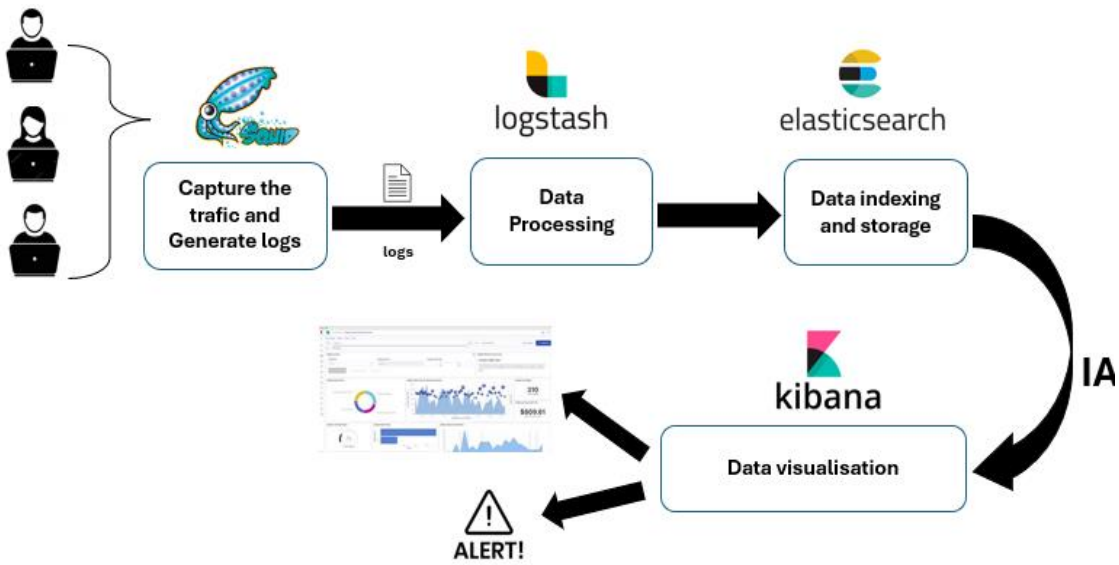
Rapport technique sur la configuration des outils du projet

Rappel de la problématique :

***D**ans le cadre de la surveillance du réseau d'entreprise, nous proposons une solution intégrée combinant **un proxy transparent**, **une analyse avancée des logs**, **un moteur de recherche** et de **visualisation des données**, ainsi **qu'un système d'intelligence artificielle** et **d'alerte**. Cette approche permet d'identifier en temps réel les activités suspectes tout en optimisant les performances du réseau.*

***L'**architecture mise en place repose sur les éléments suivants :*

- ***Squid (Proxy Transparent) :** Capture et filtre le trafic réseau HTTP/HTTPS.*
- ***Logstash :** Collecte, traite et transmet les logs vers Elasticsearch.*
- ***Elasticsearch :** Stocke et indexe les logs pour une recherche rapide et efficace.*
- ***Kibana :** Fournit des outils de visualisation et d'analyse sous forme de tableaux de bord interactifs.*
- ***Intelligence Artificielle (IA) :** Analyse comportementale pour détecter les activités suspectes.*
- ***Système d'Alerte :** Génération de notifications en cas d'anomalies détectées.*



Configuration des Outils, Collecte des Données et Autoconfiguration de Squid Proxy

🔧 **1. Configuration des Outils de Projet**

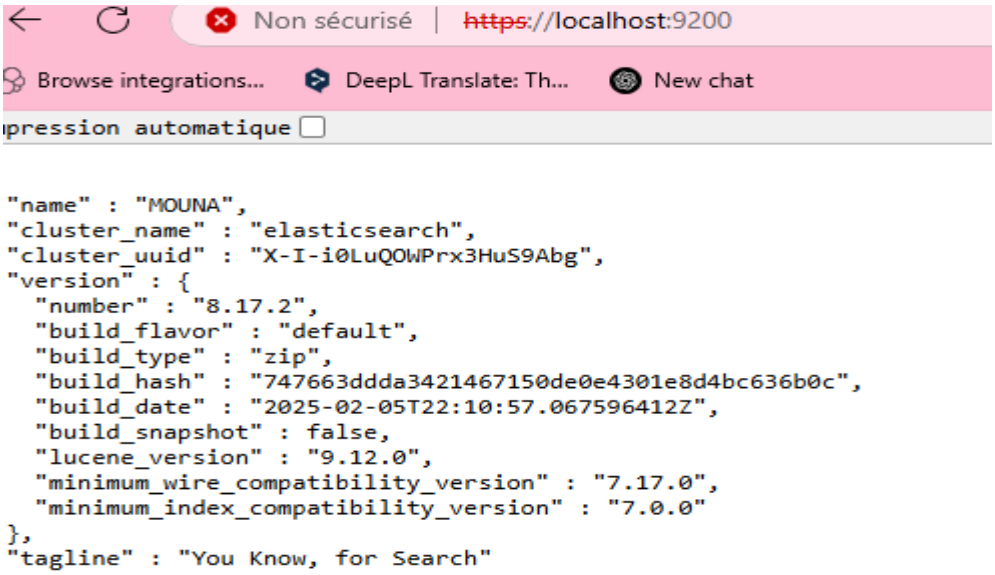
Le projet repose sur une architecture d'analyse et de filtrage intelligent du trafic réseau via :

- **Elasticsearch** : Pour configurer Elasticsearch, on utilise la commande **elasticsearch.bat**, ce qui génère un mot de passe et un nom d'utilisateur.

- **Nom d'utilisateur** : elastic
- **Mot de passe** : mDLu5n7azx_OxViC4=Fz
- **Token de connexion pour Kibana** :

  eyJ2ZXIiOiI4LjE0LjAiLCJhZHIiOlsiMTkyLjE2OC41Ni4xOjkyMDAiXSwiZmdyIjoiZTk0MGVkNTZiNTc0YjlkYWY4MGIwYzI2N2MxNzEwMDU4NmUzYmFlMzgyZWUxZGM5NDliN2ZjYzZhNTc0ZWQwNCIsImtleSI6IkVnR0tHSlVCZGpDajd1NGhRSkItOmdub2VhbWVEUm55Z3NjZkNoTlVvX3cifQ==

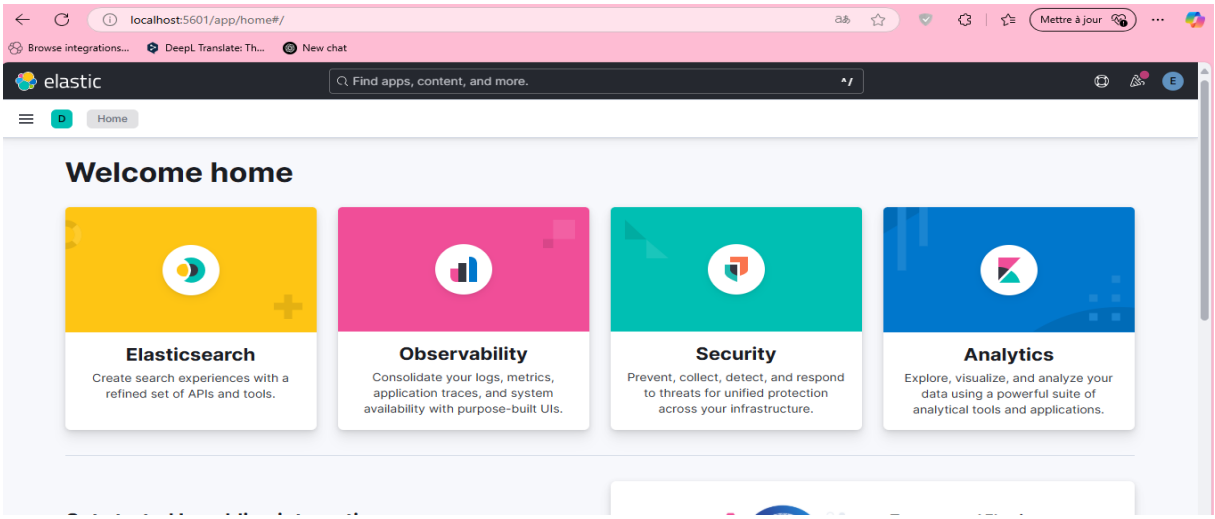
- , puis on accède à l'interface via l'URL suivante :
- 🔗 **https://localhost:9200/**



- **Kibana** : Pour configurer Kibana, on utilise la commande **kibana.bat**, puis on accède à l'interface via l'URL suivante :

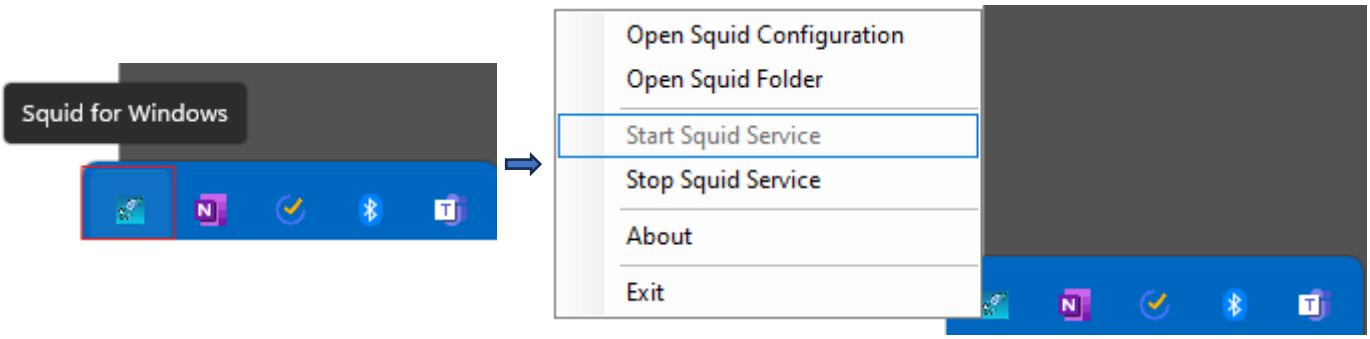
  🔗 **http://localhost:5601/**

  Cela permet d'interagir avec Elasticsearch et de visualiser les données indexées.

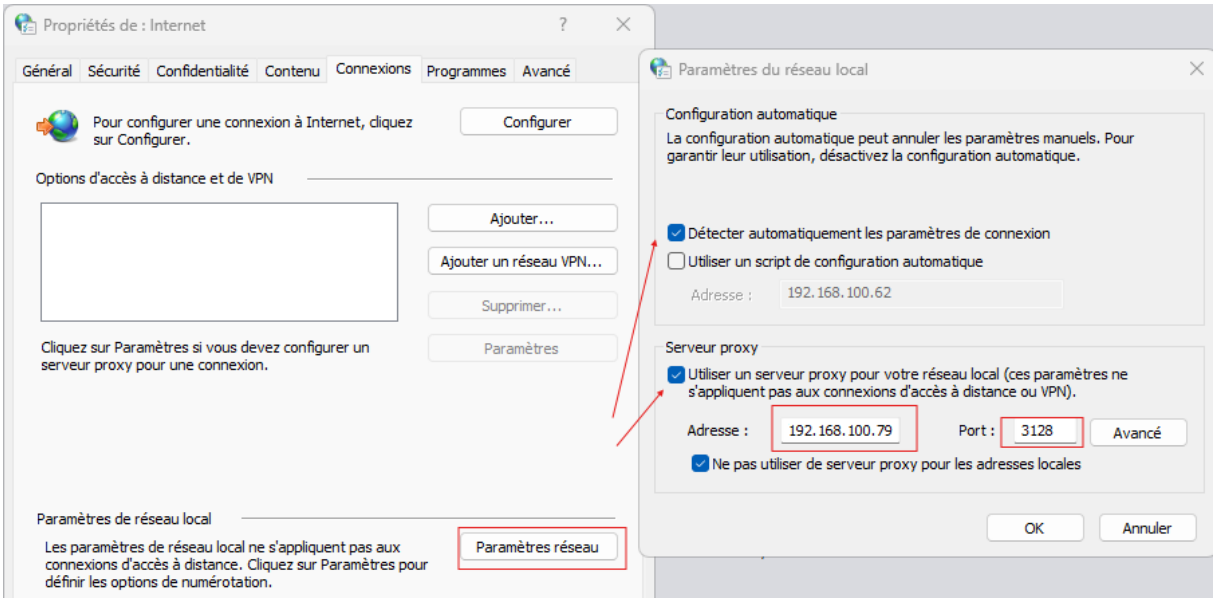


- **Squid Proxy** : pour la gestion du trafic web :

  On redémarre Squid.



Ensuite configurer le proxy sur le poste client : Internet Explorer : Outils/Options Internet/Onglet Connexions/Paramètres réseau



On envisage de connecter un ordinateur à un second PC, qui fera office de proxy chargé de collecter et traiter les logs. Pour cela, nous avons attribué l'adresse IP 192.168.100.79 au proxy, puis nous nous y sommes connectés.

- **Logstash** :

  Dans un premier temps, nous souhaitons **connecter Logstash**, qui collecte, filtre et transforme les logs générés par Squid, **à Elasticsearch** afin de les stocker et les indexer, puis les visualiser et les analyser sur Kibana pour le diagnostic.

➔ **Squid,** en tant que proxy cache, génère des logs détaillant les requêtes des utilisateurs, permettant d'analyser le trafic réseau, de détecter d'éventuelles anomalies et d'optimiser la bande passante. **Logstash** agit comme un outil d'ingestion de données, assurant le traitement et l'envoi des logs **vers Elasticsearch**, un moteur de recherche et d'analyse distribué qui permet un accès rapide et efficace aux données indexées. Enfin**, Kibana** offre une interface de visualisation intuitive qui facilite l'exploration et l'analyse des logs à travers des tableaux de bord interactifs et des graphiques.

```
input {
  file {
    # Utilisation d'un chemin sans espaces ou avec des guillemets doubles
    path => "C:/Program Files/k/squid/var/log/squid/access.log"
    start_position => "beginning"
    sincedb_path => "NUL"
    codec => plain { charset => "UTF-8" }
  }
}

filter {
  grok {
    match => {
      "message" => "%{NUMBER:timestamp} %{NUMBER:duration} %{IP:client_ip} %{WORD:method}/%{NUMBER:status} %{NUMBER:bytes} %{WORD:action} %{NOTSPACE:site}
%{NOTSPACE:other_info} %{NOTSPACE:hierarchy}/%{IP:server_ip} %{GREEDYDATA:additional_info}"
    }
    ecs_compatibility => disabled
  }

  # Filtrage de la date pour le timestamp UNIX
  date {
    match => ["timestamp", "UNIX"]
    target => "@timestamp"
  }

  # Ajout de la géolocalisation à partir de l'IP du client
  geoip {
    source => "client_ip"
    target => "geoip"
  }
}
```

Bloc input pour lire les logs depuis un fichier squid

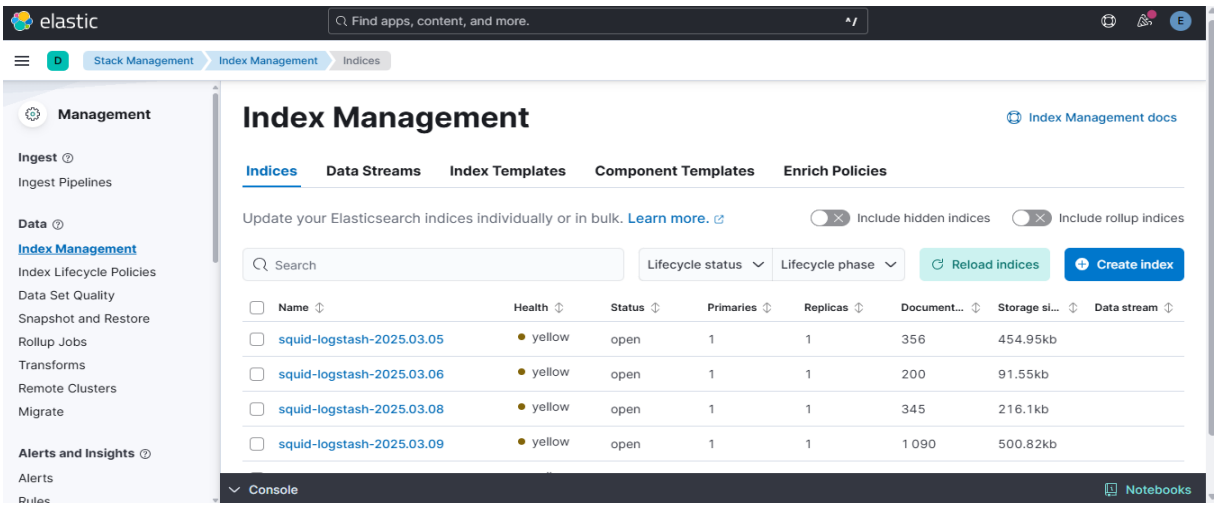
```
output {
  # Envoi des logs vers Elasticsearch
  elasticsearch {
    hosts => ["https://localhost:9200"]
    user => "elastic"
    password => "mDLu5n7azx_OxViC4=Fz"
    index => "squid-logstash-%{+YYYY.MM.dd}"
    ssl => false
  }
```

section output de Logstash.conf qui permet envoi des données traitées à Elasticsearch.

puis activer logstashe avec la commande :

```
C:\Program Files\k\logstash-8.17.2\bin>logstash.bat -f "C:/Program Files/k/logstash-8.17.2/config/logstash.conf"
```

- On peux maintenant voir les log sur kibana



2. Étapes de Collecte des Données avec Logstash

Première Méthode – Collecte via Logstash

logstash-sample - Bloc-notes

Fichier Edition Format Affichage Aide

```
input {
  file {
    path => "C:/Program Files/PFE/squid/var/log/squid/access.log"
    start_position => "beginning"
    sincedb_path => "NUL"
    codec => plain { charset => "UTF-8" }
    discover_interval => 15
    #ignore_older => 0
  }
}

filter {
  grok {
    match => {
      "message" => "%{NUMBER:timestamp}\s+%{NUMBER:duration}\s+%{IP:client_ip}\s+%{WORD:result}/%{NUMBER:status}\s+%{NUMBER:bytes}\s+%{WORD:method}\s+%{NOTSPACE:site}\s+%{NOTSPACE:rfc931}\s
    }
    # Pour continuer même si un log ne correspond pas exactement
    break_on_match => false
    # Vous pouvez également utiliser tag_on_failure pour suivre les erreurs
    tag_on_failure => ["_grokparsefailure"]
  }

  date {
    match => ["timestamp", "UNIX"]
    target => "@timestamp"
    locale => "en"
  }

  geoip {
    source => "client_ip"
    target => "geoip"
  }

  # Utiliser host.name comme nom d'hôte
  mutate {
    add_field => { "hostname" => "%{[host][name]}" }
  }

  mutate {
    add_field => { "log_date" => "%{@timestamp}" }
  }
}
```

Logstash reçoit les logs via input

Filtrage avec grok pour parser les logs

Ln 48, Col 39 100% Unix (LF) UTF-8

Taper ici pour FR 20°C 22:41 13/03/2025

logstash-sample - Bloc-notes

Fichier Edition Format Affichage Aide

```
    tag_on_failure => ["_grokparsefailure"]
  }

  date {
    match => ["timestamp", "UNIX"]
    target => "@timestamp"
    locale => "en"
  }

  geoip {
    source => "client_ip"
    target => "geoip"
  }

  # Utiliser host.name comme nom d'hôte
  mutate {
    add_field => { "hostname" => "%{[host][name]}" }
  }

  mutate {
    add_field => { "log_date" => "%{@timestamp}" }
  }
}

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    user => "elastic"
    password => "3Jx7CET1T6L8FJe_mV3v"
    index => "squid-new-logstash-%{+YYYY.MM.dd}"
    ssl => false
  }

  # Écriture dans un fichier CSV (vérifiez que tous les champs sont extraits)
  file {
    path => "C:/Program Files/PFE/ELK/logs/filtered_logs.csv"
    codec => line { format => "%{hostname},%{log_date},%{client_ip},%{site},%{status},%{method},%{bytes},%{result}" }
  }

  stdout { codec => rubydebug }
}
```

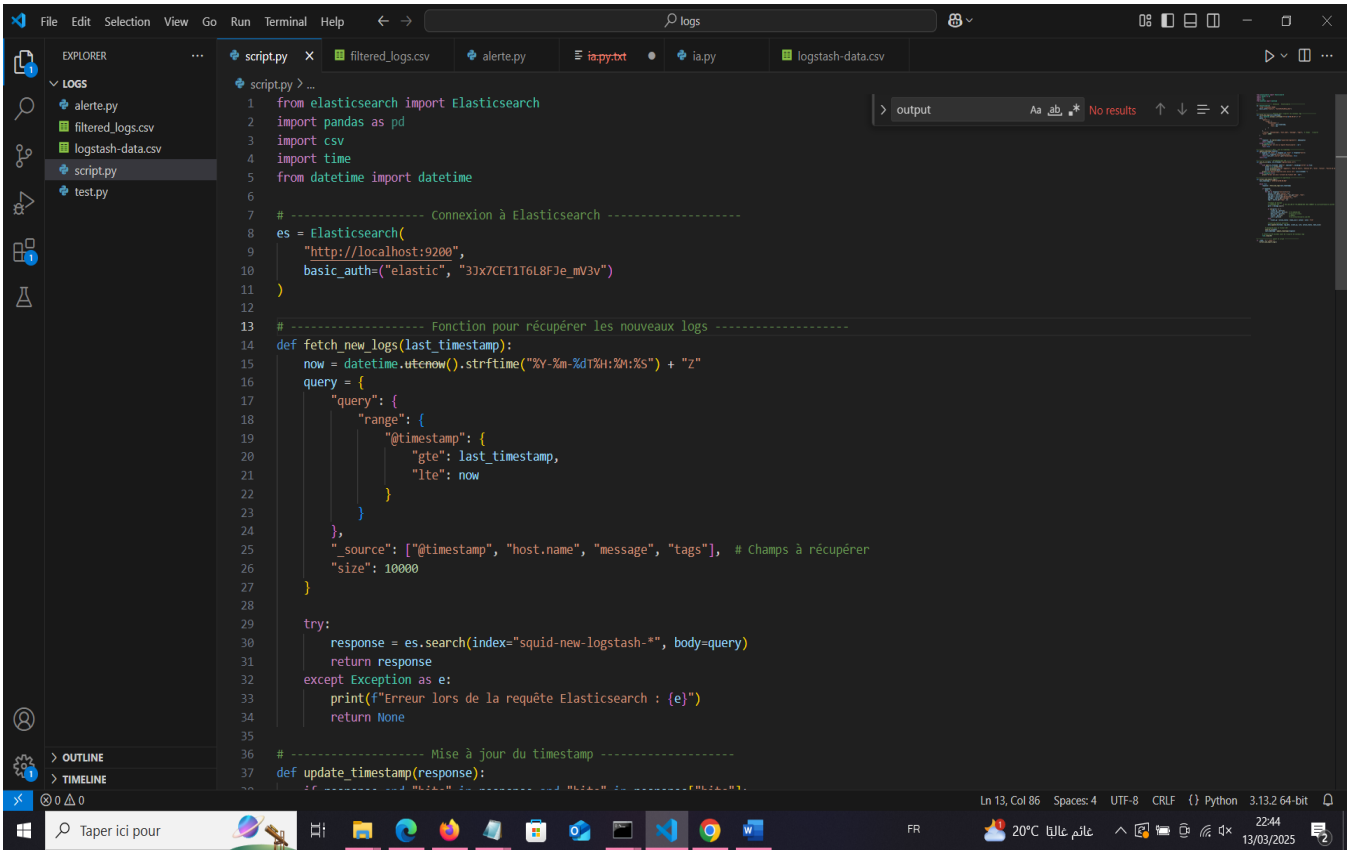
Envoi vers Elasticsearch

Ln 48, Col 39 100% Unix (LF) UTF-8

Taper ici pour FR 20°C 22:42 13/03/2025

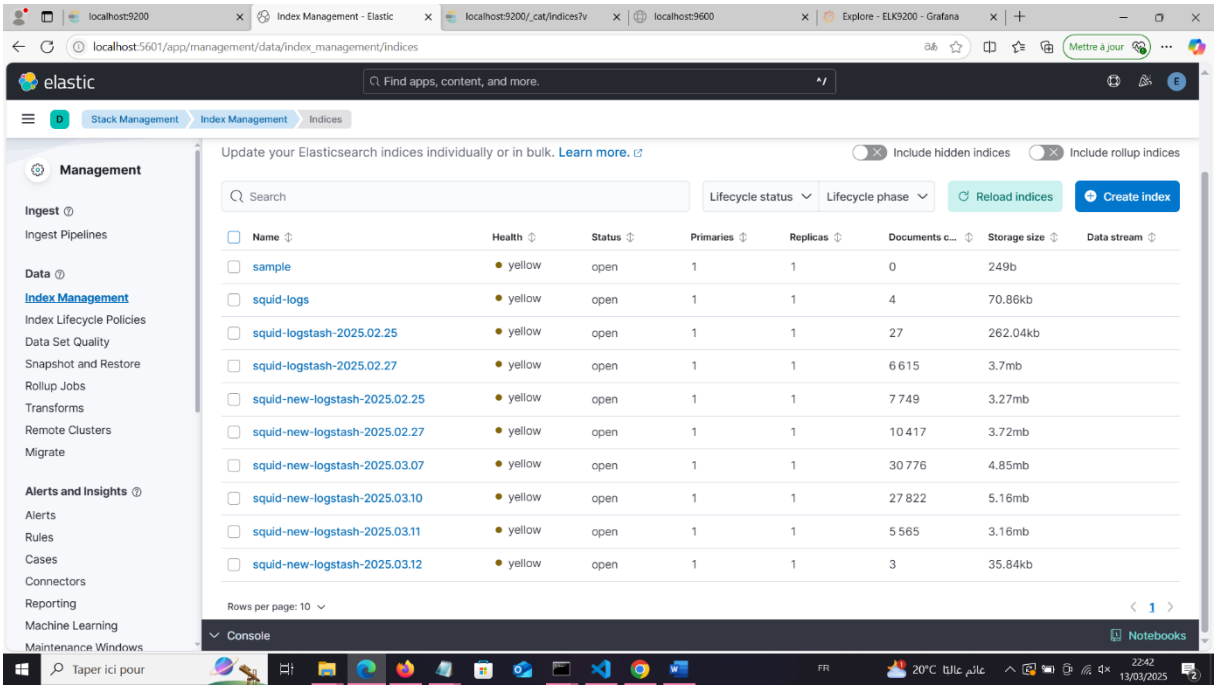
✅ Deuxième Méthode – Script Python d'Export des Logs depuis Elasticsearch

- Script pour exporter les logs indexés vers un fichier CSV pour traitement IA :

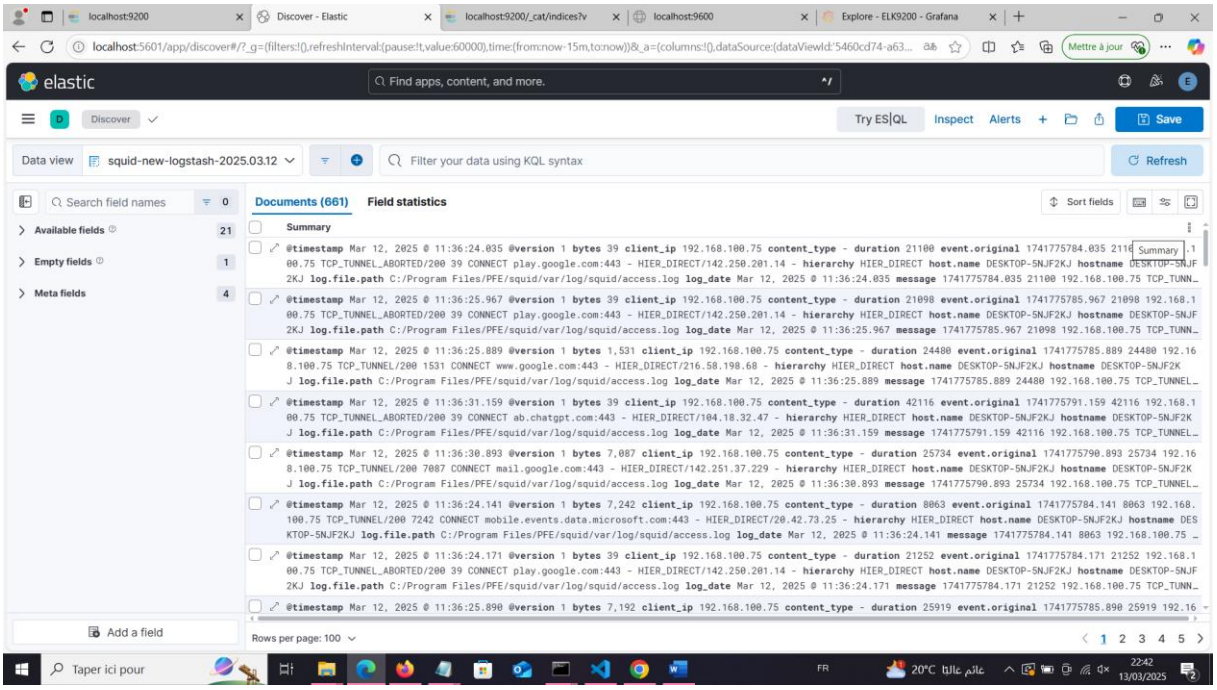


```python
from elasticsearch import Elasticsearch
import pandas as pd
import csv
import time
from datetime import datetime

# -------------------- Connexion à Elasticsearch --------------------
es = Elasticsearch(
    "http://localhost:9200",
    basic_auth=("elastic", "3Jx7CET1T6L8FJe_mV3v")
)

# -------------------- Fonction pour récupérer les nouveaux logs --------------------
def fetch_new_logs(last_timestamp):
    now = datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%S") + "Z"
    query = {
        "query": {
            "range": {
                "@timestamp": {
                    "gte": last_timestamp,
                    "lte": now
                }
            }
        },
        "_source": ["@timestamp", "host.name", "message", "tags"],  # Champs à récupérer
        "size": 10000
    }

    try:
        response = es.search(index="squid-new-logstash-*", body=query)
        return response
    except Exception as e:
        print(f"Erreur lors de la requête Elasticsearch : {e}")
        return None

# -------------------- Mise à jour du timestamp --------------------
def update_timestamp(response):
```

✅ 3. Résultats Obtenus

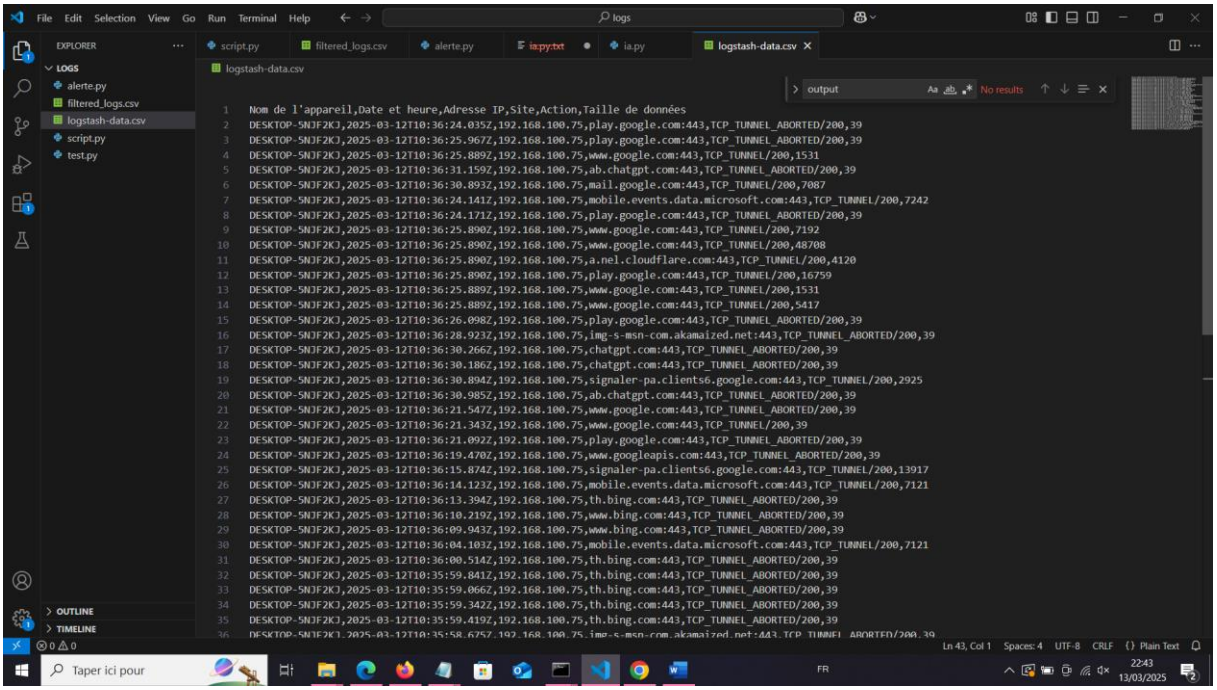


- Logs collectés en temps réel et affichés dans Kibana.

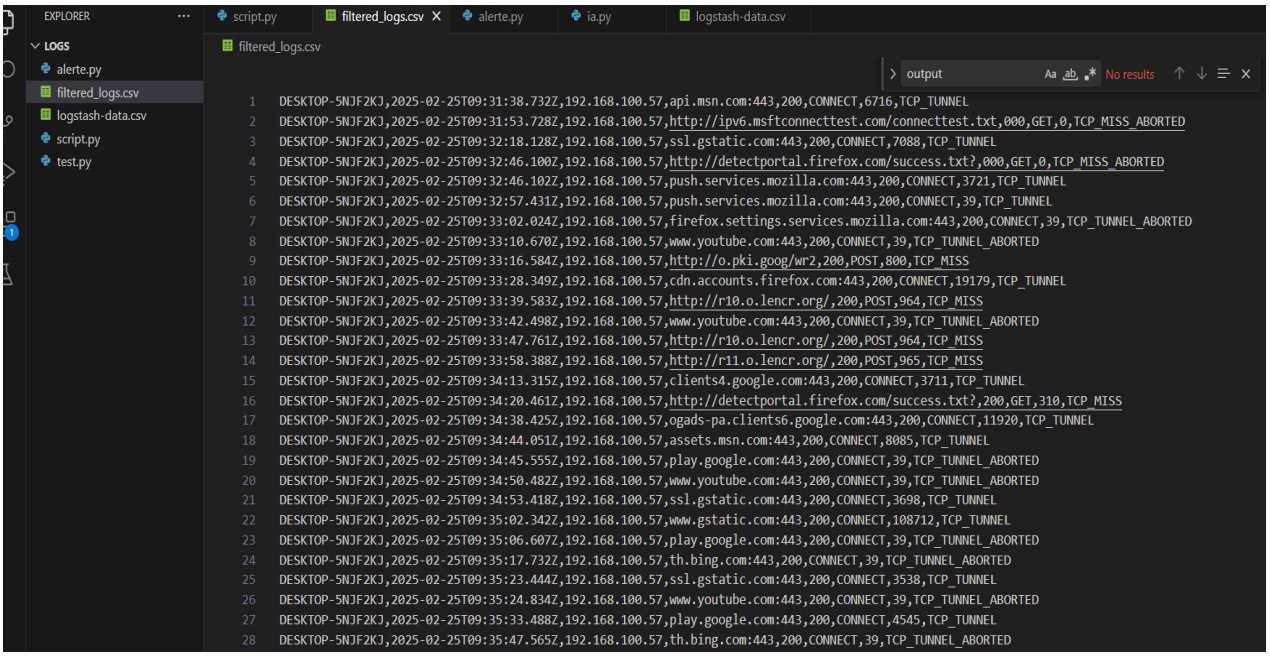


- Exportation réussie dans un fichier CSV pour traitement IA.

Par script :



Par logstsh :



- Identification de certains comportements suspects : transferts de fichiers volumineux, consultation fréquente de sites freelance ou hébergement cloud.

---

4. Idée d'Extension : Autoconfiguration Intelligente de Squid Proxy (IA)

Principe

- Le script IA doit analyser en continu le fichier CSV.
- Si :
  - Un **transfert > 100MB** est détecté,
  - Ou si l'URL appartient à une **liste noire (freelance, hébergement, etc.)**,
- Alors :
  - Une **alerte est envoyée au manager** (email ou popup).
  - Le site ou la connexion est automatiquement **ajouté dans la blacklist de Squid**.

Développement IA – Autoconfiguration de Squid Proxy

Nous sommes dans cette étapes

📁 **Structure du répertoire IA :**

/IA_Squid_Proxy/

├── data/

│ └── logstash-data.csv

```
├── ia_config/
│   └── sites_freelance.txt
├── scripts/
│   ├── analyse_ia.py
│   └── update_blacklist.bat
├── alertes/
│   └── alerte_log.txt
└── gui/
    └── ia_gui.py
```

---

1. Fichier sites_freelance.txt (liste noire initiale)

```
upwork.com
freelancer.com
fiverr.com
weworkremotely.com
```

---

2. Script Python IA : analyse_ia.py

Ce script :

- Analyse le fichier CSV des logs,
- Détecte les cas critiques (volume > 100MB ou site freelance),
- Envoie une alerte (log + popup),
- Met à jour le fichier de blacklist,
- Relance Squid via un batch.

📈 **Avantages de cette solution IA**

✅ Surveillance continue
✅ Détection proactive des abus réseau
✅ Blocage automatique sans intervention humaine
✅ Alerte directe au manager
✅ Facile à intégrer dans l'infrastructure actuelle