clair, on ouvre un port d'écoute du type « beat » sur le TCP 5044. Et si vous voulez sécuriser la connexion avec un certificat, vous n'avez qu'à en générer un.

3.1.2.2 – « Filter » sur un message « syslog »

Dans cet exemple, on a un filebeat qui scrute un fichier du type syslog.

Et on veut ajouter 2/3 bricoles avant insertion dans Elasticsearch.

Donc, on crée un fichier « 10-syslog-filter.conf » (vous aurez noté le nom explicite du fichier) contenant :

```
filter {
        if [type] == "syslog" {
        grok {
                match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %
{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: %
{GREEDYDATA:syslog_message}" }
                add_field => [ "received_at", "%{@timestamp}" ]
                add_field => [ "received_from", "%{host}" ]
        }
        syslog_pri { }
        date {
                match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
        }
        }
}
```

Ouais, je sais, c'est « grok », mais c'est puissant. Le principe est simple :

on s'appuie sur une pattern (regex) capturante, i.e « **SYSLOGTIMESTAMP** »,

et on stocke la capture dans la variable « **syslog_timestamp** ».

La liste des patterns déjà fournies par grok est accessible en « ANNEXE 3 » (bonne lecture)

3.1.2.3 – « Filter » sur un message « karaf »

Je vous ai dit plus tôt que le besoin d'ajouter des champs alimenté de données extraites du message pouvait se faire sentir. Et bien avec KARAF, parce que j'en ai un, c'est le cas. Il y a un jobid qui m'intéresse, donc je vais créer le fichier «11-karaf-filter.conf » contenant :

```
# KARAF
#17:08:35,755 | INFO  | p700599416-39369 | _0_1.Jo_mdm_Insee_WSREST_02_Pays 8003 | 325 -
mdm_insee.Jo_mdm_Insee_WSREST_02_Pays - 0.6.1.SNAPSHOT | tMap_2 - Start to work.


filter {
    if [source] =~ "tesb.log" {
        grok {
            match => { "message" => "%{GREEDYDATA:blabla} \| %{GREEDYDATA:alarmLevel} \| %
{GREEDYDATA:jobid} \| %{GREEDYDATA:blabla} \| %{GREEDYDATA:blabla} \| %{GREEDYDATA:blabla}"}
            remove_field => [blabla]
        }
    }
}
```

Comme ça, j'ajoute 2 champs à ma donnée :

- alarmLevel
- jobid

3.1.2.4 – « Output » vers un Elasticsearch

Ok, c'est bien beau mais j'en fais quoi de ma donnée ?

La solution, l'envoyer vers Elasticsearch, d'où le fichier « 30-elasticsearch-output.conf » contenant :

```
output {
        elasticsearch {
        hosts => ["elk1:9200"]
        sniffing => true
        manage_template => false
        index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
        document_type => "%{[@metadata][type]}"
        }
}
```

Mais ce pourrait être un autre Logstash.

3.2 Le démarrage

Vive le copier/coller :

- service logstash start
- service logstash stop
- service logstash status

Les traces d'une séquence de démarrage:

[2017-08-05T11:05:58,585][INFO][logstash.inputs.beats] Beats inputs: Starting input listener {:address=>"0.0.0.0:5044"}

[2017-08-05T11:05:58,884][INFO][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://elastic1:9200/]}}

[2017-08-05T11:05:58,885][INFO][logstash.outputs.elasticsearch] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://elastic1:9200/, :path=>"/"}

[2017-08-05T11:05:58,963][WARN][logstash.outputs.elasticsearch] Restored connection to ES instance {:url=>#<URI::HTTP:0x3a78fc3d URL:http://elastic1:9200/>}

[2017-08-05T11:05:58,965][INFO][logstash.outputs.elasticsearch] New Elasticsearch output {:class=>"LogStash::Outputs::ElasticSearch", :hosts=>[#<URI::Generic:0x6b78a5df URL://elastic1:9200>]}

[2017-08-05T11:05:59,050][INFO][logstash.pipeline] Starting pipeline {"id"=>"main", "pipeline.workers"=>2, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>250}

[2017-08-05T11:05:59,053][INFO][logstash.pipeline] Pipeline main started

[2017-08-05T11:05:59,091][INFO][logstash.agent] Successfully started Logstash API endpoint {:port=>9600}

[2017-08-05T11:06:04,029][INFO][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated {:changes=>{:removed=>[http://elastic1:9200/], :added=>[http://192.168.1.100:9200/]}}

[2017-08-05T11:06:04,031][INFO][logstash.outputs.elasticsearch] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://192.168.1.100:9200/, :path=>"/"}

[2017-08-05T11:06:04,041][WARN][logstash.outputs.elasticsearch] Restored connection to ES instance {:url=>#<URI::HTTP:0x5fe18259 URL:http://192.168.1.100:9200/>}

3.3 Si le démarrage est long, long, long …

Il se peut que vous ayez installé Logstash dans une VM, et bien : pas de bol.

Le problème des VM est le fait qu'elles ne fournissent pas nativement un nombre correct d'entier (l'entropie ou entropy). Dans le monde physique, ces nombres sont générés depuis/grâce à l'activité matérielle, le top du top étant une bonne vieille connexion RTC mais pas géniale de nos jours. Il faut déjà trouver un modem 56k !

L'astuce est donc d'installer « haveged ».

Ce qui dans mon cas donnait avant son installation :

viking@logstash:~$ cat /proc/sys/kernel/random/entropy_avail

57

viking@logstash:~$

Même au Loto, j'aurais perdu !

Mais avec « haveged » ,

viking@logstash:~$ cat /proc/sys/kernel/random/entropy_avail

1178

viking@logstash:~$

Pour rappel, cette entropy est aussi utilisée dans les mécanismes de sécurisation (https, ssh...)

4 – Installation d’Elasticsearch

Après avoir téléchargé le bon paquet dans la bonne version, et l’avoir installé avec votre packager préféré, on va donc pouvoir le configurer.

4.1 La configuration

Comme la plus part du temps (c’est juste pour éviter « toujours »), les fichiers de configuration sont dans « /etc/elasticseach ».

On va se contenter de 2, à savoir :

- elasticsearch.yml
- jvm.options

Ne vous inquiétez pas, on ne va pas créer une usine à gaz.

4.1.1 elasticsearch.yml

Voici les principaux paramètres à renseigner (et donc à « décommenter » au besoin) :

cluster.name: ELASTIC
node.name: elastic1
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch/
network.host: 192.168.1.100
http.port: 9200

Si vous voulez créer un vrai cluster avec 2 ou 3 voire plus de nœuds, il est important de garder le même « *cluster.name* »

Vous aurez remarquer où sont stockées les data, donc faites en sorte qu’il y ait la place suffisante. Oui je sais, mais quand même, y’en a qui...

Pour mémo, la commande Linux magique pour n’afficher que ces quelques lignes :

sed '/^#/d' /etc/elasticsearch/elasticsearch.yml

4.1.2 jvm.options

Toujours dans « /etc/elasticsearch », c'est le fichier à em...brouilles par exemple.

Il contient toutes les variables de la JVM, donc JAVA.

Nous, on va se contenter de la gestion de la mémoire, et plus particulièrement du « heap space », soit les variables :

- Xms (valeur minimale au démarrage)
- Xmx (valeur maximale)

LA REGLE : heaspace ≤ (mémoire / 2)

En clair, n'allez pas allouer 6Go sous prétexte que avez 8Go. Faudra pas venir pleurer après.

Dans ma VM qui a 6Go de RAM, les valeurs sont :

-Xms3g

-Xmx3g

=> Veillez à mettre la même valeur dans les 2 cas

4.2 Le démarrage

Ben, comme sur les Linux récents, on passe par « service » :

- service elasticsearch start
- service elasticsearch stop
- service elasticsearch status

5 – Installation de KIBANA

Après avoir téléchargé le bon paquet dans la bonne version, et l'avoir installé avec votre packager préféré, on va donc pouvoir le configurer.

4.1 La configuration

Les fichiers de configuration sont, normalement, dans « /etc/logstash».

Et il s'appelle ?

- kibana.yml

Et comme pour Elasticsearch, on va faire simple

4.1.1 kibana.yml

Voici les principaux paramètres à renseigner (et donc à « décommenter » au besoin) :

server.port: 5601
server.host: "kibana.intra"
server.name: "kibana.intra"
elasticsearch.url: "http://elastic1.intra:9200"
kibana.index: ".kibana"

Pour mémo, la commande Linux magique pour n'afficher que ces quelques lignes :

sed -e '/^#/d' -e '/^$/d' /etc/kibana/kibana.yml

4.2 Le démarrage

Ben, on passe encore par « service » :

- service kibana start
- service kibana stop
- service kibana status

4.3 NGINX, just for fun

Vous aurez noté que les ports d'écoute ne sont pas forcément super sexy (9200, 5601, 5044). Ben sur cette machine Logstash, je vous propose un truc super simple à base de NGINX.

Vous installez NGINX comme vous le pouvez (apt, rpm, yum, à la main, à la pelle…)

Chez DEBIAN & Co, on a une arborescence comme chez Apache, allez savoir pourquoi.
Donc sur cette infra, on va modifier le fichier « /etc/nginx/sites-available/default » .
Faites une copie de sauvegarde (au cas où…)

Donc, dans ce fichier, on supprime toutes les lignes (d'où la sauvegarde) que l'on remplace par :

```
upstream kibana {
        server 192.168.1.101:5601;
}
server {
    listen 80;
    location / {
       proxy_pass http://kibana;
    }
}
```

On relance ensuite nginx (service nginx restart), et au joie du sysadmin, KIBANA est accessible sans avoir besoin de connaître le port TCP.

Sinon, on va dire à 2 lignes prêts, vous avez un load-balancer à pas cher, et pas gourmand. Bon, ça va, je vous donne le truc :

```
upstream elasticsearch {
        list_conn;
        server 192.168.1.99:9200;
        server 192.168.1.100:9200;
}
```

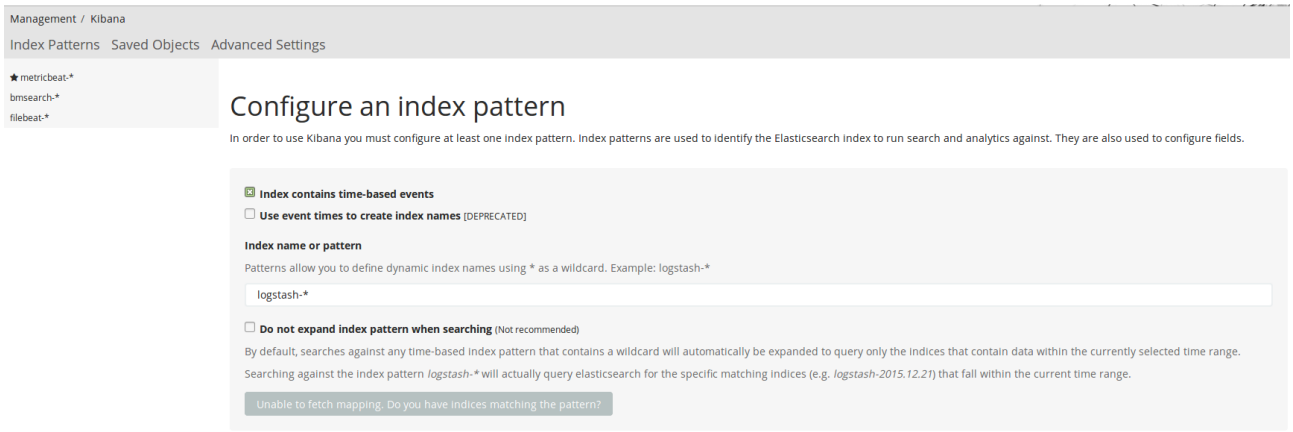
NGINX distribue les connexions selon le mode « least », c'est à dire en prenat en compte l'état des end-points, et en maintenant la connexion si besoin.

4.4 Intégrer les Index Patterns

Rappelez-vous, vous avez pousser des schémas (json) sur votre Elasticsearch.

Il faut maintenant dire à KIBANA de les prendre en comptes. Pour cela, il est préférable d'attendre un peu, le temps d'avoir des data dans votre Elasticsearch.

Il faudra alors aller dans « Management », puis dans « Index Patterns » , puis « + Add new », pour afficher :



En lieu et place de « logstash-* », saisissez :

- « metricbeat-* », pour metricbeat, si vous l'avez installé
- « filebeat-* », pour filebeat, si vous l'avez installé
- « bmsearch-* », pour bmSearch (auto-promotion oblige), parce que vous l'avez installé, si !

Il faudra aussi déterminer un index par défaut (un clic sur l'étoile).

ANNEXE 1 : metricbeat.yml

```
#========================  Modules configuration ============================
metricbeat.modules:

#------------------------------- System Module -------------------------------
- module: system
  metricsets:
    # CPU stats
    - cpu

    # System Load stats
    - load

    # Per CPU core stats
    #- core

    # IO stats
    #- diskio

    # Per filesystem stats
    - filesystem

    # File system summary stats
    - fsstat

    # Memory stats
    - memory

    # Network stats
    - network

    # Per process stats
    - process

    # Sockets (linux only)
    #- socket
  enabled: true
  period: 10s
  processes: ['.*']

#================================ General =====================================

# The name of the shipper that publishes the network data. It can be used to group
# all the transactions sent by a single shipper in the web interface.
name:  ELK.KIBANA
```

```
# The tags of the shipper are included in their own field with each
# transaction published.
#tags: ["service-X", "web-tier"]

# Optional fields that you can specify to add additional information to the
# output.
#fields:
#  env: staging

#------------------------------ Logstash output -------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.122.195:5044"]
  timeout: 300s
```

ANNEXE 2 : filebeat.yml

```
#------------------------------ Filebeat input -------------------
filebeat.prospectors:

#---- IPS
- input_type: log
  paths:
    - /var/log/bm-ips/ips.log

  fields:
    Source: BlueMind
    Soft: IPS

  multiline.pattern: '^[0-9]{4}-[0-9]{2}-[0-9]{2}'
  multiline.negate: true
  multiline.match: after
  scan_frequency: 30s



#------------------------------ Name or viewed as from -------------------------
name: BlueMind

#------------------------------ Logstash output --------------------------------
output.logstash:
  hosts: ["192.168.122.195:5044"]
```

ANNEXE 3 : grok-patterns

```
USERNAME [a-zA-Z0-9._-]+
USER %{USERNAME}
EMAILLOCALPART [a-zA-Z][a-zA-Z0-9_.+-=:]+
EMAILADDRESS %{EMAILLOCALPART}@%{HOSTNAME}
HTTPDUSER %{EMAILADDRESS}|%{USER}
INT (?:[+-]?(?:[0-9]+))
BASE10NUM (?<![0-9.+-])(?>[+-]?(?:(?:[0-9]+(?:\.[0-9]+)?)|(?:\.[0-9]+)))
NUMBER (?:%{BASE10NUM})
BASE16NUM (?<![0-9A-Fa-f])(?:[+-]?(?:0x)?(?:[0-9A-Fa-f]+))
BASE16FLOAT \b(?<![0-9A-Fa-f.])(?:[+-]?(?:0x)?(?:(?:[0-9A-Fa-f]+(?:\.[0-9A-Fa-f]*)?)|(?:\.[0-9A-Fa-f]+)))\b

POSINT \b(?:[1-9][0-9]*)\b
NONNEGINT \b(?:[0-9]+)\b
WORD \b\w+\b
NOTSPACE \S+
SPACE \s*
DATA .*?
GREEDYDATA .*
QUOTEDSTRING (?>(?<!\\)(?>"(?>\\.|[^\\"]+)+"|""|(?>'(?>\\.|[^\\']+)+')|''|(?>`(?>\\.|[^\\`]+)+`)|``))
UUID [A-Fa-f0-9]{8}-(?:[A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}
# URN, allowing use of RFC 2141 section 2.3 reserved characters
URN urn:[0-9A-Za-z][0-9A-Za-z-]{0,31}:(?:%[0-9a-fA-F]{2}|[0-9A-Za-z()+,.:=@;$_!*'/?#-])+

# Networking
MAC (?:%{CISCOMAC}|%{WINDOWSMAC}|%{COMMONMAC})
CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2}[A-Fa-f0-9]{4})
WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-){5}[A-Fa-f0-9]{2})
COMMONMAC (?:(?:[A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2})
IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-Fa-f]{1,4}|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2})|:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){4}(((:[0-9A-Fa-f]{1,4}){1,3})|((:[0-9A-Fa-f]{1,4})?:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-Fa-f]{1,4}){1,4})|((:[0-9A-Fa-f]{1,4}){0,2}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){2}(((:[0-9A-Fa-f]{1,4}){1,5})|((:[0-9A-Fa-f]{1,4}){0,3}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){1}(((:[0-9A-Fa-f]{1,4}){1,6})|((:[0-9A-Fa-f]{1,4}){0,4}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(:(((:[0-9A-Fa-f]{1,4}){1,7})|((:[0-9A-Fa-f]{1,4}){0,5}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:)))(%.+)?
IPV4 (?<![0-9])(?:(?:[0-1]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[0-9]{1,2}|2[0-4][0-9]|25[0-5]))(?![0-9])
IP (?:%{IPV6}|%{IPV4})
HOSTNAME \b(?:[0-9A-Za-z][0-9A-Za-z-]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-]{0,62}))*(\.?|\b)
IPORHOST (?:%{IP}|%{HOSTNAME})
HOSTPORT %{IPORHOST}:%{POSINT}

# paths
PATH (?:%{UNIXPATH}|%{WINPATH})
UNIXPATH (/([\w_%!$@:.,+~-]+|\\.)*)+
TTY (?:/dev/(pts|tty([pq])?)(\w+)?/?(?:[0-9]+))
WINPATH (?>[A-Za-z]+:|\\)(?:\\[^\\?*]*)+
```

```
URIPROTO [A-Za-z]+(\+[A-Za-z+]+)?

URIHOST %{IPORHOST}(?::%{POSINT:port})?

# uripath comes loosely from RFC1738, but mostly from what Firefox

# doesn't turn into %XX

URIPATH (?:/[A-Za-z0-9$.+!*'(){},~:;=@#%&_\-]*)+

#URIPARAM \?(?:[A-Za-z0-9]+(?:=(?:[^&]*))?(?:&(?:[A-Za-z0-9]+(?:=(?:[^&]*))?)?)*)?

URIPARAM \?[A-Za-z0-9$.+!*'|(){},~@#%&/=:;_?\-\[\]<>]*

URIPATHPARAM %{URIPATH}(?:%{URIPARAM})?

URI %{URIPROTO}://(?:%{USER}(?::[^@]*)?@)?(?:%{URIHOST})?(?:%{URIPATHPARAM})?


# Months: January, Feb, 3, 03, 12, December

MONTH \b(?:[Jj]an(?:uary|uar)?|[Ff]eb(?:ruary|ruar)?|[Mm](?:a|ä)?r(?:ch|z)?|[Aa]pr(?:il)?|[Mm]a(?:y|i)?|[Jj]un(?:e|i)?|[Jj]ul(?:y)?|[Aa]ug(?:ust)?|[Ss]ep(?:tember)?|[Oo](?:c|k)?t(?:ober)?|[Nn]ov(?:ember)?|[Dd]e(?:c|z)(?:ember)?)\b

MONTHNUM (?:0?[1-9]|1[0-2])

MONTHNUM2 (?:0[1-9]|1[0-2])

MONTHDAY (?:(?:0[1-9])|(?:[12][0-9])|(?:3[01])|[1-9])


# Days: Monday, Tue, Thu, etc...

DAY (?:Mon(?:day)?|Tue(?:sday)?|Wed(?:nesday)?|Thu(?:rsday)?|Fri(?:day)?|Sat(?:urday)?|Sun(?:day)?)


# Years?

YEAR (?>\d\d){1,2}

HOUR (?:2[0123]|[01]?[0-9])

MINUTE (?:[0-5][0-9])

# '60' is a leap second in most time standards and thus is valid.

SECOND (?:(?:[0-5]?[0-9]|60)(?:[:.,][0-9]+)?)

TIME (?!<[0-9])%{HOUR}:%{MINUTE}(?::%{SECOND})(?![0-9])

# datestamp is YYYY/MM/DD-HH:MM:SS.UUUU (or something like it)

DATE_US %{MONTHNUM}[/-]%{MONTHDAY}[/-]%{YEAR}

DATE_EU %{MONTHDAY}[./-]%{MONTHNUM}[./-]%{YEAR}

ISO8601_TIMEZONE (?:Z|[+-]%{HOUR}(?::?%{MINUTE}))

ISO8601_SECOND (?:%{SECOND}|60)

TIMESTAMP_ISO8601 %{YEAR}-%{MONTHNUM}-%{MONTHDAY}[T ]%{HOUR}:?%{MINUTE}(?::?%{SECOND})?%{ISO8601_TIMEZONE}?

DATE %{DATE_US}|%{DATE_EU}

DATESTAMP %{DATE}[- ]%{TIME}

TZ (?:[APMCE][SD]T|UTC)

DATESTAMP_RFC822 %{DAY} %{MONTH} %{MONTHDAY} %{YEAR} %{TIME} %{TZ}

DATESTAMP_RFC2822 %{DAY}, %{MONTHDAY} %{MONTH} %{YEAR} %{TIME} %{ISO8601_TIMEZONE}

DATESTAMP_OTHER %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{TZ} %{YEAR}

DATESTAMP_EVENTLOG %{YEAR}%{MONTHNUM2}%{MONTHDAY}%{HOUR}%{MINUTE}%{SECOND}

HTTPDERROR_DATE %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}


# Syslog Dates: Month Day HH:MM:SS

SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}

PROG [\x21-\x5a\x5c\x5e-\x7e]+

SYSLOGPROG %{PROG:program}(?:\[%{POSINT:pid}\])?

SYSLOGHOST %{IPORHOST}

SYSLOGFACILITY <%{NONNEGINT:facility}.%{NONNEGINT:priority}>
```

HTTPDATE %{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME} %{INT}

Shortcuts

QS %{QUOTEDSTRING}

Log formats

SYSLOGBASE %{SYSLOGTIMESTAMP:timestamp} (?:%{SYSLOGFACILITY})?%{SYSLOGHOST:logsource} %{SYSLOGPROG}:

COMMONAPACHELOG %{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth} \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})" %{NUMBER:response} (?:%{NUMBER:bytes}|-)

COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}

HTTPD20_ERRORLOG \[%{HTTPDERROR_DATE:timestamp}\] \[%{LOGLEVEL:loglevel}\] (?:\[client %{IPORHOST:clientip}\]){0,1}%{GREEDYDATA:errormsg}

HTTPD24_ERRORLOG \[%{HTTPDERROR_DATE:timestamp}\] \[%{WORD:module}:%{LOGLEVEL:loglevel}\] \[pid %{POSINT:pid}:tid %{NUMBER:tid}\](\(%{POSINT:proxy_errorcode}\)%{DATA:proxy_errormessage}:)?(\[client %{IPORHOST:client}:%{POSINT:clientport}\])? %{DATA:errorcode}: %{GREEDYDATA:message}

HTTPD_ERRORLOG %{HTTPD20_ERRORLOG}|%{HTTPD24_ERRORLOG}

Log Levels

LOGLEVEL ([Aa]lert|ALERT|[Tt]race|TRACE|[Dd]ebug|DEBUG|[Nn]otice|NOTICE|[Ii]nfo|INFO|[Ww]arn?(?:ing)?|WARN?(?:ING)?|[Ee]rr?(?:or)?|ERR?(?:OR)?|[Cc]rit?(?:ical)?|CRIT?(?:ICAL)?|[Ff]atal|FATAL|[Ss]evere|SEVERE|EMERG(?:ENCY)?|[Ee]merg(?:ency)?)