



École Supérieure de Technologie de Salé

Mémoire de fin d'études

Pour l'obtention du diplôme universitaire de technologie

Option : Systèmes Informatiques et Réseaux

Conception et réalisation d'un système d'analyse de trafic réseau chiffré

Réalisé par :
M. Karroud Nizar

Encadré par :
M. Amraoui Mounir

Promotion : 2024/2025

Remerciements

Je tiens tout d'abord à exprimer ma gratitude à M. Mounir Amraoui pour son encadrement.

Je souhaite également remercier les développeurs de nProbe, qui m'ont généreusement offert une licence pour utiliser leur logiciel. Leur contribution a été déterminante pour l'avancement de mon projet .

Résumé

Ce projet vise la conception d'un système d'Encrypted Network Traffic Analysis (ENTA) en exploitant une data pipeline dédiée à la collecte, au traitement et à l'analyse des flux réseau.

Les données NetFlow, générées par les appareils réseau, sont envoyées à nProbe sur un serveur dédié, où elles sont traitées avant d'être ingérées par Apache Kafka, stockées dans Apache Druid, puis visualisées avec Apache Superset.

Cette architecture permet une analyse en temps réel et offre une meilleure compréhension des comportements réseau, même sans accès au contenu chiffré des paquets.

Dans une seconde phase, ces données seront exploitées pour la détection d'anomalies à l'aide d'un système de détection d'intrusions (IDS) basé sur l'apprentissage automatique.

L'objectif est d'identifier des activités suspectes et de renforcer la sécurité du réseau en s'appuyant uniquement sur les métadonnées des flux.

Abstract

This project aims to design an Encrypted Network Traffic Analysis (ENTA) system by leveraging a dedicated data pipeline for the collection, processing, and analysis of network flows.

NetFlow data, generated by network devices, is sent to nProbe on a dedicated server, where it is processed before being ingested by Apache Kafka, stored in Apache Druid, and then visualized with Apache Superset.

This architecture enables real-time analysis and provides a better understanding of network behavior, even without access to the encrypted packet content.

In the second phase, this data will be used for anomaly detection through an Intrusion Detection System (IDS) based on machine learning.

The goal is to identify suspicious activities and enhance network security by relying solely on flow metadata.

Table des matières

Remerciements.....	I
Résumé.....	II
Abstract.....	III
Table des matières.....	IV
1 contexte et Objectifs.....	6
1.1 Introduction et Problématique.....	7
1.2 Étude de l'existant.....	8
1.2.1 Cisco Stealthwatch.....	8
1.2.2 Darktrace.....	8
1.2.3 Palo Alto Cortex XDR.....	9
1.3 Solution	10
2 Encrypted Network Traffic Analysis	11
2.1 Introduction.....	12
2.2 Méthodologie ENTA	13
2.3 Collection des Données de Trafic et Prétraitement	14
2.3.1 Packet-Based Features	16
2.3.2 Flow-Based Features	17
3 Conception.....	20
3.1 Introduction.....	21
3.2 Architecture	21
3.3 Environnement Technique et Outils	22
3.3.1 nProbe.....	22
3.3.2 Serveur de Prétraitement	23
3.3.3 Apache Kafka.....	24
3.3.4 Apache Druid.....	27
3.3.5 Apache Superset.....	33

4 Implémentation.....	34
4.1 Installation et Configuration de l'Environnement.....	35
4.1.1 Installation de VMware Workstation	35
4.1.2 Installation d'Ubuntu	35
4.2 Installation et Configuration d'Apache Druid sur Ubuntu	38
4.3 Installation et Configuration d'Apache Superset sur Ubuntu	41
4.4 Configuration et Déploiement d'un conteneur Apache Kafka	49
4.5 Installation et Configuration de nProbe sur Windows	51
4.6 Mise en Place du Serveur de Prétraitement des Flux.....	51
4.7 Connexion d'Apache Superset à Apache Druid.....	56
4.8 Connexion d'Apache Kafka à Apache Druid	58
4.9 Visualisation des Flux NetFlow avec Apache Superset.....	67
4.9.1 Intensité du trafic entrant par heure (carte thermique IN_BYTES).....	68
4.9.2 Intensité du trafic sortant par heure (carte thermique OUT_BYTES).....	69
4.9.3 Analyse de la fréquence des protocoles réseau.....	70
4.9.4 Carte de direction du trafic par protocole.....	71
4.9.5 Analyse de l'adresse IP source et de la moyenne durée des flux	72
4.9.6 Nombre de flux par jour	73
4.9.7 Carte en arbre des IP de destination dans le réseau	74
4.9.8 Carte en arbre des IP source envoyant du trafic vers le réseau	75
4.9.9 Carte en arbre des IP source envoyant du trafic hors du réseau	76
4.9.10 Carte en arbre des IP de destination recevant du trafic depuis le réseau	77
Conclusion	79
Bibliographie.....	80
Références	81

Contexte et Objectifs

1.1 Introduction et Problématique

Avec l'augmentation de l'utilisation du chiffrement dans les communications réseau, l'analyse du trafic devient un défi majeur pour la cybersécurité. Les protocoles de sécurité implémentés aux différentes couches du modèle TCP/IP protègent les données en transit contre les accès non autorisés. Toutefois, cette protection est également exploitée par des acteurs malveillants pour encapsuler des communications malicieuses, rendant leur détection plus complexe.

Dans ce contexte, les approches traditionnelles de détection, qui reposent sur l'inspection du contenu des paquets, deviennent moins efficaces face à l'opacité du chiffrement. Certaines approches consistent à déchiffrer le trafic avant de l'analyser, mais elles posent plusieurs problèmes :

- Coût en ressources : le déchiffrement consomme une quantité importante de puissance de calcul et de bande passante.
- Atteinte à la confidentialité : cette méthode compromet la vie privée des utilisateurs, car elle brise l'intégrité des communications chiffrées.

Face à ces défis, une nouvelle approche d'analyse du trafic chiffré est nécessaire. Plutôt que d'inspecter le contenu des paquets, il est possible d'exploiter des métadonnées réseau et des caractéristiques comportementales du trafic (statistiques sur les flux, temporalité, etc.) pour identifier des anomalies.

L'objectif de ce projet est donc de répondre aux questions suivantes :

- Comment analyser un trafic réseau chiffré sans compromettre la confidentialité des communications ?
- Quelles sont les métadonnées réseau les plus pertinentes pour détecter des menaces dans un flux chiffré ?
- Comment exploiter l'apprentissage automatique pour identifier des anomalies sans avoir accès au contenu des paquets ?
- Comment collecter, acheminer et traiter en temps réel les données de flux issues des différents collecteurs réseau pour alimenter l'analyse et la détection des menaces ?

Contexte et Objectifs

1.2 Étude de l'existant

L'analyse du trafic réseau est un élément essentiel pour la détection des menaces et la surveillance des infrastructures informatiques. Plusieurs solutions propriétaires proposent des approches avancées basées sur l'intelligence artificielle et l'analyse comportementale.

1.2.1 Cisco Stealthwatch

- Type : NTA (Network Traffic Analysis) & Threat Intelligence.
- Fonctionnalités :
 - Détection d'anomalies comportementales grâce au machine learning
 - Corrélation avec des sources de renseignement sur les menaces (Threat Intelligence).
 - Surveillance du trafic chiffré sans décryptage.
- Limites : Coût élevé, dépendance à l'écosystème Cisco.

1.2.2 Darktrace

- Type : NTA basé sur l'IA et l'apprentissage automatique
- Fonctionnalités :
 - Analyse en temps réel pour détecter des comportements inhabituels
 - Visualisation avancée des interactions réseau.
 - Réponse autonome avec son module Antigena (capacité de réponse automatique aux menaces).
- Limites : Boîte noire (algorithmes propriétaires), peut générer de faux positifs.

Contexte et Objectifs

1.2.3 Palo Alto Cortex XDR

- Type : Solution de détection et réponse étendue (XDR).
- Fonctionnalités :
 - Agrégation des logs réseau et analyse comportementale.
 - Automatisation des réponses aux incidents.
 - Intégration avec des sources externes pour améliorer la visibilité.
- Limites : Dépend fortement d'autres produits Palo Alto pour être pleinement efficace.

1.3 Solution

L'approche proposée ne prétend pas être révolutionnaire ni meilleure que les solutions propriétaires existantes. Cependant, elle adopte un point de vue différent en s'appuyant sur une pipeline de données pour l'analyse du trafic réseau.

Contrairement aux solutions propriétaires qui dépendent de technologies spécifiques et d'écosystèmes fermés, cette approche repose sur des outils open-source et des méthodes flexibles.

L'objectif est de permettre une analyse des flux réseau sans dépendance vis-à-vis d'un fournisseur particulier, offrant ainsi une alternative accessible et adaptable aux besoins spécifiques des entreprises et des chercheurs en cybersécurité

Encrypted Network Traffic Analysis

2.1 Introduction

Selon un rapport de Google [1], 99 % du trafic Internet est chiffré afin de garantir la confidentialité et la sécurité des données lors de leur transmission. Cependant, sous couvert de chiffrement, des acteurs malveillants exploitent cette protection pour mener diverses cyberattaques. L'analyse du trafic réseau chiffré (Encrypted Network Traffic Analysis - ENTA) est un processus qui consiste à examiner le trafic Internet protégé par différents protocoles de chiffrement afin d'en extraire des informations pertinentes sur sa nature et son comportement.

L'ENTA joue un rôle crucial pour les administrateurs réseau et les professionnels de la cybersécurité, leur permettant de protéger les données, les communications et les infrastructures réseau contre les menaces cachées dans le trafic chiffré. Les outils traditionnels comme les pare-feux applicatifs et les systèmes de détection d'intrusions (IDS) ne sont efficaces que sur le trafic non chiffré et ne peuvent donc pas traiter l'ENTA de manière adéquate. Certaines méthodes, comme l'inspection approfondie des paquets (Deep Packet Inspection - DPI), tentent de briser le chiffrement pour analyser les données, mais elles compromettent la confidentialité des utilisateurs et introduisent des risques de sécurité supplémentaires. De même, la classification du trafic basée sur les ports devient inefficace face au chiffrement.

Les recherches récentes se concentrent sur l'utilisation de l'intelligence artificielle (IA) et du machine learning (ML) pour identifier et classer le trafic chiffré malveillant sans avoir besoin de le déchiffrer. Grâce aux techniques statistiques et d'apprentissage automatique, il est possible de détecter des comportements suspects, d'identifier les communications chiffrées entre des serveurs de command-and-control (C2) et des systèmes ciblés, et d'extraire des modèles de trafic à partir de divers paramètres comme les paquets, les flux, les sessions et les connexions réseau. Ces techniques permettent de créer des empreintes numériques des applications et d'analyser les protocoles chiffrés avec une meilleure précision. Contrairement aux approches basées sur l'inspection approfondie des paquets, les techniques statistiques et d'apprentissage profond (deep learning - DL) exploitent des caractéristiques moins affectées par le chiffrement, ce qui leur permet de contourner certaines limitations.

Toutefois, l'ENTA peut être utilisée à des fins légitimes comme malveillantes. Si elle permet d'améliorer la sécurité des réseaux, elle peut également être détournée par des attaquants pour espionner les applications utilisées sur un appareil, surveiller les sites visités, même sur des réseaux anonymisés comme Tor, ou encore analyser les actions des utilisateurs.

Il est particulièrement difficile de détecter des communications malveillantes dans un trafic chiffré, car le chiffrement masque les données de charge utile, compliquant ainsi la gestion du réseau et l'efficacité des outils de cybersécurité tels que les pare-feux et les IDS. Certains attaquants exploitent cette caractéristique pour contourner la détection. Bien que l'analyse des certificats TLS et des descriptions de charges utiles puisse atténuer ces risques, cela soulève d'importants problèmes de respect de la vie privée. Trouver un équilibre entre protection des communications, sécurité du réseau et analyse du trafic chiffré reste donc un défi majeur pour les chercheurs et les professionnels de la cybersécurité.

2.2 Méthodologie ENTA

La première étape consiste à identifier l'objectif de l'analyse. En fonction de cet objectif, une approche adaptée peut être conçue.

L'ENTA vise à garantir la qualité de service, la sécurité, la modélisation des activités et comportements. Les objectifs de recherche peuvent inclure la classification du trafic chiffré, la détection des communications malveillantes, l'optimisation des ressources réseau et de la qualité de service, etc.

Une fois les objectifs définis, l'étape suivante consiste à collecter les données pertinentes. Cette phase comprend plusieurs étapes, notamment l'identification des points de collecte des données et l'utilisation d'outils spécifiques pour la capture du trafic.

Les données collectées doivent être prétraitées afin de gérer divers problèmes, tels que le bruit ou les valeurs manquantes.

La sélection et l'extraction des caractéristiques sont des étapes cruciales du processus ENTA. Les caractéristiques peuvent être directement disponibles dans les données ou dérivées à l'aide de méthodes statistiques.

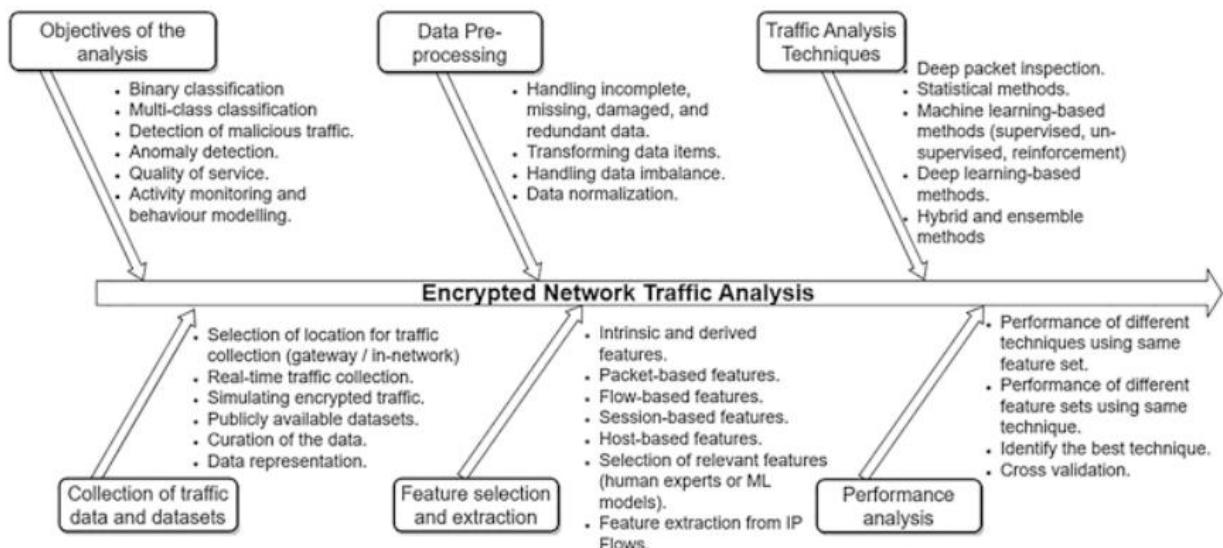


FIG. 2.1 : Méthodologie ENTA

2.3 Collection des Données de Trafic et Prétraitement

Pour l'entraînement des modèles, les données peuvent être collectées en temps réel ou simulées. Dans le cas de l'analyse en temps réel, le trafic généré par différents appareils est collecté par les passerelles, pare-feu, fournisseurs d'accès à Internet (FAI) et serveurs à l'aide de moniteurs de trafic.

Ce trafic est ensuite analysé par des systèmes exploitant différentes techniques statistiques, d'apprentissage automatique ou d'analyse de données. De nombreux outils sont disponibles pour la surveillance et l'analyse du trafic réseau [2].

L'un des principaux défis de la collecte de données en temps réel pour l'entraînement des modèles est le problème du déséquilibre des données. Obtenir une quantité suffisante de trafic malveillant est particulièrement difficile. En alternative, il existe des simulateurs de trafic réseau capables de générer du trafic selon des besoins spécifiques.

Le tableau 2.1 présente quelques simulateurs de réseau permettant de générer du trafic réseau.

Table 2.1 Outils de Simulation de Trafic Réseau

Outil	Description
NS3 [3]	Outil permettant de simuler des réseaux réels. Il génère également des fichiers PCAP pouvant être analysés avec des outils comme Wireshark.
NPing [4]	Outil open source permettant de générer des paquets réseau sur une large gamme de protocoles.
Ostinato [5]	Outil permettant de générer des fichiers PCAP et du trafic réseau de différents types.
WANkiller [6]	Outil de génération de trafic axé sur la surveillance réseau en temps réel et les tests de charge.
Netropy [7]	Outil permettant de créer du trafic de fond en utilisant des fichiers PCAP.
Bit-Twist Packet Generator [8]	Outil open source de génération et d'édition de paquets Ethernet.
TCPReplay [9]	Utilitaires open source permettant d'éditer et de rejouer du trafic réseau déjà capturé.

Encrypted Network Traffic Analysis

Le Dataset, qu'il soit réel ou simulé, doit posséder certaines caractéristiques essentielles :

- Une quantité suffisante de données d'entraînement,
- Une représentation variée des protocoles, applications et types d'attaques,
- Un équilibre des classes,
- Des données avec une vérité terrain bien confirmée.

Ces caractéristiques sont essentielles pour construire des modèles d'Analyse du Trafic Réseau Chiffré (ENTA) robustes et précis. Plusieurs dataset publics sont disponibles, spécifiques à certains types d'applications, protocoles et attaques. Le Canadian Institute for Cybersecurity (CIC) propose plusieurs ensembles de données pour les expériences (<https://www.unb.ca/cic/datasets/index.html>).

Wang et al. [10] ont composé un dataset à partir de multiples sources publiques pour surmonter ce défi. Il satisfait plusieurs critères : un bon équilibre entre trafic malveillant et non malveillant, des données issues de divers dispositifs (y compris des objets connectés) et une répartition égale des volumes de trafic.

La prochaine étape de l'ENTA consiste à extraire les caractéristiques pertinentes pour l'analyse des ensembles de données du trafic réseau.

L'aspect difficile de cette extraction est d'identifier les caractéristiques pertinentes et de comprendre leurs spécificités. La littérature disponible a classé ces caractéristiques en plusieurs catégories.

2.3.1 Packet-Based Features

Les caractéristiques basées sur les paquets sont généralement faciles à obtenir. La caractéristique de base des paquets, c'est-à-dire les adresses source et destination, les ports source et destination, ainsi que les informations sur le protocole, constitue un ensemble de caractéristiques intrinsèques. De plus, nous pouvons également dériver des caractéristiques telles que le temps d'arrivée entre les paquets et les valeurs statistiques des paquets.

Le tableau 2.2 fournit un résumé des principales caractéristiques au niveau des paquets.

La taille des paquets diffère entre les trafics non chiffrés et chiffrés. Par conséquent, cela aide à classifier le trafic. De même, le trafic chiffré aura des temps de paquets plus constants que le trafic non chiffré. Les valeurs d'entropie des paquets sont plus élevées pour le trafic chiffré que pour le trafic non chiffré. La distribution de la taille des paquets du trafic chiffré est différente de celle du trafic non chiffré. Alors que la charge utile des paquets du trafic chiffré ne révèle pas beaucoup de détails, ses métadonnées fournissent plus d'informations.

Les caractéristiques au niveau des paquets sont extraites des paquets de trafic réseau qui passent par diverses interfaces réseau. Par la suite, ces caractéristiques sont exportées et analysées.

Ainsi, la collecte et l'analyse des caractéristiques au niveau des paquets sont coûteuses et non évolutives.

Table 2.2 Caractéristiques basées sur les paquets

Caractéristique	Description
Taille du paquet	Longueur d'un paquet de trafic réseau en octets
Adresse source du paquet	Adresse IP source du paquet
Adresse de destination du paquet	Adresse IP de destination du paquet
Port source	Numéro de port de la source du paquet
Port de destination	Numéro de port de la destination du paquet
Protocole	Protocole applicatif régissant le paquet
Temps d'inter-arrivée	Temps d'arrivée du paquet
Nombre de paquets	Nombre total de paquets transférés pendant un temps donné

Encrypted Network Traffic Analysis

Caractéristiques statistiques basées sur les paquets telles que la moyenne, la médiane, le maximum, le minimum, la variance et l'écart-type	Ce sont diverses caractéristiques statistiques qui peuvent être dérivées de la taille des paquets et aussi du temps d'inter-arrivée des paquets
---	---

2.3.2 Flow-Based Features

Un flux peut être décrit comme un ensemble de paquets partageant les mêmes données de 5-tuple et observés dans un intervalle de temps donné.

Les caractéristiques basées sur les flux fournissent des détails agrégés du trafic réseau. Par conséquent, la quantité de données à analyser diminue considérablement par rapport aux caractéristiques basées sur les paquets.

Les caractéristiques au niveau des flux décrivent les propriétés du flux du trafic réseau. Elles fournissent des informations de haut niveau sur le trafic réseau à travers les adresses, les ports, les octets et les comptes de paquets.

Le tableau 2.3 fournit des caractéristiques de flux utilisées pour l'ENTA.

Table 2.3 Caractéristiques basées sur les flux

Encrypted Network Traffic Analysis

S. no	Flow features	Description
1.	Destination port	Port number of the receiver
2.	Flow duration	The total flow duration record
3.	Total Fwd packets	Total number of packets from sender to receiver
4.	Total Bwd packets	Total number of packets from receiver to sender
5.	Total length of Fwd packets	Total size of packets from sender to receiver
6.	Total length of Bwd packets	Total size of packets from receiver to sender
7.	Fwd packet length max	Maximum size of the packet from sender to receiver
8.	Fwd packet length min	Minimum size of the packet from sender to receiver
9.	Fwd packet length mean	Mean size of the packet from sender to receiver
10.	Fwd packet length std	Standard deviation size of the packet from sender to receiver
11.	Bwd packet length max	Maximum size of the packet from receiver to sender
12.	Bwd packet length min	Minimum size of the packet from receiver to sender
13.	Bwd packet length mean	Mean size of the packet from receiver to sender
14.	Bwd packet length std	Standard deviation size of the packet from receiver to sender
15.	Flow bytes/s	Number of flow in bytes per second
16.	Flow packets/s	Number of flow in packets per second
17.	Flow IAT mean	Mean time between two packets sent in the flow
18.	Flow IAT std	Standard deviation time between two packets sent in the flow
19.	Flow IAT max	Maximum time between two packets sent in the flow
20.	Flow IAT min	Minimum time between two packets sent in the flow
21.	Fwd IAT total	Total time between two packets sent from sender to receiver
22.	Fwd IAT mean	Mean time between two packets sent from sender to receiver
23.	Fwd IAT std	Standard deviation time between two packets sent from sender to receiver
24.	Fwd IAT max	Maximum time between two packets sent from sender to receiver
25.	Fwd IAT min	Minimum time between two packets sent from sender to receiver
26.	Bwd IAT total	Total time between two packets sent from receiver to sender
27.	Bwd IAT mean	Mean time between two packets sent from receiver to sender
28.	Bwd IAT std	Standard deviation time between two packets sent from receiver to sender
29.	Bwd IAT max	Maximum time between two packets sent from receiver to sender
30.	Bwd IAT min	Minimum time between two packets sent from receiver to sender

Encrypted Network Traffic Analysis

S. no	Flow features	Description
31.	Fwd PSH flags	Number of times the PSH flag set to 1 (TCP) and 0 (UDP) for the packets from source to destination
32.	Bwd PSH flags	Number of times the PSH flag set to 1 (TCP) and 0 (UDP) for the packets from destination to source
33.	Fwd URG flags	Number of times the URG flag set to 1 (TCP) and 0 (UDP) for the packets from source to destination
34.	Bwd URG flags	Number of times the PSH flag set to 1 (TCP) and 0 (UDP) for the packets from destination to source
35.	Fwd header length	Total bytes used for packet header from source to destination
36.	Bwd header length	Total bytes used for packet header from destination to source
37.	Fwd packets/s	Number of packets from source to destination per second
38.	Bwd packets/s	Number of packets from destination to source per second
39.	Min packet length	Lower limit on length of a packet
40.	Max packet length	Upper limit on length of a packet
41.	Packet length mean	Mean length of a packet
42.	Packet length Std	Standard deviation length of a packet
43.	Packet length variance	Variance length of a packet
44.	Fin flag count	Number of packets with request for connection termination
45.	Syn flag count	Number of packets in first step of connection establishment phase
46.	RST flag count	Number of packets ready to terminate the connection
47.	PSH flag count	Number of packets with request for immediate delivery
48.	ACK flag count	Number of initial packets successfully received
49.	URG flag count	Number of packets with urgent pointer field
50.	CWR flag count	Number of packets with congestion window reduced flag
51.	ECE flag count	Number of packets received with CE bit set
52.	Down/up ratio	Download and upload ratio
53.	Avg packet size	Average size of packet
54.	Avg Fwd segment size	Average size of the segment from source to destination
55.	Avg Bwd segment size	Average size of the segment from destination to source
56.	Fwd header length	Total bytes used for headers from source to destination
57.	Bwd header length	Total bytes used for headers from destination to source
58.	Fwd Avg bytes/bulk	Average number of bytes in bulk rate from source to destination
59.	Fwd Avg packets/bulk	Average number of packets in bulk rate from source to destination
60.	Fwd Avg bulk rate	Average number of bulk rate from source to destination
61.	Bwd Avg bytes/bulk	Average number of bytes in bulk rate from destination to source
62.	Bwd Avg packets/bulk	Average number of packets in bulk rate from destination to source
63.	Bwd Avg bulk rate	Average number of bulk rate from destination to source

Conception

3.1 Introduction

Ce projet vise à mettre en place une architecture de collecte, de transformation, de stockage et de visualisation des données de trafic réseau en temps réel. Le système utilise un collecteur Nprobe pour recevoir les données NetFlow v9 envoyées par des dispositifs de réseau. Ces données sont ensuite traitées et nettoyées par un serveur avant d'être envoyées à Kafka pour le traitement en temps réel.

Une fois les données disponibles dans Kafka, elles sont ingérées par Apache Druid, un moteur de données en temps réel capable d'effectuer des analyses et agrégations sur de grandes quantités de données temporelles.

Enfin, pour permettre une analyse et une visualisation interactives des données, Apache Superset est utilisé pour créer des dashboards permettant de suivre en temps réel les métriques et indicateurs de trafic réseau.

Cette solution permet de visualiser et d'analyser en temps réel les données de trafic réseau, facilitant ainsi la détection d'anomalies, la surveillance de la performance réseau,

3.2 Architecture

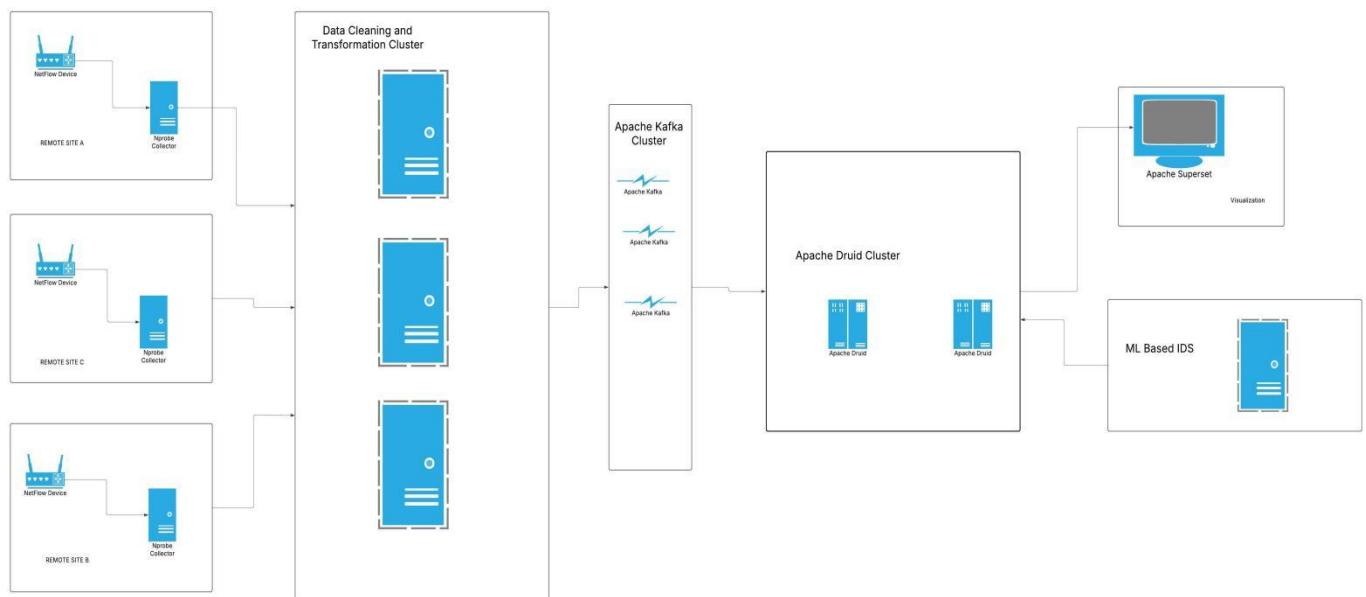


FIG. 3.1 : Architecture de la solution

3.3 Environnement Technique et Outils

3.3.1 NProbe

- Introduction**

nProbe est une sonde logicielle compatible avec les formats NetFlow v5/v9/IPFIX, permettant de collecter, analyser et exporter des rapports de trafic réseau en utilisant le format standard de Cisco. Il est disponible sur la plupart des systèmes d'exploitation du marché (Windows, BSD, Linux, MacOSX).

- Modes de fonctionnement de nProbe**

- Mode Probe**

Dans ce mode, nProbe agit comme une sonde qui capture le trafic réseau et exporte les flux vers un collecteur.

```
nprobe -i eth0 -n collector_ip:2055
```

Par défaut, les flux exportés sont envoyés via URL, mais ils peuvent également être transmis par TCP et TLS.

Exemples de syntaxe :

```
-n 2055  
-n udp://2055  
-n tcp://2055  
-n tls://2055  
-n tls://192.168.2.25:2055
```

- Mode Collector**

Dans ce mode, nProbe agit uniquement comme un collecteur de flux NetFlow envoyés par d'autres dispositifs réseau.

```
nprobe -3 2055
```

Les flux sont collectés via URL, TCP ou TLS. Si une adresse IP est spécifiée, la collecte est restreinte à cette adresse plutôt qu'à toutes les interfaces du système.

Exemples de syntaxe :

```
-3 2055  
-3 udp://2055  
-3 tcp://2055  
-3 tls://2055  
-3 tls://192.168.2.25:2055
```

- **Mode Proxy**

Dans ce mode, nProbe permet de convertir différents formats NetFlow (v5, v9, sFlow, IPFIX, jFlow) vers NetFlow v5, v9 ou IPFIX. Cela facilite la mise à niveau vers des versions plus récentes du protocole tout en conservant la compatibilité avec les versions précédentes.

```
nprobe -3 2055 -n collector_ip:2055 -V 9
```

Par exemple, il est possible de convertir les flux NetFlow v5 envoyés par un routeur en IPFIX, et inversement. Toutefois, certaines conversions (comme de v9 à v5) peuvent entraîner une perte d'informations de flux.

Dans ce mode, les flux collectés sont convertis en une représentation interne, mis en cache, puis exportés selon les paramètres par défaut ou ceux spécifiés avec -T.

3.3.2 Serveur de Prétraitement

Le serveur de prétraitement dans ce projet joue un rôle essentiel dans la collecte et la transformation des flux NetFlow envoyés par le collecteur nProbe. Ce serveur fonctionne comme un NetFlow collector en utilisant un socket TCP Python. Il reçoit les flux NetFlow générés par nProbe, nettoie et transforme ces données avant de les envoyer dans un pipeline de données via Apache Kafka.

3.3.3 Apache Kafka

- **Introduction**

Apache Kafka est une plateforme de streaming distribuée conçue pour gérer le traitement et la transmission de grands volumes de données en temps réel.

Il est utilisé pour la messagerie entre applications, la gestion des logs, l'analyse en temps réel et d'autres cas d'utilisation nécessitant un transfert rapide et fiable des données. Kafka est conçu pour être scalable, fault-tolerant et hautement performant.

- **Caractéristiques clés d'Apache Kafka**

- **Haute performance**

Kafka est optimisé pour un débit élevé et une faible latence, capable de gérer des millions de messages par seconde.

- **Scalabilité Horizontale**

Kafka s'exécute sous forme de cluster, où les brokers (serveurs Kafka) peuvent être ajoutés pour gérer une charge accrue.

- **Durabilité et Tolérance aux pannes**

Les données sont répliquées sur plusieurs brokers pour éviter la perte de messages en cas de panne d'un noeud.

- **Stockage persistant des messages**

Kafka stocke les messages sur disque et permet de conserver un historique configurable pour un replay ultérieur.

- **Modèle de messagerie Pub/Sub et Queue**

Kafka prend en charge à la fois l'architecture publish/subscribe (plusieurs consommateurs lisent un même message) et le modèle file d'attente (chaque message est traité par un seul consommateur dans un groupe donné).

- **Support natif du streaming**

Kafka Streams permet un traitement des flux en temps réel, transformant les données en mouvement.

- **Composants principaux de Kafka**

- **Broker**

Le Broker est un serveur qui exécute Kafka et gère le stockage des topics et des partitions.

Un cluster Kafka est composé de plusieurs brokers.

Les brokers partagent les topics et répliquent les partitions pour garantir la haute disponibilité.

Un broker peut être désigné leader d'une partition, les autres étant des répliques.

- **Topics**

Un topic est un canal où les messages sont publiés par les producers et lus par les consumers.

Multi-abonnés : plusieurs consommateurs peuvent écouter le même topic.

Kafka stocke les messages d'un topic sous forme d'un log immuable.

Un topic est divisé en plusieurs partitions pour la scalabilité.

- **Partition**

Chaque topic Kafka est divisé en partitions, qui sont des sous-ensembles du topic, permettant une distribution des données sur plusieurs brokers.

Conception

Kafka stocke les messages dans des partitions.

Chaque message dans une partition a un offset unique.

Les partitions permettent d'augmenter le débit en parallèle

- **Producer**

Le Producer est responsable de l'envoi des messages à Kafka. Il publie les données dans un topic spécifique et Kafka garantit que ces données sont stockées de manière fiable et distribuée.

Le Producer choisit un topic et envoie un message.

Kafka stocke ce message dans un partition du topic.

Kafka garantit la répartition des messages selon la clé du message (si spécifiée) ou en mode aléatoire.

- **Consumer**

Le Consumer est responsable de la lecture des messages à partir d'un topic Kafka. Il s'abonne à un topic et récupère les messages stockés.

Un Consumer peut appartenir à un Consumer Group.

Kafka garantit que chaque message d'une partition est lu par un seul Consumer dans un Consumer Group.

Un Consumer peut choisir de lire les messages en temps réel ou de traiter des données en différé

3.3.4 Apache Druid

- Introduction**

Apache Druid est une base de données d'analytique en temps réel conçue pour exécuter des requêtes OLAP rapides sur de grands ensembles de données. Il est particulièrement adapté aux cas d'utilisation nécessitant une ingestion en temps réel, des performances de requêtes rapides et une haute disponibilité.

- Quand utiliser Apache Druid ?**

Druid est un bon choix si votre cas d'utilisation correspond à plusieurs des critères suivants :

- 1) Taux d'insertion élevé, mais peu de mises à jour
- 2) Requêtes d'agrégation et de reporting fréquentes
- 3) Faible latence des requêtes (100ms à quelques secondes)
- 4) Données avec une composante temporelle
- 5) Analyse de colonnes à haute cardinalité
- 6) Ingestion de données depuis Kafka, HDFS, S3

- Architecture de Druid**

Druid est conçu comme une architecture distribuée et scalable, divisée en plusieurs services indépendants qui assurent différentes fonctions. Cette modularité permet une haute tolérance aux pannes et une gestion flexible du cluster.

Conception

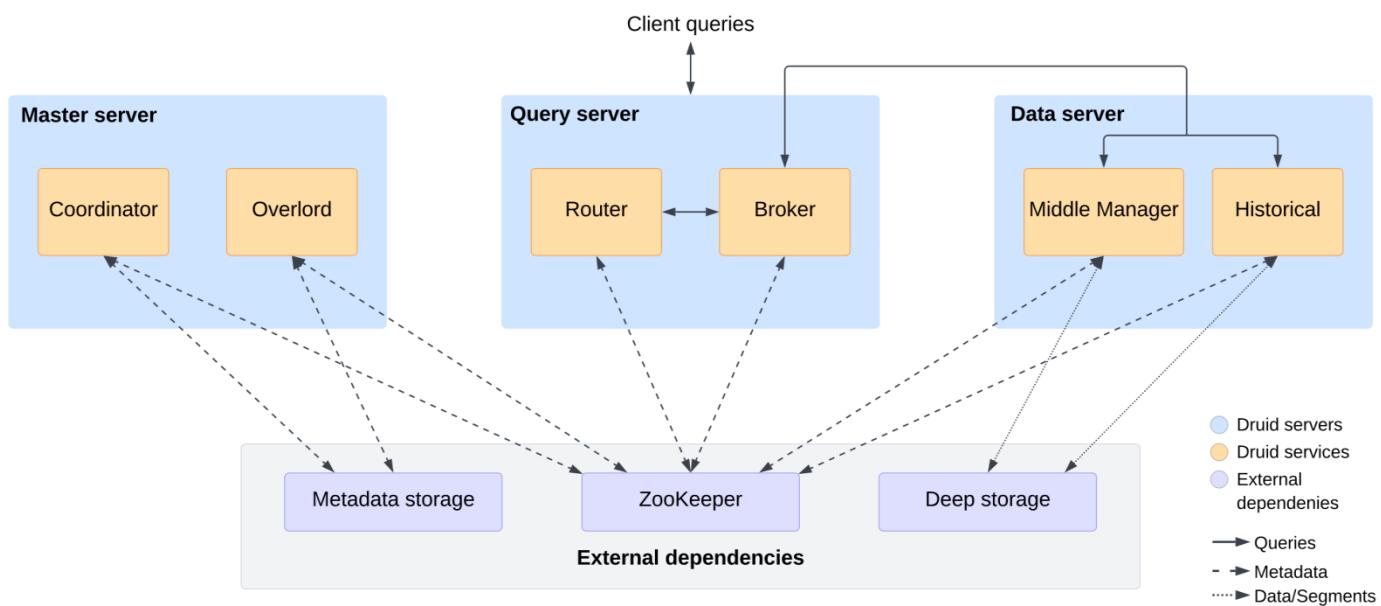


FIG. 3.2 : Architecture apache druid

Druid est structuré autour de trois types de serveurs :

- Serveurs Master (gestion de l'ingestion et de la disponibilité des données)
- Serveurs Query (exécution et routage des requêtes)
- Serveurs Data (stockage et ingestion des données)

- **Serveurs Master**

1) Coordinator Service

- Supervise les services Historicals (qui stockent les données).
- Gère l'équilibrage des segments et leur assignation aux serveurs.
- Charge les nouveaux segments et supprime ceux qui sont obsolètes.

Conception

- Assure la réPLICATION des segments sur plusieurs nœuds historiques.
- Maintient un équilibre de charge entre les nœuds en déplaçant les segments.
- dépose des informations temporaires dans ZooKeeper pour signaler un nouveau segment à charger.

2) Overlord Service

- Responsable de la gestion des tâches d'ingestion.
- Assigne les tâches d'ingestion aux Middle Managers.
- **Query Servers**

1) Broker Service

- Reçoit les requêtes des utilisateurs et les distribue aux serveurs de données.
- Agrège les résultats des sous-requêtes et les renvoie au client.
- Interprète les métadonnées publiées dans ZooKeeper pour optimiser le routage des requêtes.

2) Router Service

- Distribue les requêtes entre plusieurs Brokers.
- Permet d'isoler les requêtes en fonction de l'importance des données (par exemple, récents vs archives).
- Indispensable pour les clusters contenant des téraoctets de données.

- **Data Servers**

1) Historical Service

- Stocke les données historiques et répond aux requêtes.
- Met en cache les segments de données sur disque local et en mémoire pour accélérer les requêtes.
- Ne permet pas d'écriture ou de modification des données.

2) Middle Manager Service

- Responsable de l'ingestion des nouvelles données.
- Lit les données depuis des sources externes comme Kafka, HDFS ou S3.
- Publie les segments de données dans le cluster.

- **Stockage**

Druid stocke ses données sous forme de "datasources", similaires aux tables d'une base de données relationnelle. Chaque datasource est partitionnée par le temps et, éventuellement, par d'autres attributs.

Chaque intervalle de temps est appelé un "chunk" (par exemple, une journée si la partition est faite par jour). Chaque chunk contient un ou plusieurs segments, qui sont des fichiers autonomes pouvant contenir jusqu'à plusieurs millions de lignes de données

Conception

 Segment

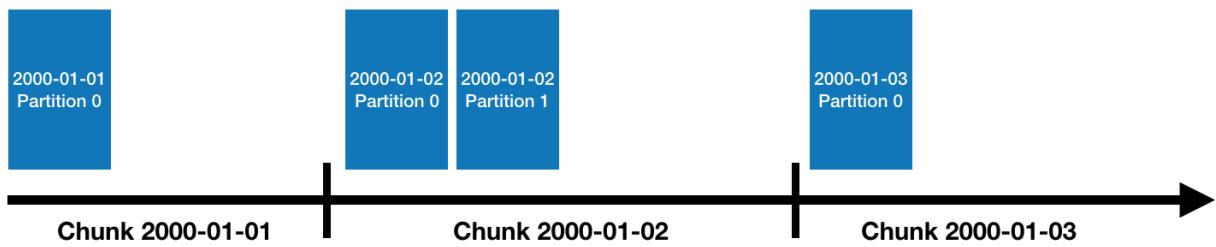


Fig. 3.3 : Storage overview

Les segments sont créés initialement comme des fichiers mutables et non validés par un Middle Manager. Ils sont immédiatement requêtables dès qu'ils sont ajoutés à un segment non validé. Le processus de construction des segments optimise les requêtes futures en produisant un fichier compact et indexé à l'aide de plusieurs techniques, telles que la conversion en format colonne, l'indexation avec des index bitmap, la compression, et l'encodage par dictionnaire.

Une fois validés, les segments sont publiés dans le deep storage, deviennent immuables et sont transférés des Middle Managers aux Historical services.

Le Coordinator utilise la base de métadonnées pour savoir quels segments sont disponibles et doivent être chargés.

- **Segments**

Apache Druid stocke ses données et indexations dans des fichiers segmentés par le temps. Chaque intervalle de segment peut contenir des données, et si un intervalle est vide, aucun segment n'est créé.

Druid peut générer plusieurs segments pour le même intervalle si différentes tâches d'ingestion sont exécutées pour cette période.

Les segments sont optimisés pour la performance et leur taille recommandée est comprise entre 300 et 700 Mo. Si un segment dépasse cette plage, il est possible de modifier la granularité de partitionnement ou d'ajuster la spécification des partitions pour limiter le nombre de lignes par segment.

Conception

Les fichiers de segment sont stockés sous un format de colonnes distinctes pour réduire la latence des requêtes en scannant uniquement les colonnes nécessaires.

Trois types de colonnes existent : timestamp, dimensions et métriques. L'organisation de ces colonnes permet une compression efficace et un traitement rapide des requêtes.

Timestamp		Dimensions				Metrics	
Timestamp		Page	Username	Gender	City	Characters Added	Characters Removed
2011-01-01T01:00:00Z		Justin Bieber	Boxer	Male	San Francisco	1800	25
2011-01-01T01:00:00Z		Justin Bieber	Reach	Male	Waterloo	2912	42
2011-01-01T02:00:00Z		Ke\$ha	Helz	Male	Calgary	1953	17
2011-01-01T02:00:00Z		Ke\$ha	Xeno	Male	Taiyuan	3194	170

Fig. 3.3 : Segment file structure

• Deep Storage

Le deep storage est l'endroit où les segments sont stockés de manière persistante.

Druid ne fournit pas cette infrastructure de stockage ; elle doit être mise en place séparément. Tant que les processus Druid ont accès au deep storage, les données ne seront pas perdues, quel que soit le nombre de nœuds en panne.

Druid supporte plusieurs options de deep storage, dont :

- Stockage local : recommandé uniquement pour un serveur unique ou un stockage réseau partagé.
- Amazon S3 et compatibles : solution évolutive et robuste.
- Google Cloud Storage : utilisé dans les infrastructures Google Cloud.
- Azure Blob Storage : adapté aux environnements Microsoft Azure.
- HDFS : utilisé dans les déploiements Hadoop.

3.3.5 Apache Superset

Apache Superset est une plateforme moderne d'exploration et de visualisation des données. Il peut remplacer ou compléter les outils propriétaires de business intelligence en offrant une alternative open-source flexible et puissante. Superset s'intègre avec de nombreuses sources de données et offre plusieurs fonctionnalités clés :

- Une interface no-code pour créer des graphiques rapidement.
- Un éditeur SQL web avancé pour des requêtes complexes.
- Une couche sémantique légère pour définir rapidement des dimensions et métriques personnalisées.
- Une compatibilité avec presque toutes les bases de données SQL.
- Une large gamme de visualisations, allant des graphiques simples aux cartes géospatiales.
- Une couche de cache configurable pour réduire la charge sur les bases de données.
- Un système de sécurité extensible avec gestion des rôles et options d'authentification avancées.
- Une API permettant une personnalisation et une automatisation.
- Une architecture cloud-native conçue pour l'évolutivité.

Implémentation

4.1 Installation et Configuration de l'Environnement

4.1.1 Installation de VMware Workstation

VMware Workstation est un hyperviseur qui permet de créer et de gérer des machines virtuelles sur un ordinateur hôte. Il est utilisé pour exécuter plusieurs systèmes d'exploitation simultanément, facilitant ainsi le développement et les tests dans des environnements isolés.

- a) Se rendre sur le site officiel de VMware : <https://www.vmware.com>
- b) Naviguer vers la section des produits et sélectionner VMware Workstation Pro ou VMware Workstation Player (version gratuite pour un usage non commercial).
- c) Télécharger l'installateur correspondant à Windows
- d) Exécuter l'installateur téléchargé (.exe).
- e) Choisir les options d'installation par défaut ou les personnaliser selon les besoins.
- f) Cliquer sur Installer et attendre la fin du processus
- g) Une fois l'installation terminée, redémarrer l'ordinateur si nécessaire
- h) Lancer VMware Workstation et entrer une clé de licence si vous utilisez la version Pro.

4.1.2 Installation d'Ubuntu

Téléchargez l'image ISO d'Ubuntu depuis le site officiel : <https://ubuntu.com/download/desktop>
Création une nouvelle VM dans VMware Workstation.

Attribution des ressources :

- Stockage : Allouez un espace disque suffisant (minimum 50 Go recommandé).
- Mémoire RAM : 8 Go recommandés pour de meilleures performances.
- Processeur : Minimum 2 cœurs, idéalement 4 cœurs pour des performances optimales.

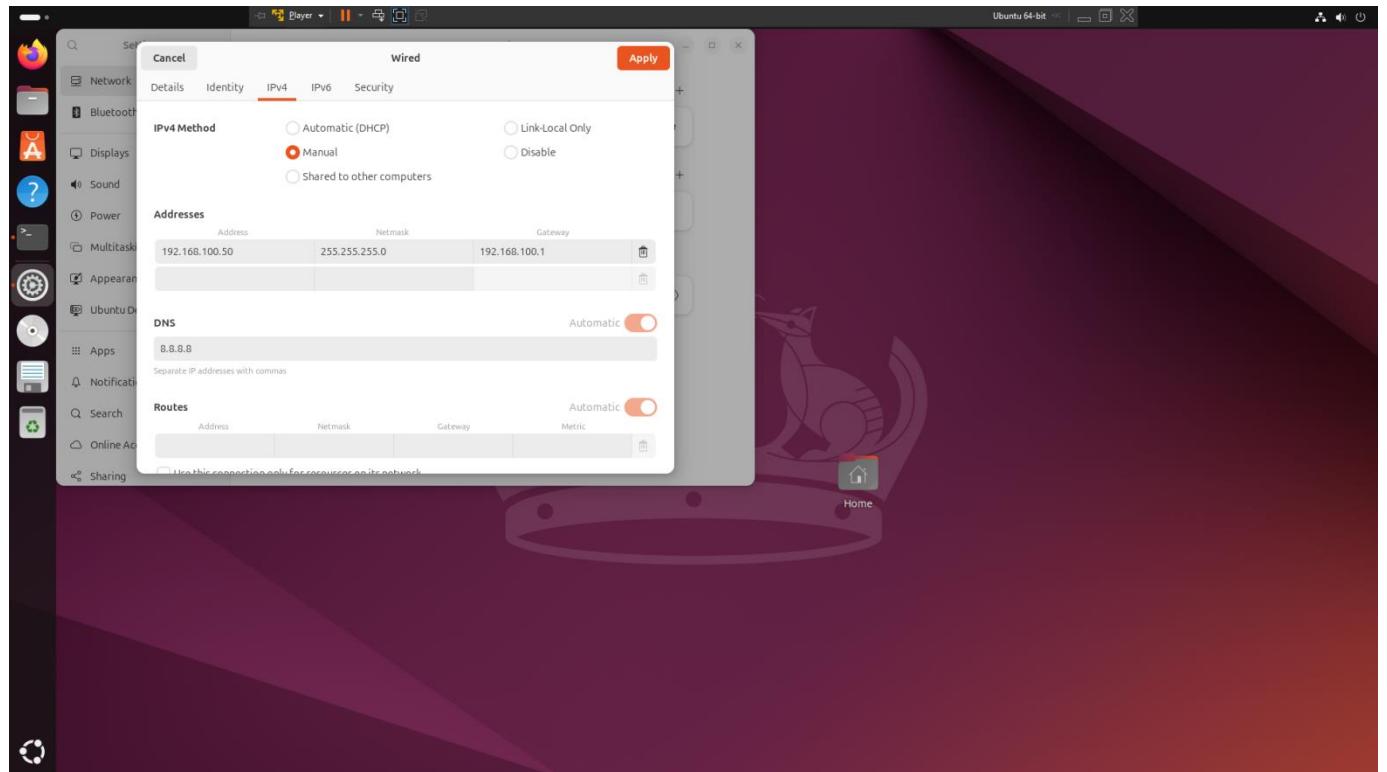
Configurez l'interface réseau en mode "Bridge" afin que la machine virtuelle obtienne une adresse IP dans le même réseau que l'hôte.

Montez l'ISO d'Ubuntu et démarrez la VM.

Suivez les instructions d'installation en choisissant les options par défaut.

Configurez un utilisateur administrateur et un mot de passe sécurisé.

Après avoir installé Ubuntu , vous devez configurer l'adresse IP de la machine virtuelle et vous assurer qu'elle peut se connecter à Internet et aux machines dans le même réseaux

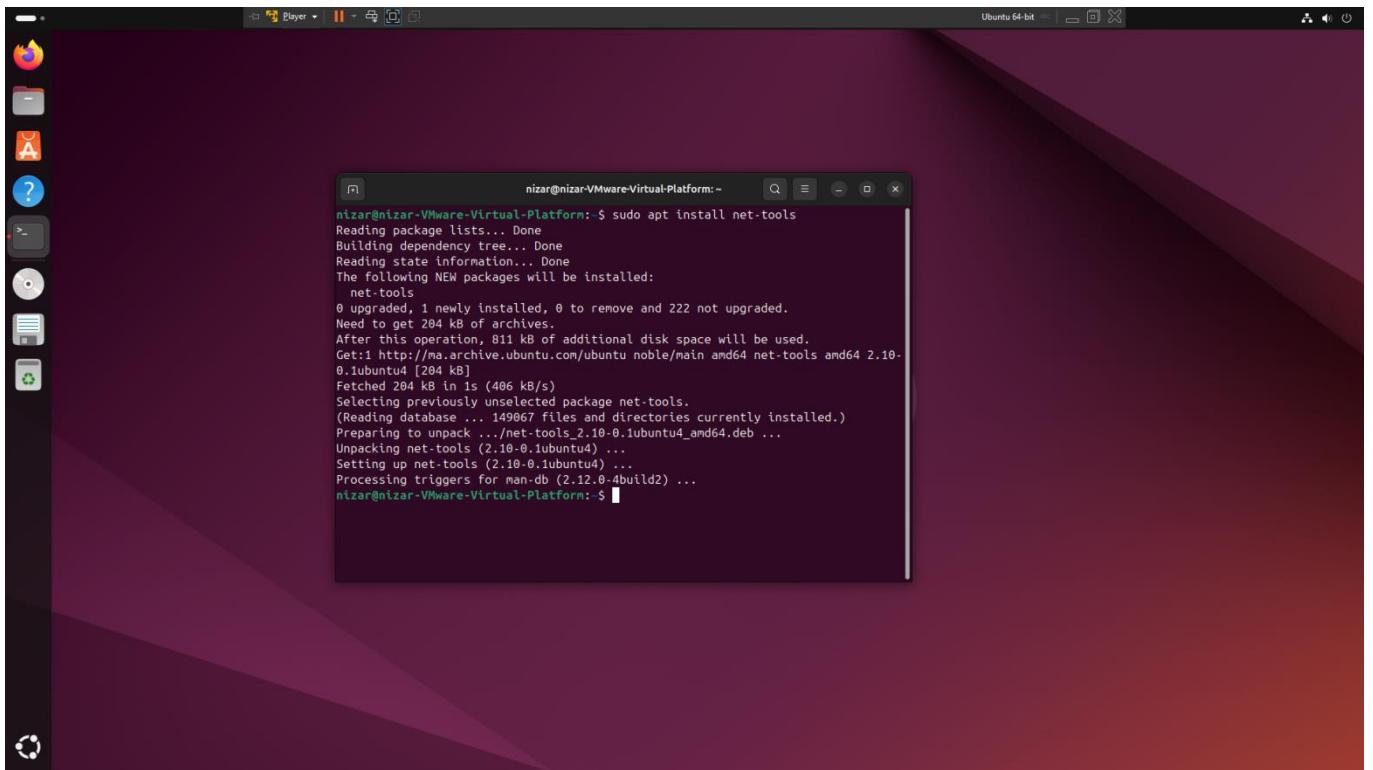


Testez si la machine virtuelle peut atteindre l'hôte (la machine physique). Utilisez la commande ping :
ping 192.168.1.1

vérifiez si la machine virtuelle peut accéder à Internet. Vous pouvez tester cela en pingant un serveur public, par exemple Google :

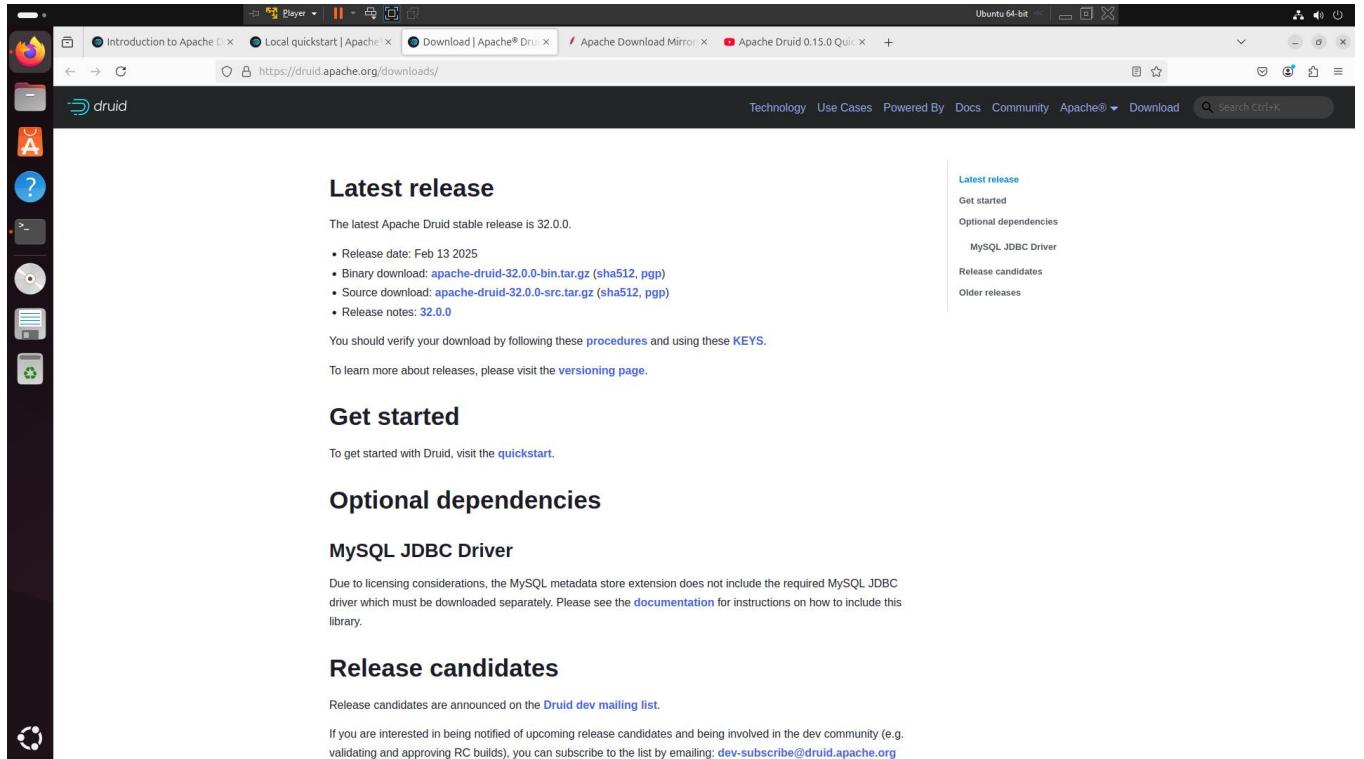
ping 8.8.8.8

installez le paquet net-tools, qui inclut des outils comme ifconfig et netstat, utiles pour vérifier la configuration réseau



4.2 Installation et Configuration d'Apache Druid sur Ubuntu

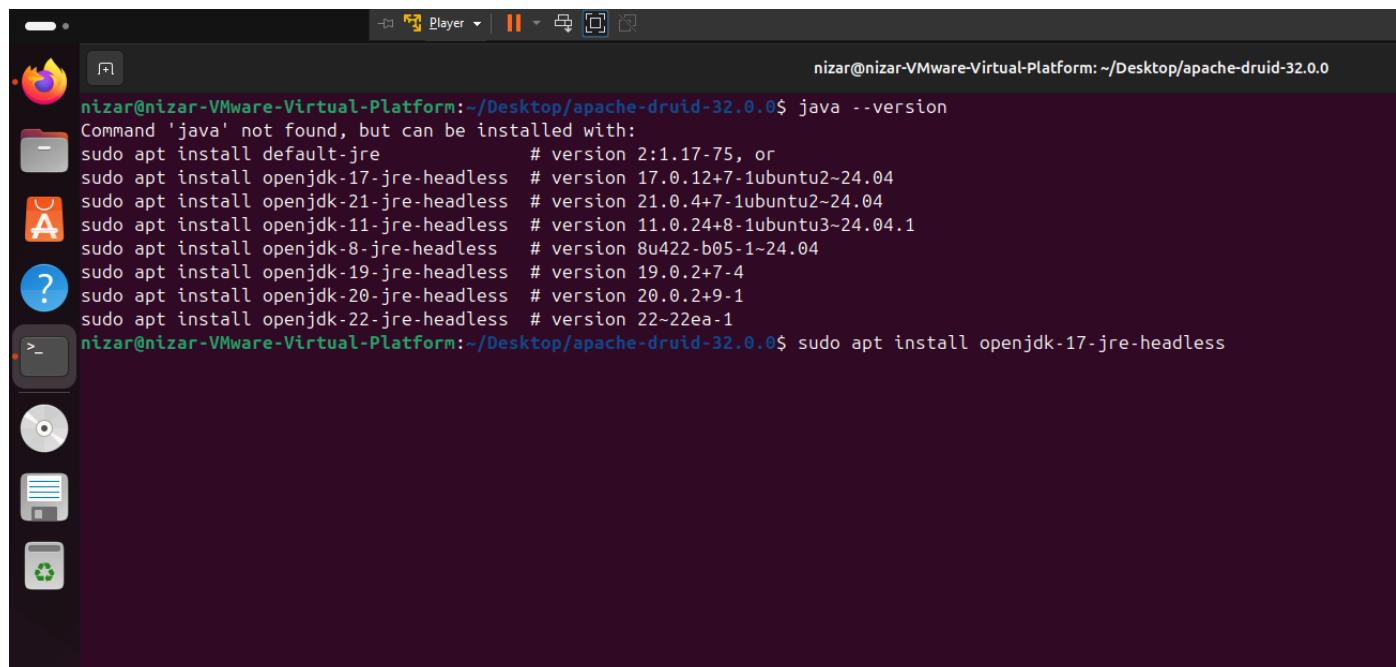
téléchargez la dernière version stable du binaire (<https://druid.apache.org/downloads/>)



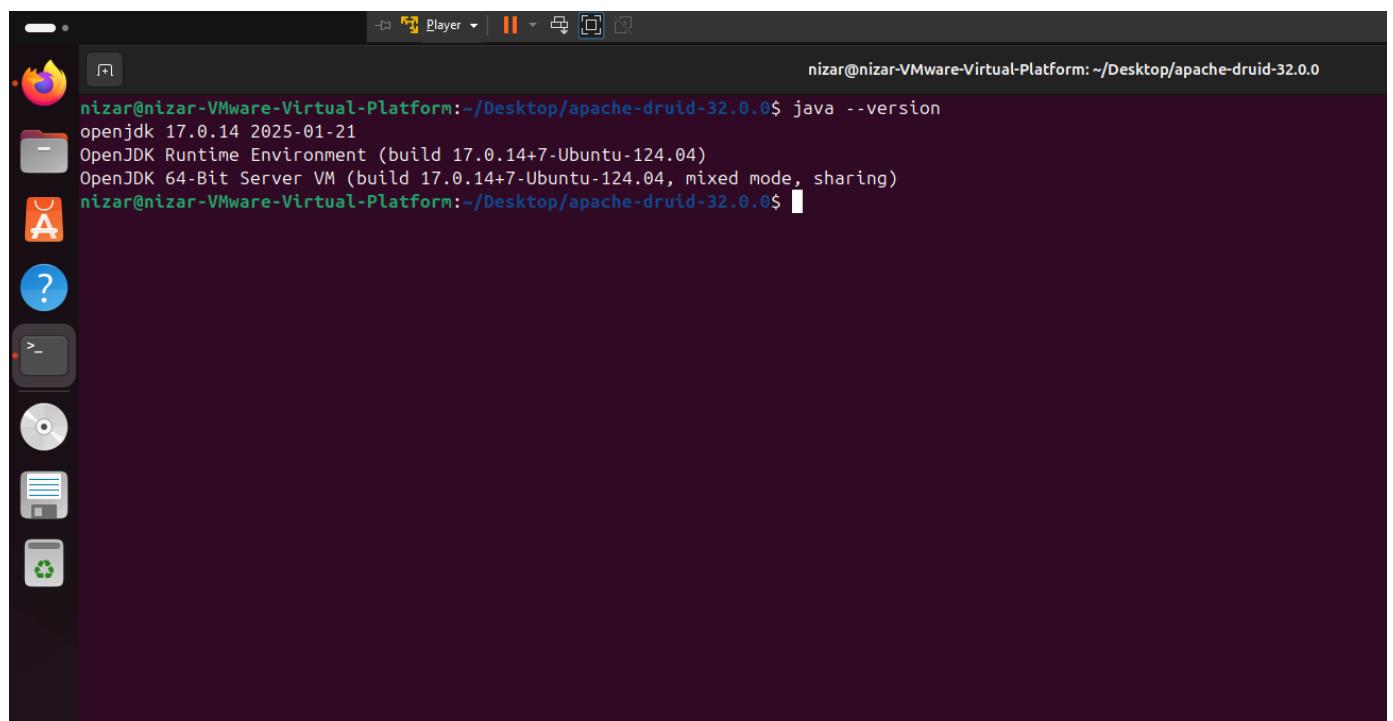
Une fois le fichier téléchargé, extrayez-le et changez de répertoire pour accéder au dossier d'installation

A screenshot of a terminal window on a Linux desktop environment. The terminal window title is "nizar@nizar-VMware-Virtual-Platform: ~/Desktop/apache-druid-32.0.0". The user has run several commands to extract the contents of the downloaded binary tarball: "ls" shows the file "apache-druid-32.0.0-bin.tar.gz"; "tar -xzf apache-druid-32.0.0-bin.tar.gz" extracts the archive; and "cd apache-druid-32.0.0/" changes the directory to the extracted root. The terminal shows the contents of the directory, including "bin", "extensions", "lib", "licenses", "quickstart", "conf", "hadoop-dependencies", "LICENSE", "NOTICE", and "README".

Avant de lancer le serveur Druid, vous devez installer Java 17.



nizar@nizar-VMware-Virtual-Platform:~/Desktop/apache-druid-32.0.0\$ java --version
Command 'java' not found, but can be installed with:
sudo apt install default-jre # version 2:1.17-75, or
sudo apt install openjdk-17-jre-headless # version 17.0.12+7-1ubuntu2~24.04
sudo apt install openjdk-21-jre-headless # version 21.0.4+7-1ubuntu2~24.04
sudo apt install openjdk-11-jre-headless # version 11.0.24+8-1ubuntu3~24.04.1
sudo apt install openjdk-8-jre-headless # version 8u422-b05-1~24.04
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-4
sudo apt install openjdk-20-jre-headless # version 20.0.2+9-1
sudo apt install openjdk-22-jre-headless # version 22~22ea-1
nizar@nizar-VMware-Virtual-Platform:~/Desktop/apache-druid-32.0.0\$ sudo apt install openjdk-17-jre-headless

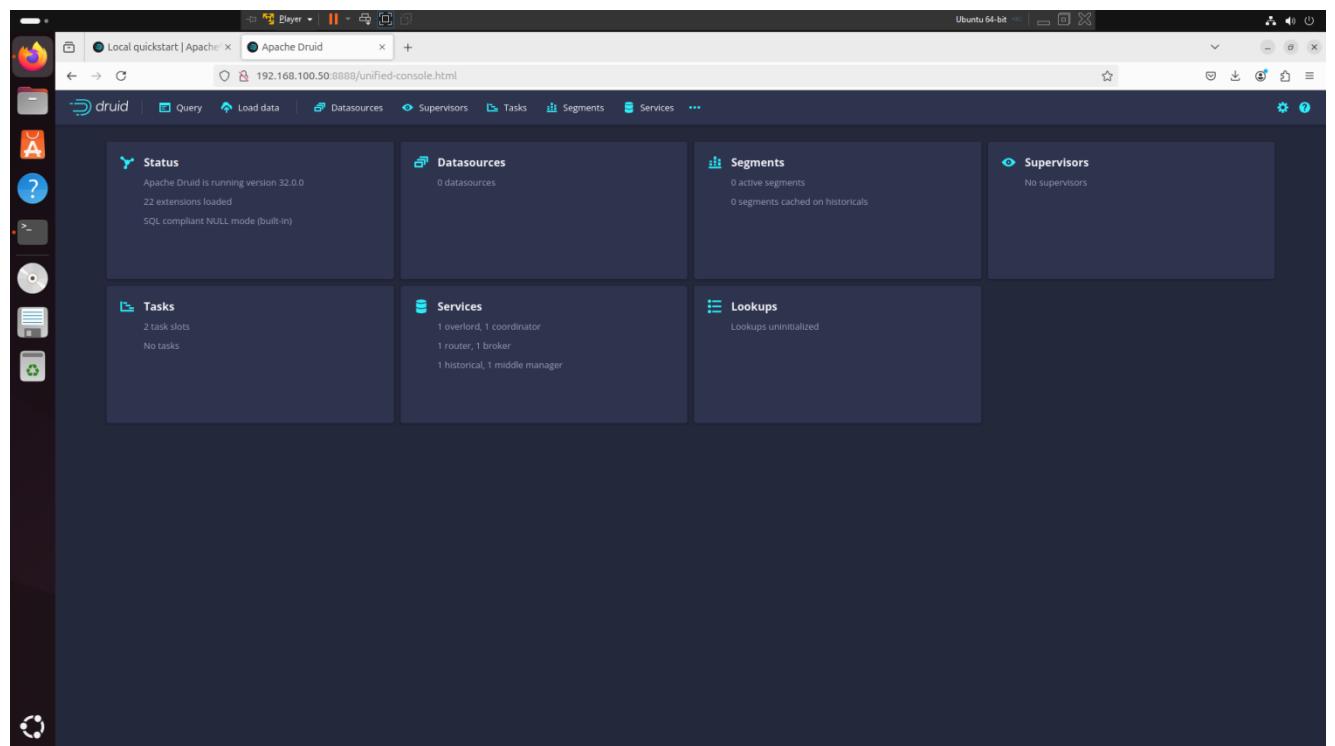


nizar@nizar-VMware-Virtual-Platform:~/Desktop/apache-druid-32.0.0\$ java --version
openjdk 17.0.14 2025-01-21
OpenJDK Runtime Environment (build 17.0.14+7-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 17.0.14+7-Ubuntu-124.04, mixed mode, sharing)
nizar@nizar-VMware-Virtual-Platform:~/Desktop/apache-druid-32.0.0\$ █

Une fois Java installé et configuré, vous pouvez démarrer le serveur Apache Druid. Pour ce faire, exécutez la commande suivante

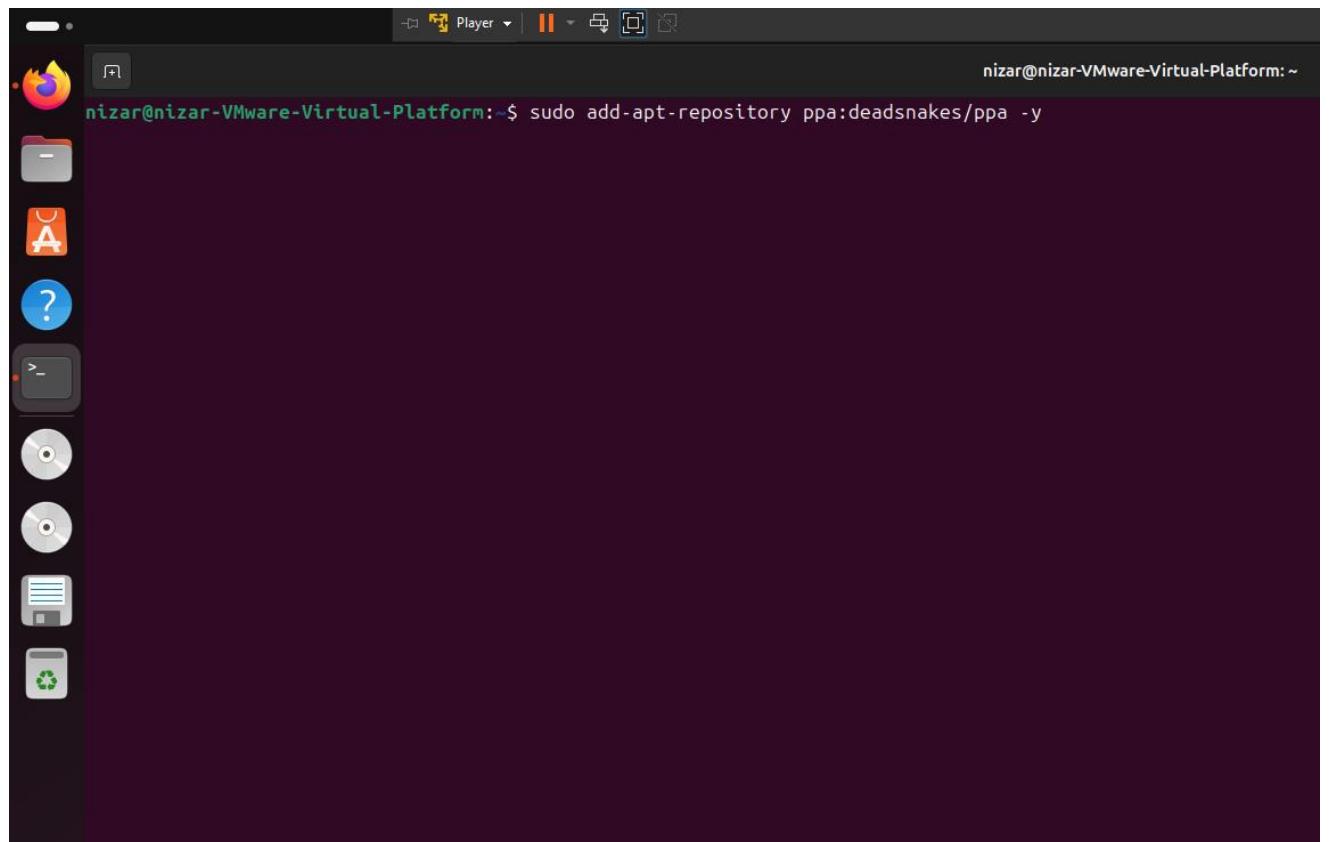
```
nizar@nizar-VMware-Virtual-Platform:~/Desktop/apache-druid-32.0.0$ ./bin/start-druid
[Thu Mar 13 17:48:02 2025] Starting Apache Druid.
[Thu Mar 13 17:48:02 2025] Open http://localhost:8888/ or http://nizar-VMware-Virtual-Platform:8888/ in your browser to access the web console.
[Thu Mar 13 17:48:02 2025] Or, if you have enabled TLS, use https on port 9088.
[Thu Mar 13 17:48:02 2025] Starting services with log directory [/home/nizar/Desktop/apache-druid-32.0.0/log].
[Thu Mar 13 17:48:02 2025] Running command[zk]: bin/run-zk conf
[Thu Mar 13 17:48:02 2025] Running command[broker]: bin/run-druid broker /home/nizar/Desktop/apache-druid-32.0.0/conf/druid/auto '-Xms852m -Xmx852m -XX:MaxDirectMemorySize=568m'
[Thu Mar 13 17:48:02 2025] Running command[coordinator-overlord]: bin/run-druid coordinator-overlord /home/nizar/Desktop/apache-druid-32.0.0/conf/druid/auto '-Xms926m -Xmx926m'
[Thu Mar 13 17:48:02 2025] Starting services with log directory [/home/nizar/Desktop/apache-druid-32.0.0/log].
[Thu Mar 13 17:48:02 2025] Running command[druidManager]: bin/run-druid middleManager /home/nizar/Desktop/apache-druid-32.0.0/conf/druid/auto '-Xms64m -Xmx64m' '-Druid.worker.capacity=2 -Druid.indexer.runner.javaOptsArray=[{"server": "-Duser.timezone=UTF-8", "file.encoding=UTF-8", "-XX:+ExitOnOutOfMemoryError", "-Djava.util.logging.manager=org.apache.logging.log4j.Slf4jLoggerManager", "-Xms256m", "-Xmx256m", "-XX:MaxDirectMemorySize=256m"}]
[Thu Mar 13 17:48:02 2025] Running command[router]: bin/run-druid router /home/nizar/Desktop/apache-druid-32.0.0/conf/druid/auto '-Xms256m -Xmx256m -XX:MaxDirectMemorySize=128m'
[Thu Mar 13 17:48:02 2025] Running command[historical]: bin/run-druid historical /home/nizar/Desktop/apache-druid-32.0.0/conf/druid/auto '-Xms988m -Xmx988m -XX:MaxDirectMemorySize=1482m'
```

Après avoir exécuté la commande, vous pouvez vérifier si les services Druid sont en cours d'exécution en accédant à l'interface web de Druid

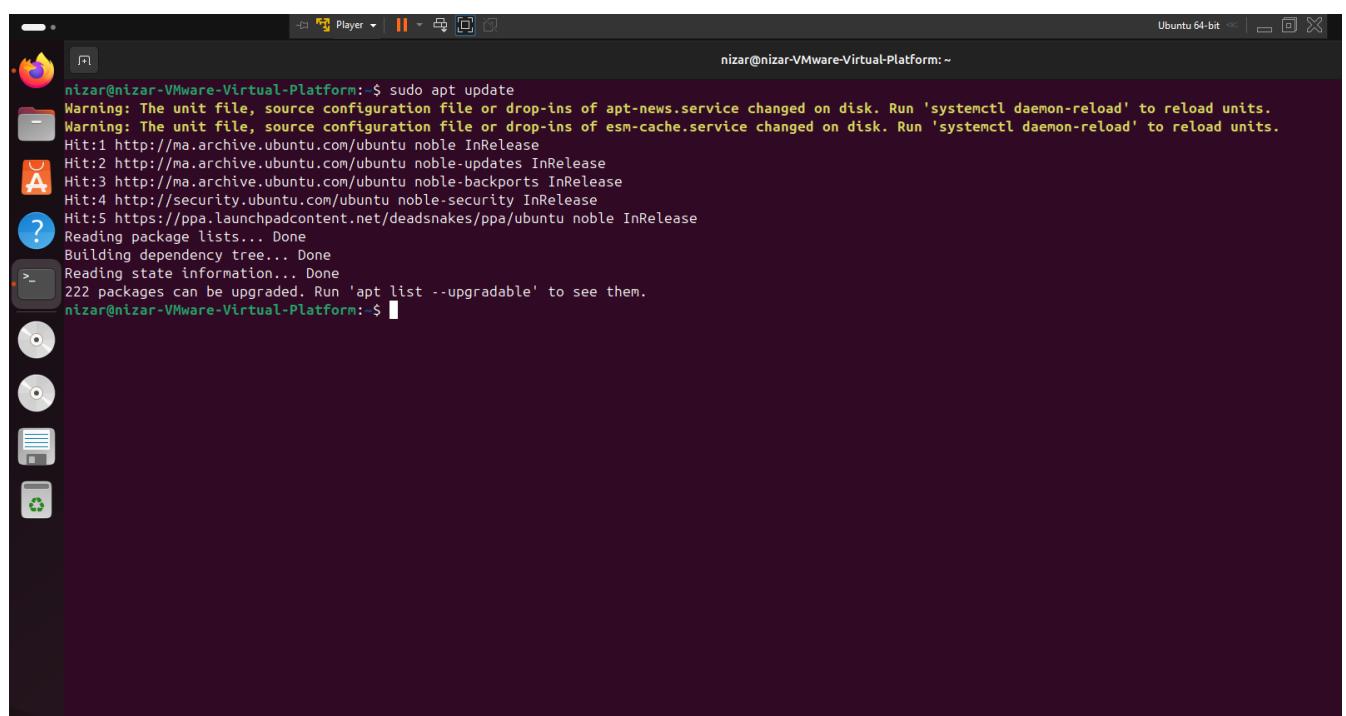


4.3 Installation et Configuration d'Apache Superset sur Ubuntu

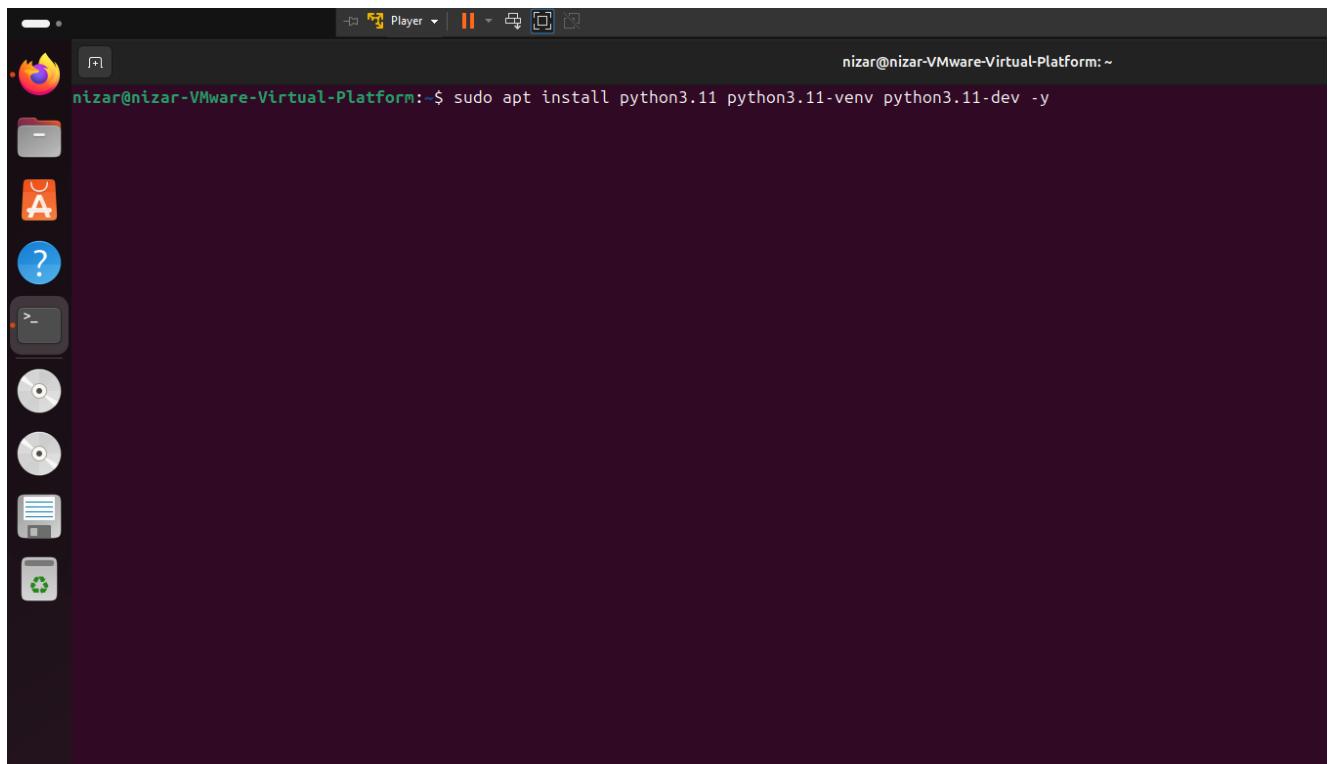
Apache Superset nécessite la version 3.11 de Python, nous devons ajouter le PPA deadsnakes, qui contient de différentes versions de Python.



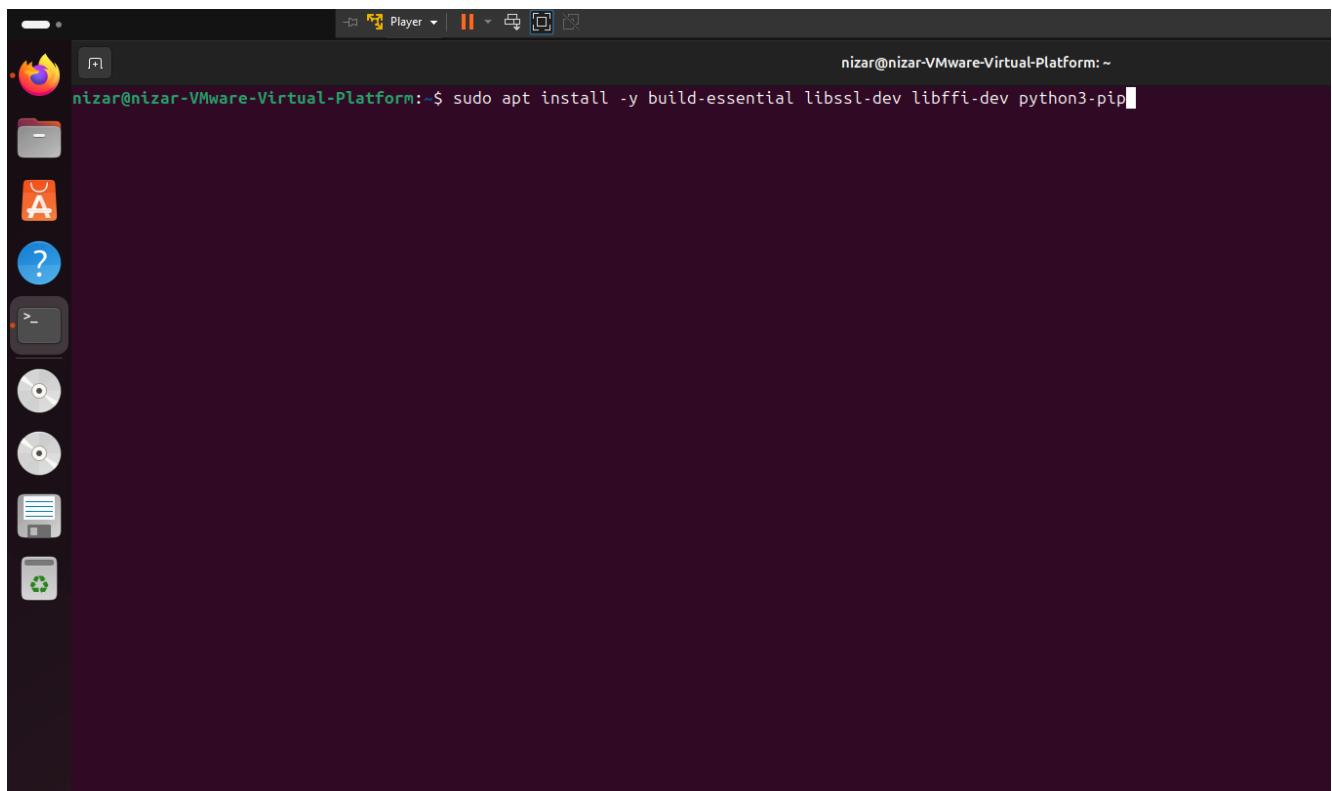
```
nizar@nizar-VMware-Virtual-Platform:~$ sudo add-apt-repository ppa:deadsnakes/ppa -y
```



```
nizar@nizar-VMware-Virtual-Platform:~$ sudo apt update
Warning: The unit file, source configuration file or drop-ins of apt-news.service changed on disk. Run 'systemctl daemon-reload' to reload units.
Warning: The unit file, source configuration file or drop-ins of esm-cache.service changed on disk. Run 'systemctl daemon-reload' to reload units.
Hit:1 http://ma.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ma.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ma.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu noble InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
222 packages can be upgraded. Run 'apt list --upgradable' to see them.
nizar@nizar-VMware-Virtual-Platform:~$
```

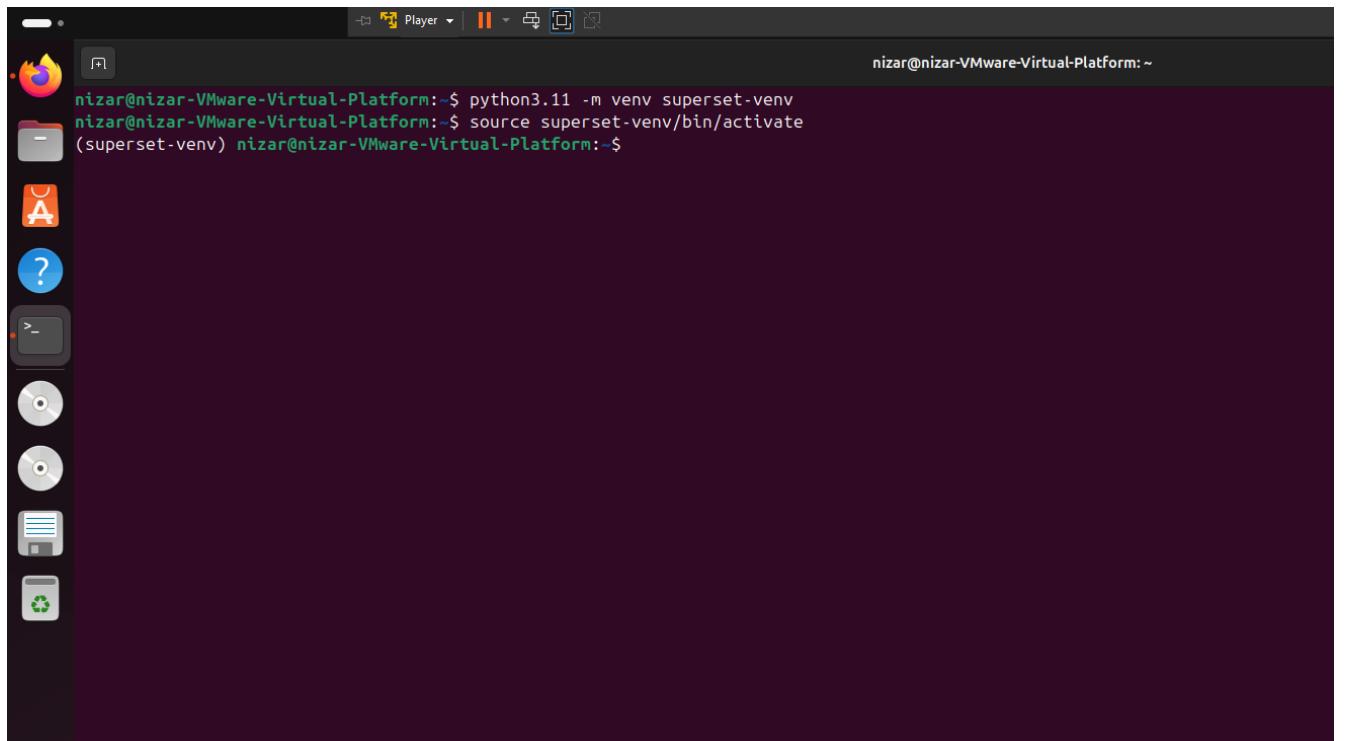


nizar@nizar-VMware-Virtual-Platform:~\$ sudo apt install python3.11 python3.11-venv python3.11-dev -y

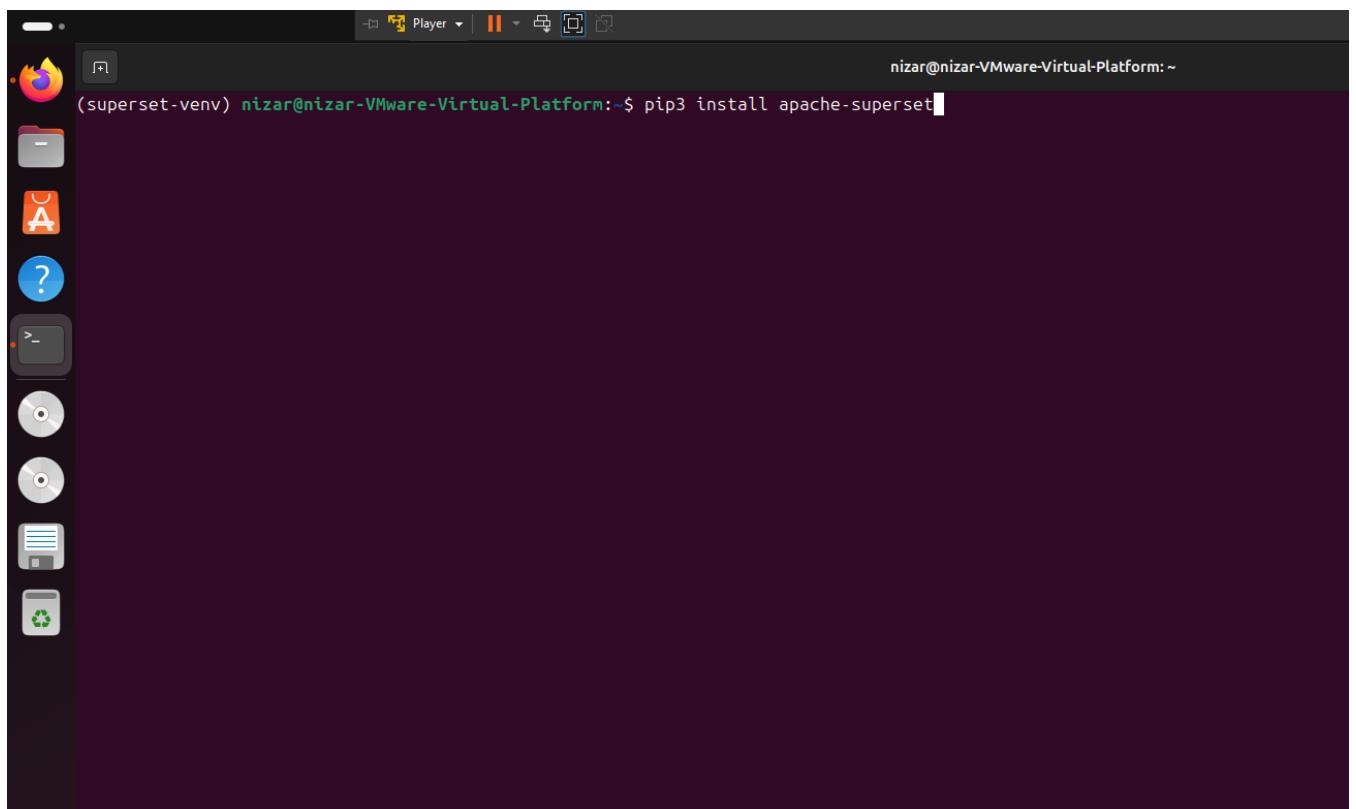


nizar@nizar-VMware-Virtual-Platform:~\$ sudo apt install -y build-essential libssl-dev libffi-dev python3-pip

Après avoir installé Python 3.11, nous allons créer un environnement virtuel (venv) afin d'isoler l'installation de Superset des autres packages système.



```
nizar@nizar-VMware-Virtual-Platform:~$ python3.11 -m venv superset-venv
nizar@nizar-VMware-Virtual-Platform:~$ source superset-venv/bin/activate
(superset-venv) nizar@nizar-VMware-Virtual-Platform:~$
```

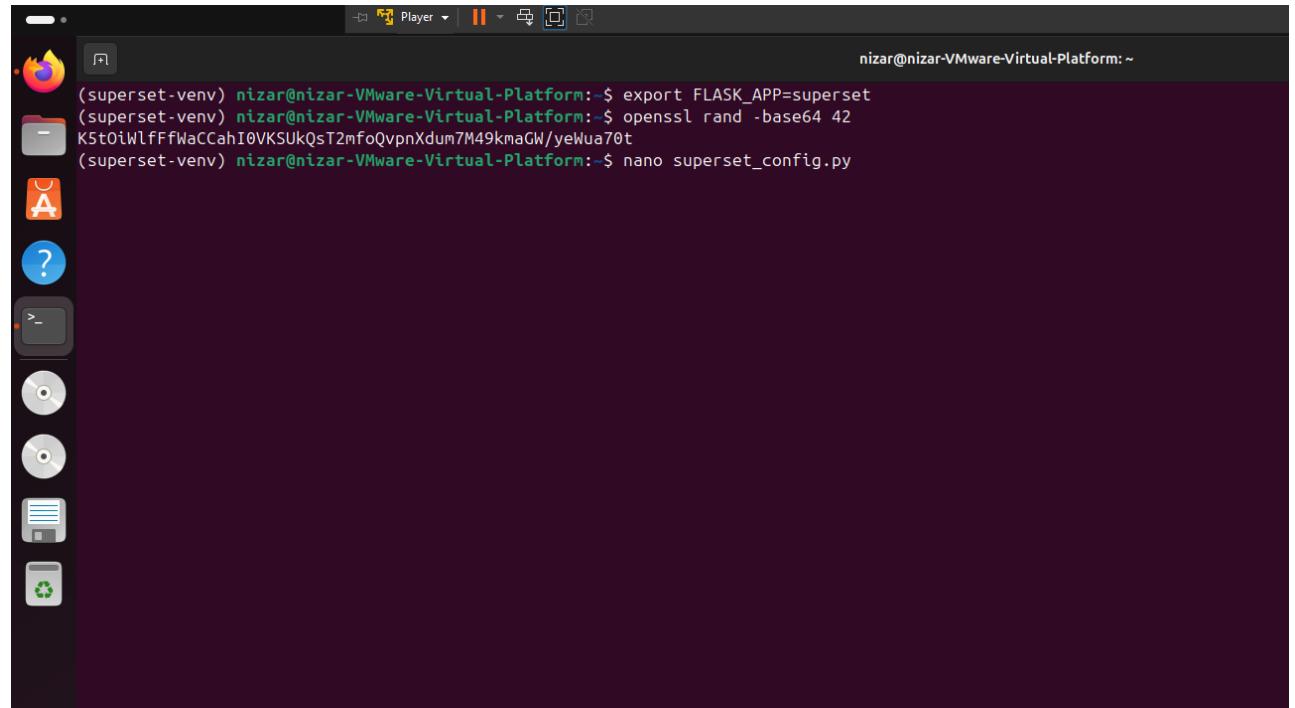


```
(superset-venv) nizar@nizar-VMware-Virtual-Platform:~$ pip3 install apache-superset
```

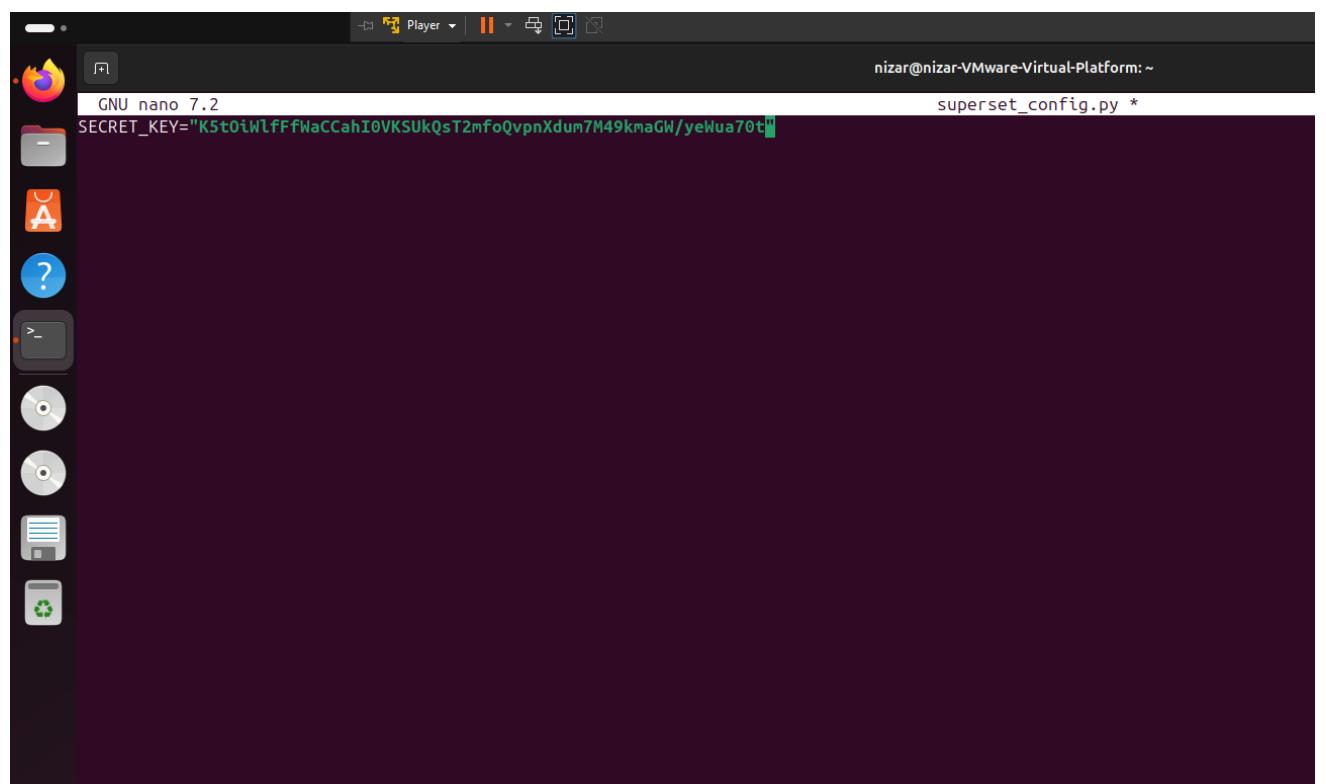
Après avoir activé l'environnement virtuel , nous devons configurer Superset avant son utilisation.

Superset est basé sur Flask, donc nous devons définir la variable d'environnement FLASK_APP

Superset nécessite une SECRET_KEY pour signer les cookies et sécuriser les sessions utilisateur. Nous allons la générer avec OpenSSL et l'ajouter dans le répertoire d'installation de Superset, dans un fichier superset_config.py

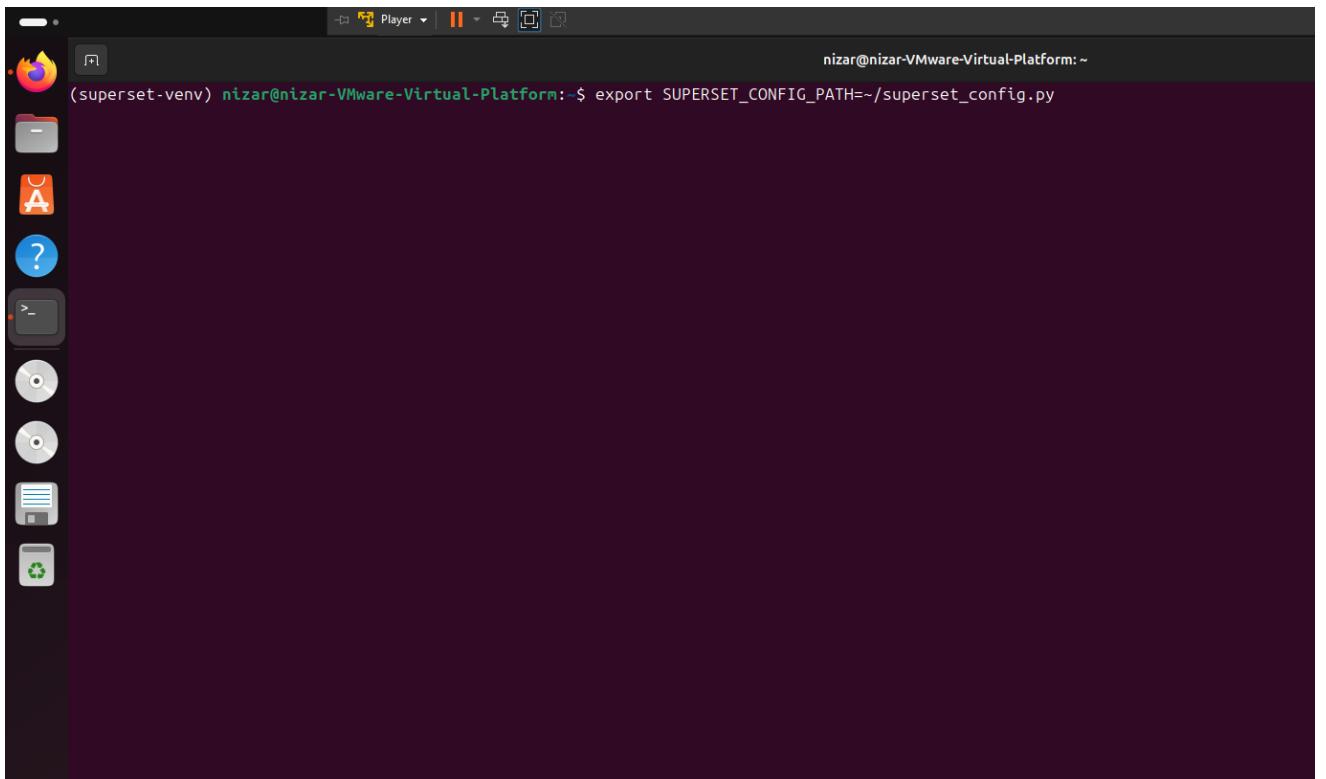


```
nizar@nizar-VMware-Virtual-Platform:~$ export FLASK_APP=superset
(nutzer-venv) nizar@nizar-VMware-Virtual-Platform:~$ openssl rand -base64 42
K5t0iWlFFFWaCCahI0VKSUkQsT2mfoQvpnXduM7M49kmaGW/yeWua70t
(nutzer-venv) nizar@nizar-VMware-Virtual-Platform:~$ nano superset_config.py
```



```
GNU nano 7.2
SECRET_KEY="K5t0iWlFFFWaCCahI0VKSUkQsT2mfoQvpnXduM7M49kmaGW/yeWua70t"
```

Superset doit connaître l'emplacement du fichier de configuration. Nous l'indiquons en définissant la variable d'environnement SUPERSET_CONFIG_PATH

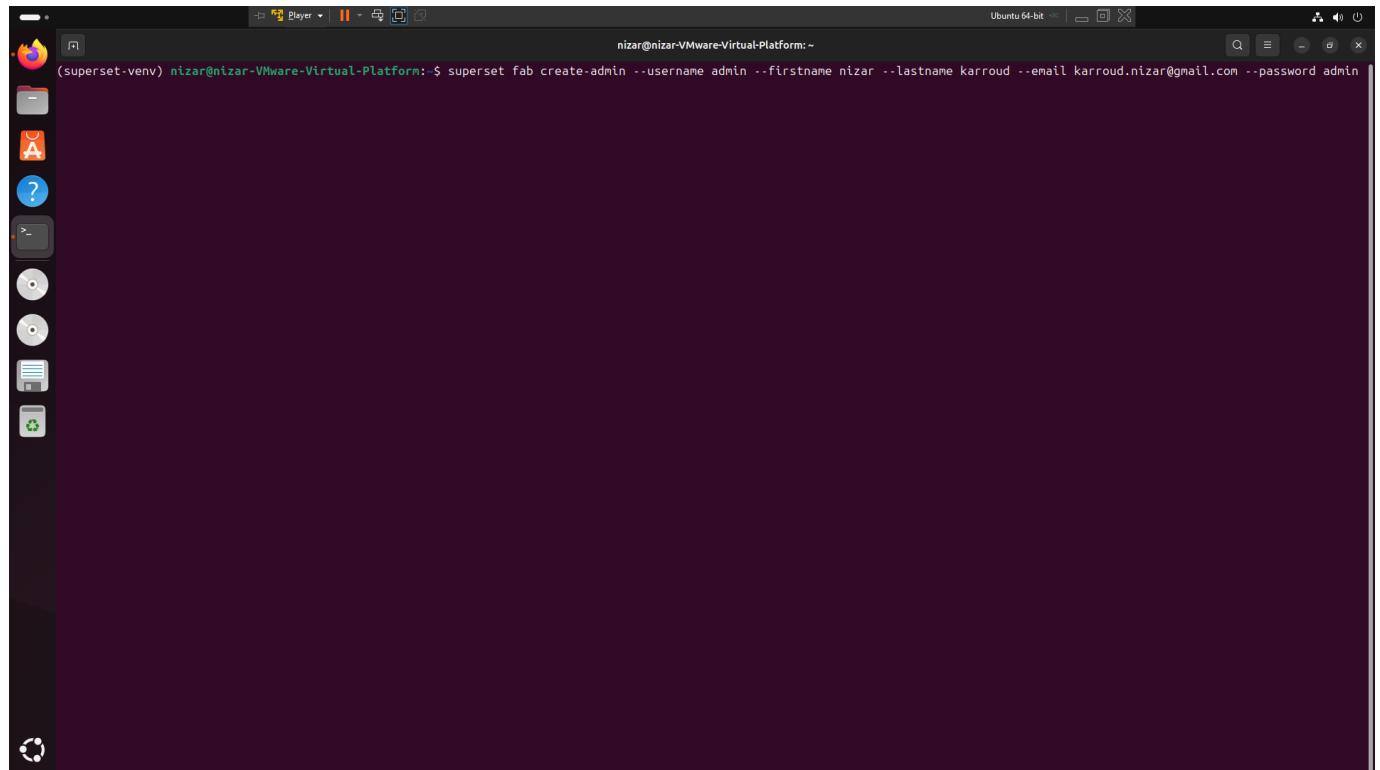


Après avoir configuré les variables d'environnement, nous allons finaliser l'installation et démarrer Superset.

Exécutez la commande suivante pour appliquer les migrations nécessaires

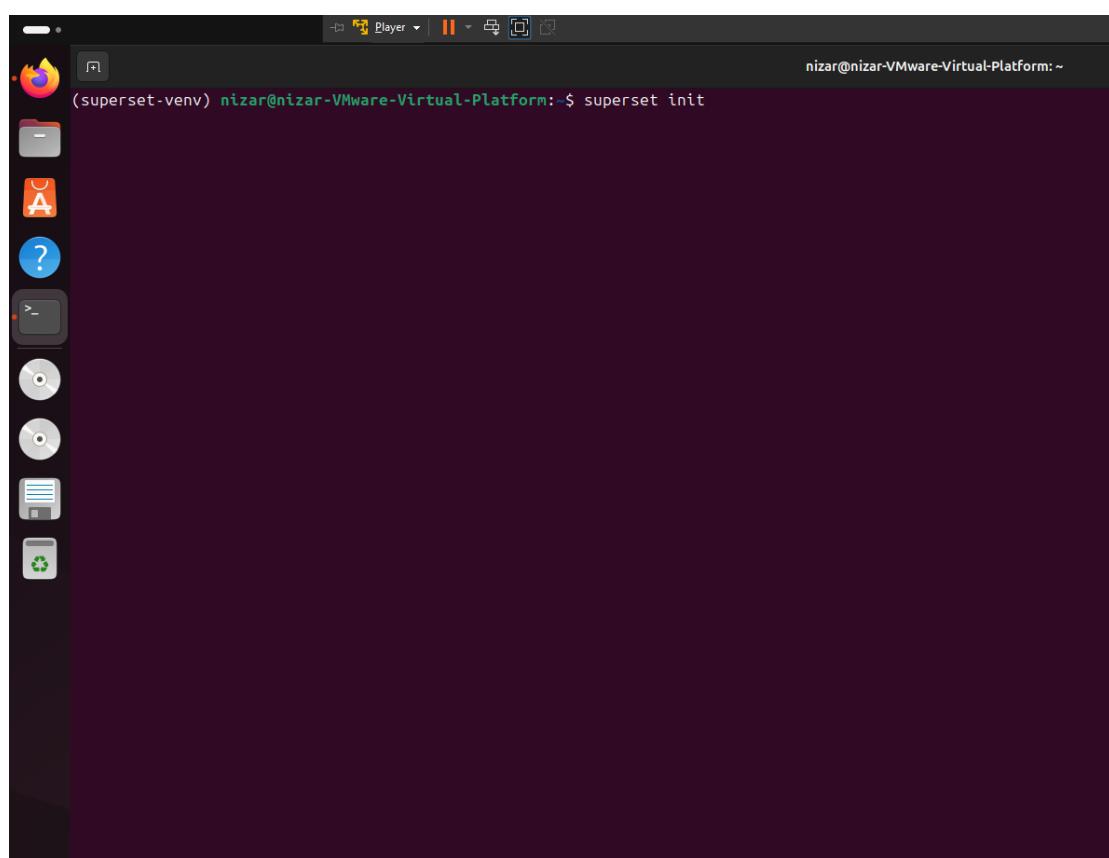
A screenshot of a terminal window titled "nizar@nizar-VMware-Virtual-Platform:~". The command "superset db upgrade" is being run, and the terminal displays a log of migration steps. The log includes messages about SQLite support being removed in the future and various alembic migration steps.

Superset nécessite un utilisateur administrateur pour accéder à l'interface



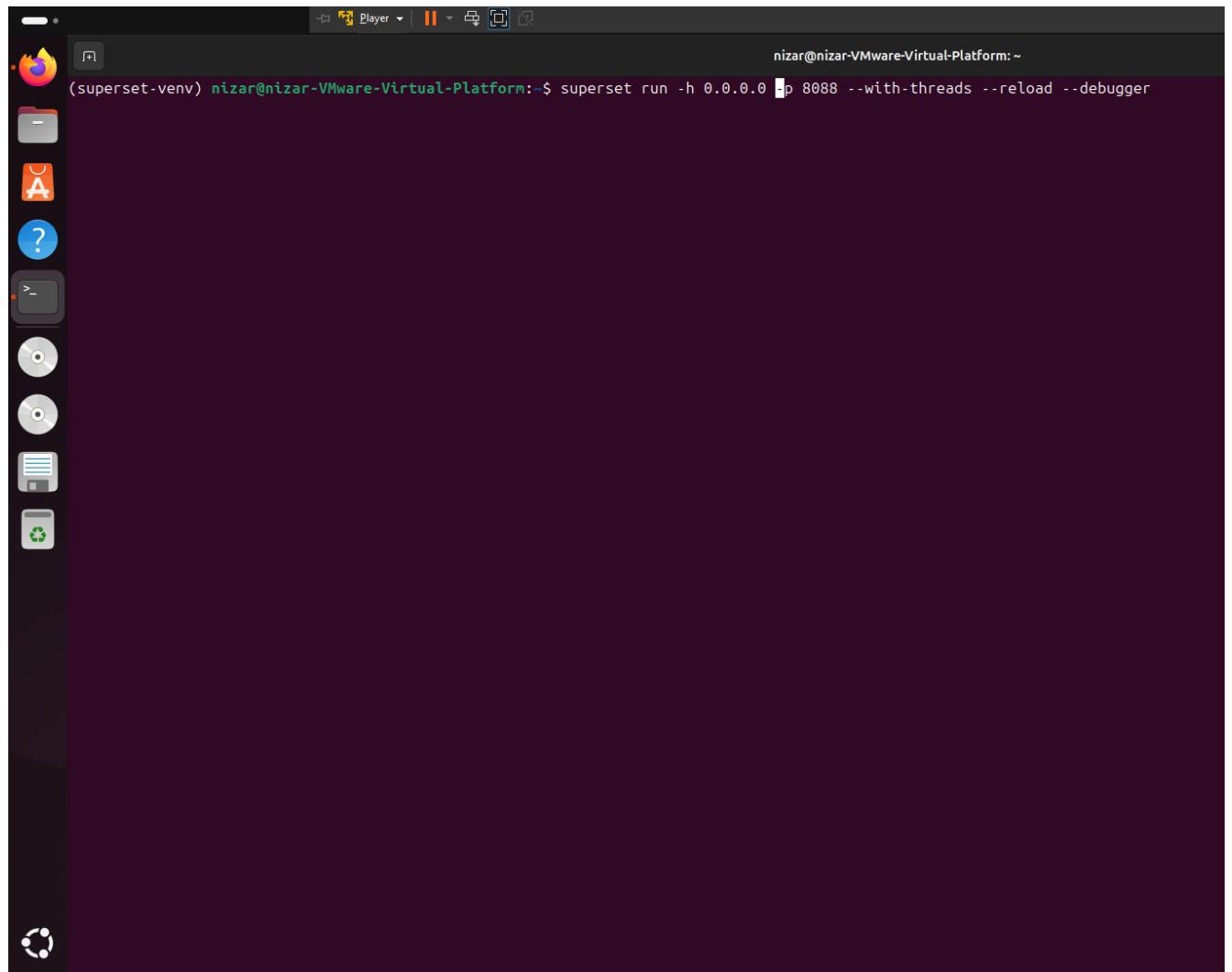
```
nizar@nizar-VMware-Virtual-Platform:~$ superset fab create-admin --username admin --firstname nizar --lastname karroud --email karroud.nizar@gmail.com --password admin
```

Une fois l'utilisateur créé, initialisez Superset avec



```
nizar@nizar-VMware-Virtual-Platform:~$ superset init
```

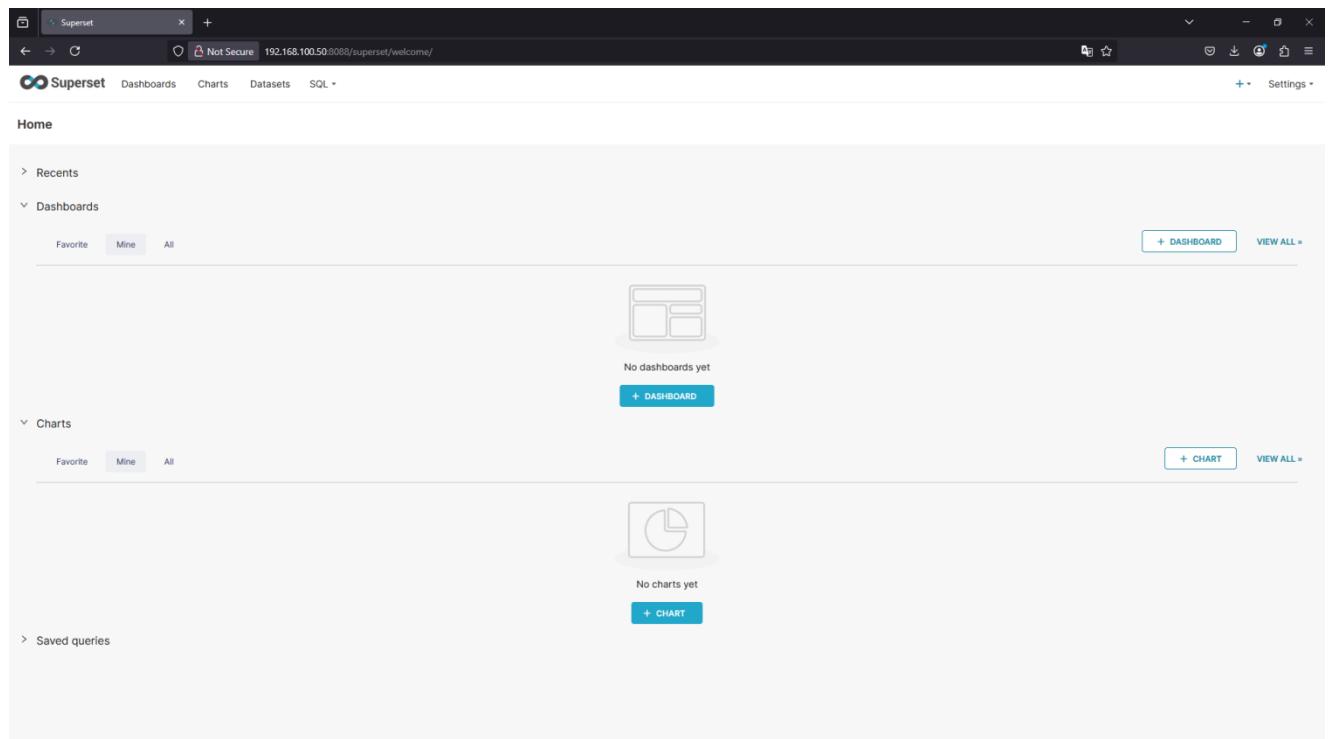
Enfin, lancez le serveur Superset en spécifiant l'IP et le port



A screenshot of a terminal window titled "Player". The terminal shows the command:

```
(superset-venv) nizar@nizar-VMware-Virtual-Platform:~$ superset run -h 0.0.0.0 -p 8088 --with-threads --reload --debugger
```

The terminal has a dark background and a vertical dock on the left side containing icons for various applications.



4.4 Configuration et Déploiement d'un conteneur Apache Kafka

Avant de déployer Apache Kafka dans un conteneur, nous devons installer Docker Desktop sur la machine hôte (<https://www.docker.com/products/docker-desktop/>)

Nous allons maintenant créer le fichier docker-compose.yml pour configurer et déployer le conteneur Kafka avec Zookeeper.

```
kafka > 🗂 docker-compose.yaml
1  version: '3'
2  services:
3    zookeeper:
4      image: confluentinc/cp-zookeeper:latest
5      environment:
6        ZOOKEEPER_CLIENT_PORT: 2181
7        ZOOKEEPER_TICK_TIME: 2000
8      ports:
9        - '2181:2181'
10     networks:
11       - etl_default
12
13   kafka:
14     image: confluentinc/cp-kafka:latest
15     depends_on:
16       - zookeeper
17     ports:
18       - '9092:9092'
19       - '9093:9093'
20     environment:
21       KAFKA_BROKER_ID: 1
22       KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
23       KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://192.168.100.4:9093
24       KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
25       KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT_HOST
26       KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
27     networks:
28       - etl_default
29
30   networks:
31     etl_default:
32       external: true
33
34
```

Ce fichier Docker Compose définit un environnement pour exécuter Apache Kafka et Zookeeper en conteneurs Docker. Zookeeper, nécessaire pour gérer la coordination des brokers Kafka, tourne sur le port 2181. Kafka dépend de Zookeeper et expose deux ports : 9092 (communication interne entre services Docker) et 9093 (accessible depuis l'hôte via l'IP 192.168.100.4). Le paramètre KAFKA_ADVERTISED_LISTENERS permet à Kafka d'être accessible à la fois dans le réseau interne Docker et depuis l'extérieur

Pour créer le réseau etl_default de type bridge, exécute la commande suivante dans un terminal sur la machine hôte :

```
docker network create etl_default
```

Pour lancer le conteneur Kafka avec Docker Compose, exécute la commande suivante dans le répertoire où se trouve le fichier docker-compose.yml

```
● PS C:\Users\Nizar\programming\Projects\etl\kafka> docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 2/2
  ✓ Container kafka-zookeeper-1 Started
  ✓ Container kafka-kafka-1      Started
○ PS C:\Users\Nizar\programming\Projects\etl\kafka>
```

L'option -d permet d'exécuter les conteneurs en arrière-plan (mode détaché). Une fois lancé, vérifie que les conteneurs sont bien en cours d'exécution avec :

```
● PS C:\Users\Nizar\programming\Projects\etl\kafka> docker ps
CONTAINER ID   IMAGE          COMMAND           CREATED        STATUS         PORTS          NAMES
a16cf72e312b   confluentinc/cp-kafka:latest   "/etc/confluent/dock..."   13 days ago   Up 30 seconds   0.0.0.0:9092->9093/tcp
7ef09382fa51   confluentinc/cp-zookeeper:latest "/etc/confluent/dock..."   13 days ago   Up 32 seconds   2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp
○ PS C:\Users\Nizar\programming\Projects\etl\kafka>
```

Oui, après être entré dans le conteneur Kafka avec :

```
○ PS C:\Users\Nizar\programming\Projects\etl\kafka> docker container exec -it kafka-kafka-1 /bin/bash
[appuser@a16cf72e312b ~]$
```

Crée le topic netflow_topic avec la commande suivante :

```
kafka-topics --bootstrap-server 192.168.100.4:9093 --topic netflow_topic --create --partitions 1
--replication-factor 1
```

Cela crée un topic Kafka nommé netflow_topic avec une seule partition et un facteur de réPLICATION de 1.

4.5 Installation et Configuration de nProbe sur Windows

Aller sur le site officiel de ntop : <https://www.ntop.org/products/netflow/nprobe/>

Télécharger la version Windows , exécuter le fichier d'installation et suivre les étapes de l'assistant d'installation.

Après l'installation de nProbe sur Windows, il est nécessaire de l'activer en créant un fichier nprobe.license dans son dossier d'installation (ex. C:\Program Files\nProbe\). Ce fichier doit contenir la clé de licence obtenue sur le site officiel de ntop. Une fois la clé collée et le fichier sauvegardé, on peut vérifier l'activation en exécutant la commande nprobe /c -v

Il suffit maintenant de lancer nProbe en mode collecteur et de configurer les appareils réseaux à exporter le flux vers nProbe (utiliser l'IP de la machine nProbe)

4.6 Mise en Place du Serveur de Prétraitement des Flux

Ce serveur de prétraitement des flux NetFlow est conçu pour collecter, transformer et publier des données en temps réel.

Il fonctionne sur l'IP 192.168.100.4 et le port 9070, où nProbe envoie les flux capturés. Lorsqu'un client (nProbe) se connecte, le serveur reçoit les données, applique des transformations (conversion des timestamps en format lisible, conversion des durées en secondes, indication de la direction du trafic TX/RX), puis publie les informations traitées dans le topic Kafka netflow_topic via quixstreams.

Grâce à asyncio, le serveur gère les connexions de manière asynchrone, permettant un traitement fluide et performant des données.

L'architecture repose sur Apache Kafka pour assurer une transmission efficace et évolutive des flux réseau.

Pour tester la pipeline le script inclut un thread dédié pour exécuter nProbe en mode probe, ce qui permet de collecter les flux réseau en continu

```
import asyncio
import json
from quixstreams import Application
from datetime import datetime, timezone
import subprocess
import os
import shutil
```

```
import threading
import logging
import colorlog

LOG_FORMAT = '[+] %(log_color)s%(asctime)s - %(levelname)s - %(message)s'
DATE_FORMAT = '%Y-%m-%d %H:%M:%S'
```

```
logger = colorlog.getLogger()
handler = logging.StreamHandler()
```

```
formatter = colorlog.ColoredFormatter(
    LOG_FORMAT, datefmt=DATE_FORMAT,
    log_colors={
        'DEBUG': 'cyan',
        'INFO': 'green',
        'WARNING': 'yellow',
        'ERROR': 'red',
        'CRITICAL': 'bold_red',
    }
)
```

```
handler.setFormatter(formatter)
logger.addHandler(handler)
logger.setLevel(logging.INFO)
```

```
def get_nprobe_path():
    exe_name = "nprobe.exe"
    exe_path = shutil.which(exe_name)
    return exe_path if exe_path else None
```

```
def run_nprobe(interface="Intel(R) Wi-Fi 6 AX201 160MHz" ,
tcp_socket="192.168.100.4" , port=9070):
    nprobe = get_nprobe_path()
```

```
    if not nprobe :
        logger.error("nprobe not found in system path.")
```

```
        return

    command = f'"{nprobe}" /c -i "{interface}" --tcp {tcp_socket}:{port} -n none -T
"%IN_SRC_MAC %OUT_DST_MAC %IPV4_SRC_ADDR %IPV4_DST_ADDR %L4_SRC_PORT %L4_DST_PORT %IP
V6_SRC_ADDR %IPV6_DST_ADDR %IP_PROTOCOL_VERSION %PROTOCOL %IN_BYTES %IN_PKTS %OUT_BYT
ES %OUT_PKTS %TCP_FLAGS %DIRECTION %FLOW_START_SEC %FLOW_END_SEC %FLOW_DURATION_MILLI
SECONDS %SRC_TO_DST_IAT_MIN %SRC_TO_DST_IAT_MAX %SRC_TO_DST_IAT_AVG %SRC_TO_DST_IAT_S
TDDEV %DST_TO_SRC_IAT_MIN %DST_TO_SRC_IAT_MAX %DST_TO_SRC_IAT_AVG %DST_TO_SRC_IAT_STD
DEV %MIN_IP_PKT_LEN %MAX_IP_PKT_LEN" '

    logger.info(command)
```

```
log_directory='logs'

if not os.path.exists(log_directory):
    os.makedirs(log_directory)

timestamp = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')

stdout_log_file = os.path.join(log_directory, f'output_{timestamp}.log')
stderr_log_file = os.path.join(log_directory, f'error_{timestamp}.log')
```

```
try:
    logger.info("Running nprobe command...")
    result = subprocess.run(command, shell=True, text=True, capture_output=True,
check=True)
```

```
    output = result.stdout
    error = result.stderr
```

```
    with open(stdout_log_file, 'a') as stdout_file:
        stdout_file.write(output)
    with open(stderr_log_file, 'a') as stderr_file:
        stderr_file.write(error)
```

```
logger.info("nprobe command executed successfully.")
```

```
except subprocess.CalledProcessError as e:
    with open(stderr_log_file, 'a') as stderr_file:
        stderr_file.write(e.stderr)
```

```
logger.error(f"Error while running nprobe command check logs directory ")
```

```
HOST = '192.168.100.4'  
PORT = 9070
```

```
app = Application(broker_address="192.168.100.4:9093", loglevel="DEBUG")
```

```
def transform_data(json_data):
```

```
    if 'FLOW_START_SEC' in json_data:  
        json_data['FLOW_START_TIME'] =  
            datetime.fromtimestamp(json_data['FLOW_START_SEC'], tz=timezone.utc).strftime('%Y-%m-%d %H:%M:%S')  
        del json_data['FLOW_START_SEC']
```

```
    if 'FLOW_END_SEC' in json_data:  
        json_data['FLOW_END_TIME'] = datetime.fromtimestamp(json_data['FLOW_END_SEC'],  
tz=timezone.utc).strftime('%Y-%m-%d %H:%M:%S')  
        del json_data['FLOW_END_SEC']
```

```
    if 'FLOW_DURATION_MILLISECONDS' in json_data:  
        json_data['FLOW_DURATION_SECONDS'] = json_data['FLOW_DURATION_MILLISECONDS']  
        / 1000  
        del json_data['FLOW_DURATION_MILLISECONDS']
```

```
    if 'DIRECTION' in json_data:  
        json_data['DIRECTION'] = 'RX' if json_data['DIRECTION'] == 0 else 'TX'
```

```
    return json_data
```

```
async def handle_client(reader: asyncio.StreamReader, writer: asyncio.StreamWriter):  
    logger.info("Handling new client connection...")
```

```
producer = app.get_producer()
```

```
try:  
    while True:
```

```
        line = await reader.readline()
        if not line:
            logger.info("Client disconnected.")
            break
```

```
    try:
        json_data = json.loads(line.decode('utf-8').strip())
        flow_data = transform_data(json_data)
```

```
        producer.produce(topic="netflow_topic", key="netflow",
value=json.dumps(flow_data))
        await asyncio.sleep(0)
```

```
    except json.JSONDecodeError:
        print("Invalid JSON received, ignoring...")
```

```
except asyncio.CancelledError:
    pass
finally:
    writer.close()
    await writer.wait_closed()
```

```
async def main():
    server = await asyncio.start_server(handle_client, HOST, PORT)
    addr = server.sockets[0].getsockname()
    logger.info(f"Starting the server on {addr}")
```

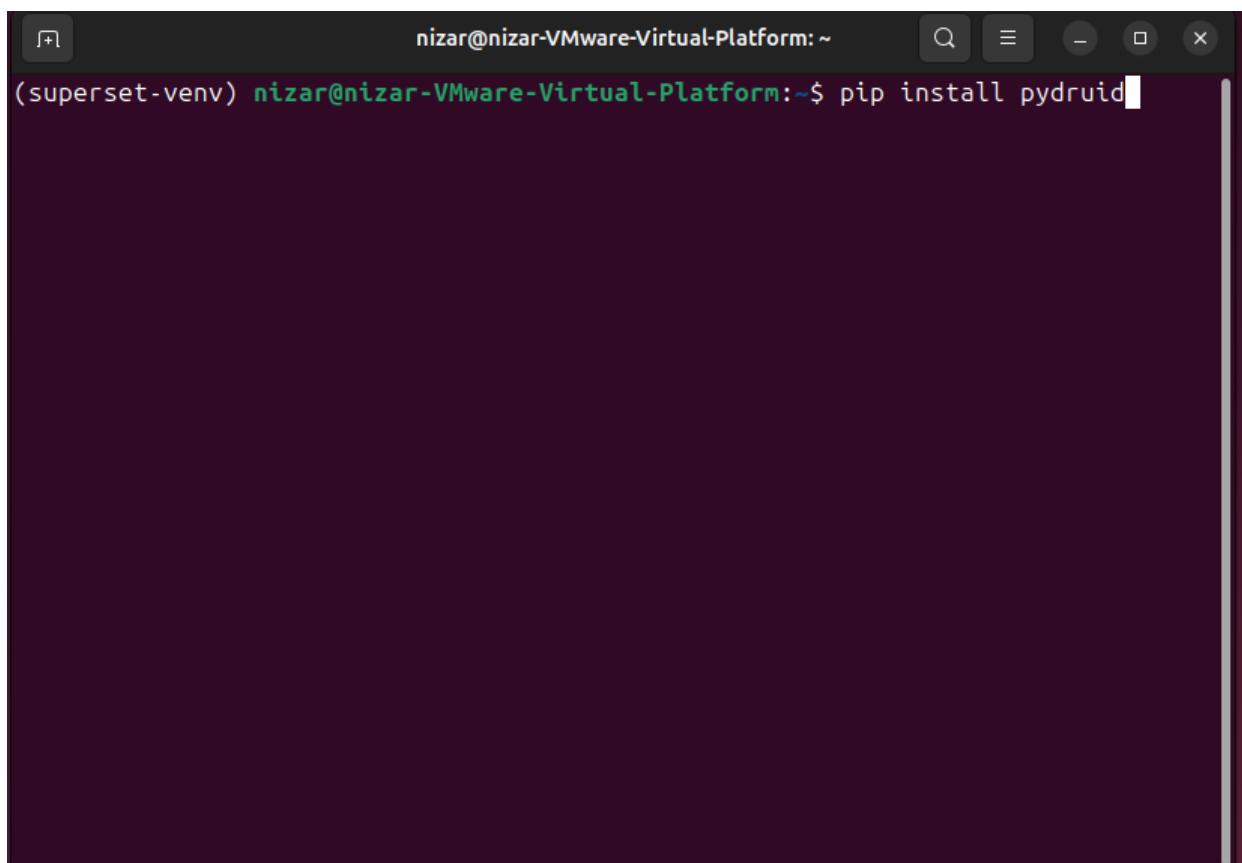
```
nprobe_thread = threading.Thread(target=run_nprobe, args=("Intel(R) Wi-Fi 6 AX201
160MHz", "192.168.100.4", 9070))
nprobe_thread.start()
```

```
async with server:
    await server.serve_forever()
```

```
if __name__ == "__main__":
    asyncio.run(main())
```

4.7 Connexion d'Apache Superset à Apache Druid

Avant de configurer la connexion, installez PyDruid, le connecteur Python pour Druid, dans votre environnement virtuel

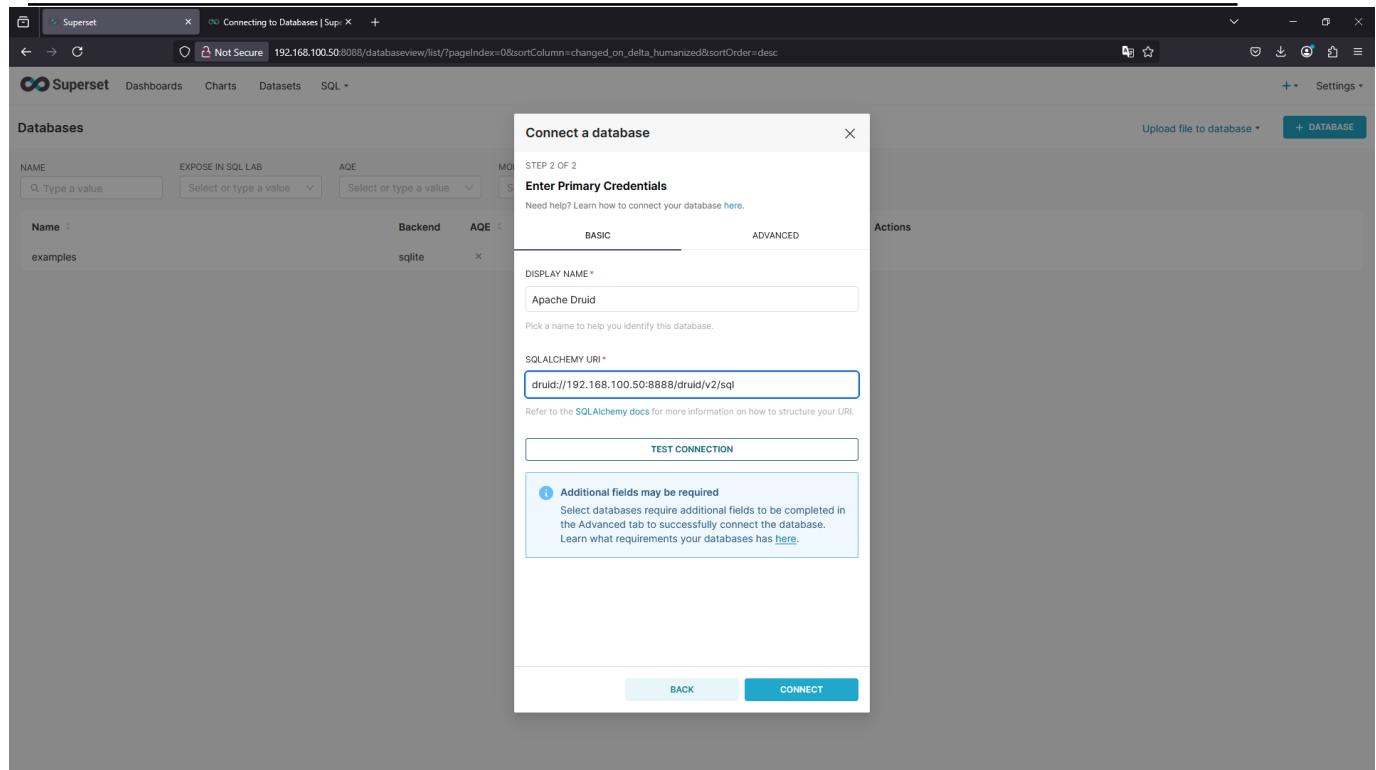


A screenshot of a terminal window titled "nizar@nizar-VMware-Virtual-Platform:~". The command "pip install pydruid" is being typed into the terminal.

```
nizar@nizar-VMware-Virtual-Platform:~$ pip install pydruid
```

Ajouter Apache Druid comme source de données dans Superset

<https://superset.apache.org/docs/configuration/databases>



NAME	EXPOSE IN SQL LAB	AQE	MODIFIED BY				
apache	Select or type a value	Select or type a value	Select or type a value				
Name	Backend	AQE	DML	File upload	Expose in SQL Lab	Last modified	Actions
Apache Druid	druid	x	x	x	v	15 seconds ago	Edit

4.8 Connexion d'Apache Kafka à Apache Druid

Il est maintenant temps d'ingérer les données de Kafka dans Druid , assurez-vous que Kafka est bien en cours d'exécution et que le topic netflow_topic reçoit des données (lancer le script)

The screenshot shows a code editor interface with multiple tabs and panes. The main pane displays a Python file named `main.py` containing asynchronous code for handling network clients and producing data to a Kafka topic. Below the code editor is a terminal window showing the execution of the script and its output, which includes logs from a Kafka consumer and a message about validating topics. The left sidebar contains a file tree with `FOLDERS`, `ETL`, `.dist`, `kafka`, `docker-compose.yaml`, `logs`, `error_2025-03-18_2...`, `error_2025-03-18_2...`, `venv`, `main.py`, and `testpy`. The bottom left pane shows an MySQL connection named `Kwang Python`. The bottom right pane displays status information like line and column numbers, and the bottom center shows the file path as `C:\Users\Wizar\programming\Projects\etl`.

```
File Edit Selection View Go Run Terminal Help etl [Administrator] 08 Folders ... test.py main.py docker-compose.yaml 101     async def handle_client(reader: asyncio.StreamReader, writer: asyncio.StreamWriter): 102         producer = app.get_producer() 103         try: 104             while True: 105                 line = await reader.readline() 106                 if not line: 107                     logger.info("Client disconnected.") 108                     break 109                 try: 110                     json_data = json.loads(line.decode('utf-8').strip()) 111                     flow_data = transform_data(json_data) 112                     producer.produce(topic="netflow_topic", key="netflow", value=json.dumps(flow_data)) 113                     await asyncio.sleep(0) 114                 except json.JSONDecodeError: 115                     print("Invalid JSON received, ignoring...") 116                 except asyncio.CancelledError: 117                     pass 118             finally: 119                 writer.close() 120                 await writer.wait_closed() 121         except asyncio.CancelledError: 122             pass 123     finally: 124         writer.close() 125         await writer.wait_closed() 126 127     async def main(): 128         server = await asyncio.start_server(handle_client, HOST, PORT) 129         return server 130 131 if __name__ == "__main__": 132     loop.create_task(main()) 133 134 PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS PS C:\Users\Wizar\programming\Projects\etl> python main.py [+ 2025-03-18 21:20:29 - INFO - Starting the server on ('192.168.100.4', 9070) [+ 2025-03-18 21:20:29 - INFO - C:\Program Files\VProber\nprobe.exe" /c -i "Intel(R) Wi-Fi 6 AX201 160MHz" --tcp 192.168.100.4:9070 -n none -T "%IN_SRC_MAC %OUT_DST_MAC %IPV4_SRC_ADDR %IPV4_DST_ADDR %4_SRC_PORT %4_DST_PORT %IPV6_SRC_ADDR %IPV6_DST_ADDR %IP_PROTOCOL_VERSION %PROTOCOL %IN_BYTES %IN_PCKTS %OUT_BYTES %OUT_PCKTS %TCP_FLAGS %DIRECTION %FLOW_START_SEC %FLOW_END_SEC %FLOW_DURATION_MILLISECONDS %SRC_TO_DST_IAT_MIN %SRC_TO_DST_IAT_AVG %SRC_TO_DST_IAT_STDEV %DST_TO_SRC_IAT_MAX %DST_TO_SRC_IAT_AVG %DST_TO_SRC_IAT_STDEV %SRC_IP_PKT_LEN %MAX_IP_PKT_LEN" [+ 2025-03-18 21:20:29 - INFO - Running nprobe command... [+ 2025-03-18 21:21:32 - INFO - Handling new client connection... [2025-03-18 21:21:32,117] [INFO] [quixstreams] : Validating Kafka topics are configured correctly [2025-03-18 21:21:32,118] [INFO] [quixstreams] : Kafka topics Validation complete > EXPLORER: MySQL OPEN EDITORS test.py main.py docker-compose... > OUTLINE x 0.0.0 Kwang Python Cloud Code - Sign in Select Postgres Server Ln 117, Col 54 (13 selected) Spaces: 4 UTF-8 CRLF () Python 3.12.6 (venv) Go
```

Connecter apache druid à kafka :

ensuite configurer le parseur de données , dans notre cas on change rien

The screenshot shows the Apache Druid streaming-data-loader interface. On the left, a large text area displays a single log entry (Original row) in JSON format. The log contains various network-related fields such as source and destination IP addresses, ports, and timestamp details. On the right, there's a configuration sidebar with tabs for 'Input format' (set to 'json'), 'JSON parser features' (with 'Parse Kafka metadata (ts, headers, key)' checked), and other settings like 'Assume newline delimited' and 'Use JSON node reader'. A note at the top right says: "Druid needs to parse data as columns. Determine the format of your data and ensure that the columns are accurately parsed." Below the configuration are buttons for 'Apply' and 'Add column flattening'.

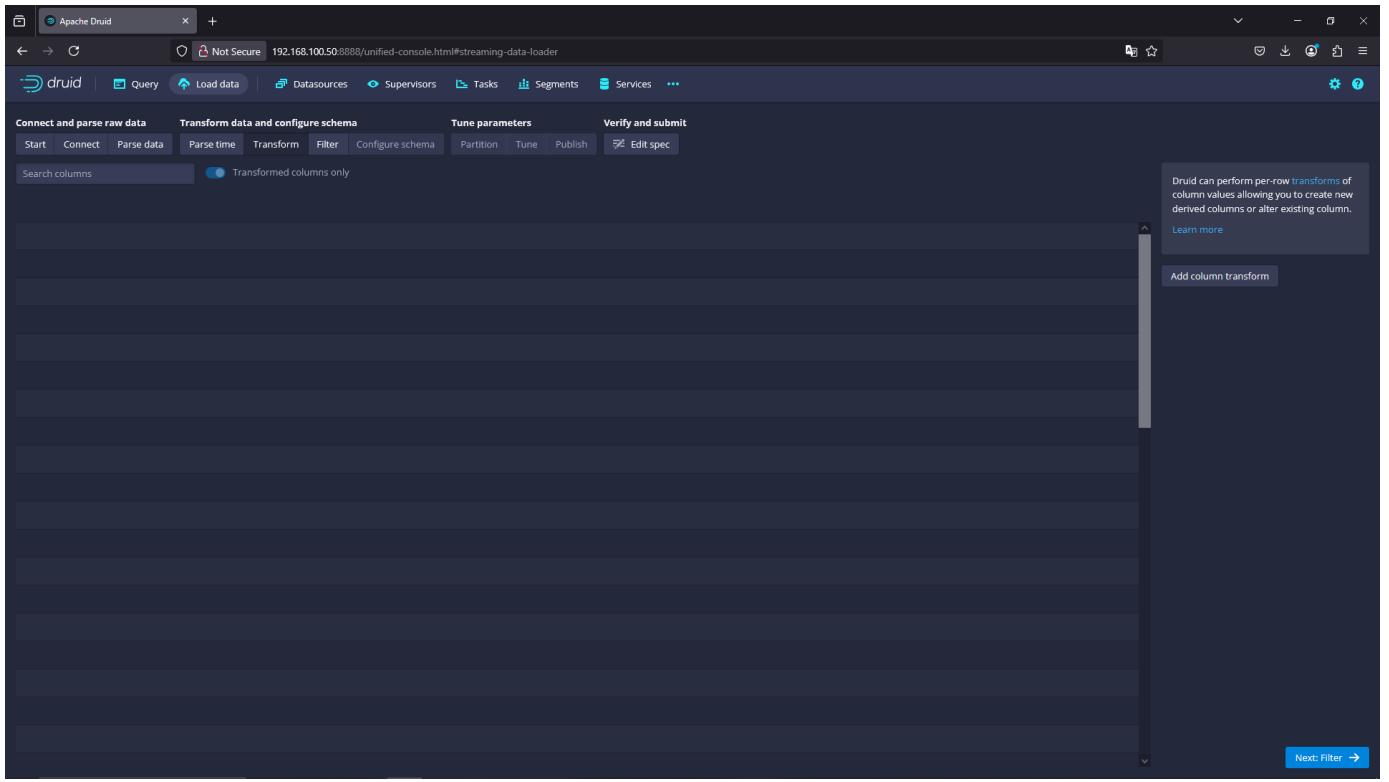
pour le parseur de temps on choisit la colonne du timestamp (FLOW_START_TIME) et le format (yyyy-MM-dd HH:mm:ss)

This screenshot shows the timestamp configuration step in the Druid interface. It highlights the 'time' column and the 'FLOW_START_TIME' column. The configuration panel on the right specifies 'Parse timestamp from' as 'Column' (set to 'FLOW_START_TIME'), 'Format' as 'auto', and 'Missing value' as '(optional)'. A note in the sidebar explains that Druid partitions data based on the primary time column and provides instructions for handling time information spread across multiple columns. Buttons for 'Apply' and 'Next: Transform' are visible at the bottom right.

Dans notre cas, l'étape de transformation est optionnelle puisque tout le prétraitement des données est déjà effectué dans le serveur de prétraitement.

Si besoin, Druid permet d'ajouter des calculs dynamiques comme :

- Création de nouvelles colonnes
- Agrégation de valeurs
- Conversion de types



L'étape de filtrage est optionnelle, car notre serveur de prétraitement effectue déjà tout le nettoyage et la structuration des données NetFlow avant l'ingestion dans Apache Kafka.

The screenshot shows the Apache Druid unified console interface with a filter applied to the data table. The filter is set to 'true'. The table contains network flow data with columns including _time, IN_SRC_MAC, OUT_DST_MAC, IPV4_SRC_ADDR, IPV4_DST_ADDR, L4_SRC_PORT, L4_DST_PORT, IPV6_SRC_ADDR, IPV6_DST_ADDR, IP_PROTOCOL_VERSI, PROTOCOL, and IN_BY.

A tooltip on the right side of the table area states:

Druid can filter out unwanted data by applying per-row filters.

[Learn more](#)

[Filter](#) ('true')

[Apply](#)

[Add column filter](#)

[Next: Configure schema →](#)

Lors de la configuration du schéma, on s'assure que chaque colonne a le bon type de données , l'ajout de métriques et dimension

The screenshot shows the Apache Druid streaming-data-loader interface. At the top, there are tabs for 'Connect and parse raw data', 'Transform data and configure schema', 'Tune parameters', and 'Verify and submit'. Below these tabs is a table with columns: _time, FLOW_END_TIME, FLOW_DURATION_SI, IN_SRC_MAC, OUT_DST_MAC, IPV4_SRC_ADDR, IPV4_DST_ADDR, IPV6_DST_ADDR, IPV6_SRC_ADDR, L4_SRC_PORT, and L4_DST_PORT. The table contains several rows of data, each representing a network flow record. To the right of the table, there is a sidebar with various configuration options and status indicators.

_time long (time column)	FLOW_END_TIME string	FLOW_DURATION_SI double	IN_SRC_MAC string	OUT_DST_MAC string	IPV4_SRC_ADDR string	IPV4_DST_ADDR string	IPV6_DST_ADDR string	IPV6_SRC_ADDR string	L4_SRC_PORT long	L4_DST_PORT long
2025-03-18 20:57:26	2025-03-18 20:57:26	0.024	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	2.16.38.4	::	::	24524	80
2025-03-18 20:57:27	2025-03-18 20:57:27	0	9C:73:70:38:1E:92	E0:2B:E9:DD:D0:7E	0.0,0.0	0.0,0.0	FF02::1	FE80::1	0	0
2025-03-18 20:57:27	2025-03-18 20:57:27	0.009	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	192.168.100.1	::	::	50920	53
2025-03-18 20:57:28	2025-03-18 20:57:28	0	9C:73:70:38:1E:92	E0:2B:E9:DD:D0:7E	20.191.95.51	192.168.100.4	::	::	443	24498
2025-03-18 20:57:27	2025-03-18 20:57:28	0.598	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	52.182.143.211	::	::	24531	443
2025-03-18 20:57:33	2025-03-18 20:57:33	0.03	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	20.199.120.85	::	::	21329	443
2025-03-18 20:57:41	2025-03-18 20:57:41	0.024	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	34.117.188.166	::	::	24495	443
2025-03-18 20:58:12	2025-03-18 20:58:12	0.026	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.50	8.8.8.8	::	::	39027	53
2025-03-18 20:58:12	2025-03-18 20:58:12	0.245	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.50	185.125.190.17	::	::	42762	80
2025-03-18 20:57:32	2025-03-18 20:58:16	44.948	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	13.107.246.77	::	::	24527	443
2025-03-18 20:57:31	2025-03-18 20:58:16	44.984	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	13.107.246.77	::	::	24526	443
2025-03-18 20:58:25	2025-03-18 20:58:25	0.007	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	192.168.100.1	::	::	55177	53
2025-03-18 20:58:25	2025-03-18 20:58:25	0.291	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	51.116.246.104	::	::	24534	443
2025-03-18 20:57:26	2025-03-18 20:59:11	105.269	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	52.84.66.110	::	::	24517	443
2025-03-18 20:57:26	2025-03-18 20:59:11	105.34	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	52.84.66.110	::	::	24519	443
2025-03-18 20:58:25	2025-03-18 20:58:26	0.779	9C:73:70:38:1E:92	E0:2B:E9:DD:D0:7E	192.168.100.1	239.255.255.250	::	::	41460	1900
2025-03-18 20:57:27	2025-03-18 20:59:25	118.127	0:72:63:D4:C8:36	FF:FF:FF:FF:FF:FF	192.168.100.79	192.168.100.255	::	::	6720	20806
2025-03-18 20:57:32	2025-03-18 20:59:18	105.349	E0:2B:E9:DD:D0:7E	0:0:C2:9E:94:76C	192.168.100.4	192.168.100.50	::	::	9093	46204
2025-03-18 20:57:32	2025-03-18 20:59:18	105.349	E0:2B:E9:DD:D0:7F	0:0:F4:94:76C	192.168.100.4	192.168.100.50	::	::	9093	46210

Nous avons choisi une granularité de segment à l'heure (HOUR) pour le partitionnement des données dans Apache Druid.

The screenshot shows the Apache Druid unified console interface. In the top navigation bar, the URL is 192.168.100.50:8888/unified-console.html#streaming-data-loader. The main area is titled 'Tune parameters' under the 'Transform data and configure schema' tab. On the left, there's a section for 'Primary partitioning (by time)' with a dropdown for 'Segment granularity' set to 'hour'. On the right, there's a section for 'Secondary partitioning' with 'Max rows per segment' set to 5000000 and 'Max total rows' set to 20000000. A tooltip on the right explains that Druid datasources are always partitioned by time into time chunks (primary partitioning) and each time chunk contains one or more segments (secondary partitioning). A 'Learn more' link is also present.

Pour le tuning, nous avons gardé les paramètres par défaut, sauf pour l'offset, où nous avons choisi de ne pas utiliser "earliest offset".

Cela permet d'éviter l'ingestion des anciens messages et de ne récupérer que les nouvelles données NetFlow arrivant sur Kafka.

The screenshot shows the Apache Druid unified console interface. In the top navigation bar, the URL is 192.168.100.50:8888/unified-console.html#streaming-data-loader. The main area is titled 'Tune parameters' under the 'Transform data and configure schema' tab. On the left, there's a section for 'Input tuning' with various configuration options: 'Use earliest offset' (set to True), 'Task duration' (set to PT1H), 'Task count' (set to 1), 'Replicas' (set to 1), 'Completion timeout' (set to PT30M), 'Poll timeout' (set to 100), 'Start delay' (set to PT5S), 'Management period' (set to PT30S), 'Late message rejection period' (set to (none)), 'Early message rejection period' (set to (none)), and 'Skip offset gaps' (set to False). On the right, there's a section for 'General tuning' with options like 'Max rows in memory' (set to 15000), 'Max bytes in memory' (Default: 1/6 of max JVM memory), 'Reset offset automatically' (set to False), 'Intermediate persist period' (set to PT10M), 'Intermediate handoff period' (set to P2147483647D), 'Worker threads' (set to min(10, taskCount)), 'Http timeout' (set to PT10S), and 'Offset fetch period' (set to PT30S). A tooltip on the right explains how Druid will ingest data. A 'Learn more' link is also present.

Pour la publication des données, nous avons nommé la datasource "netflow_topic"

The screenshot shows the Apache Druid unified console interface. The top navigation bar includes tabs for Ingestion spec reference, Apache Kafka ingestion, and What is the difference between. The main header shows the URL 192.168.100.50:8888/unified-console.html#streaming-data-loader. Below the header, there are tabs for druid, Query, Load data, Datasources, Supervisors, Tasks, Segments, Services, and more.

The main content area is divided into several sections:

- Connect and parse raw data**:
 - Datasource name: netflow_topic
 - Suspended: False (selected)
 - Use concurrent locks: True
- Transform data and configure schema**:
 - Parse time, Transform, Filter, Configure schema, Partition, Tune, Publish, Edit spec buttons.
- Tune parameters**:
 - Verify and submit button.
- Parse error reporting**:
 - Log parse exceptions: False (selected), True
 - Max parse exceptions: (unlimited)
 - Max saved parse exceptions: 0
- A sidebar on the right: "Configure behavior of indexed data once it reaches Druid."

At the bottom, there are search and filter options: Find in page, Highlight All, Match Case, Match Diacritics, Whole Words, and a Next: Edit spec button.

Après avoir configuré l'ingestion, nous visitons la page du Supervisor d'Apache Druid pour vérifier que le job d'ingestion est en cours d'exécution

The screenshot shows the Apache Druid unified console interface. The title bar indicates "Ingestion spec reference | Apache Druid" and the URL "Not Secure 192.168.100.50:8888/unified-console.html#supervisors/supervisor_id=netflow_topic". The main navigation bar includes "druid", "Query", "Load data", "Datasources", "Supervisors" (which is the active tab), "Tasks", "Segments", "Services", and "...".

The "Supervisors" table has the following columns:

Supervisor ID (datasource)	Type	Topic/Stream	Status	Configured tasks	Running tasks	Aggregate lag	Stats Rate over past 5 minutes	Recent errors	Actions
= netflow_topic	kafka	netflow_topic	RUNNING	1 (no replication)	1 active task	556		0 errors	...

On se rend dans l'onglet "Datasources" pour confirmer que la datasource "netflow_topic" a bien été créée.

The screenshot shows the Apache Druid unified console interface, specifically the "Datasources" tab. The title bar and URL are identical to the previous screenshot. The main navigation bar includes "druid", "Query", "Load data", "Datasources" (active), "Supervisors", "Tasks", "Segments", "Services", and "...".

The "Datasources" table has the following columns:

Datasource name	Availability	Historical load/drop queues	Total data size	Running tasks	Segment rows minimum / average / maximum	Total rows	Avg. row size (bytes)	Replicated size	Compaction	% Compacted bytes / segments / intervals	Left to be compacted	Re
netflow_topic	Fully available (1 segment)	No segments to load/drop	0.00 B	index_kafka: 1	0 0 0	3	0	0.00 B	Not enabled	- - -	-	Click

Apache Druid | Ingestion spec reference | Apache | +

Not Secure 192.168.100.50:8888/unified-console.html#datasources

druid Query Load data Datasources Supervisors Tasks Segments Services ...

Datasources

Datasource name: netflow_topic

Records

Columns

13/15

_time	FLOW_END_TIME	FLOW_DURATION	IN_SRC_MAC	OUT_DST_MAC	IPV4_SRC_ADDR	IPV4_DST_ADDR	IPV6_DST_ADDR	IPV6_SRC_ADDR	L4_SRC_PORT	L4_DST_PORT
2025-03-18T21:20:29.000Z	2025-03-18 21:22:14	105.085	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9,093	59,224
2025-03-18T21:20:29.000Z	2025-03-18 21:22:15	105.092	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9,093	38,530
2025-03-18T21:20:30.000Z	2025-03-18 21:20:30	0.011	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	192.168.100.1	::	::	58,141	53
2025-03-18T21:20:30.000Z	2025-03-18 21:20:31	0.743	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	20,189,173.16	::	::	24,685	443
2025-03-18T21:20:30.000Z	2025-03-18 21:22:28	118.07	00:72:63:04:C8:36	FFFF:FF:FF:FF	192.168.100.79	192.168.100.255	::	::	6,720	20,806
2025-03-18T21:20:30.000Z	2025-03-18 21:22:15	105.111	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9,093	38,534
2025-03-18T21:20:31.000Z	2025-03-18 21:22:23	112.12	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	24,674	8,888
2025-03-18T21:20:31.000Z	2025-03-18 21:22:16	105.054	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9,093	54,866
2025-03-18T21:20:32.000Z	2025-03-18 21:22:17	105.11	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9,093	54,874
2025-03-18T21:20:34.000Z	2025-03-18 21:22:27	113.114	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	5,188,148.63	::	::	21,317	443
2025-03-18T21:20:34.000Z	2025-03-18 21:21:50	75.916	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	24,677	8,888
2025-03-18T21:20:35.000Z	2025-03-18 21:21:50	75.545	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	24,675	8,888
2025-03-18T21:20:35.000Z	2025-03-18 21:21:50	75.299	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	24,678	8,888
2025-03-18T21:20:35.000Z	2025-03-18 21:21:50	75.048	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	24,679	8,888
2025-03-18T21:20:35.000Z	2025-03-18 21:22:20	105.099	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9,093	60,560
2025-03-18T21:20:35.000Z	2025-03-18 21:21:50	74.972	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	24,676	8,888

< > Showing 1-20 of 53

Actions ▾ Close

Vérifiez l'onglet "Segments", où nous devons voir les segments de données créés avec la granularité d'une heure.

Player Ubuntu 64-bit

Apache Druid | localhost:8888/unified-console.html#segments

druid Query Load data Datasources Supervisors Tasks Segments Services ...

Segments

Refresh Group by None Interval ... Show segment timeline

Columns (18/19) ▾

Start	End	Version	Time span	Shard type	Shard spec	Partiti...	Size	Num rows	Avg. row size (bytes)	Replicas (actual)	Replicat...	Is available	Is active	Is realtime	Is published	Actions
2025-03-19T17:00:00.000Z	2025-03-19T18:00:00.000Z	2025-03-19T17:01:42.904Z	Hour	numbered	No detail	0	90.02 KB	1,044	86	1	2	true	true	false	true	...
2025-03-19T17:00:00.000Z	2025-03-19T18:00:00.000Z	2025-03-19T17:01:42.904Z	Hour	numbered	No detail	1	(realtime)	89	0	1	-1	true	true	true	false	...
2025-03-19T16:00:00.000Z	2025-03-19T17:00:00.000Z	2025-03-19T16:01:44.459Z	Hour	numbered	No detail	0	83.99 KB	869	96	1	2	true	true	false	true	...
2025-03-19T15:00:00.000Z	2025-03-19T16:00:00.000Z	2025-03-19T15:29:50.961Z	Hour	numbered	No detail	0	81.40 KB	891	91	1	2	true	true	false	true	...
2025-03-18T23:00:00.000Z	2025-03-19T00:00:00.000Z	2025-03-18T23:01:05.120Z	Hour	numbered	No detail	0	15.23 KB	38	400	1	2	true	true	false	true	...
2025-03-18T22:00:00.000Z	2025-03-18T23:00:00.000Z	2025-03-18T22:11:31.809Z	Hour	numbered	No detail	0	98.86 KB	1,326	74	1	2	true	true	false	true	...

Apache Druid

Ingestion spec reference | Apache Druid

Not Secure 192.168.100.50:8888/unified-console.html#segments

druid | Query | Load data | Datasources | Supervisors | Tasks | Segments | Services | ...

Segments

Segment: netflow_topic_2025-03-18T21:00:00.000Z_2025-03-18T22:00:00.000Z_2025-03-18T21:21:33.351Z

Segment ID: 1742332829000

Metadata

Records

Showing 1-20 of 68

Actions | Close

Avg (bytes)

Segment ID	_time	FLOW_END_TIME	FLOW_DURATION_SECONDS	IN_SRC_MAC	OUT_DST_MAC	IPV4_SRC_ADDR	IPV4_DST_ADDR	IPV6_DST_ADDR	IPV6_SRC_ADDR
netflow_top	1742332829000	2025-03-18 21:22:14	105.085	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332829000	2025-03-18 21:22:15	105.092	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332830000	2025-03-18 21:20:30	0.011	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	192.168.100.1	::	::
	1742332830000	2025-03-18 21:20:31	0.743	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	20.189.173.16	::	::
	1742332830000	2025-03-18 21:22:28	118.07	00:72:63:D4:C8:36	FF:FF:FF:FF:FF:FF	192.168.100.79	192.168.100.255	::	::
	1742332830000	2025-03-18 21:22:15	105.111	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332831000	2025-03-18 21:22:23	112.12	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332831000	2025-03-18 21:22:16	105.054	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332832000	2025-03-18 21:22:17	105.11	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332834000	2025-03-18 21:22:27	113.114	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	5.188.148.63	::	::
	1742332834000	2025-03-18 21:21:50	75.916	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332835000	2025-03-18 21:21:50	75.545	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332835000	2025-03-18 21:21:50	75.299	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332835000	2025-03-18 21:21:50	75.048	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332835000	2025-03-18 21:22:20	105.099	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::
	1742332835000	2025-03-18 21:21:50	74.972	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::

Une fois la datasource "netflow_topic" confirmée et les segments visibles, vous pouvez query les données via Druid SQL ou l'interface web de Druid.

Apache Druid

Ingestion spec reference | Apache Druid

Not Secure 192.168.100.50:8888/unified-console.html#workbench/227c2151

druid | Query | Load data | Datasources | Supervisors | Tasks | Segments | Services | ...

druid

Search

netflow_topic

__time

SELECT * FROM "netflow_topic" WHERE __time >= CURRENT_TIMESTAMP - INTERVAL '1' HOUR

Run Engine: Auto [SQL (native)]

118 results in 0.17s

Showing 1-20 of 118

	__time	FLOW_END_TIME	FLOW_DURATION_SECONDS	IN_SRC_MAC	OUT_DST_MAC	IPV4_SRC_ADDR	IPV4_DST_ADDR	IPV6_DST_ADDR	IPV6_SRC_ADDR	L4_SRC_PORT	L4_DST_PORT	ts
	2025-03-18T21:20:29.000Z	2025-03-18 21:22:14	105.085	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9.093	59.224	4
	2025-03-18T21:20:29.000Z	2025-03-18 21:22:15	105.092	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9.093	38.530	4
	2025-03-18T21:20:30.000Z	2025-03-18 21:20:30	0.011	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	192.168.100.1	::	::	58.141	53	4
	2025-03-18T21:20:30.000Z	2025-03-18 21:20:31	0.743	E0:2B:E9:DD:D0:7E	9C:73:70:38:1E:92	192.168.100.4	20.189.173.16	::	::	24.685	443	4
	2025-03-18T21:20:30.000Z	2025-03-18 21:22:28	118.07	00:72:63:D4:C8:36	FF:FF:FF:FF:FF:FF	192.168.100.79	192.168.100.255	::	::	6.720	20.806	4
	2025-03-18T21:20:30.000Z	2025-03-18 21:22:15	105.111	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	9.093	38.534	4
	2025-03-18T21:20:31.000Z	2025-03-18 21:22:23	112.12	E0:2B:E9:DD:D0:7E	00:0C:29:E9:47:6C	192.168.100.4	192.168.100.50	::	::	24.674	8.888	4

4.9 Visualisation des Flux NetFlow avec Apache Superset

La première étape consiste à ajouter le dataset depuis Apache Druid. Pour cela, il faut ouvrir Superset, accéder à l'onglet Data > Datasets, puis cliquer sur + Dataset. Ensuite, il faut sélectionner Apache Druid comme source de données, choisir la datasource "netflow_topic", et l'ajouter.

The screenshot shows the Apache Superset interface for creating a new dataset. The URL in the browser is 192.168.100.50:8080/dataset/add/. The left sidebar shows the navigation: Apache Druid, Superset, Dashboards, Charts, Datasets, SQL. The main area is titled 'netflow_topic'. On the left, there are dropdown menus for DATABASE (druid - Apache Druid), SCHEMA (druid), and TABLE (netflow_topic). A search bar and a 'CREATE' button are also present. The central part displays the schema for the 'netflow_topic' table, listing columns such as _time, FLOW_END_TIME, FLOW_DURATION_SECONDS, IN_SRC_MAC, OUT_DST_MAC, IPV4_SRC_ADDR, IPV4_DST_ADDR, IPV6_DST_ADDR, IPV6_SRC_ADDR, L4_SRC_PORT, L4_DST_PORT, IP_PROTOCOL_VERSION, PROTOCOL, IN_BYTES, IN_PKTS, OUT_BYTES, OUT_PKTS, and DIRECTION. Each column has its name, type (e.g., TIMESTAMP, VARCHAR, FLOAT, BIGINT), and length or precision. At the bottom right are 'CANCEL' and 'CREATE DATASET AND CREATE CHART' buttons.

Une fois le dataset NetFlow ajouté dans Apache Superset, nous pouvons passer à l'étape de création de graphiques pour explorer, analyser et visualiser les flux réseau collectés. Superset offre une large variété de visualisations interactives adaptées aux données temporelles, catégoriques et quantitatives, ce qui permet de mieux comprendre le comportement du trafic réseau. Ces visualisations permettent de détecter des anomalies, suivre l'évolution du trafic, identifier les hôtes les plus actifs, ou encore observer la répartition des protocoles utilisés. Voici les différents graphiques que j'ai créés pour tirer parti de ces données.

4.9.1 Intensité du trafic entrant par heure (carte thermique IN_BYTES)

Cette carte thermique visualise le volume total du trafic entrant (IN_BYTES) sur les heures de la journée pour tous les jours disponibles dans l'ensemble de données.

Cette carte permet de repérer les heures de pointe, ce qui peut être utile pour la planification de la capacité ou pour détecter un comportement réseau anormal en dehors des heures de pointe.

Les pics inhabituels à des heures étranges (par exemple, à minuit) peuvent suggérer l'utilisation de botnets ou de scripts d'attaque automatisés fonctionnant à des moments spécifiques.

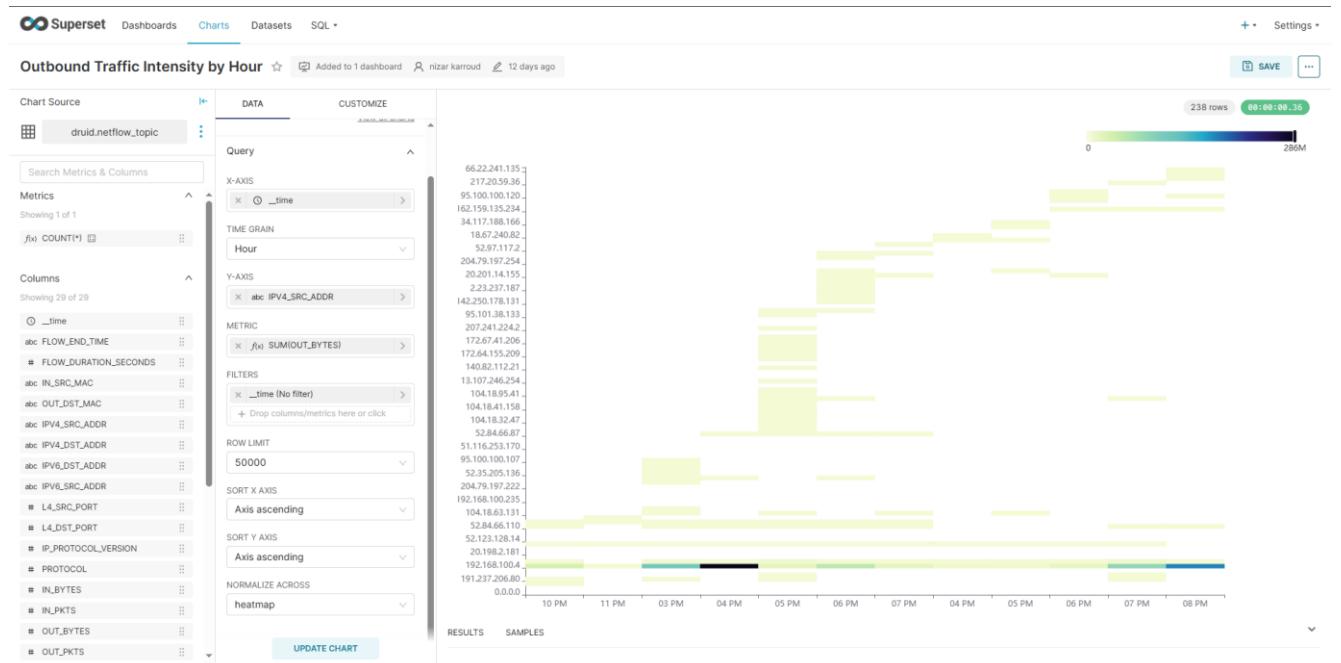


4.9.2 Intensité du trafic sortant par heure (carte thermique OUT_BYTES)

Cette carte thermique affiche le volume total du trafic sortant (OUT_BYTES) à travers les différentes heures de la journée, montrant combien de données sont envoyées depuis votre réseau à chaque heure.

Un trafic sortant élevé peut être un signe d'exfiltration de données, surtout s'il augmente de manière inattendue pendant les heures creuses ou d'une seule machine/IP.

Si un seul hôte ou une seule IP génère une quantité disproportionnée de trafic sortant par rapport aux autres, il peut être compromis et tenter d'envoyer des données sensibles à l'extérieur du réseau.

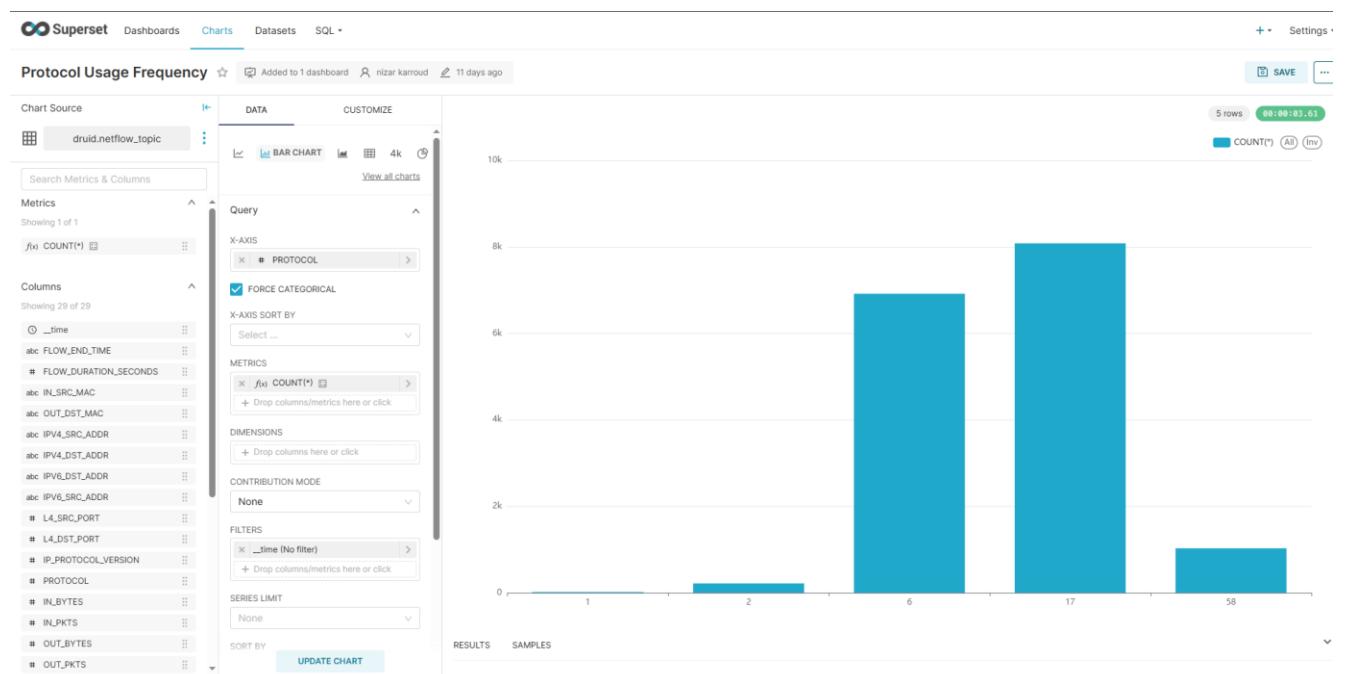


4.9.3 Analyse de la fréquence des protocoles réseau

Cette analyse vise à comprendre la distribution des différents protocoles utilisés dans votre réseau et à détecter des anomalies dans les flux de trafic .

En surveillant la fréquence de chaque protocole, vous pouvez identifier des comportements inhabituels, comme un usage anormalement élevé de certains protocoles qui pourrait être le signe d'une activité malveillante.

Si un protocole qui ne devrait pas être utilisé fréquemment dans votre réseau (par exemple, ICMP pour un réseau non administratif) présente une fréquence anormale, cela pourrait indiquer des tentatives de reconnaissance ou d'attaque réseau, comme un ping de déni de service ou des traces de scan de port.



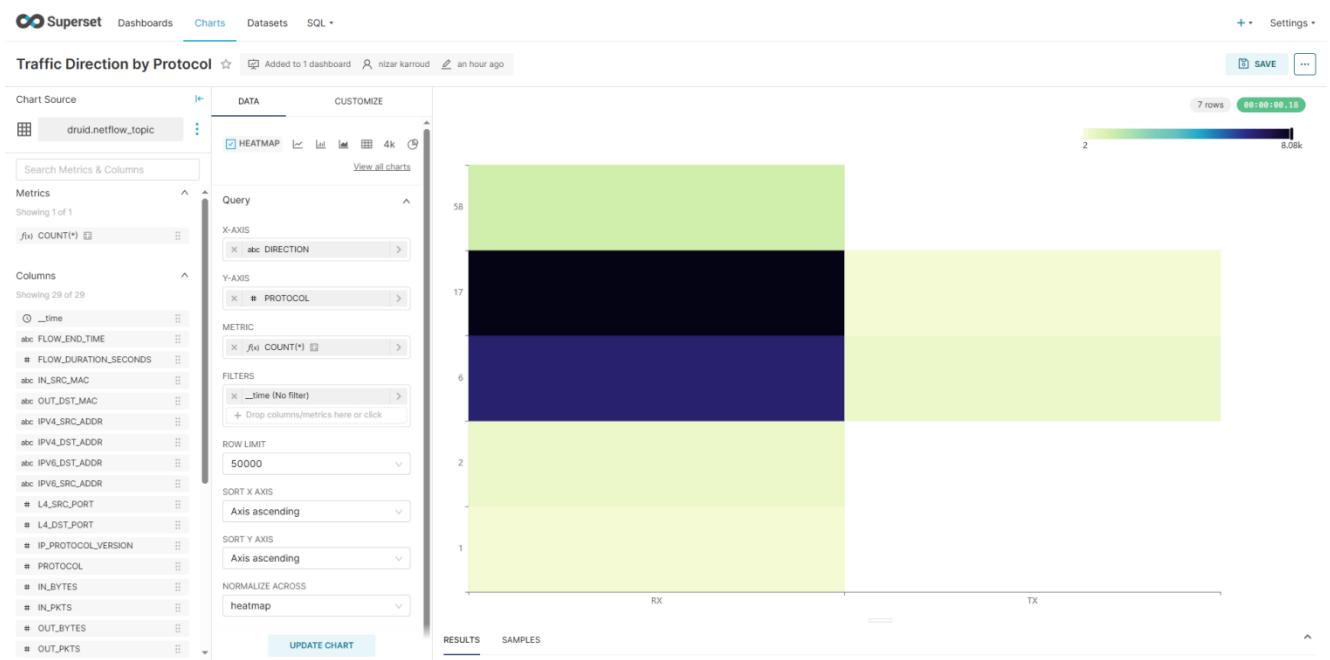
4.9.4 Carte de direction du trafic par protocole

Cette carte thermique permet de visualiser le volume ou la fréquence du trafic en fonction de la direction (entrant ou sortant) et du protocole utilisé.

Identifier des comportements anormaux, par exemple un trafic sortant élevé via un protocole inattendu comme HTTP ou DNS (exfiltration de données).

Repérer les protocoles utilisés principalement en entrée ou en sortie, ce qui peut révéler une mauvaise configuration de pare-feu .

Surveiller les modèles d'usage normaux et détecter des déviations, comme un afflux soudain de trafic entrant via un protocole rarement utilisé.

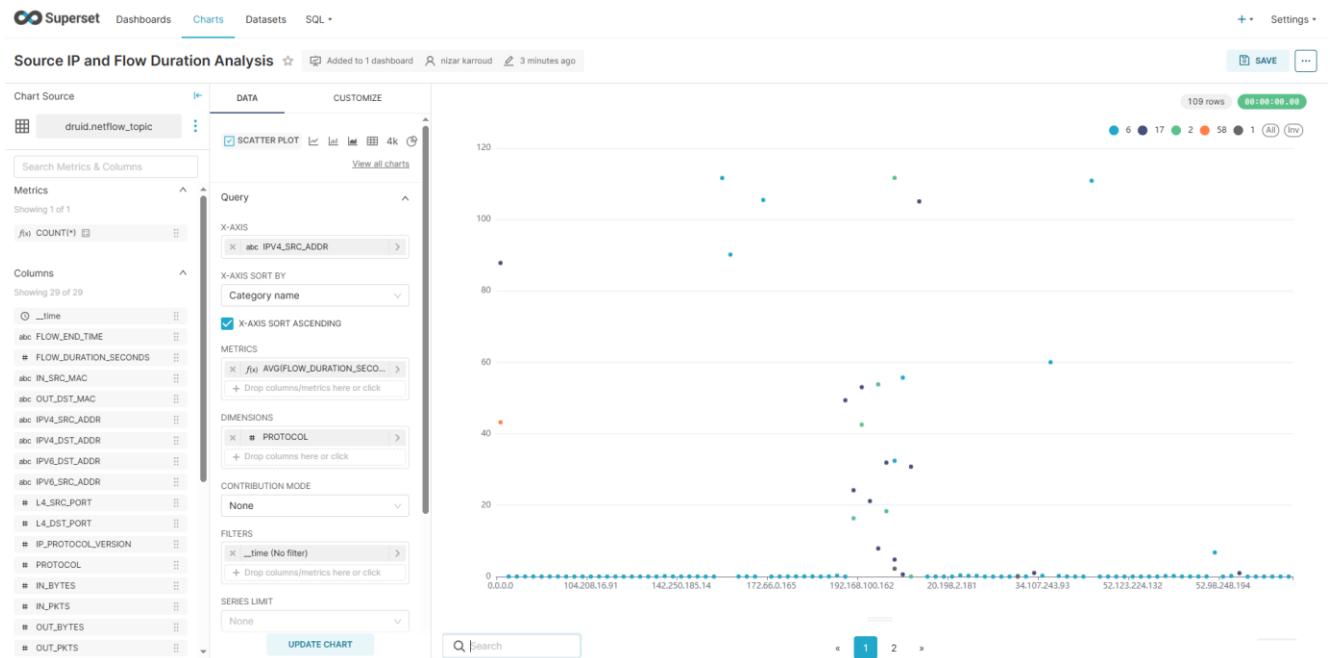


4.9.5 Analyse de l'adresse IP source et de la moyenne durée des flux

Ce nuage de points permet de visualiser la relation entre les adresses IP sources et la durée moyenne des connexions. Il aide à détecter :

Les hôtes qui génèrent des connexions longues, ce qui pourrait indiquer une activité anormale ou persistante

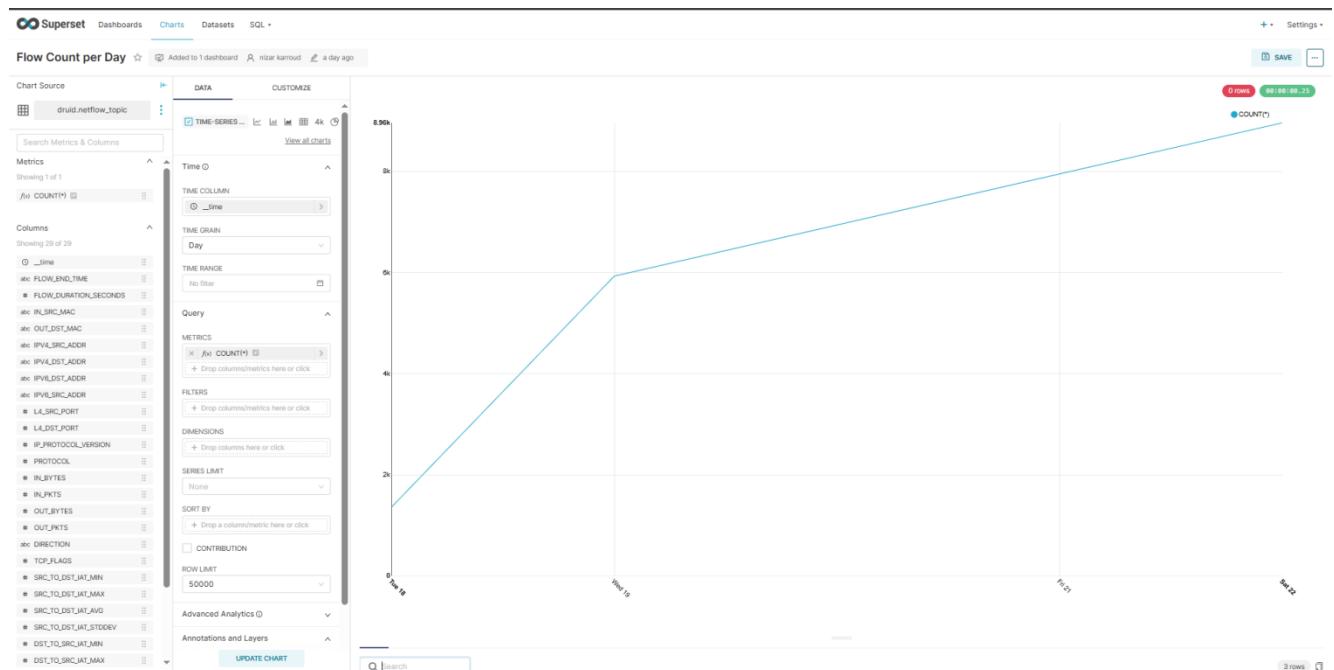
Les anomalies dans le trafic réseau, comme des connexions qui ne devraient pas durer aussi longtemps pour certains services ou utilisateurs.



nnn

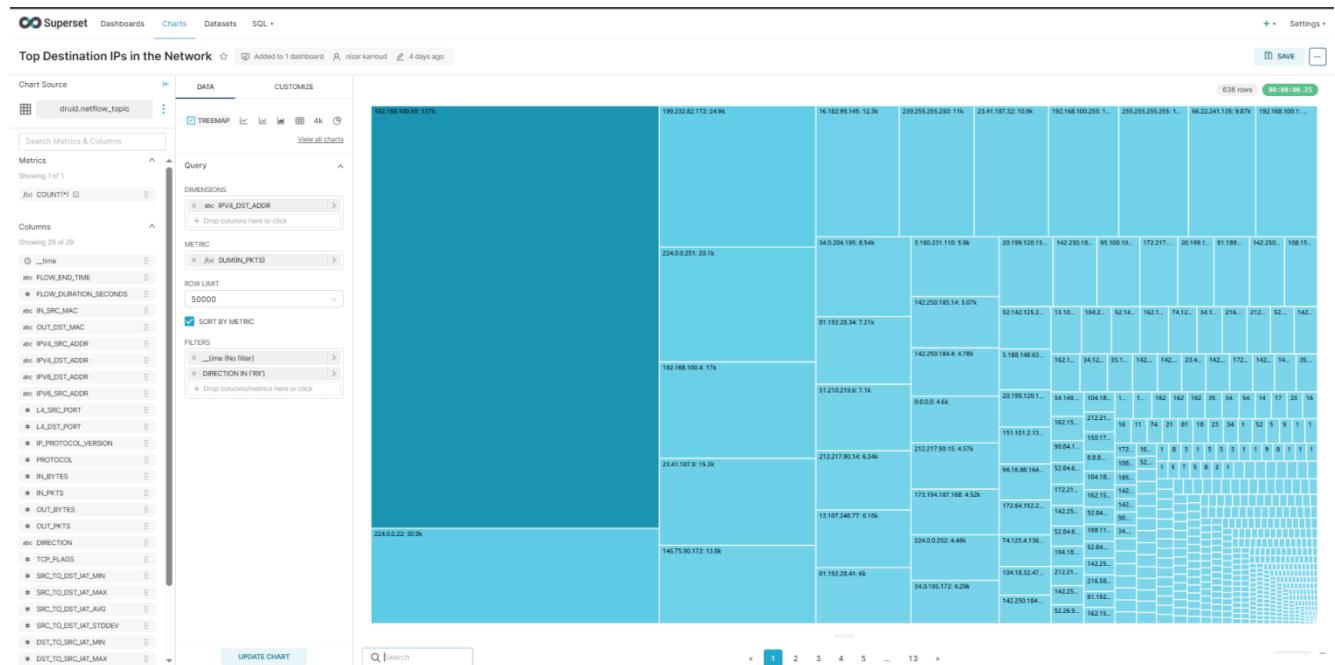
4.9.6 Nombre de flux par jour

Cette visualisation montre le nombre de flux par chaque jour, permettant ainsi d'analyser les variations du trafic réseau au quotidien. Elle aide à repérer des anomalies comme des pics de trafic inhabituels ou des périodes d'inactivité prolongées qui pourraient indiquer des activités suspectes. Par exemple, des flux élevés sur des jours spécifiques, comme un trafic anormalement élevé un jour de la semaine ou pendant le week-end, pourraient suggérer des comportements malveillants ou des attaques.



4.9.7 Carte en arbre des IP de destination dans le réseau

Cette visualisation permet de repérer les adresses IP de destination qui reçoivent le plus de trafic au sein de votre réseau. Un treemap est utile pour voir de manière hiérarchique quelles machines ou services internes sont les plus sollicités. Cela peut aider à identifier les points de charge élevés, à surveiller les ressources critiques du réseau, ou à détecter des comportements anormaux, comme une attention excessive de certaines adresses IP internes qui pourraient indiquer des actions malveillante



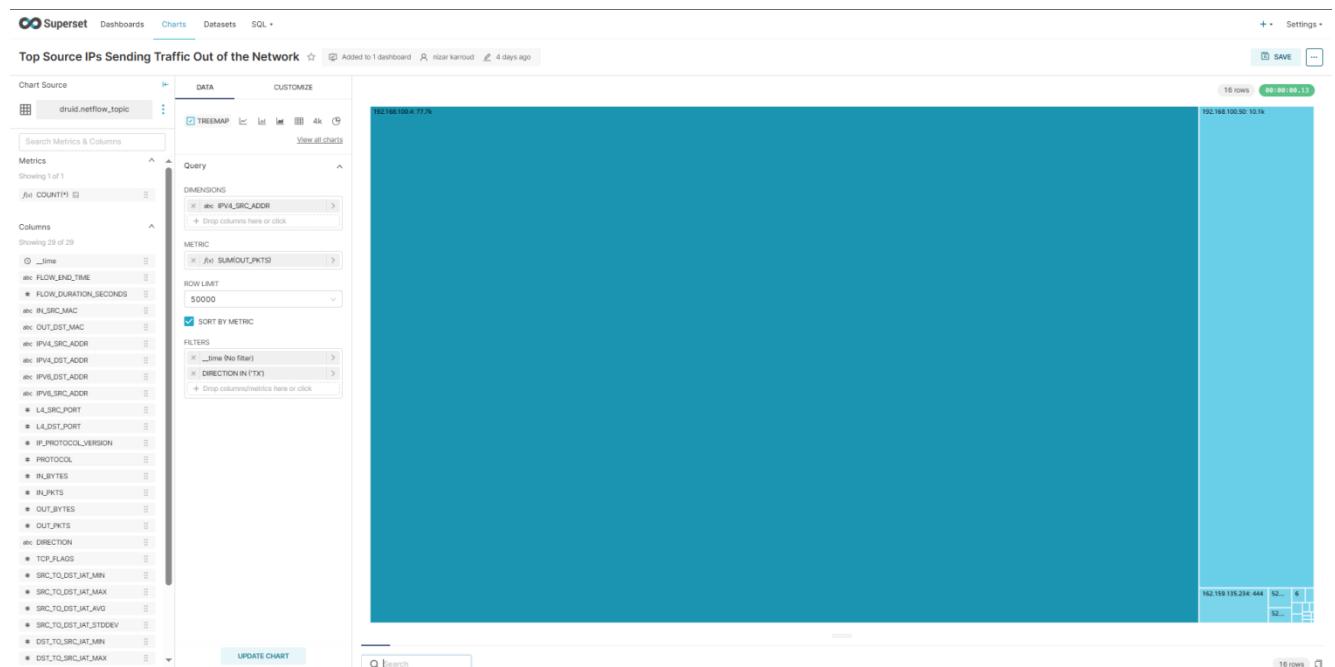
4.9.8 Carte en arbre des IP source envoyant du trafic vers le réseau

Cette visualisation permet d'identifier rapidement les sources principales de trafic qui atteignent votre réseau. Le treemap est particulièrement utile pour voir, de manière hiérarchique et visuellement distincte, quels sont les hôtes générant le plus de trafic entrant vers votre infrastructure.



4.9.9 Carte en arbre des IP source envoyant du trafic hors du réseau

Cette visualisation permet d'identifier les adresses IP sources qui génèrent le plus de trafic sortant depuis votre réseau. Un treemap aide à repérer les hôtes ou services internes qui envoient le plus de données vers l'extérieur. Cela peut être utile pour détecter des fuites de données, un comportement de type exfiltration ou d'autres anomalies, comme un serveur qui envoie une quantité de données anormalement élevée vers des destinations externes.



4.9.10 Carte en arbre des IP de destination recevant du trafic depuis le réseau

Cette visualisation permet de repérer les adresses IP de destination qui reçoivent le plus de trafic sortant depuis votre réseau. Un treemap est particulièrement utile pour identifier les hôtes externes avec lesquels votre réseau échange le plus de données. Cela peut être utile pour surveiller les activités sortantes vers des destinations spécifiques, telles que des sites web ou des serveurs externes.



On a ainsi abouti à la création du tableau de bord suivant, synthétisant visuellement les analyses effectuées à partir du trafic réseau chiffré.



Conclusion

Face à la généralisation du chiffrement des communications réseau, l'analyse traditionnelle basée sur l'inspection des paquets devient obsolète et inefficace. Ce rapport a mis en lumière les limites de ces approches, notamment en termes de coût, de respect de la vie privée, et de complexité croissante face aux nouvelles méthodes d'évasion utilisées par les acteurs malveillants.

L'étude des solutions existantes comme Cisco Stealthwatch, Darktrace ou Palo Alto Cortex XDR a permis d'identifier des approches puissantes mais souvent coûteuses, dépendantes d'un écosystème fermé ou encore opaques dans leur fonctionnement. Bien qu'efficaces, elles ne répondent pas toujours aux besoins de flexibilité, de transparence et d'accessibilité des petites structures ou des chercheurs.

Dans ce contexte, le projet proposé offre une alternative fondée sur l'analyse comportementale et l'exploitation des métadonnées réseau. En se concentrant sur des indicateurs tels que les volumes de trafic, les durées de flux, les adresses IP ou les ports, et en utilisant des visualisations adaptées et l'apprentissage automatique, il est possible de détecter des anomalies sans briser le chiffrement.

Reposant sur des outils open-source et une architecture modulaire, cette solution s'inscrit dans une démarche de souveraineté technologique et d'ouverture. Elle permet une surveillance continue du trafic chiffré tout en respectant la confidentialité, et constitue une base solide pour développer des mécanismes de détection plus intelligents, évolutifs et personnalisables.

Ce travail contribue ainsi à démontrer qu'il est possible d'analyser efficacement un trafic réseau chiffré sans en compromettre l'intégrité, et ouvre la voie à de futures explorations dans le domaine de la détection d'anomalies basée sur les comportements réseau.

Bibliographie

<https://hevodata.com/learn/kafka-clusters/>

<https://www.slingacademy.com/article/an-introduction-to-apache-kafka-event-streaming/>

<https://medium.com/@clasikas/kafka-in-docker-container-and-command-line-67bb0eb2d>

<https://www.ntop.org/guides/nprobe/index.html>

<https://kafka.apache.org/24/documentation.html>

<https://link.springer.com/book/10.1007/978-3-031-62909-9>

Références

1. <https://transparencyreport.google.com/https/overview?hl=en>
2. So-In, C. (2009). A survey of network traffic monitoring and analysis tools. In CSE 576m computer system analysis project. Washington University in St. Louis
3. <https://www.nsnam.org>
4. <https://nmap.org/nping>
5. <https://ostinato.org>
6. <https://www.solarwinds.com/engineers-toolset/use-cases/traffic-generator-wan-killer>
7. <https://www.apposite-tech.com/products/netropy-network-emulation/>
8. <https://bittwist.sourceforge.io/>
9. <https://tcpreplay.appneta.com>
10. Wang, Z., Fok, K. W., & Thing, V. L. (2022). Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. Computers & Security, 113, 102542.