

Analyse orientée business de vos logs applicatifs



Dans cet article, nous allons voir comment mettre à profit nos logs applicatifs afin de faire de l'analyse orientée "business" grâce à ElasticSearch, Logstash et Kibana.

Vincent Spiewak
Consultant Java et agile chez Xebia
@vspiewak

A la tête d'un site de vente en ligne fictif, nous essayerons notamment de monitorer l'activité de nos clients, et de répondre aux questions suivantes :

- Quel est le taux de conversion (ratio recherches / ventes) ?
- Quels sont les produits les plus consultés ?
- Où nous situons nous par rapport à nos objectifs de ventes ?

A faire chez vous

L'ensemble de la démonstration a été automatisée sur une VM Vagrant disponible sur GitHub : <https://github.com/vspiewak/elk-programmez-2014.git>. Vous devez installer uniquement VirtualBox et Vagrant (disponible gratuitement pour Windows, Linux et OS X). Vous trouverez le script d'installation ainsi que l'ensemble des fichiers de configuration.

Format des logs

Ce paragraphe vous présente le format des logs que nous allons exploiter Fig.1. Notre application génère deux types de logs, représentant respectivement une recherche ou un achat dans notre magasin de vente en ligne. Un log de vente contient l'adresse IP et le User Agent de l'acheteur, son sexe, ainsi que les informations sur le produit acheté (marque, modèle, catégorie, options et prix). Un log de recherche contient l'adresse IP et le User Agent du visiteur, et quelques informations sur le produit recherché (parfois la catégorie, parfois la catégorie et la marque, etc).

Logstash

Logstash est un ETL léger permettant de collecter, transformer et charger des logs à l'aide de connecteurs d'entrées, de filtres et de connecteurs de sorties. C'est un véritable couteau suisse qui, en version 1.4.2, vous fournit 41 entrées, 50 filtres, et 55 sorties différentes. Nous allons voir comment transformer nos lignes de logs illisibles en documents JSON structurés, et les enrichir au passage de précieuses informations (géolocalisation, version du navigateur, etc).

Premiers pas

Un agent Logstash a besoin d'un fichier de configuration pour être exécuté. Voici un exemple de configuration minimaliste : Fig.2.

Fig.1

Fig.3

L'entrée standard est utilisée comme input, la sortie standard est utilisée comme output (A noter que nous utilisons le codec rubydebug afin que le json soit formaté).

Nous pouvons lancer ensuite l'agent Logstash avec cette configuration, et écrire un message dans l'entrée standard : Fig.3.

Logstash a ainsi transformé notre message en un JSON formaté. Logstash ajoute automatiquement les champs @version, @timestamp et host. A noter que le champ @timestamp contient la date de réception du message par Logstash.

Le premier travail va consister à parser nos logs à l'aide du filtre grok...

Filtre Grok

Le filtre Grok nous permet, à l'aide d'expressions régulières, de découper nos logs et leur donner de la sémantique. Logstash vient avec plus d'une centaine d'expressions régulières prédéfinies (disponibles ici : <https://github.com/logstash/logstash/tree/master/patterns>)

Par exemple, le pattern COMBINEDAPACHELOG du fichier grok patterns permet de parser les lignes de logs d'un serveur Apache. Vous pouvez utiliser 2 syntaxes :

- %{SYNTAX:SEMANTIC} ou %{SYNTAX:SEMANTIC:TYPE}
- (? field name the pattern here)

La première syntaxe utilise un pattern Logstash prédéfini (ex: INT, NOTSPACE, LOGLEVEL, ...), SEMANTIC étant le nom du champ à mapper. Vous pouvez préciser le type du champ (integer, float, string) via le paramètre TYPE. La deuxième syntaxe vous permet de définir un pattern customisé avec ou sans l'aide de plusieurs patterns Logstash. Pour nos logs, nous utiliserons le filtre Grok suivant :

```
filter {
  grok {
    match => [ message , (?<log_date>{%MONTHDAY}-{%MONTHNUM}-{%YEAR} %{:HOUR}:%{:MINUTE}:%{:SECOND}}.[0-9]{3}) \[%{NOTSPACE:thread}\] %{:LOGLEVEL:log_level} %{:NOTSPACE:classname} - %{:GREEDYDATA:log_msg} ]
  }
}
```

Avec le filtre Grok, logstash découpe nos lignes de log correctement, ajoutant les champs log_date, thread, log_level, classname et log_msg : Fig.4.

Filtre date

Le filtre date va permettre de dater notre message correctement. En effet, par défaut logstash valorise le champ @timestamp avec la date courante. Dans le cas de nos logs, nous voulons utiliser la date du log (champ log_date). Nous utiliserons un pattern au format JodaTime (voir :

Fig.2

<http://ioda-time.sourceforge.net/apidocs/org/ioda/time/format/DateTimeFormat.html>)

```
date {
    match => [ log_date , dd-MM-YYYY HH:mm:ss.SSS ]
}
```

Filtre KV

Le filtre KV est très utile pour découper un champ au format cle1 valeur2&cle2 valeur2 (paramètres d une requête http notamment)
Nous allons utiliser ce filtre pour découper le champ log msg :

```
kv {
    field_split => &
    source => log_msg
}
```

Filtre mutate

Ajout d'un tag

Le filtre mutate est un filtre couteau suisse permettant différentes modifications sur les champs. Nous allons ajouter un tag afin de différencier nos recherches de nos ventes :

```
if [classname] =~ /SellRequest$/ {
  mutate { add_tag => sell }
} else if [classname] =~ /SearchRequest$/ {
  mutate { add_tag => search }
}
```

Conversion de type

Nous allons convertir les champs id et price respectivement en entier et nombre à virgule flottant :

```
mutate {
  convert => [ id , integer ]
}
mutate {
  convert => [ price , float ]
}
```

Suppression d'un champ

Toujours avec le filtre mutate, nous allons supprimer le champ log_msg que nous avons re découpé à l aide du filtre KV :

```
mutate {
  remove_field => [ log_msg ]
}
```

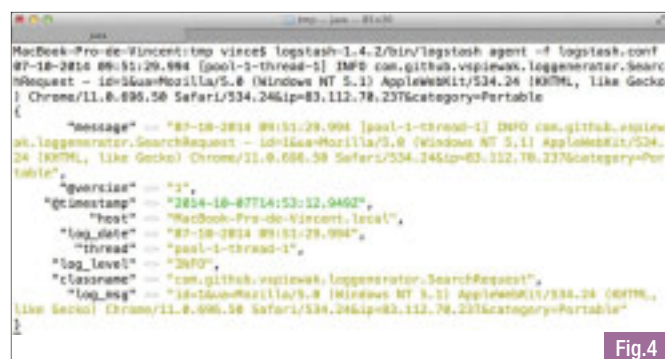


Fig.4

Split d'un champ en tableau

Pour finir avec le filtre mutate, nous allons splitter notre champ options afin d'avoir un tableau d'options :

```
mutate {
  split => [ options , | ]
}
```

Filtre GeoIP

Le filtre GeoIP permet d'ajouter des informations de géolocalisation via une adresse IP (coordonnées gps, ville et pays notamment). Logstash utilise la base de données GeoLite de Maxmind sous licence CCA ShareAlike 3.0.

```
filter {
  geoip {
    source => ip
  }
}
```

Filtre UserAgent

Le filtre UserAgent permet de parser automatiquement un User Agent et d'ajouter des informations comme la version du système d'exploitation ou du navigateur :

```
useragent {
  source => "ua"
  target => "useragent"
  remove_field => [ "ua" ]
}
```

Sortie Elasticsearch

Une fois notre log parsé, nous voulons que Logstash l'envoie à Elasticsearch. Rien de plus simple avec la sortie Elasticsearch, puisqu'il suffit simplement de déclarer :

```
output {
    elasticsearch {}
}
```

Note: la sortie est configurée pour se connecter sur l'Elasticsearch locale, avec un mode de transport de type `node` par défaut. Il n'y a donc rien à paramétrer. Logstash nous permet de collecter nos logs, les structurer, les enrichir et enfin de les envoyer à Elasticsearch : [Fig.5](#).



Fig.5

Elasticsearch

Elasticsearch est une base de données full text orientée document, distribuée et offrant de la haute disponibilité notamment. Logstash enverra nos logs dans des indices nommés `logstash YYYY MM DD` (équivalent des tables dans le monde SQL traditionnel). Pour installer Elasticsearch et le plugin `head` (offrant une interface web), rien de plus simple :

```
$ curl -O https://download.elasticsearch.org/elasticsearch/
elasticsearch/elasticsearch-1.3.4.tar.gz
$ tar xvzf elasticsearch-1.3.4.tar.gz
$ cd elasticsearch-1.3.4
$ bin/plugin --install mobz/elasticsearch-head
$ bin/elasticsearch
```

Vous pouvez surveiller l'état de santé d'Elasticsearch à l'adresse : http://localhost:9200/_plugin/head Fig.6.

Kibana

Kibana est une application de type Single Page App utilisant notamment la stack technique HTML 5, Angular JS et Twitter Bootstrap. Dans notre exemple, nous utiliserons NGINX, mais tout autre serveur Web capable de servir des fichiers statiques est possible :

```
curl -O https://download.elasticsearch.org/kibana/kibana/
kibana-3.1.1.tar.gz
tar xvzf kibana-3.1.1.tar.gz
sudo mv kibana-3.0.0milestone4 /usr/share/nginx/www/kibana
```

Vous pouvez désormais accéder à Kibana à l'URL : <http://localhost/kibana>. Fig.7. La page d'accueil de Kibana dispose d'un menu en haut à droite permettant de charger, sauvegarder et partager vos dashboards via :

- ▶ Fichier JSON
- ▶ ElasticSearch
- ▶ Gist
- ▶ Permalien


Kibana offre 4 dashboards par défaut :

- ▶ Un dashboard générique Logstash (cf. photo),
- ▶ Un dashboard générique Elasticsearch,
- ▶ Un dashboard non configuré,
- ▶ Un dashboard vide.

Un dashboard se branche sur tous les index de votre cluster Elasticsearch, un index spécifique, ou les index dont le nom matche un pattern (cas de Logstash notamment). L'interface reprend le système de grille de Twitter Bootstrap, décomposant les lignes en 12 colonnes, contenant des panels.



Fig.6

Il existe plusieurs types de panels à ajouter et configurer (recherche, période de temps, histogramme, camembert, carte, Markdown, ...) L'icône  , disponible sur chaque panel, permet de visualiser la requête Elasticsearch.


Création d'un dashboard Kibana

Partez d'un Dashboard vierge en cliquant sur le lien `Blank Dashboard`. N'oubliez pas de le sauvegarder avant de rafraîchir la page de votre navigateur ! Fig.8.

Ouvrez la fenêtre modale `Dashboard settings` en cliquant sur la roue crantée dans le coin supérieur droit :

- ▶ Onglet `General` : choisissez un titre et le thème (`dark` ou `light`),
- ▶ Onglet `Index` : sélectionnez `day` pour le champ `Timestamping`,
- ▶ Onglet `Controls` : cochez toutes les cases afin de vous donner le maximum d'options,
- ▶ Fermez la fenêtre modale en cliquant sur `save`.

Vous pouvez sélectionner une période de temps à afficher dans le menu `Time filter` (exemple: `TimeFilter Last 15m`, `TimeFilter Auto Refresh 5s`). La barre de recherche utilise le format `Lucene Query` ou une expression régulière.

En cliquant sur le bouton  du champ recherche, créez 3 barres de recherche contenant les requêtes suivantes :

- ▶
- ▶ tags: search
- ▶ tags: sell

Vous pouvez épingler des recherches et leur attribuer un alias.

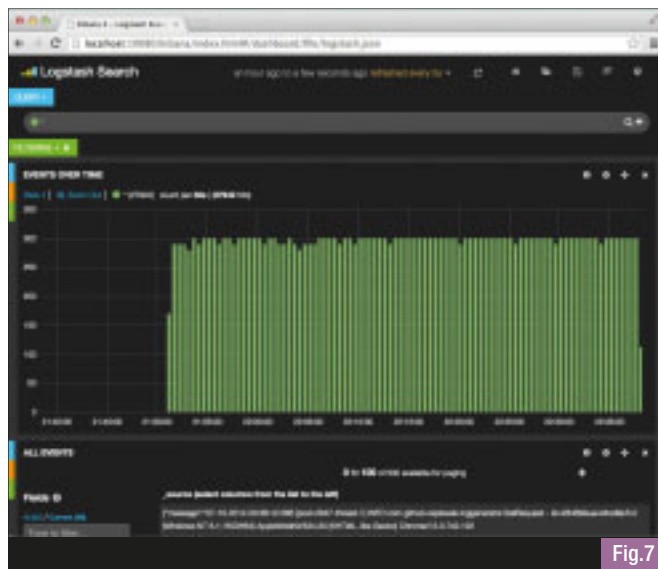


Fig.7

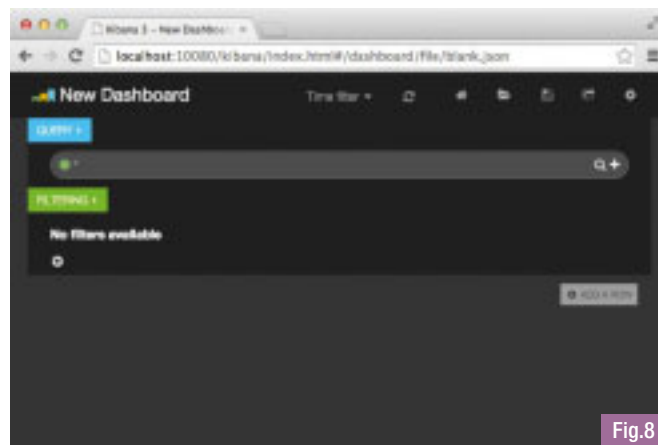


Fig.8

Cliquez sur chaque rond de couleur et sauvegardez les alias suivants :

- All
- Search
- Sell

Ligne 1 Objectifs, Histogramme et Carte

Ajoutez une ligne en cliquant sur le bouton Add row , puis :

- Un panel Histogram de taille 7
- Un panel Bettermap de taille 5

Ligne 2 Terms

Ajoutez une nouvelle ligne contenant:

- Un panel terms, field tags , taille 2
- Un panel terms, field brand.raw , length 5, taille 2

- Un panel terms, field name.raw , length 5, taille 2

- Un panel terms, field geoip.city name.raw , length 5, taille 2

- Un panel terms, field useragent.device.raw , length 5, taille 2

- Un panel terms, field useragent.name.raw , length 5, taille 2

Ligne 3 Données

Ajoutez une dernière ligne contenant uniquement un panel table de taille 12.

Drill down

En cliquant sur les différents éléments des deux dernières lignes, vous créez des filtres dynamiquement (icônes en forme de loupe ou de croix). Vous retrouverez ces filtres en dessous des barres de recherches, vous permettant de les éditer ou les désactiver. Fig.9.

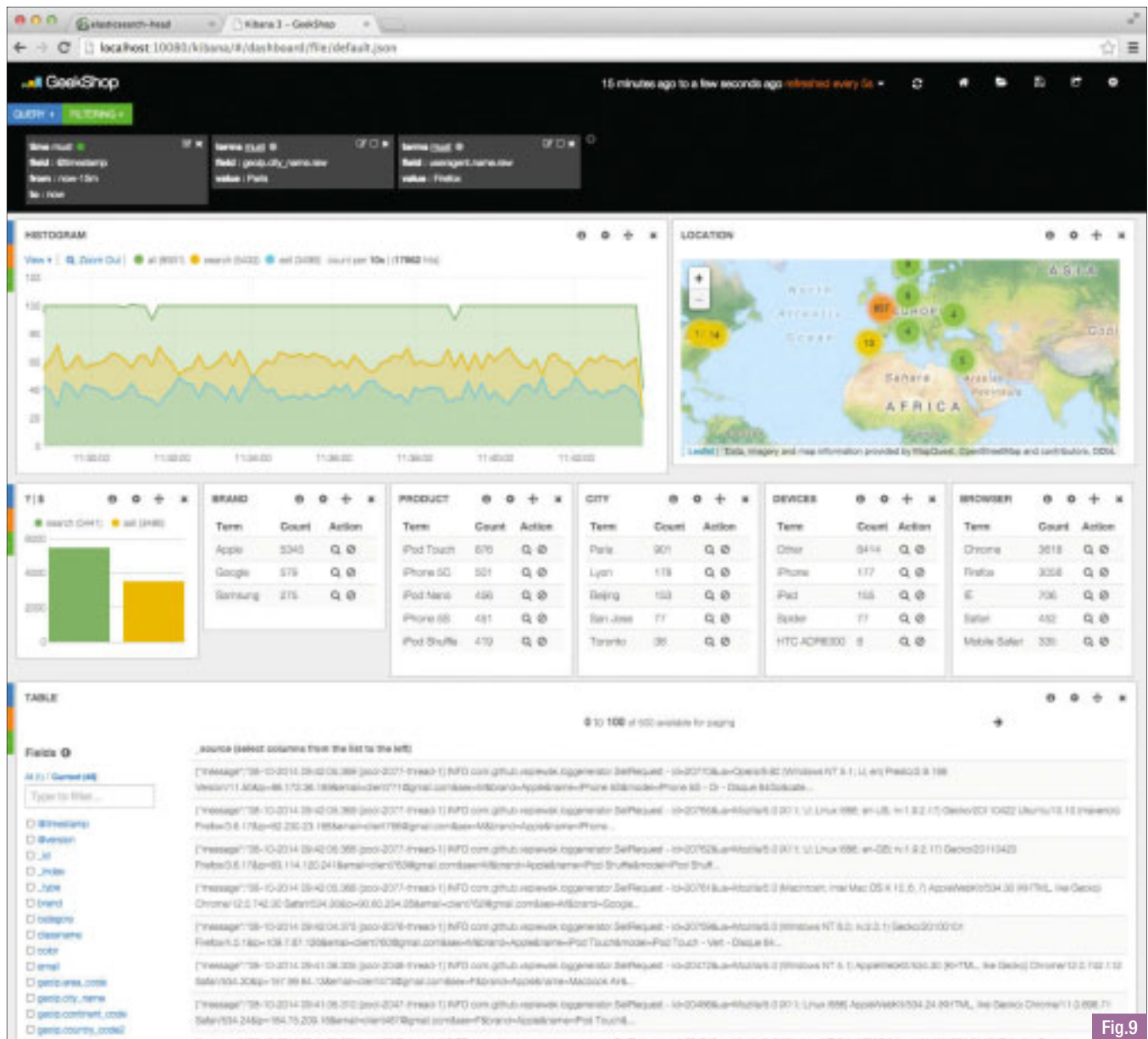


Fig.9

