Situation professionnelle en PPE 3 : Filtrage des accès internet vers internet via un PROXY

Documentation n°5 : Visualisation et interprétation des logs.

Auteur : Guillaume Orlando

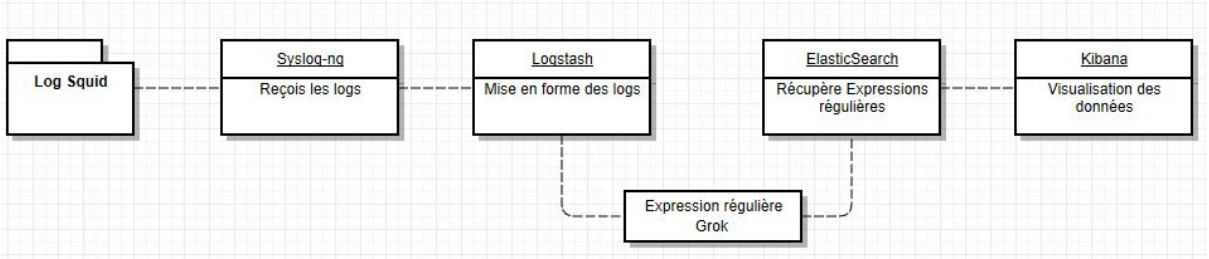
Date : 20/11/2017

I. Introduction.

La suite ELK, diminutif des applications Elasticsearch, Logstash et Kibana, permet de mettre en place une plateforme de traitement et de visualisation de logs. Les trois outils sont très puissants puisqu'ils sont capable de traiter des logs provenant de n'importe quel service. En revanche, leurs configuration nécessite beaucoup de configurations pour que l'environnement s'adapte à nos logs. Chronologiquement, les logs sont dans un premier temps traités par Logstash, Interprétés grâce un à une expression régulière Grok, recensés par ElasticSearch, puis mis en forme sous Kibana.

Ici, les logs sont ceux provenant directement de Squid. Comme vu dans les documentations précédentes, ceux-ci sont envoyés directement sur un serveur distant, et récupérés dans un fichier "pfSense.log" présent sous "/var/log/pfsense".

Le processus de traitement des logs sera donc le suivant :



I. Installation des composants.

Les applications ELK sont toutes disponibles gratuitement et en open-source sur le site des développeurs https://www.elastic.co/.Nous utiliserons la dernière version des outils, à savoir la 6.0.0.

Les fichiers téléchargés ont simplement besoins d'être décompressés dans un dossier commun. Dans notre cas, nous allons créer le dossier "ELK" dans le répertoire personnel de l'utilisateur du serveur distant : "/home/user".

Logstash est trouvable en .tar.gz ici : https://www.elastic.co/downloads/logstash

ElasticSearch en .tar ici : https://www.elastic.co/downloads/elasticsearch.

Et Kibana en version 64 bits ici : https://www.elastic.co/downloads/kibana.

Il s'agira uniquement de décompresser les trois fichiers dans notre dossier "ELK".

II. Configuration de Logstash.

Logstash fonctionne avec deux fichiers de configurations que nous allons devoir créer de toute pièce. Le premier comprendra les instructions à suivre pour disséquer nos logs, et le second à appliquer un "template" pour récupérer les expressions régulières.

Avant de créer nos fichiers de configurations, nous allons tester l'installation de Logstash. Pour ce faire, il suffira d'ouvrir un terminal et de taper "*/home/user/ELK/bin/logstash -e 'input { stdin { } } output { stdout {} }'* ". une fois les service démarrés, essayez d'entrer un mot au clavier dans le prompt. Si un message du type *"2016-03-16T12:46:40.283Z User-PC test"* s'affiche, Logstash est prêt à fonctionner correctement.

```
user@user-virtual-machine ~ $ ELKv2/logstash-6.0.0/bin/logstash -e 'input { stdin { } } output {
 stdout {} }'
Sending Logstash's logs to /home/user/ELKv2/logstash-6.0.0/logs which is now configured via log4
j2.properties
[2017-11-18T22:21:47,773][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"
fb_apache", :directory=>"/home/user/ELKv2/logstash-6.0.0/modules/fb_apache/configuration"}
[2017-11-18T22:21:47,778][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"
netflow", :directory=>"/home/user/ELKv2/logstash-6.0.0/modules/netflow/configuration"}
[2017-11-18T22:21:49,166][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml'
 file because modules or command line options are specified
[2017-11-18T22:21:49,513][INFO ][logstash.agent           ] Successfully started Logstash API en
dpoint {:port=>9600}
[2017-11-18T22:21:50,936][INFO ][logstash.pipeline        ] Starting pipeline {:pipeline_id=>"ma
in", "pipeline.workers"=>4, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max
_inflight"=>500, :thread=>"#<Thread:0xff6306a@/home/user/ELKv2/logstash-6.0.0/logstash-core/lib/
logstash/pipeline.rb:290 run>"}
[2017-11-18T22:21:50,972][INFO ][logstash.pipeline        ] Pipeline started {"pipeline.id"=>"ma
in"}
The stdin plugin is now waiting for input:
[2017-11-18T22:21:50,992][INFO ][logstash.agent           ] Pipelines running {:count=>1, :pipel
ines=>["main"]}
test
2017-11-18T21:22:24.021Z user-virtual-machine test
```

Nous allons maintenant créer le premier fichier de configuration “pfsense_log.conf” :

```
  GNU nano 2.5.3        Fichier : ELKv2/logstash-6.0.0/bin/pfsense_log.conf          Modifié

input {
  file {
    path => "/var/log/pfsense/pfSense.log"
    start_position => beginning
    sincedb_path => "/dev/null"
  }
}

filter {
  grok {
    match => {
        "message" => ""
    }
  }
}

output {
   elasticsearch {
       hosts => ["127.0.0.1:9200"]
       index => "pfsense_log"
       template => "/home/user/ELKv2/logstash-6.0.0/bin/pfsense_template.json"
       template_name => "pfsense_log"
       template_overwrite => true
   }
        stdout {}
}
```

Le fichier est découpé en trois balises principales : “*Input*”, “*Filter*” et “*Output*”. le premier est ici configuré de façon à récupérer manuellement les logs provenants du fichier situé sous “/var/log/pfsense/pfSense.log”.
La balise Filter comprendra l’expression régulière que nous allons créer en partie 2. Tous les logs provenants de l’Input vont se voir appliquer ce filtre.
Enfin, la balise Output va définir où les logs traités vont être envoyés. Dans notre cas, ce sera au service ElasticSearch, opérant sur la même machine et sur le port 9200. Nous spécifions un fichier d’index qui va contenir les données filtrés. Enfin, nous spécifions le chemin relatif du fichier de template Logstash à utiliser.

III. Création du filtre Grok.

La création du filtre est la partie la plus délicate, puisqu'il faut être capable de filtrer les logs en fonction de leur syntaxe et des variations possibles. Ici, un log type de Squid à la syntaxe suivante :

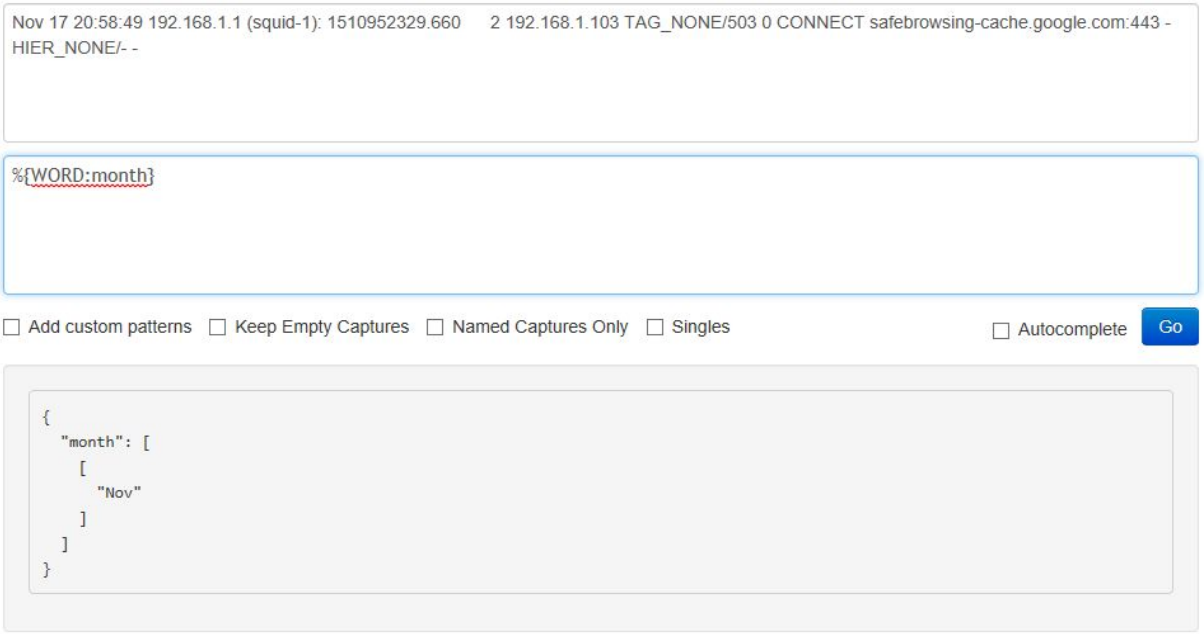
Nov 17 21:01:10 192.168.1.1 (squid-1): 1510952470.370 233176 192.168.1.103 TCP_TUNNEL/200 5931 CONNECT etherpad.fr:443 - ORIGINAL_DST/195.154.57.241 -

Avant de créer le filtre, observons les différentes parties du log, et ce que nous allons en extraire :

- Le mois : *Nov*
- Le jour : *17*
- L'heure : *21:01:10*
- L'IP du proxy : *192.168.1.1*
- Le nom du proxy : *squid*
- La version de Squid : *1*
- Un premier numéro non-identifié : *1510952470.370*
- Un second numéro non-identifié : *233176*
- L'IP du PC émetteur de la requête : *192.168.1.103*
- La méthode utilisée : *TCP_TUNNEL*
- Le code HTTP / HTTPS retourné : *200*
- Un troisième numéro non-identifié : *5931*
- Le résultat de la requête : *CONNECT*
- Le nom du site de destination : *etherpad.fr*
- Le port de destination : *443*
- Un label : *ORIGINAL_DST*
- L'ip de destination : *195.154.57.241*

Maintenant que nous savons ce que nous allons extraire du fichier de log, attardons nous sur la création d'expression Grok pour Logstash. Premièrement, il est important de tester en temps réel nos expressions pour limiter les erreurs. Pour ce faire, le site https://grokdebug.herokuapp.com/ va nous permettre de passer en paramètre une ligne de log typique, et de construire le filtre dessus, tout en visualisant le résultat.
Un élément Grok à la syntaxe suivante : %{TYPE:nom}. Le type peut être un nombre "NUMBER", un mot "WORD", un entier "INT", une IP "IP", des données "DATA", etc… Le nom servira à identifier les éléments correspondant à l'expression régulière.

Par exemple, %{WORD:month} nous retournera “Nov” :



Sachant que les expression grok sont sensibles à la casse, même avec les espaces, et que le caractère d'échappement est un “\”, nous pouvons construire cette expression régulière :

```
%{WORD:month} %{NUMBER:date} %{TIME:timestamp} %{IP:proxy} \(%{WORD:squid_version}-%{INT:http_status_code}\)\: %{DATA:header1} \s*%{NUMBER:header2} %{IP:ip_source} %{WORD:protocole}\/%{INT:code} %{NUMBER:header3} %{WORD:status} %{DATA:destination}\:%{NUMBER:port_destination} \- %{WORD:label_destination}\/%{DATA:ip_destination} \-
```

Les quelques spécificitées de l’expression sont les suivantes :

- Puisque le nombre d’espace entre l'élément “numéro non identifié 1” et “numéro non identifié 2” est différent en fonction des logs, nous utilisons le paramètre “ \s* ” qui permet de dépasser les espaces, peu importe leurs nombres, et d’aller directement à la prochaine expression.
- Les parenthèse, double points et tirets sont précédés du caractères d'échappement.

En essayant notre filtre grok sur le debugger en ligne, nous obtenons biens nos différents éléments :

Nous pouvons maintenant ajouter notre filtre dans le fichier pfsense_log.conf, dans la rubrique “filter” :

```
filter {
  grok {
    match => {
        "message" => "%{WORD:month} %{NUMBER:date} %{TIME:timestamp} %{IP:proxy} \(%{WO
RD:squid_version}-%{INT:http_status_code}\)\: %{DATA:header1} \s*%{NUMBER:header2} %{IP
:ip_source} %{WORD:protocole}\/%{INT:code} %{NUMBER:header3} %{WORD:status} %{DATA:dest
ination}\:%{NUMBER:port_destination} \- %{WORD:label_destination}\/%{DATA:ip_destinatio
n} \-"
    }
  }
}
```

Passons maintenant au fichier “pfsense_template.json”. Celui-ci contient notre index ElasticSearch. l’index nous servira pour créer les graphiques Kibana.
Dans ce fichier, nous décrivons comment traiter les données filtrés provenant de Logstash. La configuration du template est la suivante (trop longue pour être placée ici) :

https://github.com/HomardBoy/Squid-to-Kibana/blob/master/pfsense_template.json

Nous pouvons maintenant lancer Logstash avec nos deux fichiers de configurations. La commande est la suivante : “*logstash -f pfsense_log.conf*”. La console devrait afficher ce message :

Settings: Default pipeline workers: 4
Logstash startup completed

Cela signifie que les fichiers ont bien été interprétés par logstash, et qu’il n’y a pas d’erreurs. Si l’on fait un hit dans Logstash, en ajoutant un log dans notre fichier de log, nous devrions voir dans la console que ce log est traité et filtré correctement.
Si tout se passe correctement jusqu’ici, Logstash à terminé d’être configuré, et nous pouvons passer à la partie suivante.

IV. Configuration d’ElasticSearch.

La configuration d’ElasticSearch est minime, puisque les données n’ont plus qu’a être stockés dans un index. Nous lançons simplement le service en l’appelant, dans le dossier bin de elasticsearch : “*elasticsearch*”.
Celui-ci prend un certain temps pour se lancer, mais un fois opérationnel, nous devrions être capable d’accéder à l’URL *127.0.0.1:9200* (le port elasticsearch) :

```
{
  "name" : "Marvel Man",
  "cluster_name" : "elasticsearch",
  "version" : {
    "number" : "2.2.1",
    "build_hash" : "d045fc29d1932bce18b2e65ab8b297fbf6cd41a1",
    "build_timestamp" : "2016-03-09T09:38:54Z",
    "build_snapshot" : false,
    "lucene_version" : "5.4.1"
  },
  "tagline" : "You Know, for Search"
}
```


Si une page similaire s'affiche, le service est correctement lancé. Essayons maintenant d'atteindre notre index : *127.0.0.1:9200/pfsense_log*
Si une page du type s'affiche, alors ElasticSearch alimente bien notre fichier d'index.
Nous pouvons dès maintenant s'occuper de Kibana :

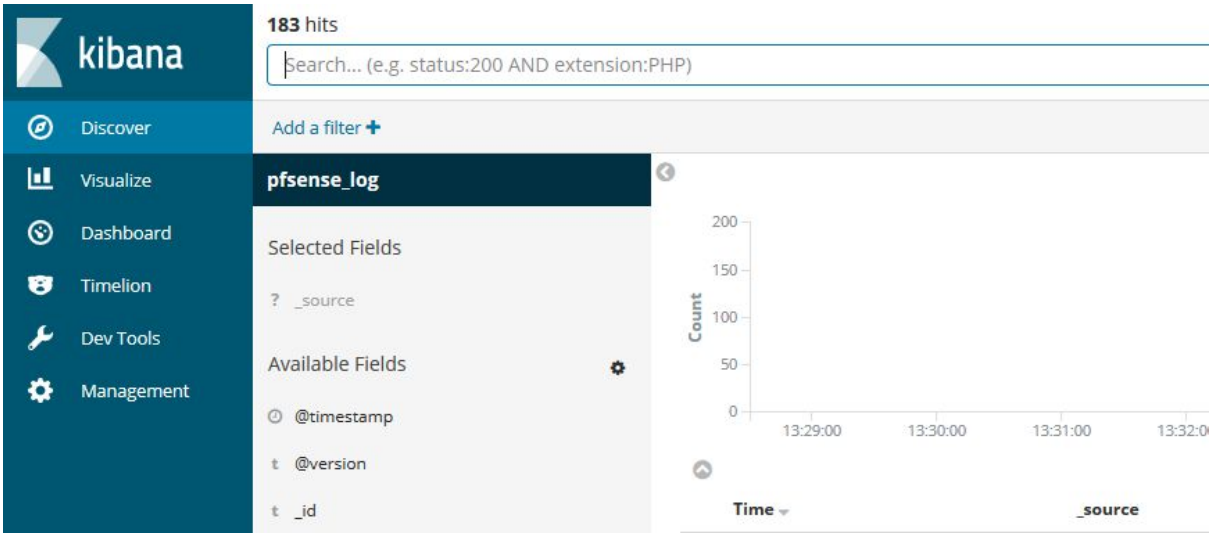

```
{"pfsense_log":{"aliases":{},"mappings":{"doc":{"properties":{"@timestamp":
{"type":"date"},"@version":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"code":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"date":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"destination":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"fqdn_destination":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"header1":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"header2":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"header3":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"host":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"http_status_code":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"ip_destination":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"ip_source":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"label_destination":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"message":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"month":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"path":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"port_destination":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"protocole":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"proxy":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"squid_version":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"status":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"tags":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"timestamp":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}}}}}},"settings":{"index":
{"creation_date":"1511166150614","number_of_shards":"5","number_of_replicas":"1","uuid":"4iXC7idCRD
G7jkmrcqst5Q","version":{"created":"6000099"},"provided_name":"pfsense_log"}}}}
```


V. Configuration de Kibana.

Le service Kibana n'a pas besoin d'énormément de configurations de notre part, puisque les trois services sont faits pour fonctionner nativement ensemble. L'unique chose pouvant être faite au niveau de fichier de configuration sera de permettre l'accès à l'interface web de Kibana à distance.
En effet, par défaut, il est uniquement possible d'y accéder depuis la machine sur lequel le service tourne en local. Nous éditons le fichier "*/conf/kibana.yml*" dans le dossier de Kibana. Nous ajoutons la ligne suivante :

```
server.host: 172.16.55.17
```

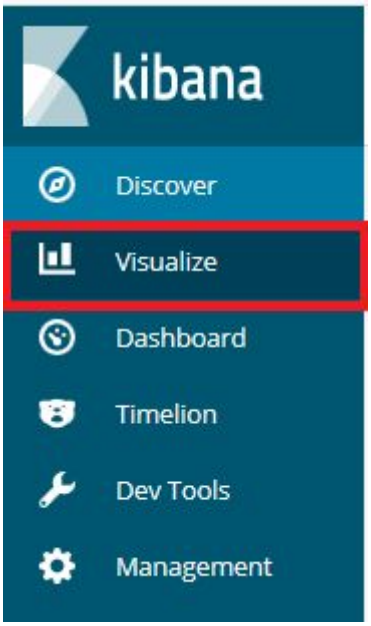
Essayons maintenant d'accéder à l'interface web de Kibana à l'adresse : 127.0.0.1:5601 :



Nous sommes désormais prêt à créer notre Dashboard pour administrer les connexions des utilisateurs de notre LAN.

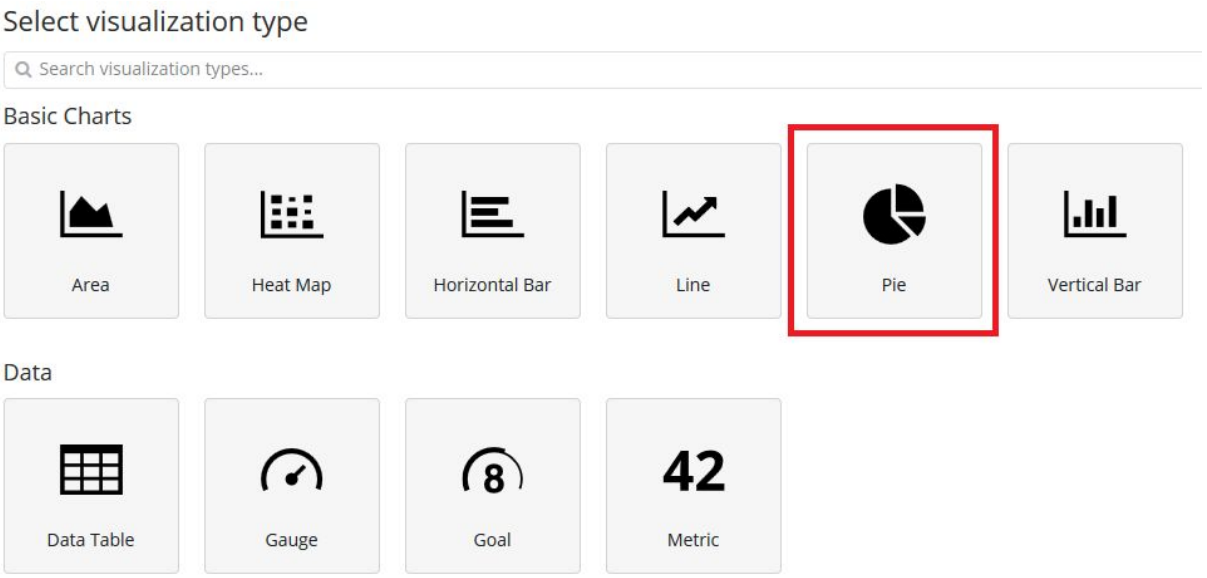
VI. Création du Dashboard.

Avant de créer notre DashBoard, nous allons paramétrer les différentes modules un par un :

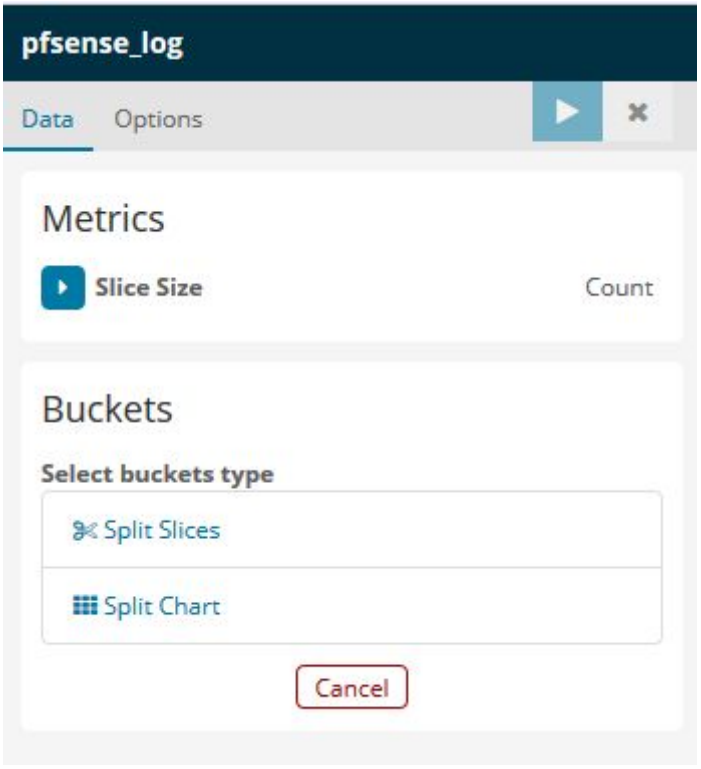


Une fois dans le module "Visualize", nous ajoutons un nouveau graphique :

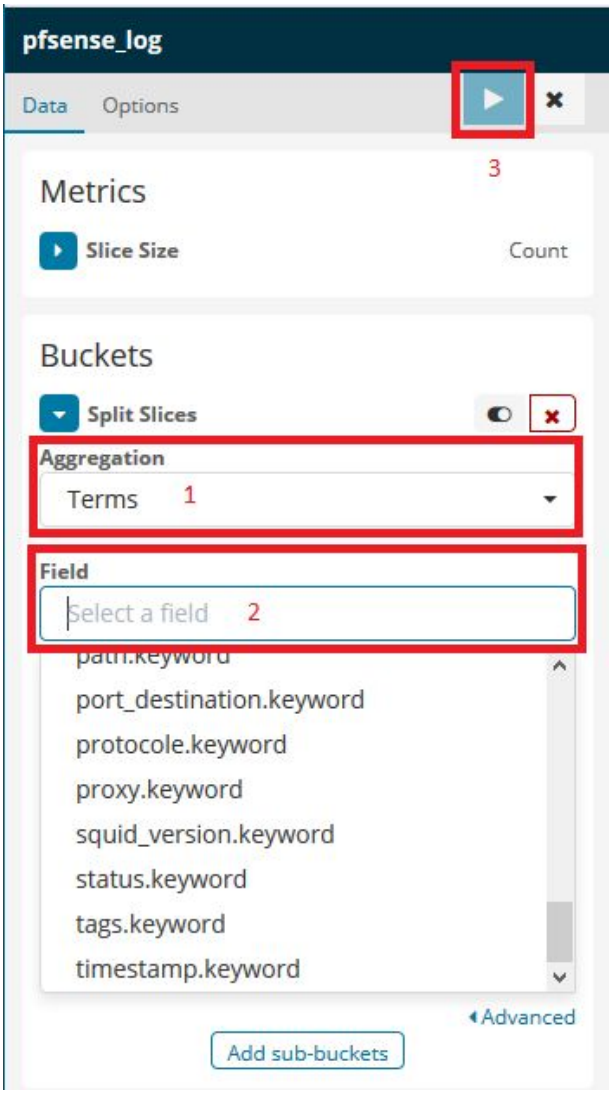
Nous choisissons le type de schémas que nous souhaitons ajouter. Le plus intuitif et visuel s'avère être le diagramme en cercle :



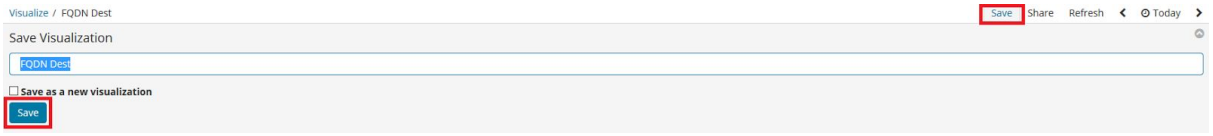
Une nouvelle fenêtre de création va ensuite s'ouvrir. Nous sélectionnons l'option "Split Splice" dans le menu de gauche :



Enfin, nous indiquons que le filtrage se fait sur les termes filtrés par ElasticSearch, avant de choisir le terme en question. Pour générer un aperçu du graphique, nous cliquons sur la flèche en haut à droite de la boîte d'options :



Si le graphique est convainquant, nous le sauvegardons via les options présentes dans la partie supérieure de la fenêtre :



Uns fois que nos différents graphiques sont générés, nous allons les regrouper au sein d'un DashBoard. Pour créer un dashboard, rendez-vous dans le menu "Dashboard" de gauche, et ajoutez les graphiques avec le bouton "Add" présent dans la partie supérieur droite de l'écran. Notre DashBoard type permettre d'observer le nombres de requêtes traités, les destinations les plus sollicités, les IP dont le plus grand nombre de requête sont émises, le nombre de requêtes en fonction du temps, etc …

Le Dashboard est le suivant :

Nous sommes maintenant capables d'observer précisément l'étendu des requêtes sortantes de notre réseau LAN.
Quelques petites informations additionnels :
- Il sera parfois nécessaire de rafraîchir la liste des filtres logstash. Pour ce faire, au sein de Kibana, accédez à l'onglet Management / Index Patterns, puis sur le bouton de refresh en haut à droite.
- Si aucunes données n'est affichées sur le dashBoard, ou s'il n'y a plus aucuns hits sur le menu principal, pas de panique, l'interval de temps d'observation par défaut n'est pas très long. Pour le modifier, rendez-vous sur le menu "Discover" puis sur l'expression temporelle affichée en haut à droite de l'écran. Vous serez alors capable de modifier la période temporelle à traiter.
- L'ensembles des fichiers de configurations utilisés ici sont récupérable dans un dépôt github créé spécialement pour l'occasion : https://github.com/HomardBoy/Squid-to-Kibana