

SPECIALITE : SSIR-2D

RAPPORT DE PROJET CLOUD

Implémentation d'une architecture ELK

Réalisée par : Wissem nasri

Encadré par : mourad melliti

Année Universitaire : 2024 – 2025

Table de matière

1.	Introduction	1
1.1.	Contexte et Objectifs	1
2.	Présentation de l'Architecture ELK	1
2.1.	Description des Composants ELK (Elasticsearch, Logstash, Kibana, Metricbeat, Filebeat, Setup Certificates,fleet server,apm server,elastic agent)	1
2.2.	Avantages et Limitations de l'Architecture ELK	2
2.2.1.	Avantages :	2
2.2.2.	Limitations :	2
3.	Analyse des Besoins et Objectifs	3
3.1.	Identification des Besoins Techniques et Fonctionnels.....	3
3.2.	Spécifications Non-Fonctionnelles (Sécurité, Performances, Évolutivité)	3
3.3.	Enjeux et Contraintes du Projet.....	3
4.	Conception de l'Architecture	4
4.1.	Schéma d'Architecture ELK	4
5.	Mise en Œuvre de l'Architecture ELK.....	5
5.1.	Préparation de l'Environnement (Docker et Docker Compose)	5
5.2.	Déploiement des services	6
5.2.1.	Déploiement de Setup Certificates	7
5.2.2.	Déploiement de Elasticsearch	7
5.2.3.	Déploiement de Kibana.....	7
5.2.4.	Déploiement de Logstash, Filebeat, et Metricbeat.....	7
5.2.5.	Déploiement de Fleet Server et Elastic Agent.....	8
5.2.6.	Déploiement de APM Server	8
6.	Tests et Validation de l'Architecture	8
6.1.	Generation des certificats	8
6.1.1.	Elasticsearch (es01)	8
6.1.2.	Kibana	9
6.1.3.	Fleet Server	9
6.2.	Test des fonctionnement des différents services.....	12

Table des figure

Figure 1 : Schéma d'Architecture ELK	5
Figure 2:installation docker & docker-compose	6
Figure 3:fichier de configuration des services	6
Figure 4:déploiement des services Elk	6
Figure 5:: certificats générer par Setup certificates	10
Figure 6:verification de génération des certificats auto signée	11
Figure 7:fonctionnement de metricbeat	12
Figure 8: fonctionnement de filebeat.....	13
Figure 9:fonctionnement de logstash	15
Figure 10:fonctionnement de fleet server	15
Figure 11:ca.crt	16
Figure 12: Empreinte de confiance de l'autorité de certification Elasticsearch	16
Figure 13:integration de serveur fleet	17
Figure 14: Politique du serveur Fleet	18
Figure 15:fonctionnement d'une application python	18
Figure 16:teste de fonctionnement de l'application et son traitement via serveur APM.	19

1. Introduction

1.1. Contexte et Objectifs

Les systèmes modernes génèrent de grandes quantités de logs, contenant des informations précieuses pour le suivi, la sécurité, et l'optimisation des performances. Gérer ces données efficacement est un défi majeur pour les administrateurs et les équipes de sécurité. Dans ce contexte, l'architecture ELK (Elasticsearch, Logstash, Kibana) est une solution largement utilisée pour centraliser, traiter et visualiser ces données de manière rapide et accessible.

L'objectif de ce rapport est de documenter la mise en œuvre d'une architecture ELK, en partant de l'analyse des besoins jusqu'à l'évaluation des résultats, afin de faciliter l'analyse des logs et la détection d'anomalies.

2. Présentation de l'Architecture ELK

2.1. Description des Composants ELK (Elasticsearch, Logstash, Kibana, Metricbeat, Filebeat, Setup Certificates, fleet server, apm server, elastic agent)

L'architecture ELK repose sur trois composants principaux : Elasticsearch, Logstash, et Kibana. Elle est souvent étendue avec d'autres outils comme Metricbeat , Filebeat ,setup certificates,fleet server ,apm server et elastic agent pour la collecte de données et l'analyse.

- **Elasticsearch** : Un moteur de recherche et d'analyse distribué utilisé pour stocker et indexer les données de logs. Il permet de rechercher rapidement des informations spécifiques dans de grandes quantités de données.
- **Logstash** : Un pipeline de traitement de données qui collecte, filtre et envoie les logs vers Elasticsearch. Il est hautement configurable et peut traiter les données issues de plusieurs sources.
- **Kibana** : Un outil de visualisation qui permet d'afficher les données stockées dans Elasticsearch sous forme de graphiques, tableaux de bord, et alertes. Kibana rend les données plus accessibles pour une analyse en temps réel.
- **Metricbeat** : Un outil léger qui collecte les métriques des systèmes et services (CPU, mémoire, réseau, etc.) et les envoie à Elasticsearch pour surveillance et analyse.

- **Filebeat** : Un collecteur léger qui envoie les logs des fichiers directement vers Elasticsearch ou Logstash, facilitant l'ingestion des données de journaux sans surcharge du système.
- **Setup Certificates** : La configuration des certificats SSL/TLS pour sécuriser les échanges de données entre les composants de l'architecture ELK, garantissant ainsi la confidentialité et l'intégrité des informations transmises.
- **Elastic Agent** : Une solution unifiée pour remplacer les Beats individuels (Metricbeat, Filebeat, etc.), capable de collecter des logs, des métriques et d'autres données.
- **Fleet Server** : Un serveur centralisé qui simplifie la gestion des Elastic Agents dans une infrastructure distribuée.
- **APM Server** : Spécifiquement conçu pour l'Application Performance Monitoring (APM), il collecte des données sur la performance des applications, telles que la latence, les erreurs, et les transactions, afin de faciliter leur optimisation.

2.2. Avantages et Limitations de l'Architecture ELK

2.2.1. Avantages :

- **Centralisation** : Facilite la collecte et la gestion des logs de différentes sources en un seul endroit.
- **Analyse en Temps Réel** : Avec Kibana, les utilisateurs peuvent surveiller et analyser les données en temps réel.
- **Scalabilité** : Elasticsearch est conçu pour gérer des volumes importants de données et peut être étendu pour répondre aux besoins croissants.

2.2.2. Limitations :

- **Consommation de Ressources** : Les composants ELK peuvent être gourmands en mémoire et CPU, surtout avec de gros volumes de données.
- **Configuration Complexe** : La configuration et le déploiement d'ELK peuvent nécessiter des compétences techniques et du temps, surtout pour des fonctionnalités avancées comme la sécurité.
- **Sécurisation** : Nécessite des configurations supplémentaires, comme les certificats SSL/TLS, pour assurer la sécurité des données en transit.

3. Analyse des Besoins et Objectifs

3.1. Identification des Besoins Techniques et Fonctionnels

Dans cette partie, nous identifions les besoins techniques et fonctionnels que l'architecture ELK doit satisfaire pour réussir. Ces besoins incluent :

- **Collecte de Logs** : Le système doit pouvoir collecter des logs de différents serveurs, applications et dispositifs réseau.
- **Analyse des Données** : Une capacité d'analyse avancée pour extraire des informations pertinentes à partir des logs collectés.
- **Visualisation** : Les utilisateurs doivent pouvoir visualiser les données sous forme de tableaux de bord interactifs et graphiques dans Kibana.
- **Alertes et Notifications** : L'architecture doit être capable de générer des alertes en fonction d'événements ou d'anomalies détectées dans les données.

3.2. Spécifications Non-Fonctionnelles (Sécurité, Performances, Évolutivité)

Les spécifications non-fonctionnelles de l'architecture ELK comprennent des exigences telles que :

- **Sécurité** : Le système doit assurer la protection des données, en particulier via des protocoles sécurisés comme SSL/TLS pour le chiffrement des communications.
- **Performances** : La capacité de traiter et d'analyser un grand volume de logs en temps réel tout en maintenant des temps de réponse optimaux.
- **Évolutivité** : L'architecture doit être capable de s'adapter à une augmentation du volume de données collectées et analysées, et de s'étendre en fonction des besoins futurs.

3.3. Enjeux et Contraintes du Projet

Plusieurs enjeux et contraintes doivent être pris en compte pour assurer la réussite du projet, notamment :

- **Volume de Données** : La gestion efficace d'un grand nombre de logs en temps réel, sans nuire aux performances du système.
- **Sécurité** : Garantir l'intégrité des données collectées et le respect des meilleures pratiques de sécurité.
- **Scalabilité** : S'assurer que l'architecture puisse évoluer pour s'adapter à la croissance future des données ou du nombre d'utilisateurs.

4. Conception de l'Architecture

4.1. Schéma d'Architecture ELK

L'architecture **ELK** a été déployée sur une machine unique fonctionnant sous **Red Hat 9.3**, en utilisant **Docker** et **Docker Compose** pour orchestrer les différents services. Les composants suivants ont été installés et configurés sur cette machine :

- **Elasticsearch** : Assure l'indexation, le stockage et la recherche des données.
- **Kibana** : Permet la visualisation et l'analyse des données indexées par Elasticsearch.
- **Logstash** : Collecte, transforme et envoie les logs vers Elasticsearch pour une gestion centralisée.
- **Filebeat** : Collecte les logs à partir des systèmes cibles et les transmet à Logstash ou directement à Elasticsearch.
- **Metricbeat** : Collecte les métriques système (CPU, mémoire, réseau, etc.) et les envoie à Elasticsearch ou Logstash.
- **Fleet Server** : Sert de point centralisé pour gérer et superviser les **Elastic Agents** dans l'infrastructure.
- **Elastic Agent** : Unifie la collecte des logs, des métriques et d'autres données, simplifiant ainsi la gestion des différents beats.
- **APM Server** : Collecte des données sur les performances des applications, notamment la latence, les erreurs, et les transactions, afin d'offrir une surveillance approfondie des applications critiques.
- **Setup Certificates** : Sécurise les communications entre tous les composants grâce à des certificats SSL/TLS, garantissant des échanges chiffrés et fiables.

Le schéma d'architecture illustre comment ces composants interagissent :

1. **Filebeat** et **Metricbeat** collectent les logs et métriques des systèmes distants.
2. Ces données sont ensuite envoyées à **Logstash** ou directement à **Elasticsearch**, où elles sont traitées et indexées.
3. **Fleet Server** centralise la gestion des **Elastic Agents**, simplifiant le déploiement et la configuration de ces derniers.
4. **APM Server** fournit des informations sur les performances des applications critiques.
5. **Kibana** offre une interface utilisateur pour visualiser et analyser ces données.

6. Les communications entre tous les services sont protégées par le **Setup Certificates**, garantissant un chiffrement de bout en bout.

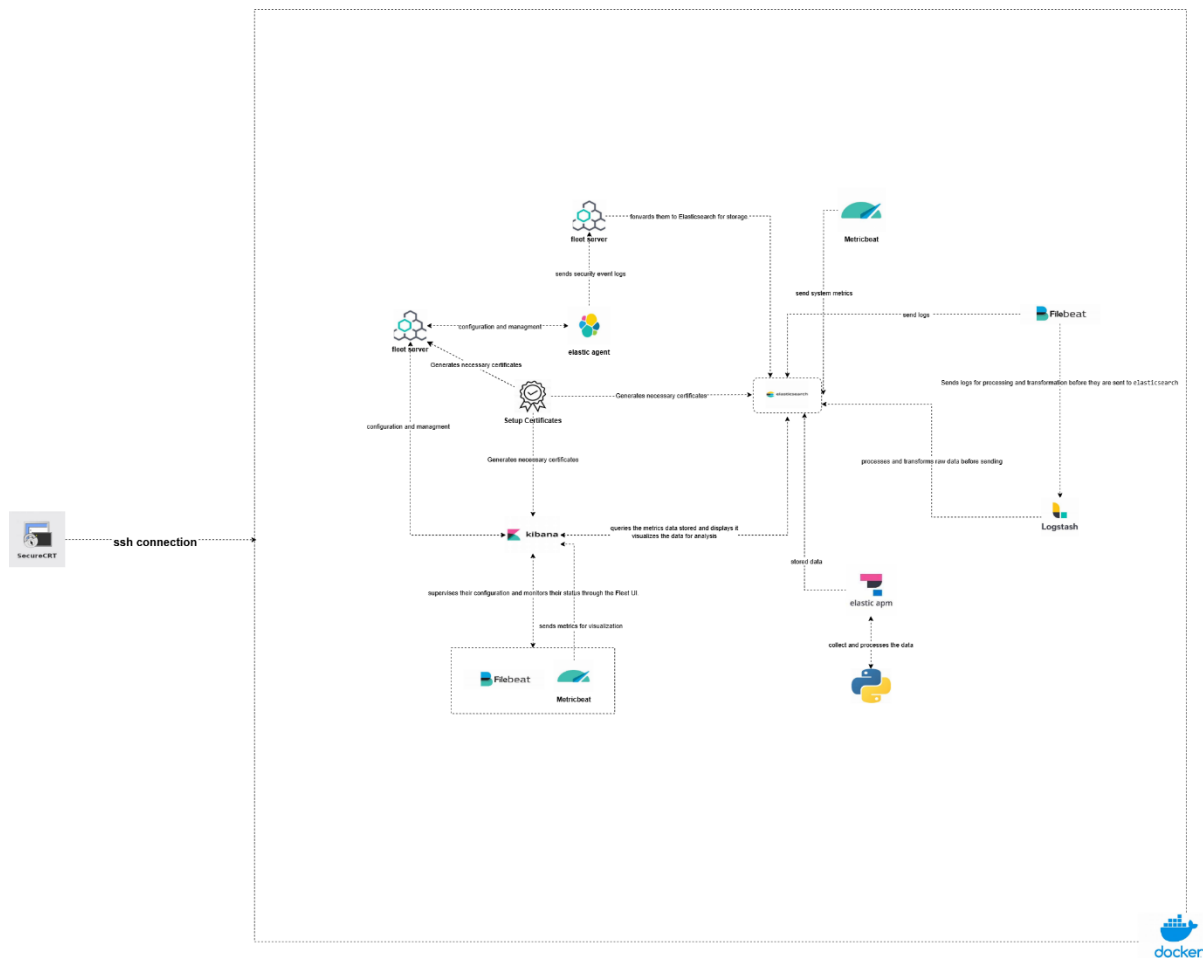


Figure 1 : Schéma d'Architecture ELK

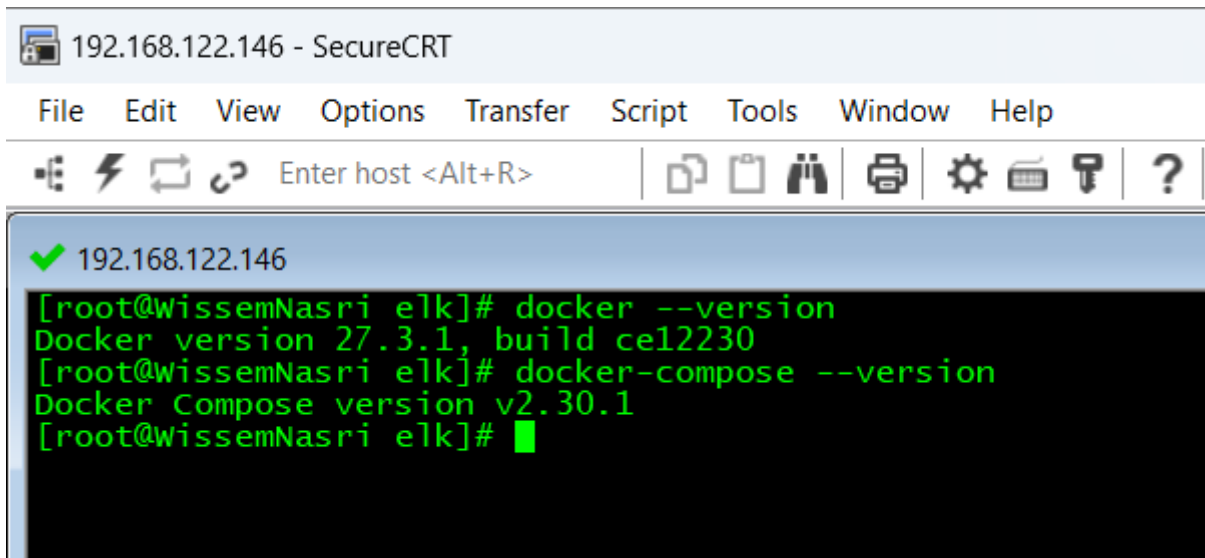
5. Mise en Œuvre de l'Architecture ELK

5.1. Préparation de l'Environnement (Docker et Docker Compose)

La mise en œuvre de l'architecture ELK repose sur une machine virtuelle RedHat 9.3 où Docker et Docker Compose sont utilisés pour déployer les différents services de l'architecture. Voici les étapes principales de la préparation de l'environnement :

- **Installation de Docker** : Docker a été installé sur la machine virtuelle pour permettre l'exécution des conteneurs des différents services de l'architecture ELK (Elasticsearch, Logstash, Kibana, Filebeat, Metricbeat).
- **Installation de Docker Compose** : Docker Compose est utilisé pour simplifier l'orchestration des services et leur interconnexion. Il permet de définir tous les

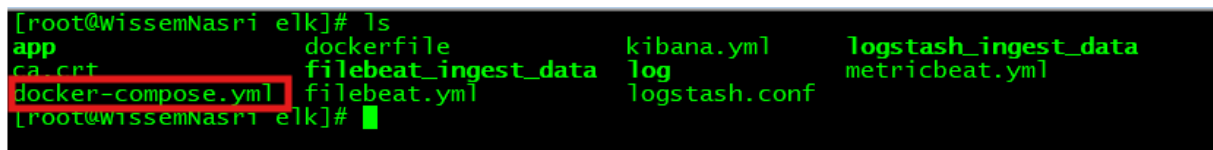
services nécessaires dans un seul fichier de configuration (docker-compose.yml), facilitant ainsi leur gestion et déploiement.



```
192.168.122.146 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
192.168.122.146
[root@WissemNasri elk]# docker --version
Docker version 27.3.1, build ce12230
[root@WissemNasri elk]# docker-compose --version
Docker Compose version v2.30.1
[root@WissemNasri elk]#
```

Figure 2: installation docker & docker-compose

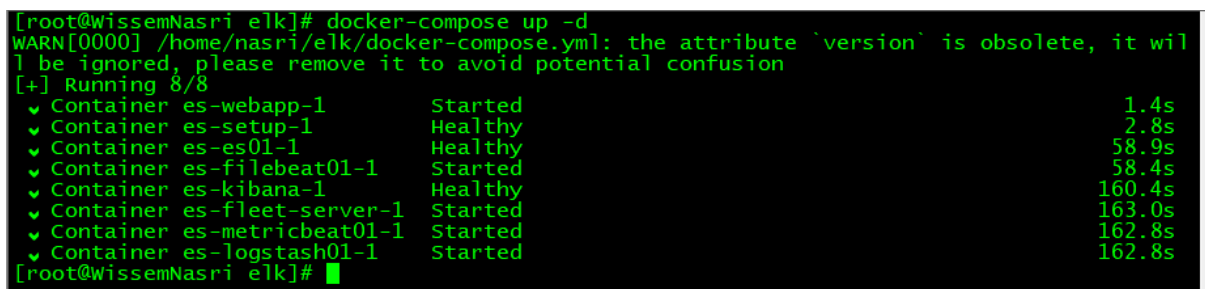
Un fichier docker-compose.yml a été préparé pour définir et orchestrer l'ensemble des services (setup certificates, Elasticsearch, Kibana, Logstash, metricbeat, filebeat)



```
[root@WissemNasri elk]# ls
app                dockerfile          kibana.yml          logstash_ingest_data
ca.crt             filebeat_ingest_data log                  metricbeat.yml
docker-compose.yml filebeat.yml         logstash.conf
```

Figure 3: fichier de configuration des services

5.2. Déploiement des services



```
[root@WissemNasri elk]# docker-compose up -d
WARN[0000] /home/nasri/elk/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 8/8
✔ Container es-webapp-1           Started           1.4s
✔ Container es-setup-1           Healthy          2.8s
✔ Container es-es01-1           Healthy          58.9s
✔ Container es-filebeat01-1     Started          58.4s
✔ Container es-kibana-1         Healthy          160.4s
✔ Container es-fleet-server-1   Started          163.0s
✔ Container es-metricbeat01-1   Started          162.8s
✔ Container es-logstash01-1     Started          162.8s
[root@WissemNasri elk]#
```

Figure 4: déploiement des services Elk

5.2.1. Déploiement de Setup Certificates

Le service Setup Certificates a été déployé pour générer et fournir des certificats nécessaires à la sécurisation des communications entre les différents services de l'architecture ELK. Ces certificats permettent à Elasticsearch, Kibana, Logstash, Metricbeat, Filebeat, Elastic Agent, Fleet Server, et APM Server de communiquer de manière sécurisée. Grâce à ces certificats, toutes les communications entre ces composants sont protégées par des canaux chiffrés, assurant ainsi la confidentialité et l'intégrité des données échangées au sein de l'architecture.

5.2.2. Déploiement de Elasticsearch

Grâce aux certificats fournis par le service Setup Certificates, Elasticsearch a été déployé et configuré pour accepter des connexions sécurisées. Le service a été configuré pour indexer et stocker les données provenant des autres composants, y compris les logs, les métriques, et les traces d'applications collectées par Elastic Agent et APM Server. Cette configuration permet à Elasticsearch de centraliser et d'indexer toutes les informations critiques en toute sécurité, tout en permettant une recherche et une analyse performantes des données.

5.2.3. Déploiement de Kibana

Kibana a été configuré pour se connecter à Elasticsearch de manière sécurisée, en utilisant les certificats fournis par le service Setup Certificates. Grâce à cette connexion sécurisée, Kibana permet aux utilisateurs de visualiser les données collectées et indexées par Elasticsearch dans des dashboards interactifs. Il offre également des fonctionnalités avancées de visualisation des logs, des métriques système et des traces d'application collectées par Filebeat, Metricbeat, et Elastic Agent.

5.2.4. Déploiement de Logstash, Filebeat, et Metricbeat

Logstash, Filebeat, et Metricbeat ont été déployés pour collecter, transformer et envoyer les données vers Elasticsearch. Tous ces services sont configurés pour utiliser les certificats générés par le service Setup Certificates pour garantir une communication sécurisée.

Logstash collecte et transforme les logs provenant de diverses sources avant de les envoyer à Elasticsearch.

Filebeat est responsable de la collecte des logs des systèmes distants et de leur envoi vers Elasticsearch ou Logstash.

Metricbeat collecte les métriques systèmes (CPU, mémoire, réseau, etc.) et les envoie à Elasticsearch ou à Logstash pour un traitement supplémentaire.

Ces services assurent la collecte et le traitement des données provenant des systèmes, des applications et des infrastructures, avant leur stockage et leur analyse dans Elasticsearch.

5.2.5. Déploiement de Fleet Server et Elastic Agent

Fleet Server a été déployé comme un composant central de gestion des Elastic Agents dans l'architecture. Il permet d'enregistrer, de configurer et de surveiller tous les Elastic Agents à travers un tableau de bord centralisé dans Kibana. Le Fleet Server se connecte à Elasticsearch pour stocker les informations sur les agents et gérer leur configuration à l'aide de Kibana.

Les Elastic Agents ont été déployés sur les hôtes cibles pour collecter les logs, les métriques, et les traces des applications. Ces agents sont enregistrés et gérés via Fleet Server. Une fois installés, les agents collectent et envoient les données vers Elasticsearch en utilisant des communications sécurisées grâce aux certificats SSL/TLS fournis.

5.2.6. Déploiement de APM Server

Le APM Server a été déployé pour collecter les traces et les données de performance des applications. Ces données sont ensuite envoyées vers Elasticsearch pour y être indexées et analysées. Comme les autres services, APM Server utilise les certificats générés par le service Setup Certificates pour sécuriser les communications avec Elasticsearch et Kibana.

Ainsi, tous les services de l'architecture ELK, y compris Elasticsearch, Kibana, Logstash, Filebeat, Metricbeat, Elastic Agent, Fleet Server, et APM Server, ont été déployés et configurés pour fonctionner de manière sécurisée et efficace, garantissant la collecte, l'indexation, et l'analyse des données dans un environnement protégé.

6. Tests et Validation de l'Architecture

Pour garantir le bon fonctionnement et la fiabilité de l'architecture ELK déployée, une série de tests a été effectuée.

6.1. Generation des certificats

les certificats sont utilisés principalement pour trois services : **Elasticsearch (es01)**, **Kibana**, et **Fleet Server**. Ces services sont configurés pour utiliser des certificats SSL/TLS pour sécuriser les communications, tant en interne qu'avec les utilisateurs externes.

6.1.1. Elasticsearch (es01)

- **Certificats utilisés :**

- Certificat pour les connexions HTTP (xpack.security.http.ssl) et pour les connexions de transport (xpack.security.transport.ssl).
- Ces certificats sont créés lors de l'initialisation du service Elasticsearch dans la section setup. Les certificats sont générés via la commande `elasticsearch-certutil` et stockés dans le volume `certs`.
- **Emplacement des certificats dans le service :**
 - Le certificat `es01.crt` et sa clé privée `es01.key` sont utilisés pour sécuriser les connexions HTTP et de transport.
 - Le certificat de l'autorité de certification (CA) `ca.crt` est utilisé pour vérifier la validité des connexions SSL.

6.1.2. Kibana

- **Certificats utilisés :**
 - Kibana utilise des certificats SSL pour sécuriser les connexions entre Kibana et les utilisateurs (clients) ainsi qu'avec Elasticsearch.
- **Emplacement des certificats dans le service :**
 - Le certificat et la clé privée de Kibana (`kibana.crt` et `kibana.key`) sont utilisés pour les connexions HTTPS sécurisées à Kibana.
 - Le certificat CA `ca.crt` est utilisé pour vérifier les connexions SSL entre Kibana et Elasticsearch.

6.1.3. Fleet Server

- **Certificats utilisés :**
 - Le Fleet Server utilise des certificats pour sécuriser la communication avec Kibana et Elasticsearch.
- **Emplacement des certificats dans le service :**
 - Le certificat et la clé privée du Fleet Server (`fleet-server.crt` et `fleet-server.key`) sont utilisés pour les connexions sécurisées entre le Fleet Server et les autres services.
 - Le certificat de l'autorité de certification (`ca.crt`) est utilisé pour vérifier la validité des connexions SSL.
- **Création des certificats :**

Lors de l'initialisation du service setup, des certificats sont créés pour les services suivants :

- **CA (Certificate Authority)** : utilisé pour signer les autres certificats.
- **Certificats pour Elasticsearch (es01.crt, es01.key), Kibana (kibana.crt, kibana.key), et Fleet Server (fleet-server.crt, fleet-server.key)** sont générés et stockés dans le volume certs pour être utilisés par ces services.

En résumé, les certificats sont créés principalement pour **Elasticsearch, Kibana, et Fleet Server**. Ces certificats sont utilisés pour sécuriser les communications entre les services et garantir l'intégrité des données transmises. Les autres services comme **Metricbeat, Filebeat, Logstash**, utilisent ces certificats pour s'assurer de la sécurité des connexions lors de l'envoi de données vers Elasticsearch ou Kibana.

```
[root@WissemNasri _data]# pwd
/var/lib/docker/volumes/es_certs/_data
[root@WissemNasri _data]# ls
ca ca.zip certs.zip es01 fleet-server instances.yml kibana
[root@WissemNasri _data]# cd es01/
[root@WissemNasri es01]# ls
es01.crt es01.key
[root@WissemNasri es01]# cd ..
[root@WissemNasri _data]# cd fleet-server/
[root@WissemNasri fleet-server]# ls
fleet-server.crt fleet-server.key
[root@WissemNasri fleet-server]# cd ..
[root@WissemNasri _data]# cd kibana/
[root@WissemNasri kibana]# ls
kibana.crt kibana.key
[root@WissemNasri kibana]#
```

Figure 5:: certificats générés par Setup certificates

Cela justifie l'utilisation d'un certificat SSL auto-signé.

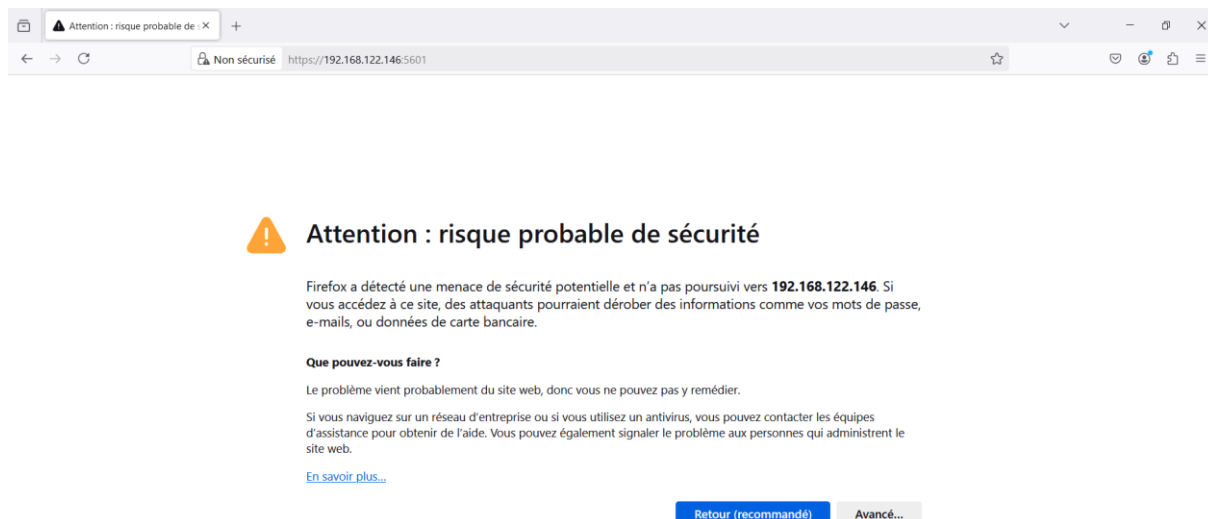
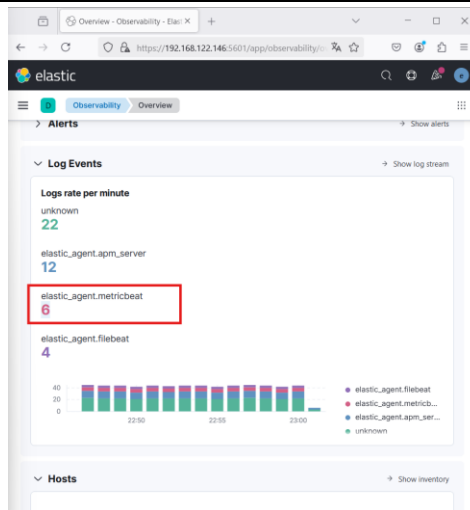


Figure 6:verification de génération des certificats auto signée

6.2. Test des fonctionnement des différents services

```
[root@WissemNasri elk]# docker ps
CONTAINER ID   IMAGE                                PORTS          COMMAND
CREATED       STATUS
NAMES
aa666fd1b7cc   docker.elastic.co/beats/metricbeat:8.8.2   Up 2 hours     "/usr/bin/tini -- /u..."
5 days ago    es-metricbeat01-1
987348de0054   docker.elastic.co/beats/elastic-agent:8.8.2   Up 2 hours     "/usr/bin/tini -- /u..."
5 days ago    0.0.0.0:8200->8200/tcp, :::8200->8200/tcp, 0.0.0.0:8220->8220/tcp, :::8220->8220/tcp
es-fleet-server-1
d4a288a1971e   docker.elastic.co/logstash/logstash:8.8.2   Up 2 hours     "/usr/local/bin/dock..."
5 days ago    5044/tcp, 9600/tcp
es-logstash01-1
277d2204156b   docker.elastic.co/kibana/kibana:8.8.2       Up 2 hours (healthy) 0.0.0.0:5601->5601/tcp, :::5601->5601/tcp
5 days ago    es-kibana-1
4caaf7c1aab    docker.elastic.co/beats/filebeat:8.8.2      Up 2 hours     "/usr/bin/tini -- /u..."
5 days ago    es-filebeat01-1
8988416e6c30   docker.elastic.co/elasticsearch/elasticsearch:8.8.2   Up 2 hours (healthy) 0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 9300/tcp
5 days ago    es-es01-1
6fca5cc501a7   es-webapp                                   Up 2 hours     "uvicorn main:app --..."
5 days ago    0.0.0.0:8000->8000/tcp, :::8000->8000/tcp
es-webapp-1
[root@WissemNasri elk]#
```



```
[root@WissemNasri elk]# docker ps
CONTAINER ID   IMAGE                                PORTS          COMMAND
CREATED       STATUS
NAMES
aa666fd1b7cc   docker.elastic.co/beats/metricbeat:8.8.2   Up 2 hours     "/usr/bin/tini -- /u..."
5 days ago    es-metricbeat01-1
987348de0054   docker.elastic.co/beats/elastic-agent:8.8.2   Up 2 hours     "/usr/bin/tini -- /u..."
5 days ago    0.0.0.0:8200->8200/tcp, :::8200->8200/tcp, 0.0.0.0:8220->8220/tcp, :::8220->8220/tcp
es-fleet-server-1
d4a288a1971e   docker.elastic.co/logstash/logstash:8.8.2   Up 2 hours     "/usr/local/bin/dock..."
5 days ago    5044/tcp, 9600/tcp
es-logstash01-1
277d2204156b   docker.elastic.co/kibana/kibana:8.8.2       Up 2 hours (healthy) 0.0.0.0:5601->5601/tcp, :::5601->5601/tcp
5 days ago    es-kibana-1
4caaf7c1aab    docker.elastic.co/beats/filebeat:8.8.2      Up 2 hours     "/usr/bin/tini -- /u..."
5 days ago    es-filebeat01-1
8988416e6c30   docker.elastic.co/elasticsearch/elasticsearch:8.8.2   Up 2 hours (healthy) 0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 9300/tcp
5 days ago    es-es01-1
6fca5cc501a7   es-webapp                                   Up 2 hours     "uvicorn main:app --..."
5 days ago    0.0.0.0:8000->8000/tcp, :::8000->8000/tcp
es-webapp-1
[root@WissemNasri elk]#
```

Hosts					
Uptime	Hostname	CPU %	Load 15	RX	TX
1h 50m	aa666fd1b7cc	27.47%	3.16	98KB/s	60KB/s
1h 56m	987348de0054	27.39%	3.17	6KB/s	10KB/s

Figure 7: fonctionnement de metricbeat

```

GNU nano 5.6.1 cron.log
Nov 24 00:01:01 localhost CROND[3441]: (root) CMD (run-parts /etc/cron.hourly)
Nov 24 00:01:01 localhost run-parts[3444]: (/etc/cron.hourly) starting 0anacron
Nov 24 00:01:01 localhost anacron[3454]: Anacron started on 2024-11-24
Nov 24 00:01:01 localhost anacron[3454]: Normal exit (0 jobs run)
Nov 24 00:01:01 localhost run-parts[3456]: (/etc/cron.hourly) finished 0anacron
Nov 24 00:01:01 localhost CROND[3440]: (root) CMDEND (run-parts /etc/cron.hourly)
Nov 28 19:21:48 localhost crond[1208]: (CRON) STARTUP (1.5.7)
Nov 28 19:21:48 localhost crond[1208]: (CRON) INFO (Syslog will be used instead of sendmail.)
Nov 28 19:21:48 localhost crond[1208]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 49% if used.)
Nov 28 19:21:48 localhost crond[1208]: (CRON) INFO (running with inotify support)
Nov 28 20:01:01 localhost CROND[19740]: (root) CMD (run-parts /etc/cron.hourly)
Nov 28 20:01:01 localhost run-parts[19743]: (/etc/cron.hourly) starting 0anacron
Nov 28 20:01:01 localhost anacron[19753]: Anacron started on 2024-11-28
Nov 28 20:01:01 localhost anacron[19753]: Will run job 'cron.daily' in 20 min.
Nov 28 20:01:01 localhost anacron[19753]: Will run job 'cron.weekly' in 40 min.
Nov 28 20:01:01 localhost anacron[19753]: Jobs will be executed sequentially
Nov 28 20:01:01 localhost run-parts[19755]: (/etc/cron.hourly) finished 0anacron
Nov 28 20:01:01 localhost CROND[19739]: (root) CMDEND (run-parts /etc/cron.hourly)
Nov 28 20:21:01 localhost anacron[19753]: Job 'cron.daily' started
Nov 28 20:21:01 localhost anacron[19753]: Job 'cron.daily' terminated
Nov 28 20:41:01 localhost anacron[19753]: Job 'cron.weekly' started
Nov 28 20:41:01 localhost anacron[19753]: Job 'cron.weekly' terminated
Nov 28 20:41:01 localhost anacron[19753]: Normal exit (2 jobs run)
Nov 28 21:01:01 localhost CROND[39608]: (root) CMD (run-parts /etc/cron.hourly)
Nov 28 21:01:01 localhost run-parts[39611]: (/etc/cron.hourly) starting 0anacron
Nov 28 21:01:01 localhost run-parts[39611]: (/etc/cron.hourly) finished 0anacron
Nov 28 21:01:01 localhost CROND[39607]: (root) CMDEND (run-parts /etc/cron.hourly)
Nov 28 21:12:33 WissemNasri crond[1326]: (CRON) STARTUP (1.5.7)
Nov 28 21:12:33 WissemNasri crond[1326]: (CRON) INFO (Syslog will be used instead of sendmail.)
Nov 28 21:12:33 WissemNasri crond[1326]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 13% if used.)
Nov 28 21:12:33 WissemNasri crond[1326]: (CRON) INFO (running with inotify support)
Nov 28 22:01:01 WissemNasri CROND[24312]: (root) CMD (run-parts /etc/cron.hourly)
Nov 28 22:01:01 WissemNasri run-parts[24315]: (/etc/cron.hourly) starting 0anacron
Nov 28 22:01:01 WissemNasri run-parts[24321]: (/etc/cron.hourly) finished 0anacron
Nov 28 22:01:01 WissemNasri CROND[24311]: (root) CMDEND (run-parts /etc/cron.hourly)
Nov 28 23:01:01 WissemNasri CROND[44698]: (root) CMD (run-parts /etc/cron.hourly)
Nov 28 23:01:01 WissemNasri run-parts[44701]: (/etc/cron.hourly) starting 0anacron
Nov 28 23:01:02 WissemNasri run-parts[44707]: (/etc/cron.hourly) finished 0anacron
Nov 28 23:01:02 WissemNasri CROND[44697]: (root) CMDEND (run-parts /etc/cron.hourly WissemNasri.SSIR.D )

[root@WissemNasri filebeat_ingest_data]#

```

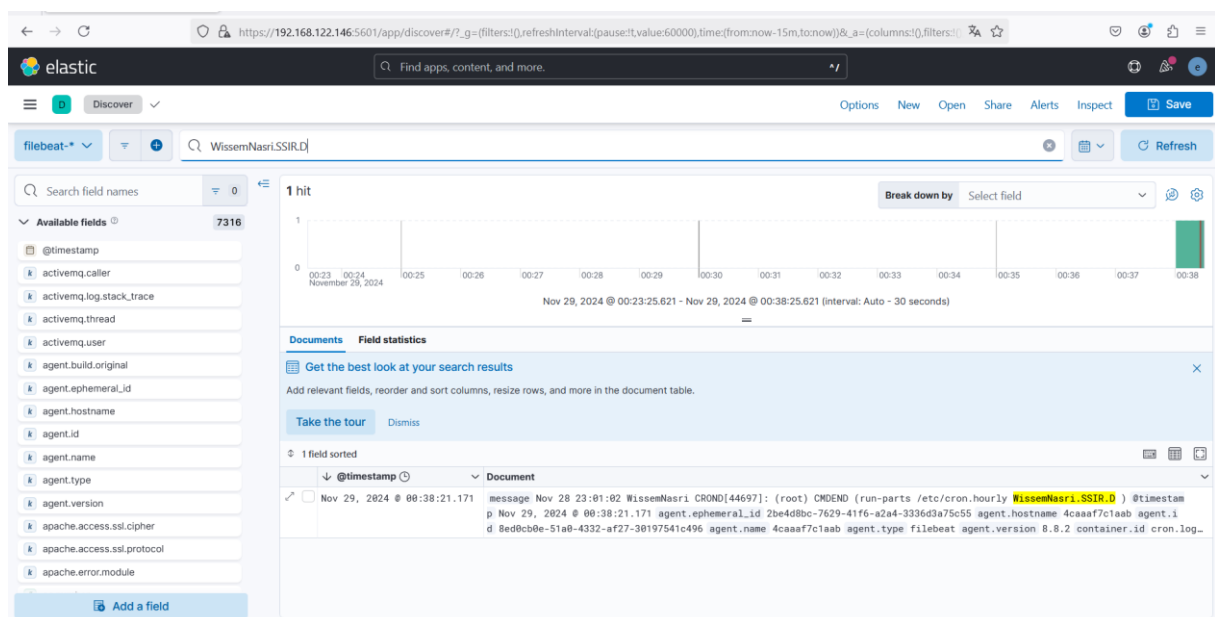
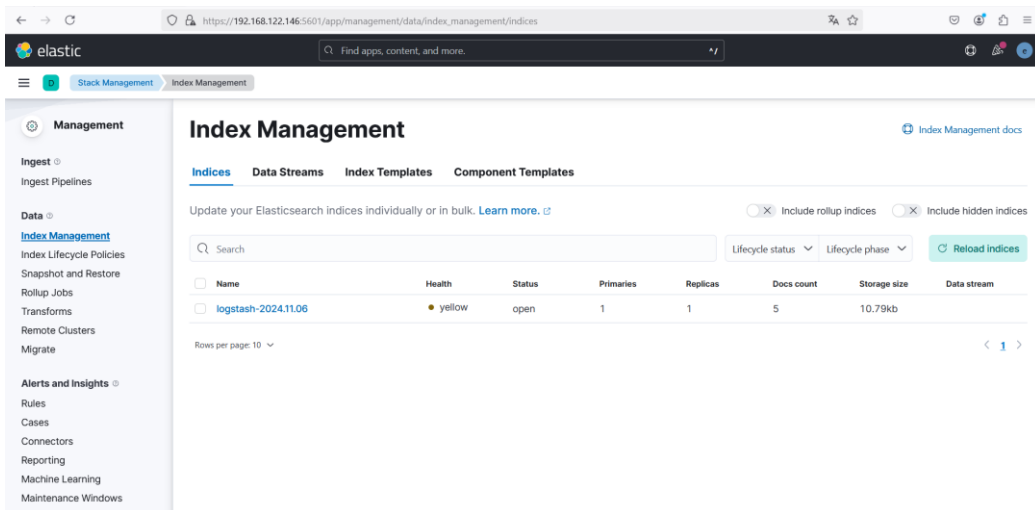


Figure 8: fonctionnement de filebeat



```

GNU nano 5.6.1 cron.log
Nov 3 15:27:24 localhost crond[1407]: (CRON) STARTUP (1.5.7)
Nov 6 23:01:01 localhost CROND[18605]: (root) CMDEND (run-parts /etc/cron.hourly project-elk)
Nov 7 00:01:02 localhost CROND[44900]: (root) CMDEND (run-parts /etc/cron.hourly wissem-nasri)

```

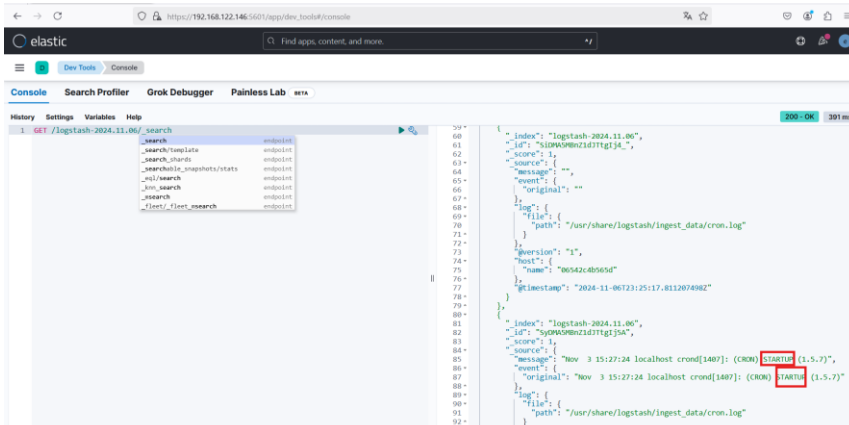
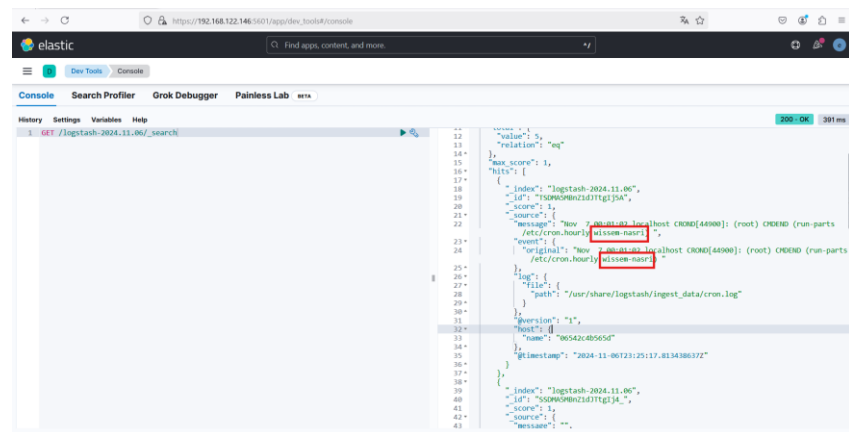


Figure 9: fonctionnement de logstash

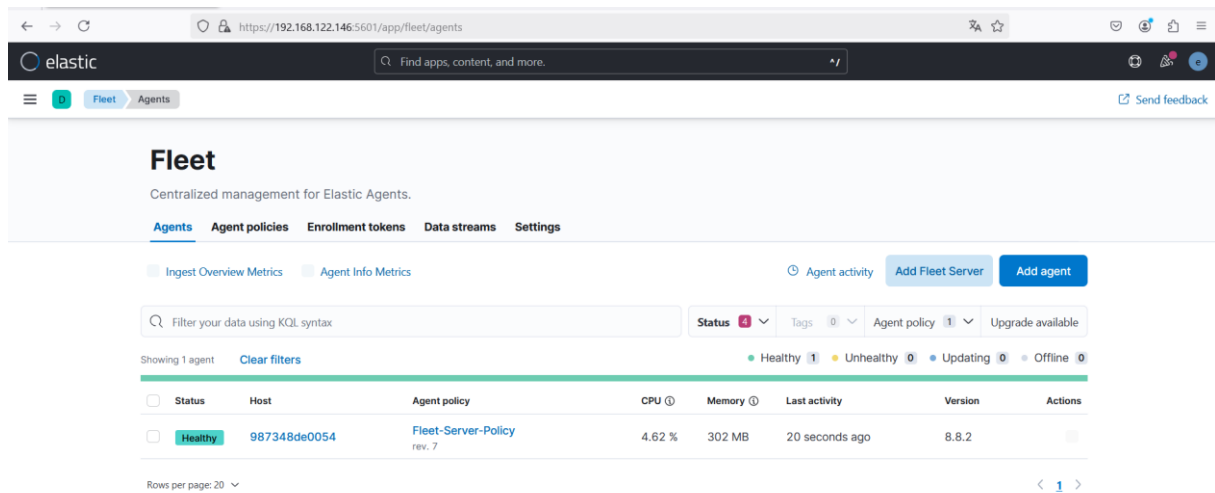


Figure 10: fonctionnement de fleet server

```

[root@wissemNasri elk]# docker ps
CONTAINER ID   IMAGE                                PORTS          COMMAND
CREATED        STATUS                                     NAMES
aa666fd1b7cc   docker.elastic.co/beats/metricbeat:8.8.2    "/usr/bin/tini -- /u..."
5 days ago    Up 3 hours                               es-metricbeat01-1
987348de0054   docker.elastic.co/beats/elastic-agent:8.8.2    "0.0.0.0:8200->8200/tcp, :::8200->8200/tcp, 0.0.0.0:8220->8220/tcp, :::8220->8220/tcp"
5 days ago    Up 3 hours                               es-fleet-server-1
d4a288a1971e   docker.elastic.co/logstash/logstash:8.8.2    "5044/tcp, 9600/tcp"
5 days ago    Up 3 hours                               es-logstash01-1
277d2204156b   docker.elastic.co/kibana/kibana:8.8.2    "0.0.0.0:5601->5601/tcp, :::5601->5601/tcp"
5 days ago    Up 3 hours (healthy)                     es-kibana-1
4caaf7c1aab    docker.elastic.co/beats/filebeat:8.8.2    "/usr/bin/tini -- /u..."
5 days ago    Up 3 hours                               es-filebeat01-1
8988416e6c30   docker.elastic.co/elasticsearch/elasticsearch:8.8.2    "0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 9300/tcp"
5 days ago    Up 3 hours (healthy)                     es-es01-1
6fca5cc501a7   es-webapp                                "0.0.0.0:8000->8000/tcp, :::8000->8000/tcp"
5 days ago    Up 3 hours                               es-webapp-1
[root@wissemNasri elk]# docker cp es-es01-1:/usr/share/elasticsearch/config/certs/ca/ca.crt /
Successfully copied 3.07kB to /home/nasri/elk/.
[root@wissemNasri elk]# ls
app      dockerfile      kibana.yml      logstash_ingest_data
ca.crt   filebeat_ingest_data  log            metricbeat.yml
docker-compose.yml  filebeat.yml    logstash.conf
[root@wissemNasri elk]# cat ca.crt
-----BEGIN CERTIFICATE-----
MIIDSTCAjGgAwIBAgIUbbZsuRwmVvH8XpYwtt7h45YS1pEwDQYJKoZIhvcNAQEL
BQAwNDEyMDA1UEAxMwRwXhc3RyYyBDZXJ0awZpY2F0ZSBub29sIEF1dG9nZW51
cmF0ZWQgQ0EwHhcNMjQxMTAzMTU0NTA3WhcNMjQxMTAzMTU0NTA3WjA0MTIwMAYD
VQDEYlFbGFZdGZlIjEENlcnRpZmljYXR1IFRvb2wgcXV0b2d1bmVvYXR1ZCBBDQTC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALafPK6MG6BV1TcAmd2FuhCA
p+EMVMQTI02qDtAkVLA7c9qaYxzG2CIDuVJ/JuLdJoHVKIdq1S/Uoo2XbVYUdmcz
EN464Nw+KfUAhzAM9XRYU7BDmv7HkhpZN9V1haw7paEyKI1kgL6DRFn8mM+qn23a
4e7qr0hbi84VqwVnDTQBFguxe8OHATFJbWYgwXFG6g/HSp+igbtEvX4LTxdUcb5z
Iw6V3y/ojcyXVs62k6TuvKpKGfMrAws/WefK86fvRtPBWG5CJgYKDP1Ax/1rbTR
oa0LmASUDTnKYI6yx8F33myHHinxXHuB+LkCcCic6HGwpfLpRdB/KLCJ7uebEwUC
AwEAAaNTMFEwHQYDVR0OBBYEFKuXsMSvaexXN+8UF8sJwz9iH6SjMB8GA1UdIwQY
MBAwAFKuXsMSvaexXN+8UF8sJwz9iH6SjMA8GA1UdEwEB/wQFMAMBAF8wDQYJKoZI
hvcNAQELBQADggEBAI/BnzWT8EE1Lrx1MjdVuqXXY1LC8Tf4tXP2BiP4fCj2FV0a
00xRCMHmbpwpMZIKphZzRjaLlmonyepN10/f4dh3dSS1AbAljTSZQSn9P0h3yP
5gmjszcBV+jCFwPW5f7LaE+LGFsB/ymeyBOPz/wF2Kg7iUYHgGYItcf/0OZ6ojk1
bo4dC7ek/r9iV9Bsexv+791x8A9mD+kJ+u1c+gcDdCrAUDV53EstCkpc9WRCvHAM
H07YT4FpxqzXkFwRgueb4UR6BRnPE4FKUhB3BQRTWA1cmZlYo82J3LQzOecErH/G
2dea3eiRbdc3UibYGLU6Wfa0U6QfEWfd3T0xG2I=
-----END CERTIFICATE-----
[root@wissemNasri elk]#

```

Figure 11:ca.crt

```

[root@wissemNasri elk]# openssl x509 -fingerprint -sha256 -noout -in ./ca.crt | awk -F"=" '{p
rint $2}' | sed 's:/://g'
B87FEA90B847B9EE79DBD0D7C93559A54B07C3968B8F27EF02A5F1730D9EF45C
[root@wissemNasri elk]#

```

Figure 12: Empreinte de confiance de l'autorité de certification Elasticsearch

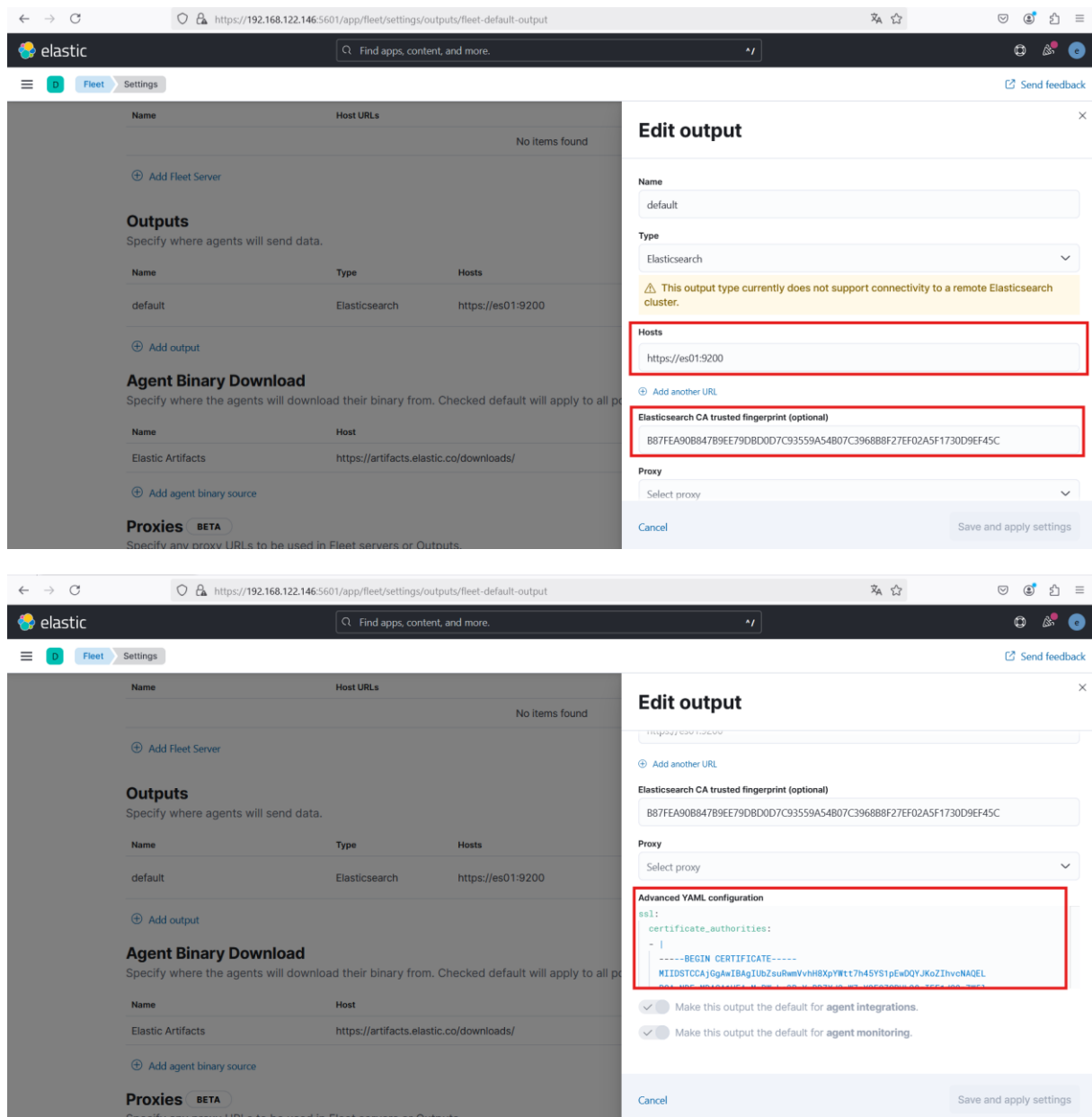


Figure 13: integration de serveur fleet

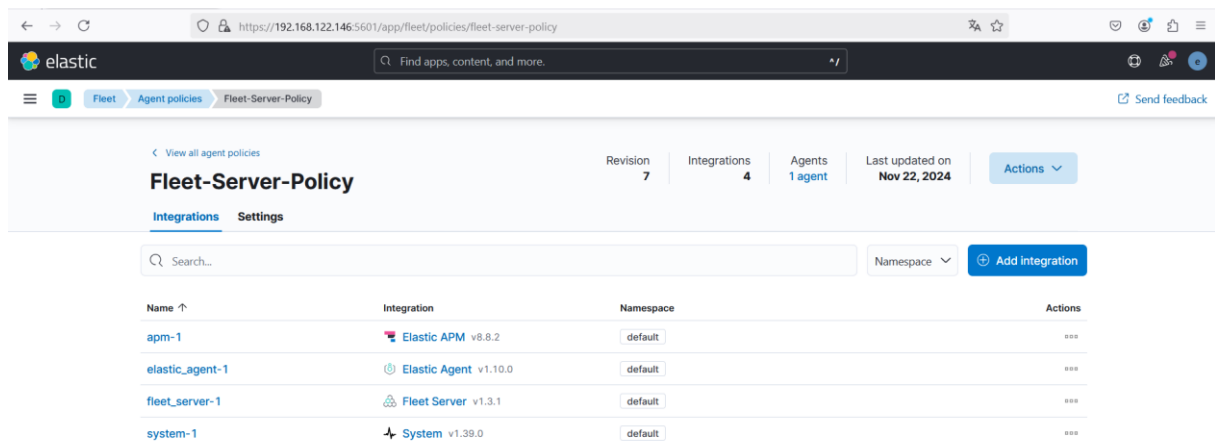


Figure 14: Politique du serveur Fleet

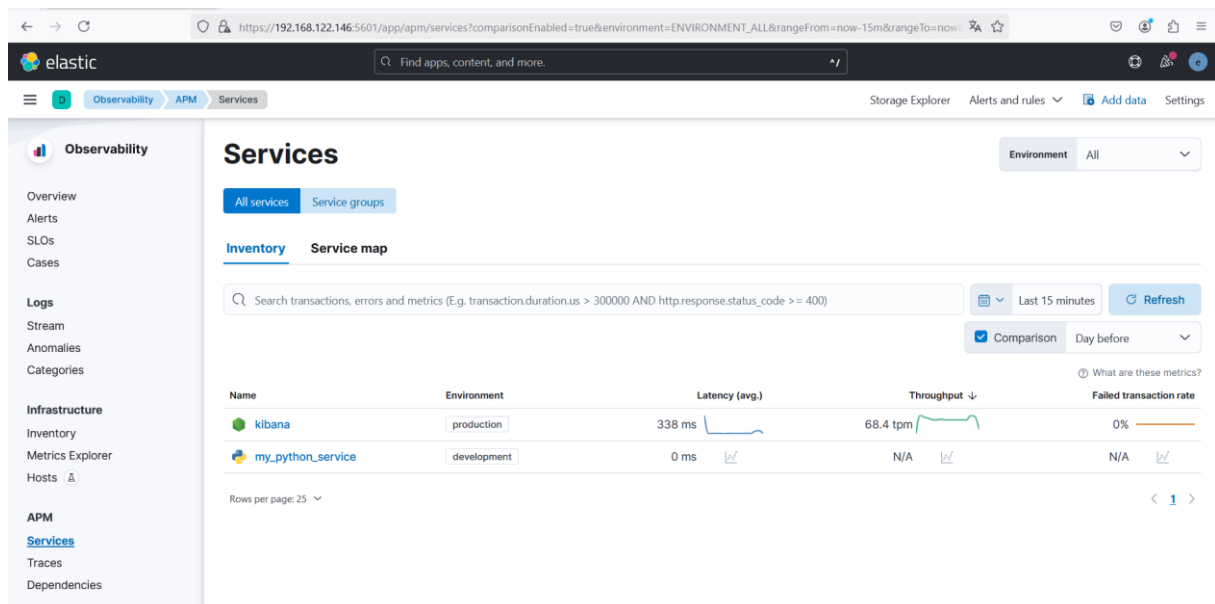


Figure 15: fonctionnement d'une application python

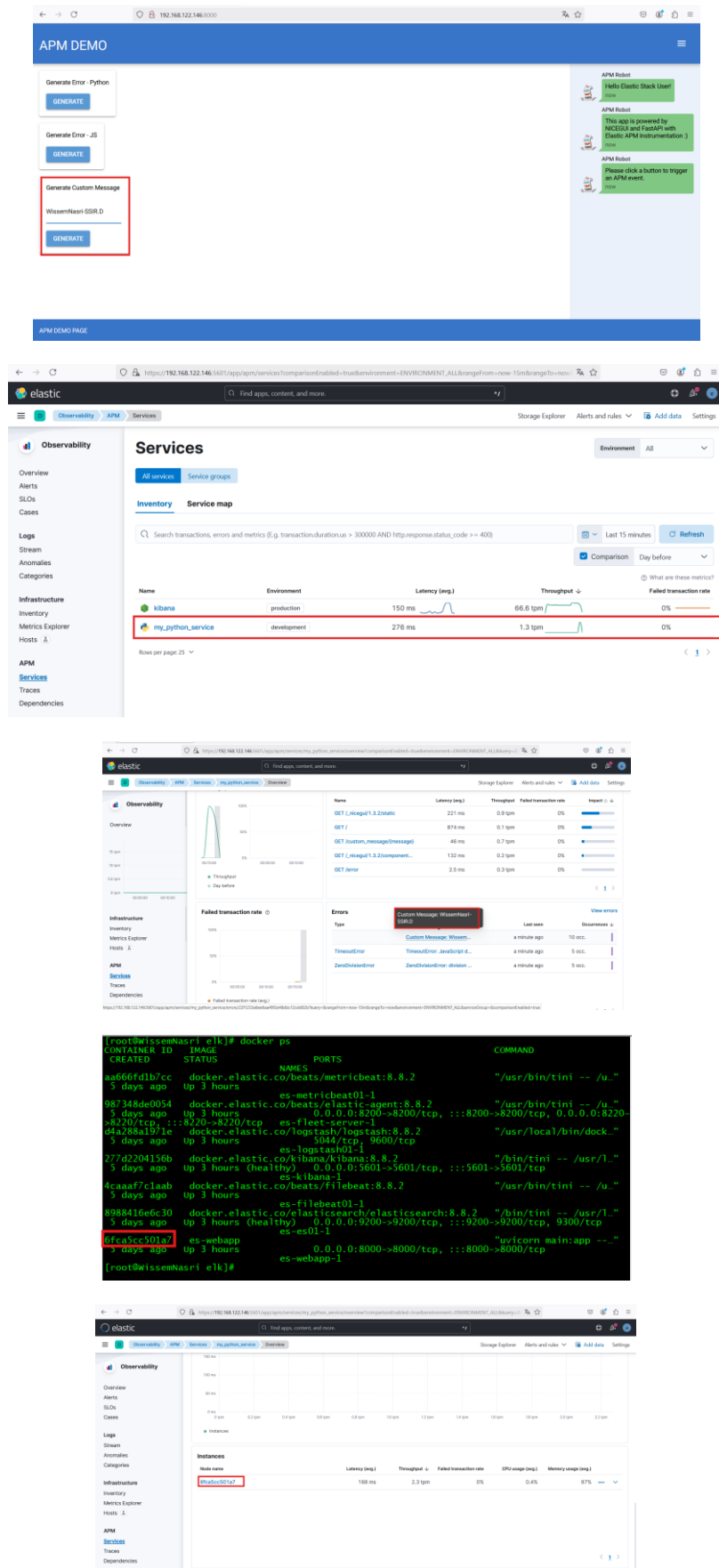


Figure 16: teste de fonctionnement de l'application et son traitement via serveur APM