SIEM : TP Projet

Locqueneux Owen

3C

Contexte :

L'objectif de ce TP est de tout d'abord mettre en place un conteneur docker (ça peut être un server web comme par exemple apache, nginx….) via un fichier docker-compose.yml.

Une fois le conteneur docker créé on va faire en sorte de récupérer les logs de ce container et les placer dans la stack ELK.

Préparation (Set up) :

On commence par cloner un repo github sur notre machine avec cette commande :

```
git clone https://github.com/ashishtiwari1993/elastic-docker.git
```

Pour ma part j'utilise une VM kalilinux pour la réalisation de ce TP projet.

Une fois le repo github cloné on se retrouve avec ces fichiers :

```
┌──(owenⓀkali)-[~/Documents/SIEM]
└─$ ls
elastic-docker

┌──(owenⓀkali)-[~/Documents/SIEM]
└─$ cd elastic-docker

┌──(owenⓀkali)-[~/Documents/SIEM/elastic-docker]
└─$ ls
docker-compose.yml  pipeline  README.md
```

Ce qui nous interesse ici c'est le fichier docker-compose.yml que l'on va par la suite modifier.

Si l'on regarde de plus près le contenu de ce fichier.yml on remarque qu'il s'agit d'un fichier de configuration pour déployer des conteneurs Docker de la suite Elastic Stack : Elasticsearch, Kibana, logstash et APM Server, qui forment une suite de logiciels de gestion de logs, de métriques et de traces.

On va lancer ce fichier de configuration .yml avec la commande suivante, avec comme option -d si l'on veut éviter que les logs s'affichent et défient sur notre cmd :

```
sudo docker-compose up -d
```

Cependant on rencontre une erreur lors de l'execution du fichier :

```
┌──(owen㉿kali)-[~/Documents/SIEM/elastic-docker]
└─$ sudo docker-compose up -d
Creating network "elastic-docker_default" with the default driver
Creating elastic-docker_setup_1 ... done
Creating elastic-docker_es01_1  ... done
Creating elastic-docker_logstash_1 ... done
Creating elastic-docker_apm_1      ... done

ERROR: for kibana  Container "f21070a2969a" is unhealthy.
ERROR: Encountered errors while bringing up the project.
```

On va maintenant essayer de regler le problème. Tout d'abord on va regarder l'état des conteneurs qui tournent sur notre machine :

```
┌──(owen㉿kali)-[~/Documents/SIEM/elastic-docker]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED         STATUS
4e28d730fddd   docker.elastic.co/apm/apm-server:8.6.1                 "/usr/bin/tini -- /u…"   7 minutes ago   Up 7 minutes
686d1f6d65dc   docker.elastic.co/logstash/logstash:8.6.1              "/usr/local/bin/dock…"   7 minutes ago   Up 7 minutes
f21070a2969a   docker.elastic.co/elasticsearch/elasticsearch:8.6.1    "/bin/tini -- /usr/l…"   7 minutes ago   Exited (78) 7 minutes ag
d899e8a12e70   docker.elastic.co/elasticsearch/elasticsearch:8.6.1    "/bin/tini -- /usr/l…"   8 minutes ago   Up 8 minutes (healthy)
```

On remarque que celui ayant l'ID "f21070a2969a" n'a pas réussi à s'installer, ce qui rejoint l'erreur que l'on a rencontré precedemment.

On continue à chercher en regardant les logs de ce container pour trouver le problème :

```
┌──(owen㉿kali)-[~/Documents/SIEM/elastic-docker]
└─$ sudo docker logs f21070a2969a | grep "r1"
bootstrap check failure [1] of [1]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]
ERROR: Elasticsearch did not exit normally - check the logs at /usr/share/elasticsearch/logs/docker-cluster.log

ERROR: [1] bootstrap checks failed. You must address the points described in the following [1] lines before starting Elasticsearch.
```

L'erreur nous indique que la valeur de la mémoire virtuel max "vm.max_map_count" est trop basse, on va donc l'augmenter :

```
sudo sysctl -w vm.max_map_count=262144
```

Le problème étant qu'à chaque démarage de ma machine virtuelle il va falloir entrer cette commande pour ne pas avoir d'erreur, pour éviter cela, on va tout simplement implémenter cette commande "vm.max_map_count=262144" dans notre fichier de configuration sysctl.conf.

```
sudo nano /etc/sysctl.conf
```

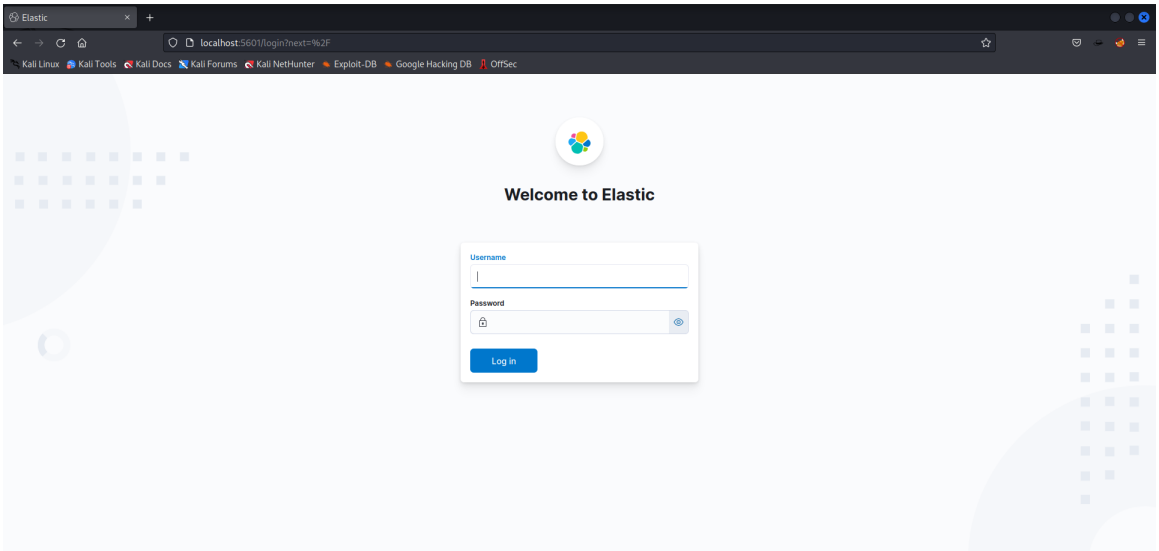
Lorsque l'on réexcute le fichier docker-compose on ne retrouve plus d'erreur et tout nos conteneurs sont lancés :

```
┌──(owenⓀkali)-[~/Documents/SIEM/elastic-docker]
└─$ sudo docker-compose down
Stopping elastic-docker_apm_1      ... done
Stopping elastic-docker_logstash_1 ... done
Stopping elastic-docker_setup_1    ... done
Removing elastic-docker_apm_1      ... done
Removing elastic-docker_logstash_1 ... done
Removing elastic-docker_es01_1     ... done
Removing elastic-docker_setup_1    ... done
Removing network elastic-docker_default

┌──(owenⓀkali)-[~/Documents/SIEM/elastic-docker]
└─$ sudo docker-compose up -d
Creating network "elastic-docker_default" with the default driver
Creating elastic-docker_setup_1 ... done
Creating elastic-docker_es01_1  ... done
Creating elastic-docker_kibana_1   ... done
Creating elastic-docker_logstash_1 ... done
Creating elastic-docker_apm_1      ... done

┌──(owenⓀkali)-[~/Documents/SIEM/elastic-docker]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE                                                 COMMAND                  CREATED              STATUS
43db28f2f8c2   docker.elastic.co/logstash/logstash:8.6.1             "/usr/local/bin/dock…"   9 seconds ago        Up 3 seconds
26ebfda92d54   docker.elastic.co/apm/apm-server:8.6.1                "/usr/bin/tini -- /u…"   9 seconds ago        Up 2 seconds
819c71c676ce   docker.elastic.co/kibana/kibana:8.6.1                 "/bin/tini -- /usr/l…"   9 seconds ago        Up 3 seconds (healt
a5b54750a63b   docker.elastic.co/elasticsearch/elasticsearch:8.6.1   "/bin/tini -- /usr/l…"   About a minute ago   Up About a minute (
9e74c82f00a9   nginx:latest                                          "/docker-entrypoint.…"   About a minute ago   Up About a minute
f7db231a4225   docker.elastic.co/elasticsearch/elasticsearch:8.6.1   "/bin/tini -- /usr/l…"   About a minute ago   Up About a minute (
```

Il est ainsi possible d'accéder à l'interface graphique de la stack ELK en tapant localhost:5601 dans la barre de recherche de notre navigateur (Username: elastic, Password: pass@123) :



Création du serveur nginx:

Pour la création de ce conteneur on va éditer le fichier docker-compose.yml :

```
nginx:
    image: nginx:latest
    container_name: nginx
    ports:
      - "80:80"
   volumes:
      - ./nginx/logs:/var/log/nginx
```

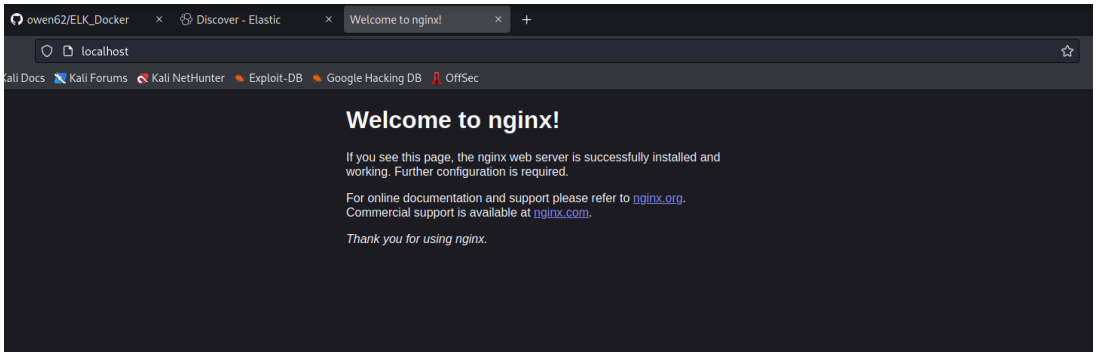
Une fois le fichier enregistré et le conteneur nginx lancé, on vérifie que le serveur nginx tourne bien avec un simple curl :

```
┌──(owenⓀkali)-[~/Documents/SIEM/elastic-docker]
└─$ curl localhost 80:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

On peut tout aussi bien taper dans la barre de recherche localhost:80 qui corresponds au port que l'on a choisi dans notre docker-compose :



La commande ci-dessous montre les logs du serveur, on remarquera que ces logs indiquent que le server a bien réussi à s'installer et qu'un utilisateur a lancé la commande "curl" sur ce serveur :

```
┌──(owenⓀkali)-[~/Documents/SIEM/elastic-docker]
└─$ sudo docker logs -f nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/03/24 14:02:56 [notice] 1#1: using the "epoll" event method
2023/03/24 14:02:56 [notice] 1#1: nginx/1.23.3
```

```
2023/03/24 14:02:56 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/03/24 14:02:56 [notice] 1#1: OS: Linux 5.19.0-kali2-amd64
2023/03/24 14:02:56 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/03/24 14:02:56 [notice] 1#1: start worker processes
2023/03/24 14:02:56 [notice] 1#1: start worker process 29
2023/03/24 14:02:56 [notice] 1#1: start worker process 30
2023/03/24 14:02:56 [notice] 1#1: start worker process 31
2023/03/24 14:02:56 [notice] 1#1: start worker process 32
172.19.0.1 - - [24/Mar/2023:14:13:39 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/7.85.0" "-"
2023/03/24 14:19:01 [error] 31#31: *2 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.19
172.19.0.1 - - [24/Mar/2023:14:19:01 +0000] "GET /favicon.ico HTTP/1.1" 404 153 "http://localhost/" "Mozilla/5.0 (X11; Linux x86_64; r
```

Installation de Filebeat :

Filebeat fait partie de la suite Elastic autrement dit, il est conçu pour fonctionner de manière homogène avec Logstash, Elasticsearch et Kibana.

Filebeate est équipé de plusieurs modules pour les sources de données d'observabilité et de sécurité qui simplifient la collecte, l'analyse et la visualisation des formats de logs les plus courants. Le tout, via une seule commande. Pour ce faire, ces modules associent les valeurs automatiques par défaut basées sur le système d'exploitation avec les définitions du pipeline d'ingestion Elasticsearch Ingest Node et les tableaux de bord Kibana.

On va donc créer un conteneur filebeat dans notre fichier docker-compose.yml :

```
175
176   filebeat:
177     depends_on:
178       es01:
179         condition: service_healthy
180     image: docker.elastic.co/beats/filebeat:${STACK_VERSION}
181     container_name: filebeat
182     volumes:
183       - ./filebeat.yml:/usr/share/filebeat/filebeat.yml
184       - ./test.log:/var/log/app_logs/test.log
185       - certs:/usr/share/elasticsearch/config/certs
186     environment:
187       - ELASTICSEARCH_HOSTS=https://es01:9200
188       - ELASTICSEARCH_USERNAME=elastic
189       - ELASTICSEARCH_PASSWORD=${ELASTIC_PASSWORD}
190       - ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES=config/certs/ca/ca.crt
191
192
```

Ce container va être utilisé pour envoyer les logs d'une application vers Elasticsearch. Il va lire les logs de notre conteneur nginx, qui est un serveur Web, et les envoyer à Elasticsearch. La configuration de filebeat est stockée dans un fichier "filebeat.yml" qui est monté en tant que volume dans le conteneur filebeat. Le fichier de configuration contient des informations sur les chemins des fichiers de logs à lire, ainsi que les paramètres de connexion pour se connecter à Elasticsearch, tels que les noms d'utilisateur et les mots de passe. On va également utilisé dans ce conteneur, des volumes afin d'accéder aux fichiers de logs de l'application nginx, ainsi qu'aux certificats SSL nécessaires pour se connecter à Elasticsearch de manière sécurisée.

On va également rajouter quelques lignes de code dans la partie setup de notre fichier docker-compose.yml afin de générer un certificat pour notre container filebeat :

```
services:
  setup:
    image: docker.elastic.co/elasticsearch/elasticsearch:${STACK_VERSION}
    volumes:
      - certs:/usr/share/elasticsearch/config/certs
    user: "0"
    command: >
      bash -c '
        if [ x${ELASTIC_PASSWORD} == x ]; then
          echo "Set the ELASTIC_PASSWORD environment variable in the .env file";
          exit 1;
        elif [ x${KIBANA_PASSWORD} == x ]; then
          echo "Set the KIBANA_PASSWORD environment variable in the .env file";
          exit 1;
        fi;
        if [ ! -f config/certs/ca.zip ]; then
          echo "Creating CA";
          bin/elasticsearch-certutil ca --silent --pem -out config/certs/ca.zip;
          unzip config/certs/ca.zip -d config/certs;
        fi;
        if [ ! -f config/certs/certs.zip ]; then
          echo "Creating certs";
          echo -ne \
          "instances:\n"\
          "  - name: es01\n"\
          "    dns:\n"\
          "      - es01\n"\
          "      - localhost\n"\
          "    ip:\n"\
          "      - 127.0.0.1\n"\
          "  - name: filebeat\n"\
          "    dns:\n"\
          "      - es01\n"\
          "      - localhost\n"\
          "    ip:\n"\
          "      - 127.0.0.1\n"\
          > config/certs/instances.yml;
```

Voici les commandes que l'on rajoute:

```
"  - name: filebeat\n"\
          "    dns:\n"\
          "      - es01\n"\
          "      - localhost\n"\
          "    ip:\n"\
          "      - 127.0.0.1\n"\
          > config/certs/instances.yml;
```

Ensuite on configure le fichier filebeat.yml :

filebeat.yml

home > owen > Documents > SIEM > elastic-docker > filebeat.yml

```
filebeat.inputs:
- type: filestream
  id: my-application-logs
  enabled: true
  paths:
    - /var/log/app_logs/*.log
output.elasticsearch:
  hosts: '${ELASTICSEARCH_HOSTS:elasticsearch:9200}'
  username: '${ELASTICSEARCH_USERNAME:}'
  password: '${ELASTICSEARCH_PASSWORD:}'
  ssl:
    certificate_authorities: "/usr/share/elasticsearch/config/certs/ca/ca.crt
    certificate: "/usr/share/elasticsearch/config/certs/filebeat/filebeat.crt
    key: "/usr/share/elasticsearch/config/certs/filebeat/filebeat.key"
```

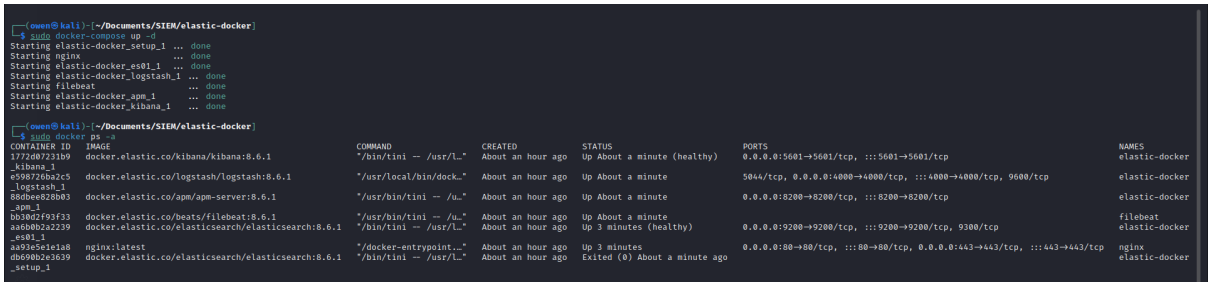
Ce fichier de configuration est utilisé par le système de collecte de logs et dans notre cas il se compose de deux sections principales: "filebeat.inputs" et "output.elasticsearch".

Dans la section "filebeat.inputs", nous définissons l'entrée des logs que nous souhaitons collecter. Dans cet exemple, nous utilisons le type "filestream" pour spécifier que nous collectons des logs de fichiers. L'ID est un identifiant unique pour cette entrée de logs, et "enabled" est défini sur true pour activer la collecte de ces logs. Nous spécifions également le chemin des fichiers de logs que nous souhaitons collecter, dans ce cas-ci "/var/log/app_logs/*.log".

Dans la section "output.elasticsearch", nous définissons où nous souhaitons envoyer les logs collectés. Nous spécifions le ou les hôtes Elasticsearch sur lesquels nous voulons envoyer les logs via la propriété "hosts". Si la variable d'environnement "ELASTICSEARCH_HOSTS" est définie, elle prendra le pas sur la valeur définie dans le fichier de configuration. Nous spécifions également l'authentification via les propriétés "username" et "password", qui peuvent être lues à partir des variables d'environnement "ELASTICSEARCH_USERNAME" et "ELASTICSEARCH_PASSWORD". Enfin, nous spécifions les chemins des fichiers de certificat SSL pour établir une connexion sécurisée avec Elasticsearch.

Lancement des conteneurs

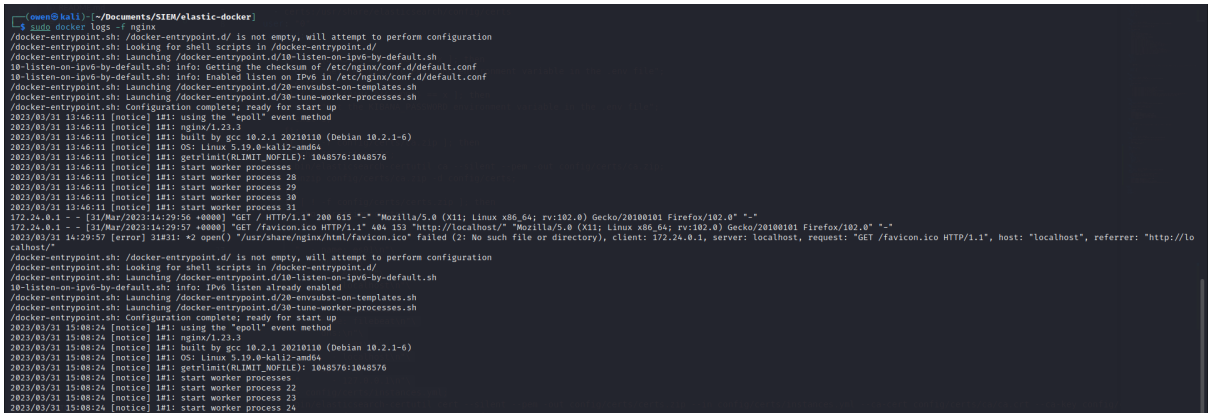
Une fois nos fichiers configurés, on va pouvoir lancer nos conteneurs avec la commande docker-compose -d et en vérifiant qu'ils sont tous bien lancés, ce qui est le cas ci-dessous :



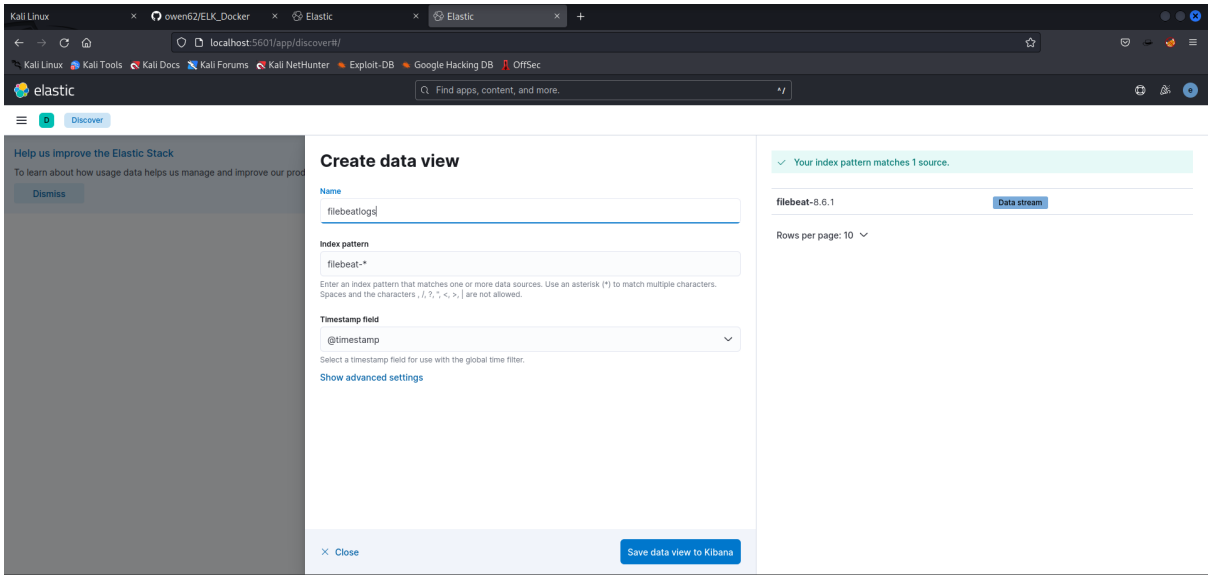
Maintenant vérifions la présence des logs des conteneurs nginx et filebeat :

- Filebeat :

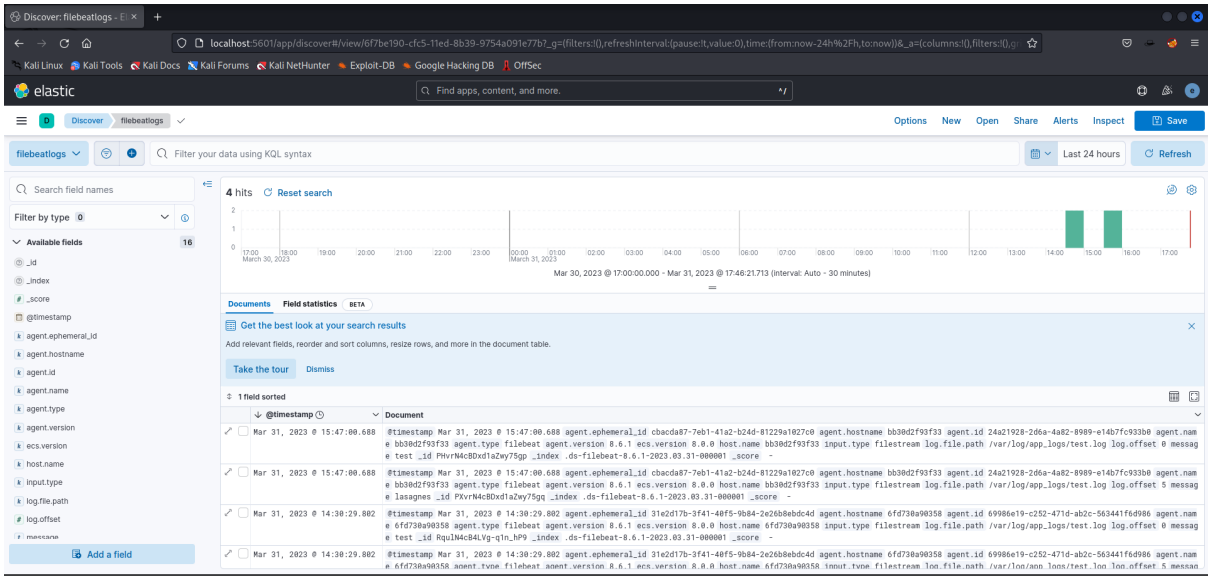
- Nginx :



Désormais, tout est configuré et lancé, on peut aller sur l'interface kibana en se connectant à la page de login d'elastic en tapant localhost:5601 dans la barre de recherche de notre navigateur, on se dirige vers la page "discover" et on créé un "visuel de nos données" :



Une fois que c'est créé, un tableau de bord nous affiche ce que l'on vient d'envoyer de notre conteneur nginx vers elasticsearch.



Conclusion :

En somme, l'Elastic Stack est un ensemble d'outils puissant, mais qui peut parfois s'avérer complexe à utiliser. Docker permet de faciliter grandement l'exécution de cet ensemble.Cependant, il est également important de souligner son importance dans le domaine de la cybersécurité.

La stack ELK est utilisée par de nombreuses organisations pour détecter et prévenir les attaques de sécurité. En collectant et en analysant les logs de différents systèmes, elle peut aider à identifier les comportements suspects et à prendre des mesures proactives pour protéger les données de l'entreprise.

De plus, l'utilisation de la stack ELK avec des outils de sécurité supplémentaires tels que Beats, Logstash et Kibana permet aux équipes de sécurité d'avoir une vue d'ensemble des événements de sécurité en temps réel. Cela permet une détection rapide des menaces, une réponse plus rapide aux incidents de sécurité et une amélioration globale de la posture de sécurité de l'entreprise.

En résumé, la stack Elastic joue un rôle important dans la cybersécurité en aidant les entreprises à surveiller leurs systèmes, à détecter les menaces et à prendre des mesures proactives pour protéger leurs données.