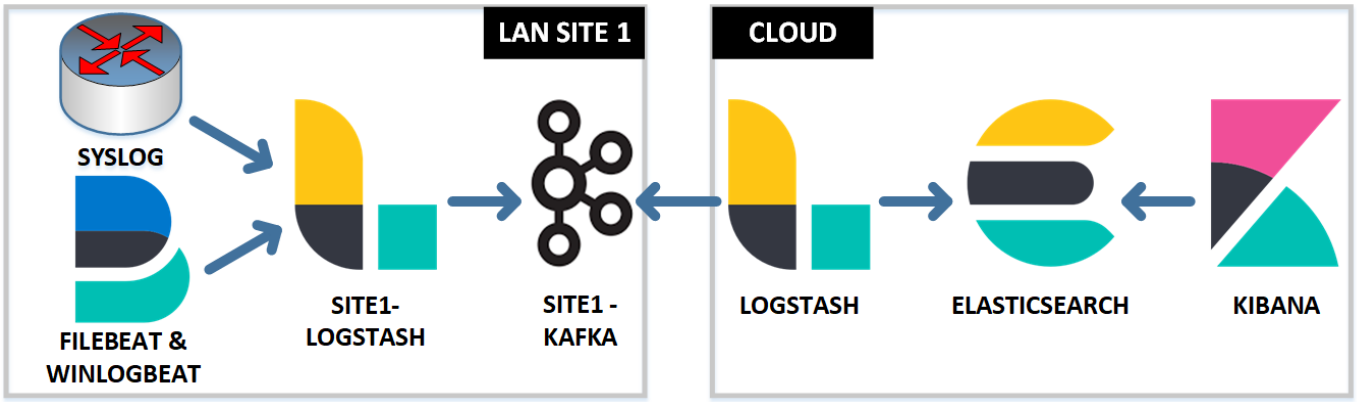
Censured

RAPPORT DE PROJET

Mise en place d'une infrastructure basée sur la stack ELK



Année universitaire 2020-2021
Master 2 R&T ASR-alt

Tuteur Entreprise : Censured
Tuteur MASTER : Censured

REMERCIEMENTS

Pour commencer, je tiens à adresser mes remerciements aux membres de la formation Master R&T ASR et plus particulièrement à Censured et Censured. En effet ils ont su s'adapter pour continuer à nous proposer des modules de qualités malgré cette période un peu particulière. (Surtout au niveau des intervenants professionnels).

Je me dois aussi de remercier mon Responsable d'entreprise Censured, pour la confiance qu'il m'a portée tout au long de ces trois années d'alternance. Ce qui m'a permis de développer mon autonomie ainsi qu'a monté en compétences dans le domaine IT.

Je remercie également l'ensemble du personnel CFA BTP GRAND EST pour leur rigueur ainsi que pour les différentes aides apportées lors de mes périodes d'entreprise.

SOMMAIRE

INTRODUCTION

Je suis actuellement en deuxième année de Master Informatique Mention Réseaux Télécom, parcours Administration et Sécurité des Réseaux (ASR) à Reims en alternance. Comme pour la première année de master une demande de réalisation d'un projet m'a été confié.

Tout d'abord, il faut savoir que j'effectue mon alternance dans le service informatique de l'association régionale CFA BTP GRAND EST depuis presque trois années maintenant. Le nombre de sites a gérer ayant considérablement augmenté, le service informatique se doit de mettre en place de nouveaux outils permettent de gérer au mieux ces nouveaux établissements.

Afin de répondre à ses besoins, j'avais eu l'occasion de mettre en place un outil de supervision complet lors de mon projet de première année de Master. Cet outil de supervision regroupe aujourd'hui l'ensemble de nos équipements Systèmes et Réseaux tels que les commutateurs, bornes wifi, imprimantes, routeurs, bande passante ect …Ainsi qu'une cartographie de chaque sites (Centreon + Nagvis)

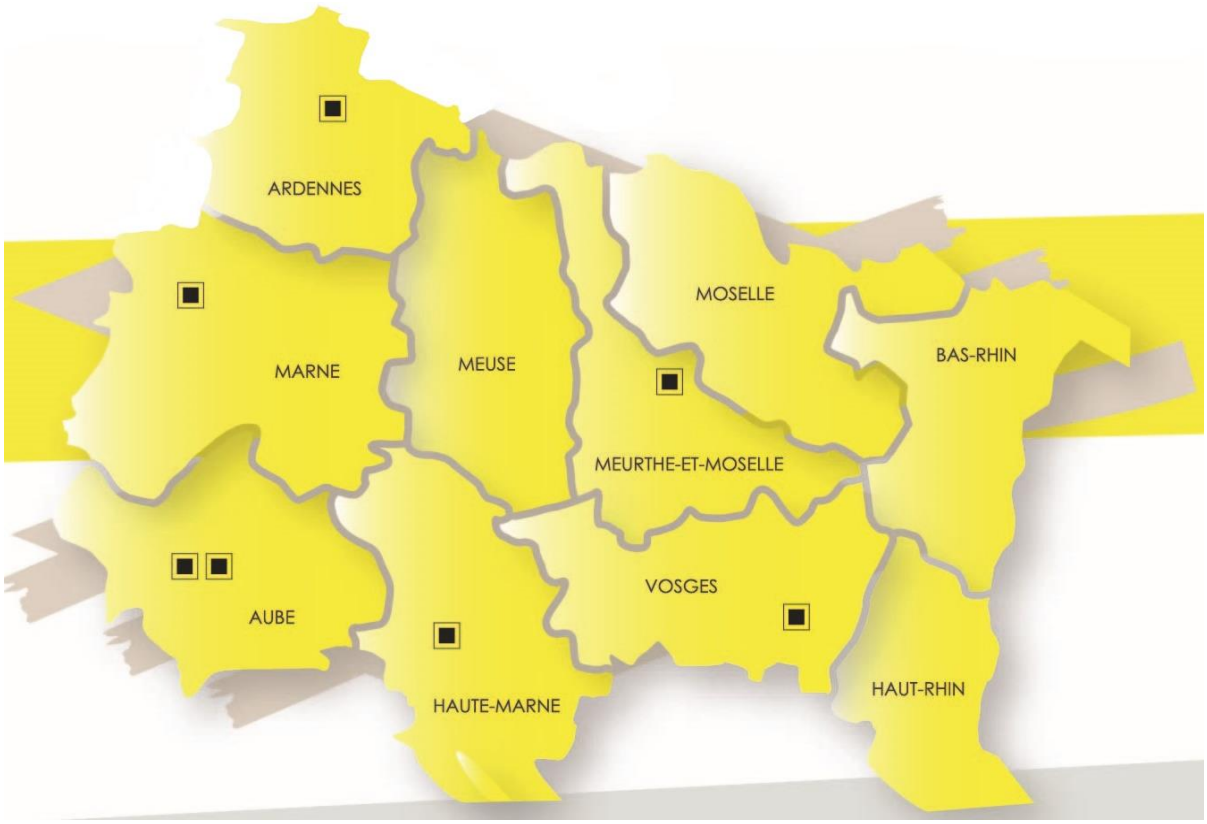
Suite à ce projet de supervision actif, la suite logique selon moi été de mettre en place un système de monitoring de logs et d'événements. C'est pourquoi j'ai choisi pour mon projet de deuxième année de mettre en place une infrastructure basée sur la stack ELK dans un lab test.

Je vais dans un premier temps vous présenter rapidement mon entreprise en vous indiquant les différents besoins et contraintes menant à ce projet. Par la suite je vous ferai part d'une explication générale du projet (avec un schéma détaillé voir annexe 1). Pour ensuite vous donner une explication détaillée de chaque partie, enfin je vous indiquerai la réponse aux besoins et contraintes, se clôturant par une conclusion.

I. LE CFA BTP GRAND EST

a. PRÉSENTATION

LE CFA BTP GRAND EST est une **association régionale**, elle fut créée suite à la fusion avec BTP CFA Champagne-Ardenne et BTP CFA Lorraine le 1er janvier 2018 et avec Pont-à-Mousson le 9 janvier. Cette association regroupe maintenant **six centres de formations d'apprentis aux Métiers du Bâtiment et des Travaux Publics et Institut Universitaire des Métiers** : BTP CFA ARDENNES, BTP CFA AUBE, BTP CFA MARNE, BTP CFA HAUTE-MARNE, BTP CFA MEURTHE-ET-MOSELLE ET MEUSE, BTP CFA VOSGES ET L'IUMP DE TROYES. Le siège du CFA BTP GRAND EST a pour but d'apporter un support sur ces différents établissements.



Ces différents CFA proposent des **formations initiales** et **continues aux apprenants** afin de faciliter leurs insertions professionnelles. De nombreux diplômes y sont disponibles tel que le Certificat d'aptitude professionnelle (**CAP**), la Mention Complémentaire (**MC**), le Brevet Professionnel (**BP**), le Baccalauréat Professionnel (**BAC PRO**) et le Brevet de Technicien Supérieur (**BTS**). Comprenant un choix varié de métiers du Bâtiment et des Travaux publics comme par exemple les métiers de Maçon, Couvreur, Installateur Thermique, Canalisateurs….

b. LE CFA BTP GRAND EST EN QUELQUES CHIFFRES

c. LES BESOINS ET CONTRAINTES MENANT A CE PROJET

Comme évoqué lors de mon rapport de projet de master 1, le CFA BTP GRAND EST comprend maintenant **sept établissements ainsi qu'un siège social** répartis dans la région GRAND EST. **Le service informatique doit être en mesure d'apporter un support efficace sur ces sites.**

Pour répondre à ses besoins grandissants j'ai eu l'occasion dans le cadre mon projet de Master 1 de mettre en place un outil de supervision actif permettant de répondre aux besoins suivants :

- **Centraliser** la supervision active de tous les établissements
- **Référencer** l'ensemble des équipements réseaux et serveurs
- **Apporter des informations** sur l'état de santé de ces hôtes en temps réel
- **Fournir une cartographie** de la topologie réseau de chaque site afin de facilité la visualisation des incidents pouvant survenir.

Aujourd'hui mon projet de master 2 s'intitule : Mise en place d'une infrastructure basée sur la stack ELK. Ce projet est un LAB TEST, c'est-à-dire qu'il n'est pas encore déployé au sein de mon entreprise. Il aura pour but de se combiner à l'outil de supervision actif, en effet il va permettre de répondre aux besoins suivants :

- **Centraliser** un ensemble de logs/évènements de tous les établissements
- **Être informé en temps réel** de ce qu'il se passe sur le réseau et les serveurs
- **Recherche et Analyse** de l'ensemble de données relatives aux entités de mon entreprise
- **Apporter une couche de sécurité** avec la gestion d'alertes de sécurité

Cependant ce projet doit être en mesure de garantir :

- **La sécurité de notre réseau** (ex : Ne pas faire fuiter les logs/events)
- **Impacter le moins possible** réseau de production (ex : Disponibilité)
- **L'évolutivité de l'infrastructure** (ex : Si intégration de plusieurs sites)

II. EXPLICATION GÉNÉRALE DU PROJET

A SAVOIR : L'ensemble des ressources de ce projet se trouve sur mon github : https://github.com/blesseux/PROJET-ELK

a. INTRODUCTION DU PROJET

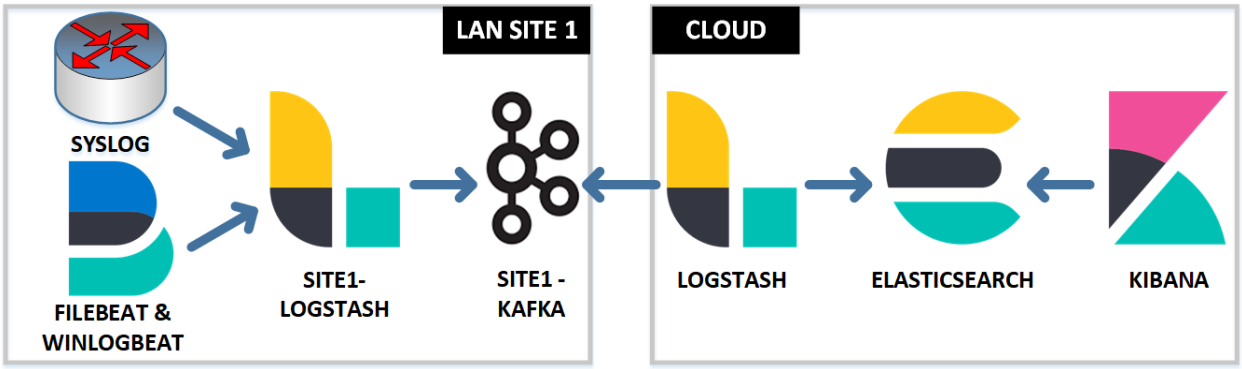


Schéma très simplifié du projet

>> VOIR Annexe 1 SCHÉMA DÉTAILLÉ DE L'INFRASTRUCTURE <<

Mon projet de cette deuxième année de master correspond à la **mise en place d'une infrastructure basée sur la stack ELK**. Ce **LAB TEST** (VMs locales) s'appuie sur les besoins et contraintes de mon entreprise. En effet, il permet d'avoir **une approche pratique à échelle réduite** (pour un site) d'un déploiement d'une telle infrastructure. À l'avenir si ce projet était intégré au CFA BTP GRAND EST il aurait pour missions de **centraliser l'ensemble des logs/événements de nos sept établissements**.

Lors de mes prochaines parties je vais vous expliquer le fonctionnement de ce LAB TEST étape par étape, c'est pourquoi je vous invite à vous référer à : Annexe 1 SCHÉMA DÉTAILLÉ DE L'INFRASTRUCTURE . Qui schématise en « détail » ce projet (le principal s'y trouve).

Points supplémentaires abordés dans ce projet :

- **Sécurisation des échanges :** chiffrement SSL entre site1-kafka et logstash
- **Sécurisation des accès :**
  - Entre site1-kafka (local) et logstash (cloud) : authentification bidirectionnelle par certificat + authentification unidirectionnelle SASL (par login)
  - Entre site1-logstash et site1-kakfa : authentification unidirectionnelle SASL (par login)
- **Contrôle de la consommation :** La bande passante du consommateur logstash est limité à 1Mb/s pour les topics Kafka.

b. LES ENTITÉS INTERVENANTES

Ce projet est composé de 6 entités hébergés en local sur ma machine physique via VirtualBox et GNS3 qui sont les suivantes :

Partie DATA SOURCE :

- **VM WEB :** hébergeant un serveur web NGINX + Service FILEBEAT (Agent de transfert léger conçu pour les logs)
- **ROUTER :** émulé via GNS3
- **VM WINDOWS :** Système exploitation windows 10 Pro + Service WINLOGBEAT (Agent léger conçu pour le transfert des logs d'événements Windows)

Partie DATA PARSE & BUFFER :

- VM BUFFER :
  - **Docker site1-logstash** : Parse log (Grosse partie du traitement des logs)
  - **Docker kafka** : Zone tampon par topics (dépendant de zookeeper-kafka)
  - **Docker zookeeper-kafka** : Agit comme une base de données pour kafka (stockage de configuration, états des partitions…)

Partie DATA PROCESSING, STORAGE et VISUALIZATION :

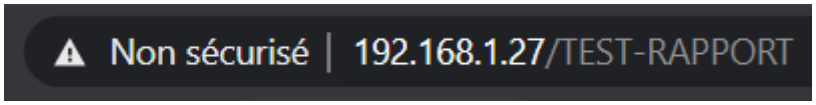
- VM ELK :
  - **Docker logstash :** Parse log (la plus grosse partie de l'analyse est géré par site1-logstash)
  - **Docker elasticsearch :** Stockage et Recherche
  - **Docker Kibana :** Visualisation des logs

Partie AUTRE :

- **VM rootCA :** Utilisé pour la communication SSL (signature de certificat+ truststore)

Voici un exemple simplifié d'un acheminement d'un log dans cette infra (voir annexe1 pour avoir un visuel) :

1. Un utilisateur consulte la page du serveur web Nginx hébergé sur **VM WEB**



2. Cela génère un log sur **VM WEB** dans access.log

```
192.168.1.90 - - [24/May/2021:14:21:23 +0200] "GET /TEST-
RAPPORT HTTP/1.1" 404 199 "-
" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/90.0.4430.212 Safari/537.36"*
```

3. L'agent Filebeat détecte un nouveau log, il va donc le prendre et l'envoyer à **docker site1-logstash** (VM BUFFER).

```
INFO        [publisher_pipeline_output]        pipeline/output.go:15
1        Connection to backoff(async(tcp://BUFFER:5044)) established
```

4. **Docker site1-logstash** analyse ce message en le filtrant, transformant afin de le rendre plus facilement utilisable par la suite. (ex : Associe chaque partie du log à des variables) => On le rend plus facilement exploitable et enrichi. Extrait :

5. Puis envoie le message traité/enrichi dans un topic **Docker Kafka.** Cette zone tampon stock le log.


```
{"response":"404","message":"192.168.1.90 - - [24/May/2021:14:21:23 +0200] \"GET /TEST-
RAPPORT HTTP/1.1\" 404 199 \"-
\" \"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
e/90.0.4430.212 Safari/537.36\"","auth":"-","tags":["site1-
web","beats_input_codec_plain_applied"],"clientip":"192.168.1.90","@timestamp":"2021-05-
24T12:21:27.939Z","verb":"GET","ident":"-","host":{"name":"ELK-
PRODUCTOR","architecture":"x86_64","ip":["192.168.1.27","fe80::a00:27ff:fecb:d1db"],"os":{"
kernel":"4.19.0-16-
amd64","version":"10 (buster)","platform":"debian","type":"linux","codename":"buster","fami
ly":"debian","name":"Debian GNU/Linux"},"containerized":false,"mac":["08:00:27:cb:d1:db"],"
id":"ad589957bc9945738bdaf239751c176f","hostname":"ELK-PRODUCTOR"},"request":"/TEST-
RAPPORT","log":{"file":{"path":"/var/log/nginx/access.log"},"offset":1031},"ecs":{"version"
:"1.8.0"},"timestamp":"24/May/2021:14:21:23 +0200","input":{"type":"log"},"agent":{"hostnam
```

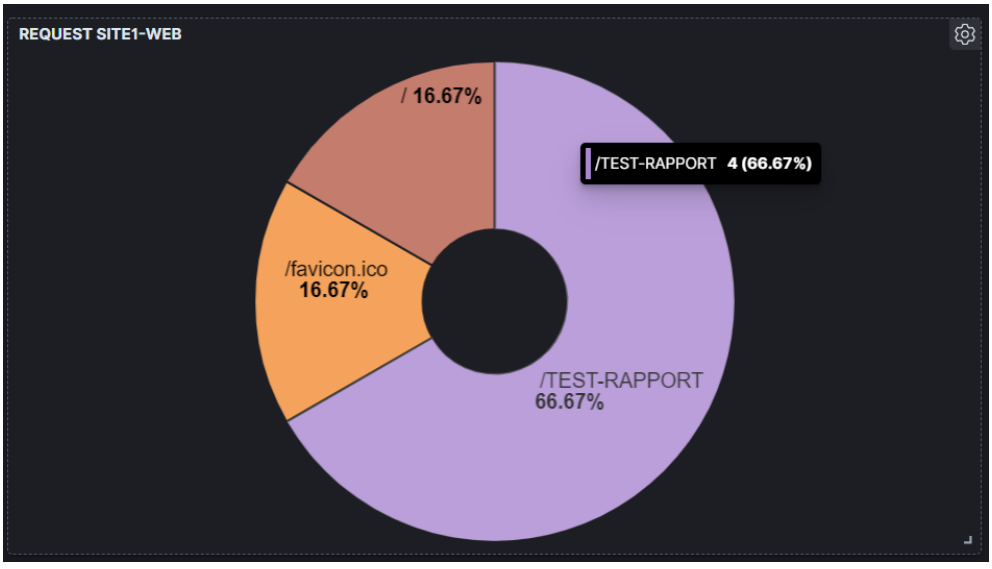
```
e":"ELK-PRODUCTOR","version":"7.12.0","ephemeral_id":"706bc331-5c86-47ab-94ab-92370abd69e9","type":"filebeat","id":"64795570-d0ad-48d2-9eb4-49b7ad087ff8","name":"ELK-PRODUCTOR"},"httpversion":"1.1","@version":"1"}
```


6. Docker logstash situé dans la VM ELK, étant en écoute des topics Kafka va aller chercher ce log traité/enrichi.

7. Docker logstash aura ensuite la possibilité de refaire une phase de traitement de log (exemple ajouté un ou plusieurs champs, plugin ect) mais dans ce cas actuel il n'aura pas à le filtrer. Il va l'envoie directement à Docker Elasticsearch en indiquant l'index a crée en fonction du tagg contenu dans le log (Ce tagg a été ajouté par le service Filebeat auparavant).

8. Docker Kibana qui héberge un outil de virtualisation web pourra ensuite aller chercher cet index pour par exemple l'afficher sur un graphique. Extrait :

May 24, 2021 @ 14:21:27.939 @timestamp: May 24, 2021 @ 14:21:27.939 @version: 1 @version.keyword: 1 agent.ephemeral_id: 706bc331-5c86-47ab-94
92370abd69e9 agent.ephemeral_id.keyword: 706bc331-5c86-47ab-94ab-92370abd69e9 agent.hostname: ELK-PRODUCTOR
agent.hostname.keyword: ELK-PRODUCTOR agent.id: 64795570-d0ad-48d2-9eb4-49b7ad087ff8 agent.id.keyword: 64795570-d0
49b7ad087ff8 agent.name: ELK-PRODUCTOR agent.name.keyword: ELK-PRODUCTOR agent.type: filebeat agent.type.keyword:
agent.version: 7.12.0 agent.version.keyword: 7.12.0 auth: - auth.keyword: - clientip: 192.168.1.90



III. EXPLICATION PARTIE : DATA SOURCE

Comme évoqué pour cette partie trois sources de logs :

- **VM WEB :** hébergeant un serveur web NGINX + Service FILEBEAT
- **ROUTER :** émulé via GNS3
- **VM WINDOWS :** Système exploitation Windows 10 Pro + Service WINLOGBEAT

a. PRODUCTEUR : VM WEB (Nginx)

A SAVOIR : FILEBEAT est un Agent de transfert léger conçu pour les logs. Dans ce projet je propose un type d'utilisation mais il faut savoir qu'il existe une multitude d'utilisation possible avec différents types d'input, d'output ainsi que de module pour faciliter son utilisation. Voir Documentation officielle pour plus de détails.

La **VM WEB** comprend un **serveur web NGINX** écoutant sur le port 80. Un agent **FILEBEAT** va récupérer tous les logs qui arrivent dans « **access*.log** ». Par exemple lorsqu'un utilisateur consulte une page du serveur Web cela génère un log qui va dedans. Voici la configuration du fichier filebeat.

Filebeat.yml :

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/nginx/access*.log
  tags: ["site1-web"]

output.logstash:
  hosts: ["BUFFER:5044"]

processors:
  - add_host_metadata: ~
  - add_cloud_metadata: ~
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~
```

Ce fichier de configuration comporte 2 parties :

- **Input :** L'endroit où il écoute dans mon cas type : **log => access*.log** + Ajout d'**un tag site1-web** à chaque log.
- **Output :** Le destinataire du log qui est **Docker site1-logstash** qui écoute sur le port 5044.

b. PRODUCTEUR : CISCO-ROUTER

A SAVOIR : Les logs CISCO ont un format Syslog c'est-à-dire que chaque log à une syntaxe tel que : « 1.date émis 2.hostname 3.info-processus 4.niveau-gravité 5.id-processus 6.corp-du-message ». Il est donc tout à fait possible par exemple de n'envoyer que les logs de niveau-gravité élevé. Dans mon cas j'envoie tous les logs. Voir documentation intéressante.

Pour des raisons pratiques le **ROUTER CISCO** est émulé sur GNS3 mais cela ne change en rien son fonctionnement dans ce LAB TEST. Cet équipement va envoyer au format SYSLOG **l'ensemble de ses syslogs** au Docker site1-logstash écoutant sur le port **5140**. Voici la configuration relative à l'envoie de log sur le router :

```
R1(config)#logging host 192.168.59.10 transport udp port 5140
```

a. PRODUCTEUR : VM-WINDOWS

A SAVOIR : WINLOGBEAT est un agent léger conçu pour le transfert des logs d'événements Windows. Comme pour FILEBEAT il existe une multitude d'utilisations possible. Dans mon cas je l'utilise pour faire remonter uniquement les évènements « échecs et succès » d'authentifications Windows. Voir Documentation officielle pour plus de détails.

La **VM WINDOWS** n'héberge aucun service particulier externe au système Windows 10 mis à part **l'agent WINLOGBEAT**.

Winlogbeat.yml :

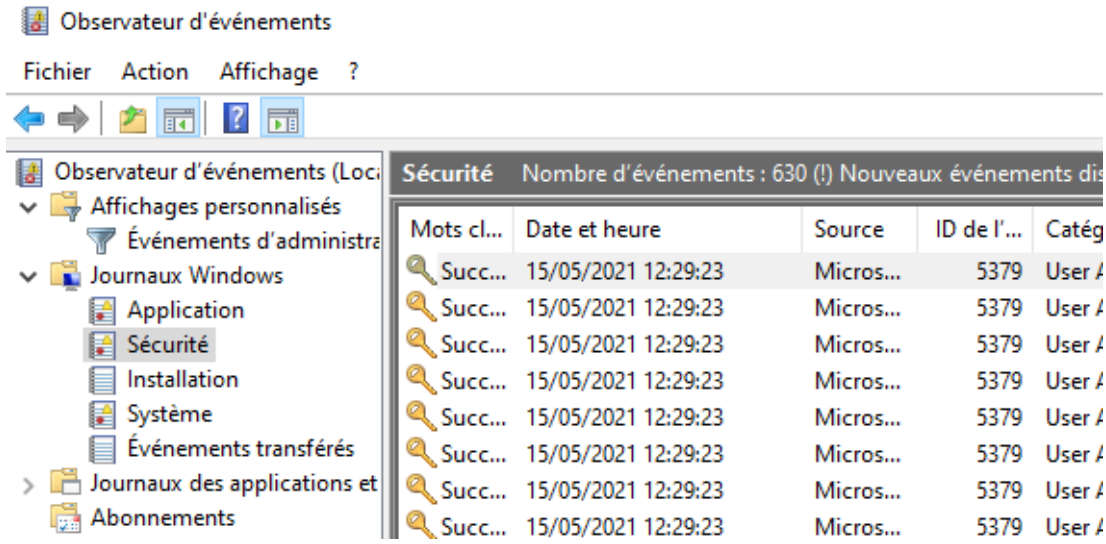
```
winlogbeat.event_logs:
  - name: Security
    #On va recuperer uniquement connexion échoué et succes (id_evenement)
    event_id: 4624, 4625
    processors:
      - add_tags:
          tags: ["site1-windows"]

output.logstash:
  hosts: ["192.168.1.24:5044"]

processors:
  - add_host_metadata:
      when.not.contains.tags: forwarded
  - add_cloud_metadata: ~
```

Ce fichier de configuration comporte 2 parties :

- **Input :** La partie d'écoute, elle va récupérer uniquement les id événements dans Sécurité suivant :
  - 4524 : Connexion réussite d'un utilisateur à l'ordinateur
  - 4525 : Echec de connexion d'un utilisateur à l'ordinateur

  + Ajout d'un **tag site1-web** à chaque événement
- **Output :** Le destinataire du log qui est **Docker site1-logstash** qui écoute sur le port 5044.

Voici l'arborescence des événements Windows pour mieux comprendre :



IV. EXPLICATION PARTIE : DATA PARSE & BUFFER

a. TRAITEMENT : SITE1-LOGSTASH

A SAVOIR : LOGSTASH est un centralisateur et transformateur de données. C'est un outil très puissant qui peut écouter et traiter une multitude de données différentes. Il peut être considéré comme un ETL (Extract, Transform, Load). Il dispose d'un nombre important de fonctionnalités tel que le module GEOIP (mapping IP public => Zone géographique, intéressant pour les cartographies Kibana). Un dernier point il dispose d'un nombre de pattern par défaut listé dans ce lien : defaut-patterns-logstash que j'ai utilisé mais il est tout à fait possible d'en ajouter de nouveaux (comme j'ai fait pour nginx). Voir Documentation officielle pour plus de détails.

Le docker Site1-logstash va recevoir les logs/events des différents DATA SOURCE (dans mon cas 3) pour ensuite procéder à une phase de traitement de ces données. Ce traitement est géré dans le pipeline « site1-logstash.conf » VOIR ANNEXE 2. PIPELINE DU SITE1-LOGSTASH

PIPELINE SITE 1 (schématisée)

```
Input :
  udp port: 5140 (syslog)
  beats port: 5044 (windows + web)


Filter En fonction du type ou tagg
:
  Grok : Parse log
  Date : Format date
  mutate : modif field (ex:add_tag)
  ...


Output En fonction du tagg:
  kafka :
     host : BUFFER:9092
     topic id: topic_name
```

Ci-dessus une capture simplifiée du « site1-logstash.conf sur site1-logstash », ce fichier comporte 3 parties :

- Input : Partie réception/écoute dans mon cas :
  - Ecoute SYSLOG : UDP port 5140 pour les remontés SYSLOG du ROUTER
  - Ecoute FILEBEAT et WINLOGBEAT : port 5044 pour le WINDOWS et le serveur WEB

- Filter : Va effectuer un traitement en fonction du type de log :
  - Si c'est un log type = site1-syslog-cisco
    - Utilisation d'un Grok cisco qui va associer à chaque partie du log une variable. (Utile par la suite pour filtrer par niveau de gravité sur Kibana)
    - Ajout de champ via nom host et date réception via « add_field ». Permet d'enrichir le log en ajouter des champs qui ne sont pas envoyé par le ROUTER
    - Modification du format de la date syslog via « date »
    - Modification du champ tag en site1-syslog-cisco via « mutate »
  - Si c'est un log tag = site1-web
    - Utilisation d'un Grok propre au log nginx qui comme pour cisco va associer à chaque partie du log une variable.
    - patterns_dir : Fait appel à un pattern qui n'existe pas de base dans logstash. C'est un fichier « nginx » que j'ai créé et qui comporte une variable associée à un regex. « NGUSERNAME [a-zA-Z\.\@\-\+_%]+ » (on indique uniquement le NGUSERNAME dans le pipeline => permet de rendre le pipeline plus lisible et moins lourde).
  - Si c'est un log tag = site1-windows
    - Pas de traitement car WINLOGBEAT s'occupe déjà de cette partie
- Output : Partie d'envoi des logs traités à KAFKA :
  - Si le log tag = site1-syslog-cisco
    - Envoie dans le topic : site1-syslog-cisco
    - Authentification unidirectionnelle via SASL (explication détaillée dans les parties suivantes).
    - Codec = format json
  - Si le log tag = site1-web
    - Envoie dans le topic : site1-web
    - Authentification unidirectionnelle via SASL (explication détaillée dans les parties suivantes).
    - Codec = format json
  - Si le log tag = site1-windows
    - Envoie dans le topic : site1-windows
    - Authentification unidirectionnelle via SASL (explication détaillée dans les parties suivantes)
    - Codec = format json

J'ai choisi de mettre en place un Logstash local au site 1 sans passer directement par le KAFKA pour plusieurs raisons que je détaillerais davantage par la suite :

- Plus de flexibilité car le logstash peut écouter une multitude de source de log contrairement au Kafka qui est plus limité.
- Moins d'impact sur le lien WAN car les logs sont déjà traités avant de sortir du site 1, c'est-à-dire qu'on ne remonte que les informations essentielles (log moins volumineux).

b. TAMPON : SITE1-KAFKA

A SAVOIR : KAFKA est un système de stockage de flux de messages. Kafka organise les messages en catégories appelées topics. Le producteur est un client qui va déposer des messages dans un ou plusieurs topics. Le consommateur est celui qui va aller récupérer les messages dans ces topics. Dans ce projet il va agir comme une zone tampon + sauvegarde 7 jours de rétention. Voir documentation intéressante.

Dans mon cas **Site1-logstash** est considéré comme un producteur pour le **docker Kafka** mais ce n'est pas réellement le cas dans cette architecture. En effet l'origine de la production de ces données, comme je l'ai expliqué dans la partie DATA SOURCE sont les 3 entités (ROUTER, VM WEB et VM WINDOWS) qui vont envoyer leurs logs au site1-logstash.

Docker Kafka est dépendant de **docker zookeeper-kafka**, en effet le zookeeper va agir comme une base de données pour kafka en stockant par exemple la configuration des sujets (nombre de partition, emplacement des répliques), les listes de contrôles d'accès, les appartenances aux cluster (pas dans mon cas car nœud kafka unique).

Liste des topics KAFKA:

```
/ # kafka-topics.sh --bootstrap-server BUFFER:9094 --list kafkacat
__consumer_offsets
site1-syslog-cisco
site1-web
site1-windows
```

Kafka server.properties (partie écoute) :

```
#PARTIE ECOUTE (SASL_PLAINTEXT : Filebeat > Kafka ; SASL_SSL : Kafka < Logstash ;
PLAINTEXT : broker > broker (en plaintext car pas de cluster kafka))
listener.security.protocol.map=INTERNAL:SASL_PLAINTEXT,EXTERNAL:SASL_SSL, INTE
RBROKER:PLAINTEXT
listeners=INTERNAL://0.0.0.0:9092, EXTERNAL://0.0.0.0:9093, INTERBROKER://0.0.
0.0:9094
advertised.listeners=INTERNAL://BUFFER:9092,EXTERNAL://BUFFER:9093, INTERBROKE
R://BUFFER:9094
inter.broker.listener.name= INTERBROKER
#PARTIE SASL ( Auth par login )
...Vu dans une autre partie
#PARTIE SSL ( Keystore + Trustore )
...Vu dans une autre partie
#PARTIE CONSOMMATOR LIMIT Mb/s (1Mb MAX)
...Vu dans une autre partie
```

3 protocoles d'écoutes dans mon kafka:

- INTERNAL : **SASL_PLAINTEXT (communication entre site1-logstash et site1-kafka)**
- EXTERNAL : **SASL_SSL (communication entre site1-kafka et logstash) => WAN**
- INTERBROKER : **PLAINTEXT (communication inter-broker mais dans mon cas pas de cluster)**

V. EXPLICATION PARTIE : CLOUD

a. DATA PROCESSING : LOGSTASH

Docker Logstash situé dans la partie cloud (dans ce projet en local sur la machine) comme indiqué précédemment va agir comme un **consommateur**. Etant abonnés aux topics de site1-kafka il va récupérer les différents messages. Cette phase d'input et d'Ouput est géré dans le pipeline « site1-logstash.conf » VOIR ANNEXE 3. PIPELINE DU LOGSTASH (CLOUD)

PIPELINE SITE 1 (schématisée)

```
Input :
  kafka :
    topic : site1-web
    topic : site1-syslog-cisco
    topic : site1-windows


Filter :
    #Site1-logstash parse déjà
les logs

Output En fonction du tagg:
  Elasticsearch :
     host : ["ELK:9200"]
     index: topic_name-%{+YYYY.MM.dd}
```

Ci-dessus une capture simplifiée du « site1-logstash.conf sur logstash », ce fichier comporte 2 parties :

- Input : Partie réception/écoute dans mon cas :
  - Ecoute les **3 topics** KAFKA :
    - Site1-web, Site1-syslog-cisco, Site1-windows
    - **Authentification unidirectionnelle** via **SASL** (explication détaillée dans les parties suivantes)
    - **Chiffrement SSL + Authentification bidirectionnelle par certificat** (explication détaillée dans les parties suivantes)
    - **Codec = format json**

- Filter : Dans mon cas pas de partie filter car site1-logstash s'occupe de cette partie :
  - Mais toujours la possibilité d'enrichir les logs exemple :
    - Ajouter des champs (**add_tagg**)
    - Ajouter un module (**GEOIP**) pour attribuer une Localisation Géographique en fonction de l'IP public origine du log (utile pour la cartographie KIBANA)

- Output : Partie d'envoi des logs à Elasticsearch :
  - Si le log **tag = site1-syslog-cisco**
    - Envoie sur **Elasticsearch** port 9200
      - Dans **Index** "site1-syslog-cisco-date"
  - Si le log **tag = site1-web**
    - Envoie sur **Elasticsearch** port 9200
      - Dans **Index** "site1-web-date"
  - Si le log **tag = site1-windows**
    - Envoie sur **Elasticsearch** port 9200
      - Dans **Index** "site1-windows-date"

b. DATA STORAGE : ELASTICSEARCH

A SAVOIR : ELASTICSEARCH est un moteur de recherche et d'analyse RESTful distribué, conçu pour répondre à une multitude de cas d'utilisation. C'est l'outil d'indexation **NOSQL** le plus performant pour des recherches (très utiles pour les logs). **Voici quelques termes importants :**

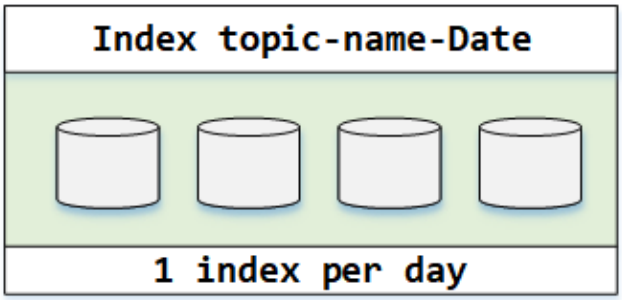
Cluster : ensemble de nœuds Elasticsearch (+ nœuds = + performances + redondance)

Index : une instance de base de données ;

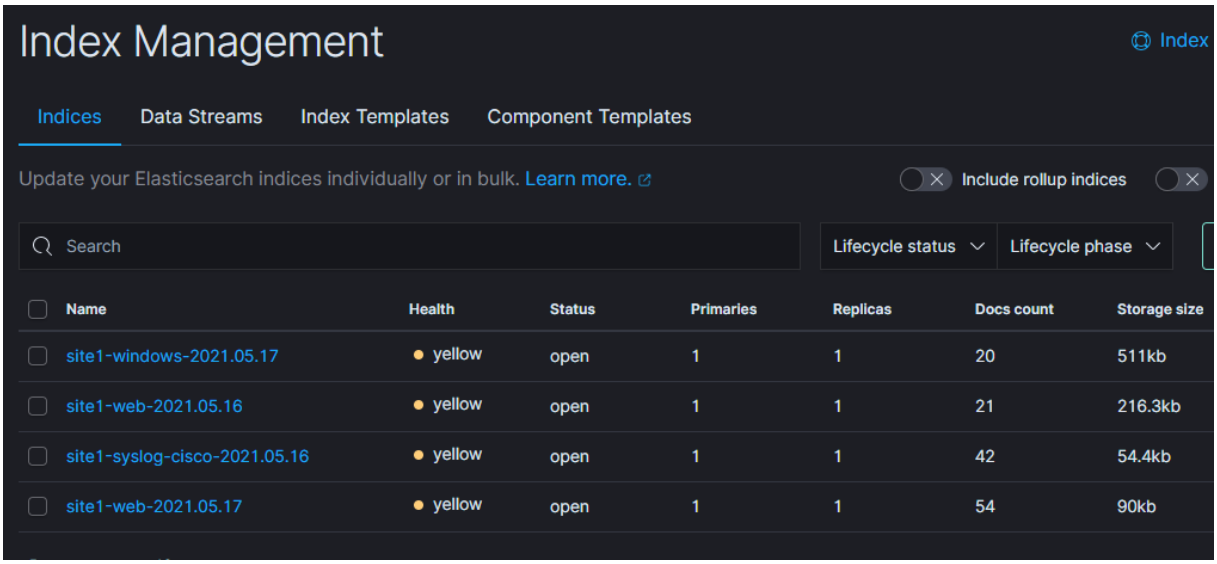
Shards : découpage logique d'un index (un ou plusieurs shards, utile pour la répartition shards sur plusieurs nœuds)

Réplicas : Réplicas de shards d'un index (redondance et performances)

Voir Documentation officielle pour plus de détails.



Le **docker Elasticsearch** comme indiqué précédemment va recevoir les logs/events par le biais de logstash. Dans ce projet Elasticsearch n'est pas en cluster, c'est-à-dire que je n'ai qu'un **seul nœud**. Le déploiement d'un cluster pourrait être un axe d'évolution. Mais cela demande une étude préalable, **un cluster de 3 nœuds serait-il judicieux dans le cadre de mon entreprise ?**



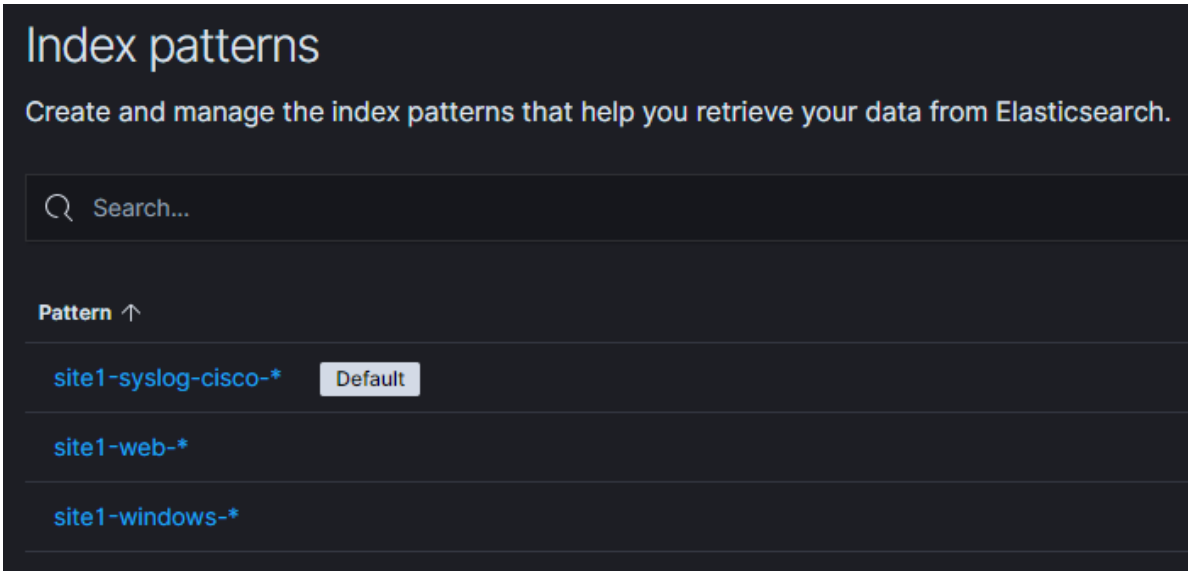
Comme vous pouvez le constater sur la capture ci-dessus qui représente les index Elasticsearch disponible sur Kibana. **Un index** va entrer crée pour chaque nom de topic Kafka **par jour**. On peut y voir la taille occupée, le nombre de réplicas (dans mon cas 1 car pas de cluster), le nombre de documents… Vous verrez par la suite comment Kibana les utilises afin de visualiser ses données.

c. DATA VISUALIZATION : KIBANA

A SAVOIR : KIBANA est une interface qui permet de visualiser les données Elasticsearch et de naviguer dans la Suite Elastic. C'est aussi un assistant d'installation metrics/logs/SIEM.

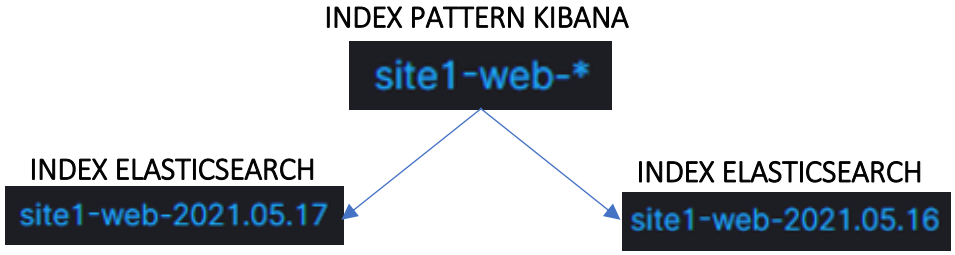
Voir Documentation officielle pour plus de détails.

- Onglet Management Kibana

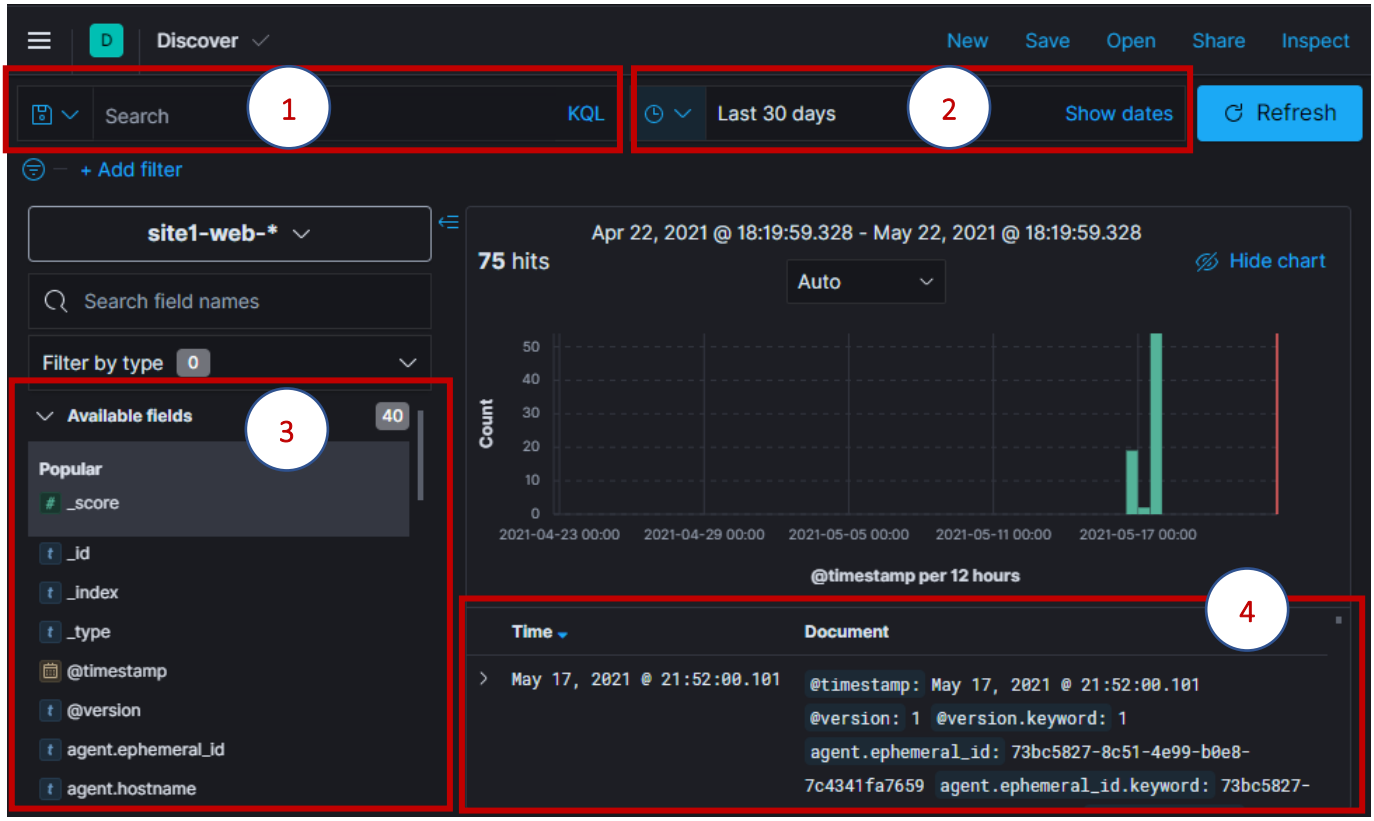


Le **docker Kibana** va aller questionner les index Elasticsearch afin de pouvoir les traiter sous forme d'Index patterns. Comme vous pouvez le voir sur la capture ci-dessus qui représente ses index patterns (qui ont dû être créée manuellement).

Chacun de ces trois index patterns vont regrouper les index Elastisearch exemple pour site1-web :



- Onglet Discover Kibana



L'onglet Discover de kibana permet de voir l'ensemble des logs contenus dans un Index pattern. Ci-dessus un exemple avec site1-web-* :

- **1 : Partie Filter** permettant par exemple de filtrer tous les logs contenants un champ spécifique
- **2 : Partie Affichage par date**, possibilité d'afficher les logs de ces 30 derniers jours par exemple
- **3 : Partie affichant les champs disponibles** pour cet index pattern
- **4 : Partie listant le détail** de l'ensemble des logs matchant avec la partie 1 et 2 pour un index Pattern.

- Onglet Dashboard Kibana

L'onglet Dashboard de Kibana permet d'apporter une visualisation plus intuitive des logs dont voici un exemple pour chacun des trois patterns :

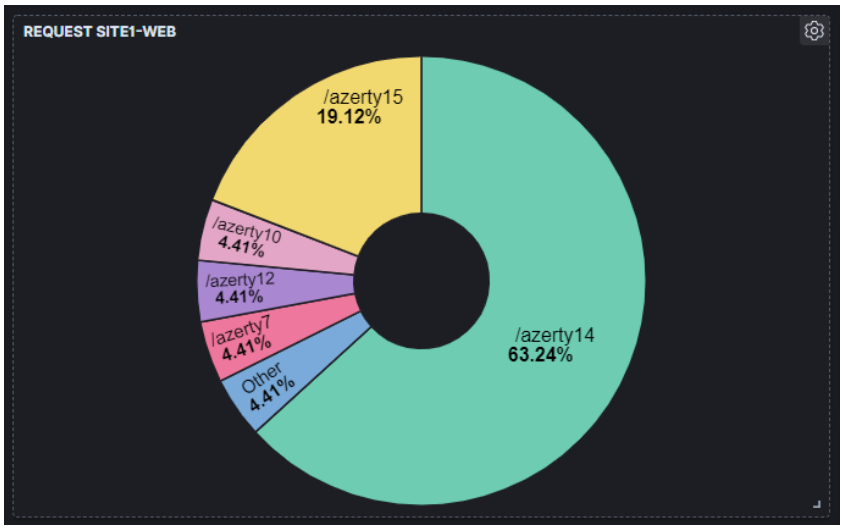


Diagramme des requests site1-web

SITE1-WINDOWS ECHEC LOGON

IP machine	Nombre connexion échoué à cet ordinateur
192.168.1.160	4
fe80::55c1:8c5b:6e19:6cbb	4

SITE1-WINDOWS SUCCES LOGON

IP machine	Nombre connexion succès à cet ordinateur
192.168.1.160	5
fe80::55c1:8c5b:6e19:6cbb	5

Compteurs échec/succès connexion ordinateur pour chaque host Windows

TOP VALUES MESSAGE SITE1-SYSLOG-CISCO

Top values of message.keyword	Top values of host.key...	Count of records
<186>30: *Mar 1 00:17:22.343: %SYS-2-LOGMSG: Message from 0(): "bye"	192.168.59.11	1
<186>31: *Mar 1 00:20:15.219: %SYS-2-LOGMSG: Message from 0(): "bye"	192.168.59.11	1
<186>32: *Mar 1 00:20:16.379: %SYS-2-LOGMSG: Message from 0(): "bye"	192.168.59.11	1
<186>33: *Mar 1 00:20:43.595: %SYS-2-LOGMSG: Message from 0(): "bye"	192.168.59.11	1
<186>34: *Mar 1 00:22:06.363: %SYS-2-LOGMSG: Message from 0(): "bye2"	192.168.59.11	1
Other	192.168.59.11	37

Top des valeurs syslog cisco

VI. LA REPONSE AUX CONTRAINTES ET BESOINS

a. POINT SUR LES CONTRAINTES

>> VOIR Annexe 4 SCHÉMA COMMUNICATION DE L'INFRASTRUCTURE <<

Rappel des contraintes du projet :

- **La sécurité de notre réseau** (ex : Ne pas faire fuiter les logs/events)
- **Impacter le moins possible** réseau de production (ex : Disponibilité)
- **L'évolutivité de l'infrastructure** (ex : Si intégration de plusieurs sites)

• SÉCURITÉ : ÉCHANGES ET ACCÈS

A SAVOIR : J'ai eu l'occasion de faire une documentation détaillés sur la mise en place de SSL et SASL dans ce projet que vous retrouverez sur mon GITHUB au lien suivant : SSL-SASL.pdf

La garantit de la sécurité sur le réseau est géré par deux technologies : SSL et SASL.

Voici les différents types de communication de ce projet :

- **Entre site1-logstash et site1-kafka : PLAINTEXT_SASL**
  - PLAINTEXT : les messages sont en claire (car on est sur notre réseau local)
  - SASL : Authentification par login unidirectionnel vers site1-kafka pour pourvoir déposer dans les topics.
- **Entre site1-kafka et logstash : SSL_SASL**
  - SSL : les messages sont chiffrés (car on passe sur un réseau WAN) + authentification bidirectionnelle par certificat (les deux entités valident l'identité de chacun grâce au certificats SSL signé par rootCA)
  - SASL : Authentification par login unidirectionnel vers site1-kafka pour pourvoir consommer dans les topics.
- **Entre les data source et site1-logstash : PLAINTEXT**
  - PLAINTEXT : les messages sont en claire (car on est sur notre réseau local)
- **Entre Logstash, Elasticsearch et Kibana : PLAINTEXT**
  - PLAINTEXT : les messages sont en claire (car on est sur une même VM)

Kafka server.properties (partie SASL et SSL) :

```
#PARTIE SASL ( Auth par login )
sasl.enabled.mechanisms=PLAIN
listener.name.internal.plain.sasl.jaas.config=org.apache.kafka.common.security
.plain.PlainLoginModule required username="admin" password="Azerty77" user_pro
ductor="Azerty77";
```

```
listener.name.external.plain.sasl.jaas.config=org.apache.kafka.common.security
.plain.PlainLoginModule required username="admin" password="Azerty77" user_con
sommator="Azerty77";
#PARTIE SSL ( Keystore + Trustore )
ssl.keystore.location=/opt/kafka/SSL/keystore.jks
ssl.keystore.password=Azerty77
ssl.key.password=Azerty77
ssl.truststore.location=/opt/kafka/SSL/trustchain.jks
ssl.truststore.password=Azerty77
ssl.client.auth=required
```

- ## IMPACT : RESSOURCES ET COMMUNICATION

Un autre point étais d'impacter le moins possible le réseau de production. Pour ce faire j'ai dû faire certains choix :

1. **J'ai choisi de ne pas chiffrer (PLAINTEXT) les messages sur le réseau LAN site 1** car je pars du principe que ce réseau Local est déjà bien sécurisé et que nous maitrisons ce qu'il s'y passe. De plus le chiffrement SSL demande davantage de **ressources CPU**, ce qui est assez contraignant pour des machines en productions qui sont déjà fortement sollicités.
2. **J'ai aussi fait le choix de limiter le débit de consommation à 1 Mb/s sur le réseau WAN pour les consommateurs** de site1-kakfa. Logstash se trouve donc limiter à cette valeur qu'il ne peut pas dépasser pour consommer dans les topics kafka. Cela permet d'avoir une maitrise du flux en cas de grosse rafale de logs par exemple.
3. **Un autre point est l'externalisation de la stack ELK vers le cloud (pour ce LAB TEST reste en local)**, afin d'éviter un engorgement des logs sur un seul site.

Kafka server.properties (partie LIMIT CONSO) :

```
#PARTIE CONSOMMATOR LIMIT Mb/s (1Mb MAX)
quota.consumer.default=1048576
```

- ## UNE SOLUTION ÉVOLUTIVE

La possibilité de faire évoluer l'infrastructure est un point primordial surtout si elle tend à être intégré dans mon entreprise comprenant sept établissements :

1. **L'externalisation de la stack ELK vers le cloud** m'a semblait intéressante car elle permet d'avoir **une centralisation des logs ne dépendent pas du réseau de mon entreprise**. L'aspect **convergence est moins un problème** surtout sur les offres cloud actuel(facilité de disposer d'une bande passante de 100Mega et d'une VM évolutive à faible coûts)
2. Ce projet étant **entièrement virtualisé (+ Dockerisé)** il sera très **simple d'ajouter de nouvelles entités les autres sites**. (Possibilité d'exporter rapidement et simplement les dockers et machines virtuelles)

3. De plus dû au fait que les différentes entités se combinent très bien ensemble il est tout à fait envisageable de faire évoluer presque tous les aspects de ce projet par exemple :
    a. **Déployer un cluster** (Kubernetes ?) Elasticsearch, Kafka, Logstash… (via docker-compose)
    b. **Intégrer de nouveaux DATA SOURCES** (nécessite uniquement la modification des pipelines des différents logstash + Kibana)
    c. **Faire du monitoring d'éléments externe** au réseau LAN l'entreprise grâce aux multitudes d'input possible sur logstash (Flux twitter du compte CFA GRAND EST, outil de paye SAGE, outil d'incident GLPI…)

b. POINT SUR LES BESOINS

La réponse aux besoins en quelques lignes :

- **Centraliser** un ensemble de logs/évènements de tous les établissements
    - ✓ **Externalisation de la stack ELK** sur un VM CLOUD
    - ✓ Ce projet comporte un seul site mais cette infrastructure peut **supporter l'ensemble des sept établissements** sans aucun problème (export, import des entités + ajout d'un pipeline sur Logstash pour chaque site + Création Index Pattern sur Kibana).

- **Être informé en temps réel** de ce qu'il se passe sur le réseau et les serveurs
    - ✓ **Les DATA SOURCE envoie** leurs logs dès qu'un évènement se produit
    - ✓ **En cas de surcharge du lien/Rafale** de log le site1-kafka est là pour apporter une **zone tampon** (fait aussi office de s**auvegarde** 7 jours de rétention + permet au logstash de prendre **le temps de traiter** les logs)

- **Recherche et Analyse** de l'ensemble des données relatives aux entités de mon entreprise
    - ✓ L'intervention de **Logstash dans l'analyse** et **d'Elasticsearch dans la recherche** optimise cette partie car nous avons deux outils complémentaires.
    - ✓ Quant à **Kibana** il permet d'apporter **une visualisation des toutes ces données** (aspect ergonomique).

- **Apporter une couche de sécurité** avec la gestion d'alertes de sécurité
    - ✓ **Utilisation technologie SSL et SASL** pour la communication des différentes entités
    - ✓ **La gestion d'alertes de sécurité** est un point abordé mais à développer (dans ce projet VM-WINDOWS remonte les échecs et succès de connexion à l'ordinateur)

VII. CONCLUSION

Ce projet de deuxième année de Master m'a permis de **monter en compétence** dans le domaine du **monitoring de logs** mais aussi de peaufiner mes connaissances en **dockerisation**. C'est un domaine que j'apprécie et que je compte approfondir davantage, afin d'exploiter le **potentiel grandissent** de ces technologies.

J'ai eu l'occasion de **prendre conscience** de certaines choses surtout au niveau de **l'analyse des prérequis** avant de déployer un tel système. En effet il faut déjà avoir une très bonne connaissance de son parc informatique et de savoir **quel type de logs monitorer** pour ne pas être submergé par des logs "inutiles".

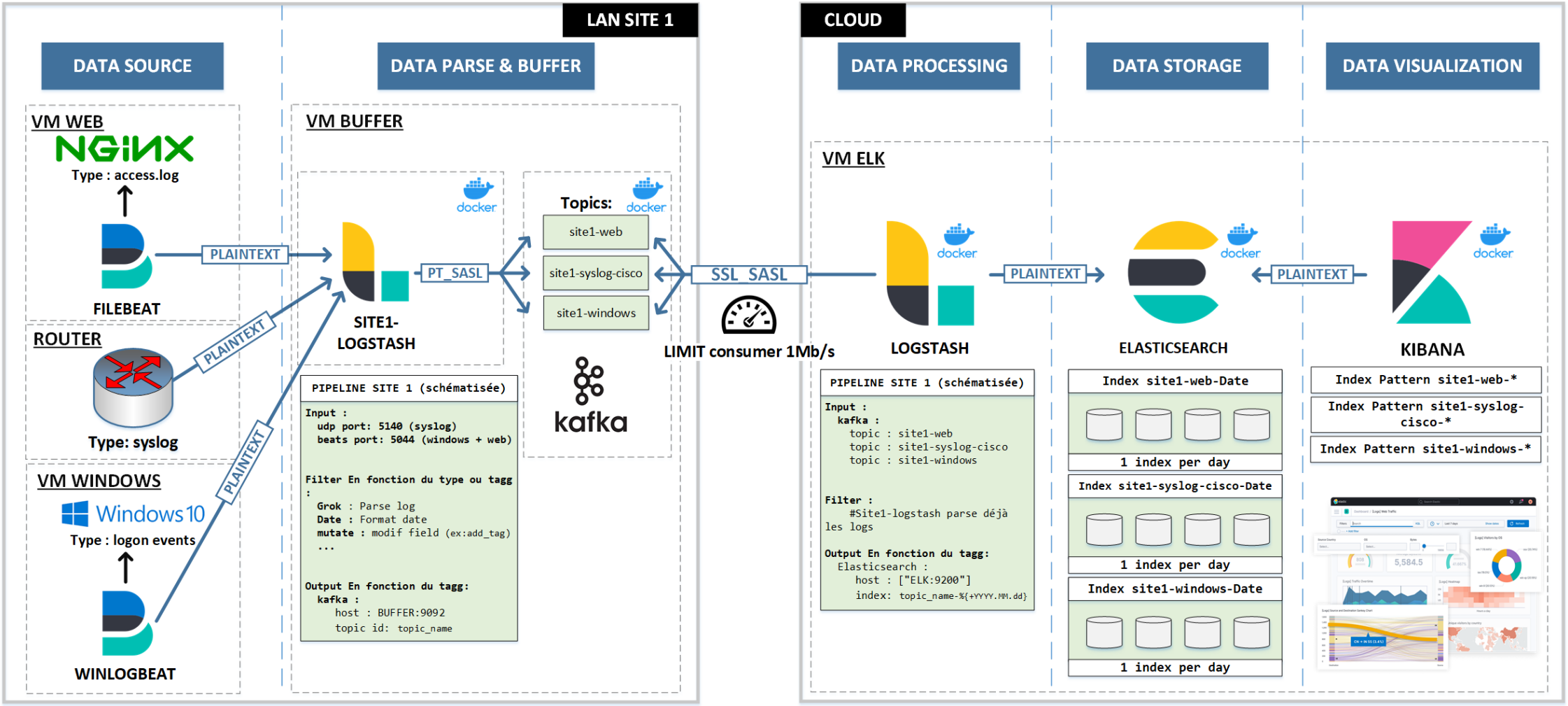
En me basant sur les **besoins et contraintes de mon entreprise**, j'ai pu apporter une **plus-value** à ce projet. Par exemple avec la mise en place d'une autorité de certification pour le **chiffrement SSL** mais aussi avec la **limitation du consommateur** Logstash sur le buffer Kafka.

L'un des objectifs de ce **LAB TEST** était d'avoir une **solution évolutive** afin de pouvoir **s'adapter à mon entreprise**. Je pense que ce point a été respecté, en effet pour ajouter d'autres sites il suffira de dupliquer (après un sizing) la partie LAN SITE 1 sachant que quelques modifications devront être faites (exemple : adapter pipeline de sitex-logstash, les noms topics et ajouter un pipeline sitex sur logstash).

Ma grande difficulté a été de **choisir un type d'infrastructure** qui me semblait **cohérente** car il faut savoir qu'il existe une **multitude de possibilités** sur ces technologies. Un point que j'aurais aimé aborder est la **gestion des alertes** de sécurité au niveau de Kibana.

.

ANNEXE 1. SCHÉMA DÉTAILLÉ DE L'INFRASTRUCTURE



ANNEXE 2. PIPELINE DE SITE1-LOGSTASH

```
input {
#ECOUTE SYSLOG
  udp {
    port => 5140
    type => "site1-syslog-cisco"
  }
#ECOUTE FILBEAT
  beats {
    port => 5044
  }
}

filter {
#MATCH SI TYPE = syslog-cisco
  if [type] == "site1-syslog-cisco" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOS
T:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: %{GREE
DYDATA: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" ]
    }
    mutate {
      add_tag => ["site1-syslog-cisco"]
    }
  }
#MATCH SI TAG = site1-web
  if "site1-web" in [tags] {
    grok {
      patterns_dir => "/usr/share/logstash/config/pattern/"
      match => { "message" => "%{IPORHOST:clientip} %{NGUSER:ident} %{NGUSER:a
uth} \[%{HTTPDATE:timestamp}\] \"%{WORD:verb} %{URIPATHPARAM:request} HTTP/%{N
UMBER:httpversion}\" %{NUMBER:response}" }
    }
  }
}

output {
  if "site1-syslog-cisco" in [tags] {
    kafka {
      #Envoie a kafka sur le port 9092
      bootstrap_servers => "BUFFER:9092"
      #Topic d'envoie
      topic_id => "site1-syslog-cisco"
```

```
      #Type de communication
      security_protocol => "SASL_PLAINTEXT"
      #Login pour auth SASL
      sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginMo
dule required username='productor' password='Azerty77';"
      sasl_mechanism => "PLAIN"
      codec => "json"
    }
  }

  if "site1-web" in [tags] {
    kafka {
      #Envoie a kafka sur le port 9092
      bootstrap_servers => "BUFFER:9092"
      #Topic d'envoie
      topic_id => "site1-web"
      #Type de communication
      security_protocol => "SASL_PLAINTEXT"
      #Login pour auth SASL
      sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginMo
dule required username='productor' password='Azerty77';"
      sasl_mechanism => "PLAIN"
      codec => "json"
    }
  }

  if "site1-windows" in [tags] {
    kafka {
      #Envoie a kafka sur le port 9092
      bootstrap_servers => "BUFFER:9092"
      #Topic d'envoie
      topic_id => "site1-windows"
      #Type de communication
      security_protocol => "SASL_PLAINTEXT"
      #Login pour auth SASL
      sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginMo
dule required username='productor' password='Azerty77';"
      sasl_mechanism => "PLAIN"
      codec => "json"
    }
  }
}
```

ANNEXE 3. PIPELINE DE LOGSTASH (CLOUD)

```
input {
    kafka {
      #Ecoute le kafka avec communication en SASL_SSL
      bootstrap_servers => "BUFFER:9093"
      #Topic d'ecoute
      topics => ["site1-syslog-cisco", "site1-web", "site1-windows"]
      #Type de communication
      security_protocol => "SASL_SSL"
      #Login pour auth SASL
      sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginMo
dule required username='consommator' password='Azerty77';"
      sasl_mechanism => "PLAIN"
      #Partie SSL (keystore + Truststore)
      ssl_keystore_location => "/usr/share/logstash/SSL/keystore.jks"
      ssl_keystore_password => "Azerty77"
      ssl_truststore_location => "/usr/share/logstash/SSL/trustchain.jks"
      ssl_truststore_password => "Azerty77"
      #desactive verification du nom d'hote niveau certificat SSL
      ssl_endpoint_identification_algorithm => ""
      codec => "json"
    }
}
#filter {
#Site1logstash prossède deja au filter (mais possiblité d'enrichir encore logs ex: GEOIP)
#}
output {
  if "site1-syslog-cisco" in [tags] {
    elasticsearch {
      hosts => ["ELK:9200"]
      index => "site1-syslog-cisco-%{+YYYY.MM.dd}"
    }
  }

  if "site1-web" in [tags] {
    elasticsearch {
      hosts => ["ELK:9200"]
      index => "site1-web-%{+YYYY.MM.dd}"
    }
  }

  if "site1-windows" in [tags] {
    elasticsearch {
      hosts => ["ELK:9200"]
      index => "site1-windows-%{+YYYY.MM.dd}"
    }
  }
}
```

ANNEXE 4. SCHÉMA COMMUNICATION DE L'INFRASTRUCTURE

PLAINTEXT_SASL : *no crypted & Auth unidirectionnal per login with kafka* | SSL_SASL : *crypted with Auth bidirectionnal by certificat & Auth unidirectionnal per login with kafka*