

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет» РТУ МИРЭА**

**Институт кибербезопасности и цифровых технологий**

*(наименование института, филиала)*

**Кафедра КБ-2«Информационно-аналитические системы кибербезопасности»**

*(наименование кафедры)*

**ОТЧЕТ**

по дисциплине: «Технологии хранения в системах кибербезопасности»

Задание получил:

III курс, группа БИСО-02-23

*Подпись*

Макаревич Сергей Витальевич

*ФИО*

Проверил:

|  |  |
| --- | --- |
| « » 2025 г. |  |
| *Дата* | *Подпись* |

|  |  |
| --- | --- |
|  | Селин А. А. |
| *Отметка / результат* | *ФИО* |

Москва 2025 г.

Оглавление

[1. Запуск виртуальной машины с Ubuntu Server 24.04, подключение к ней по ssh используя ПО Termius для удобства взаимодействия. 3](#_Toc213428865)

[2. Создание пользователя и настройка окружения 3](#_Toc213428866)

[3. Установка Docker и Docker Compose 4](#_Toc213428867)

[4. Настройка параметров ядра(Постоянное изменение) 5](#_Toc213428868)

[5. Создание docker-compose.yml для OpenSearch 5](#_Toc213428869)

[6. Запуск OpenSearch 6](#_Toc213428870)

[7. Установка TShark 7](#_Toc213428871)

[8. Скачивание образцов трафика 7](#_Toc213428872)

[9. Обработка PCAP-файлов с помощью TShark 8](#_Toc213428873)

[10. Загрузка данных используя Python скрипт. 10](#_Toc213428874)

[11. Запуск панели, подготовка OpenSearchDashboards. 11](#_Toc213428875)

[12. Выполним поиск в разделе Discover. 12](#_Toc213428876)

[13. Создадим несколько визуализаций. 14](#_Toc213428877)

[14. Изучим OpenSearch Alerting. 18](#_Toc213428878)

[15. Изучим возможности OpenSearch Dashboards по анализу гео-данных 19](#_Toc213428879)

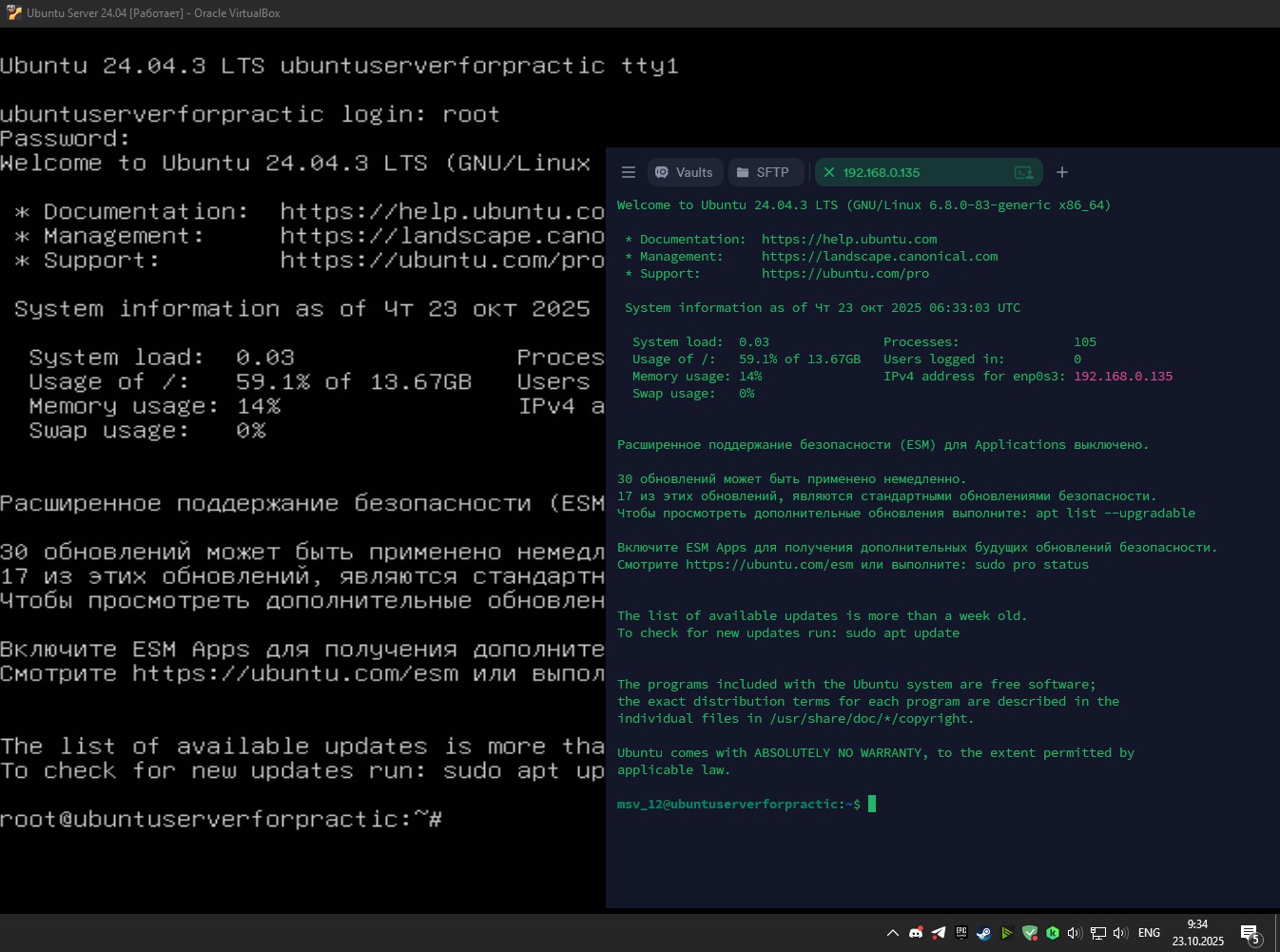
[16. Заключение 22](#_Toc213428880)

ПРАКТИЧЕСКАЯ РАБОТА № 4

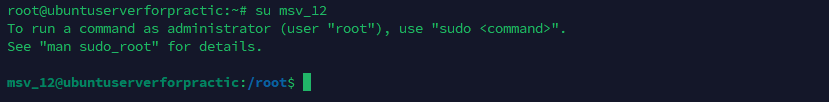
«Знакомство с инструментами для работы с частично структурированными данными на примере документо-ориентированной СУБД MongoDB»

Цель работы – получение навыков развертывания приложений с использованием Docker.

# Запуск виртуальной машины с Ubuntu Server 24.04, подключение к ней по ssh используя ПО Termius для удобства взаимодействия.

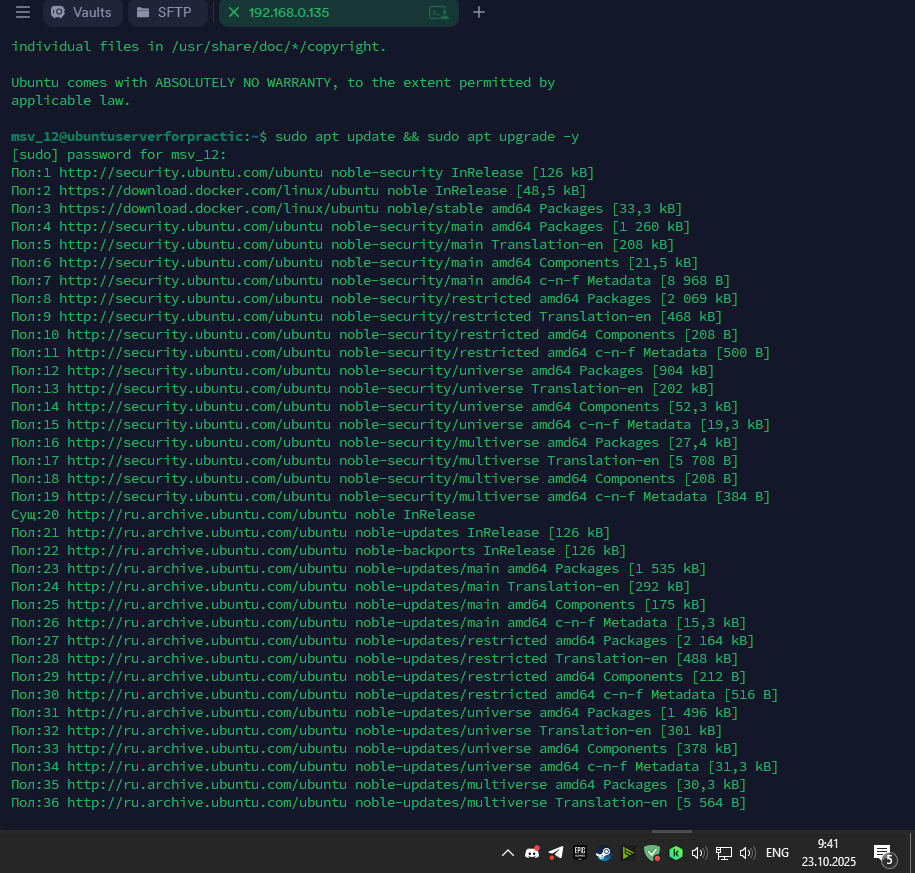


# Создание пользователя и настройка окружения

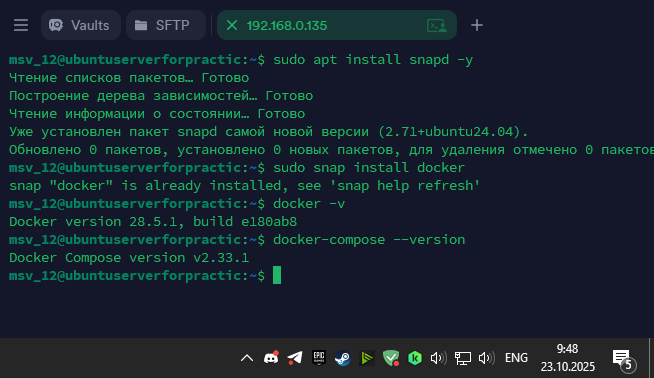


# Установка Docker и Docker Compose

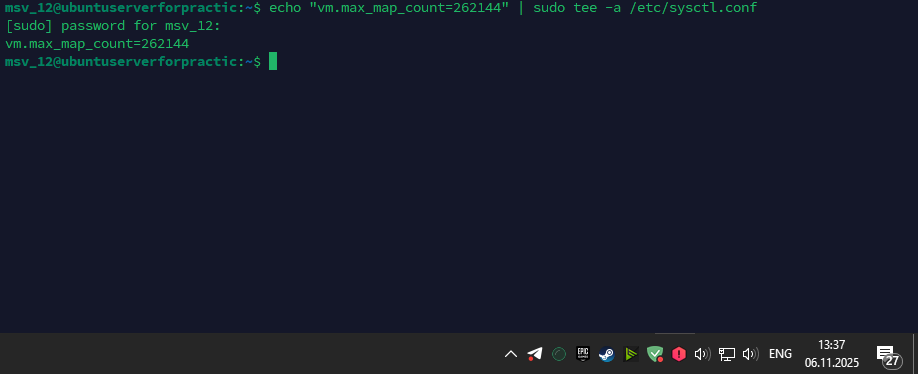
* 1. Обновление пакетов и установка зависимостей



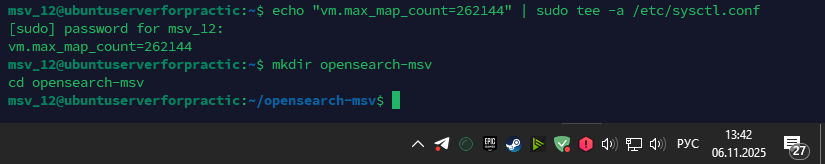
* 1. Установка Docker и Docker Compose через snap. Проверка установки.

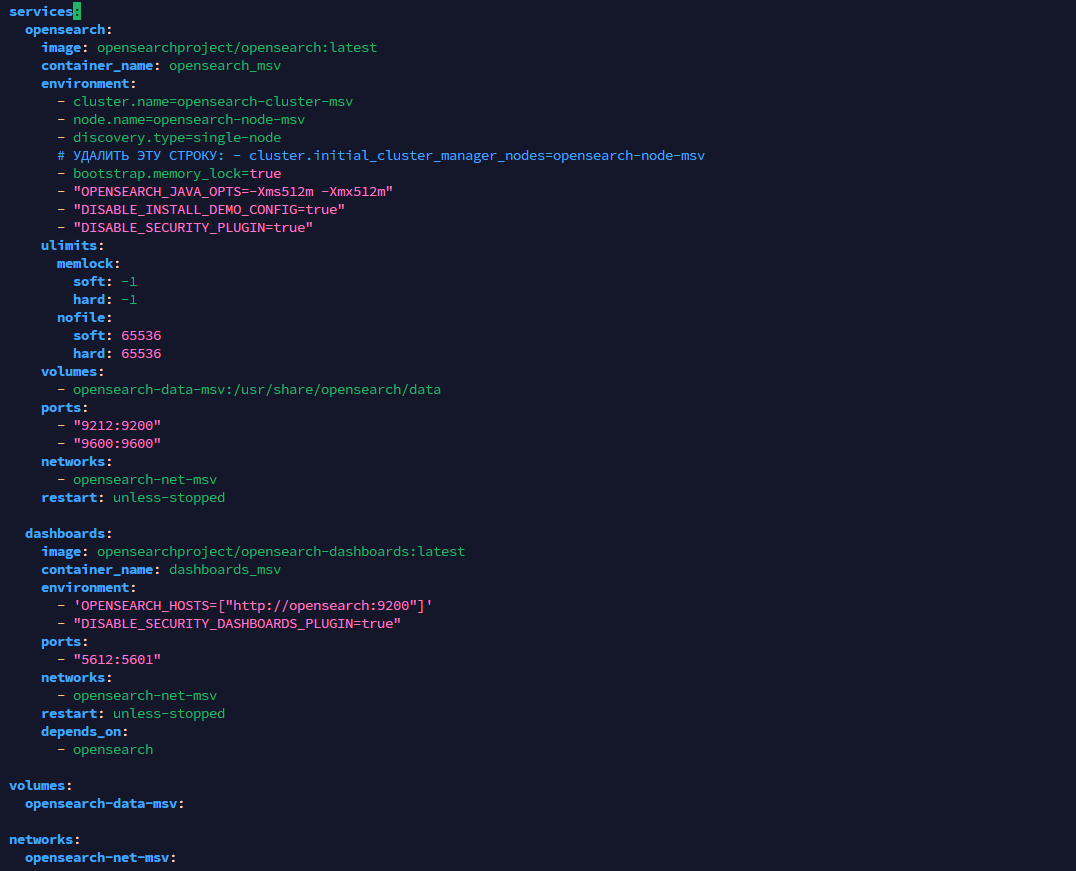


# Настройка параметров ядра(Постоянное изменение)

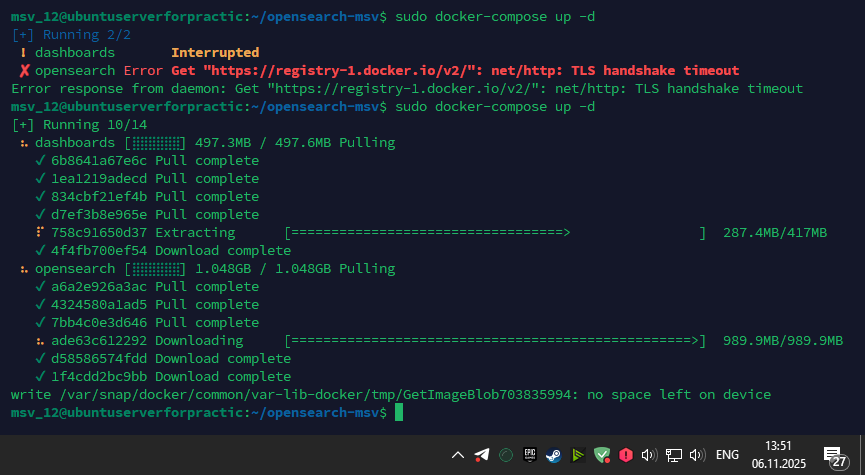


# Создание docker-compose.yml для OpenSearch

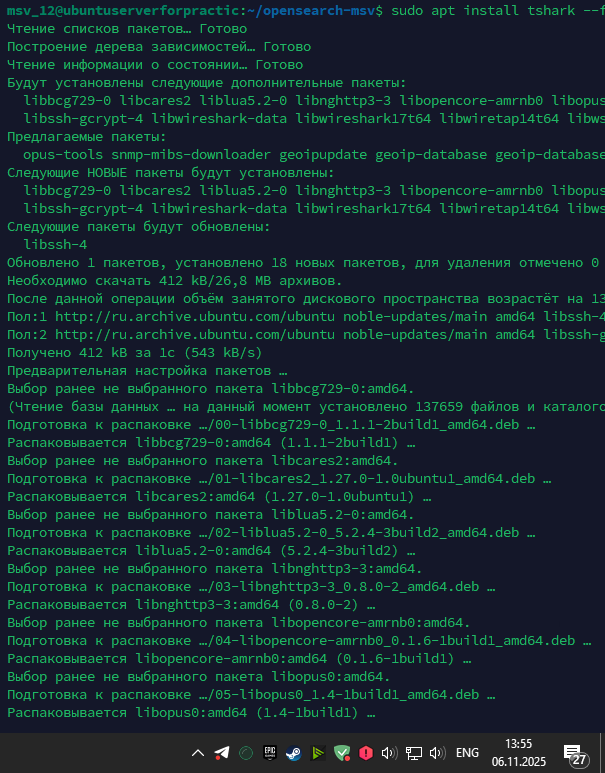




# Запуск OpenSearch



# Установка TShark



# Скачивание образцов трафика

В процессе поиска файлов на сайте <https://www.malware-traffic-analysis.net> были найдены три наиболее подходящих по условию задания:  
1. <https://www.malware-traffic-analysis.net/2023/02/13/2023-02-13-IcedID-traffic-carved-and-sanitized.pcap.zip>

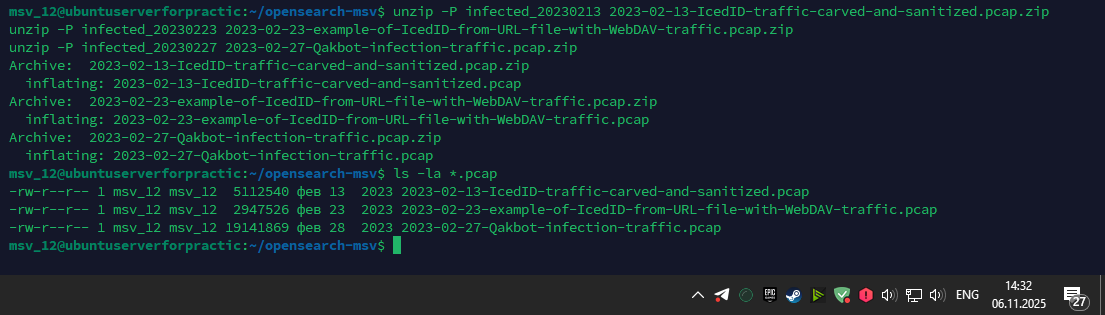
2. <https://www.malware-traffic-analysis.net/2023/02/23/2023-02-23-example-of-IcedID-from-URL-file-with-WebDAV-traffic.pcap.zip>

3. <https://www.malware-traffic-analysis.net/2023/02/27/2023-02-27-Qakbot-infection-traffic.pcap.zip>

Скачиваем архивы используя wget:

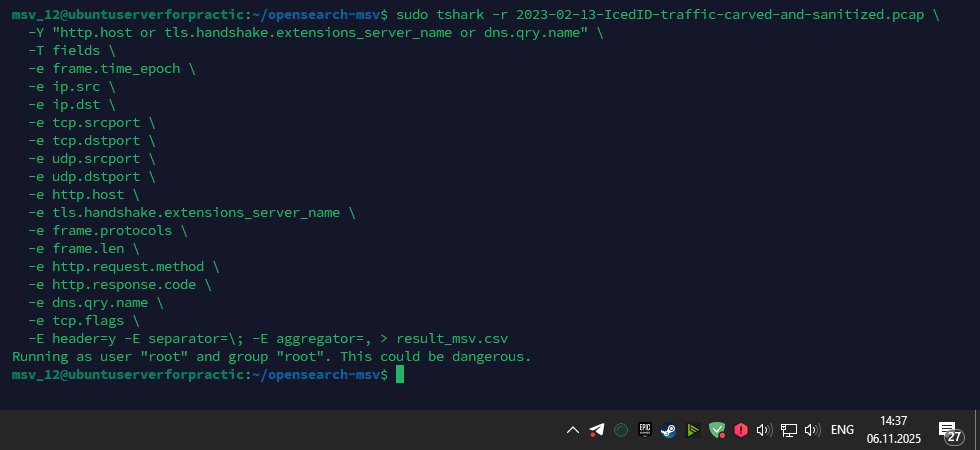


Теперь распакуем архивы:

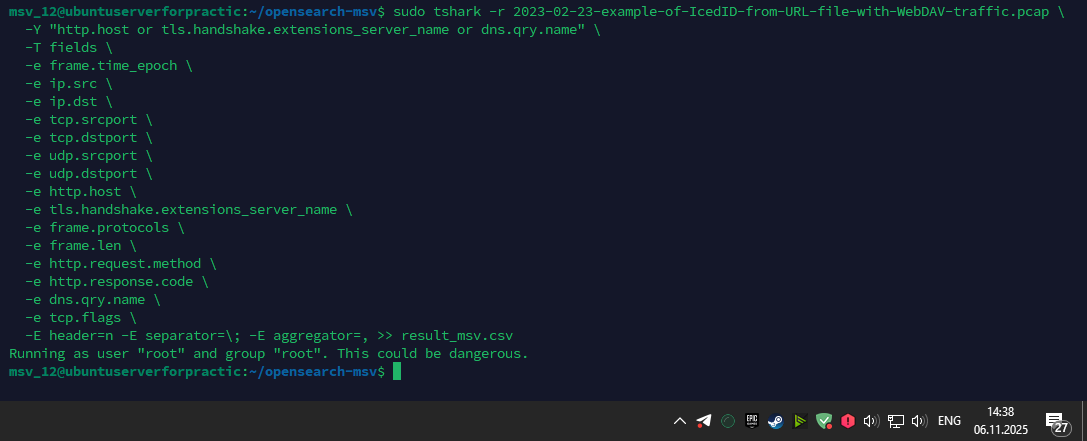


# Обработка PCAP-файлов с помощью TShark

* 1. Первый файл



* 1. Второй файл



* 1. Третий файл

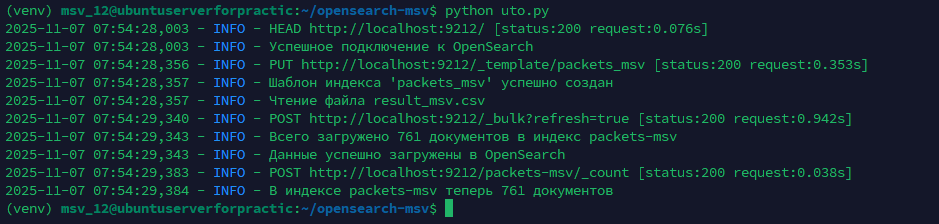


Обоснование по каждому дополнительно выбранному полю:

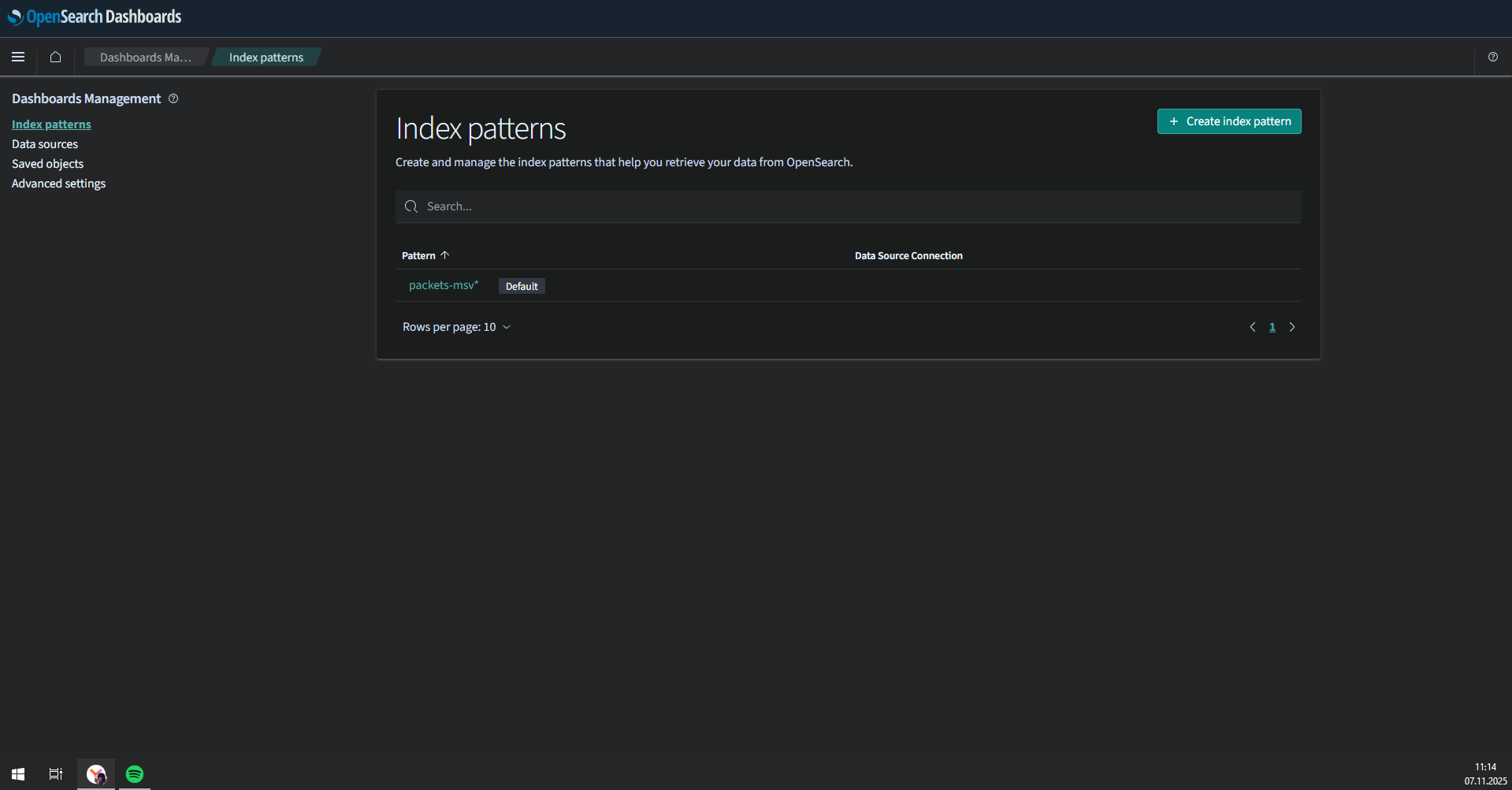
1. http.request.method:  
   Позволяет определить тип выполняемого HTTP-запроса (GET, POST, PROPFIND и др.). Помогает выявить подозрительную активность, например, использование методов WebDAV (PROPFIND, PUT) или массовые автоматизированные GET-запросы, характерные для сканирования и C2-трафика.
2. http.response.code:  
   Отражает результат обработки запроса сервером (200, 404, 500 и др.). Позволяет оценить успешность выполнения операций и обнаружить ошибки, связанные с атаками или сканированием. Анализ сочетания с методом запроса помогает понять контекст действий (например, POST + 200 — возможная передача данных).
3. dns.qry.name:  
   Показывает доменные имена, к которым выполняются обращения. Используется для выявления вредоносных доменов, DGA-активности и аномальных запросов. Помогает сопоставить IP-адреса с доменами и анализировать сетевое взаимодействие на уровне имён.
4. ssh.protocol:  
   Содержит информацию о версии и типе SSH-протокола. Позволяет отслеживать активность удалённого доступа, выявлять несанкционированные подключения или попытки подбора паролей, а также использование устаревших версий SSH.
5. ftp.request.command:  
   Отражает команды FTP (USER, PASS, STOR, RETR и др.). Полезно для анализа передачи файлов, обнаружения попыток эксфильтрации данных или брутфорса. Позволяет отследить загрузку или выгрузку подозрительных файлов.

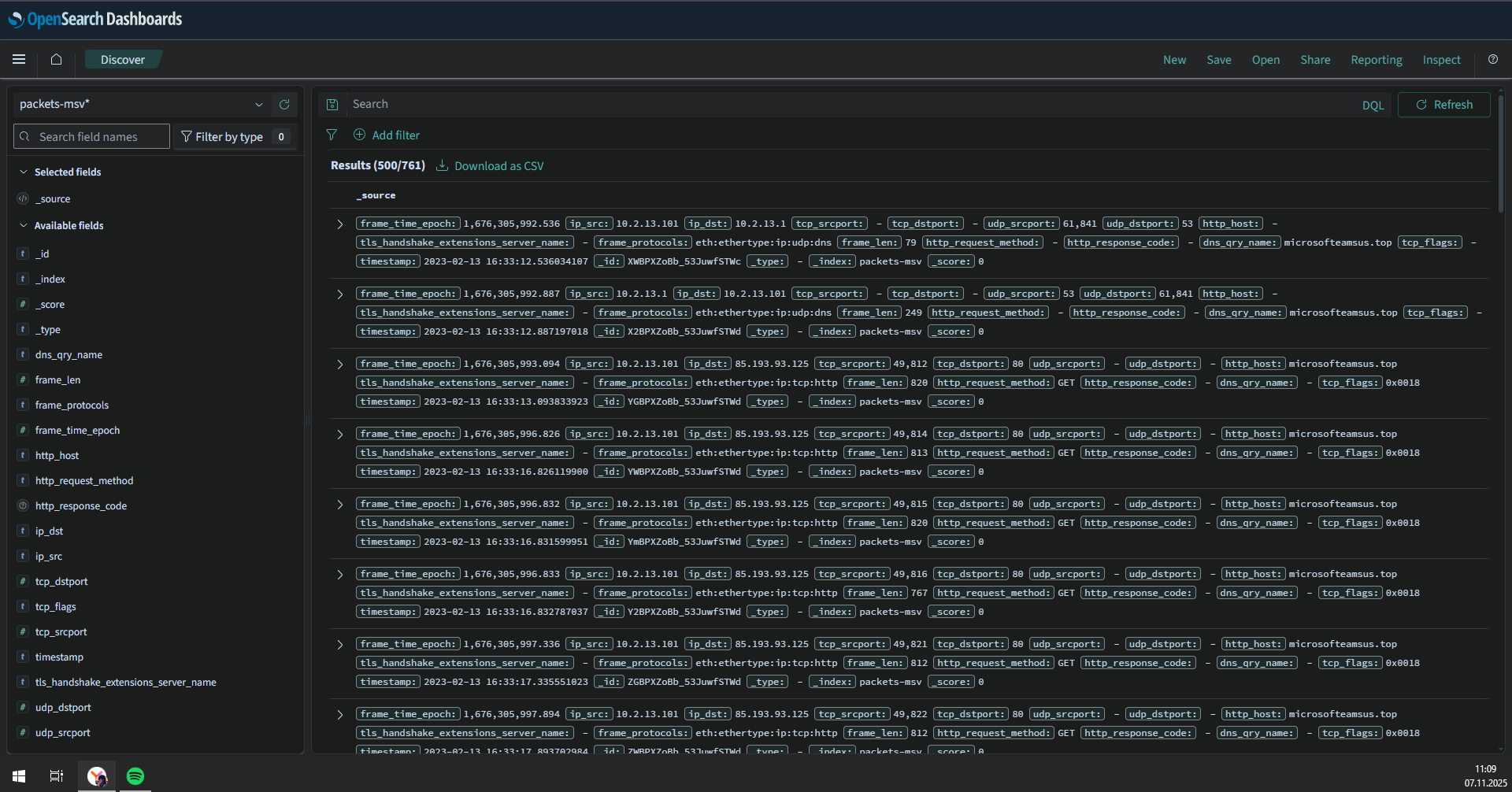
# Загрузка данных используя Python скрипт.

Используемый Python скрипт:  
<https://github.com/l1ratch/WC_BISO/blob/main/5_sem/ТХвСК/practic/pr_5/uto.py>

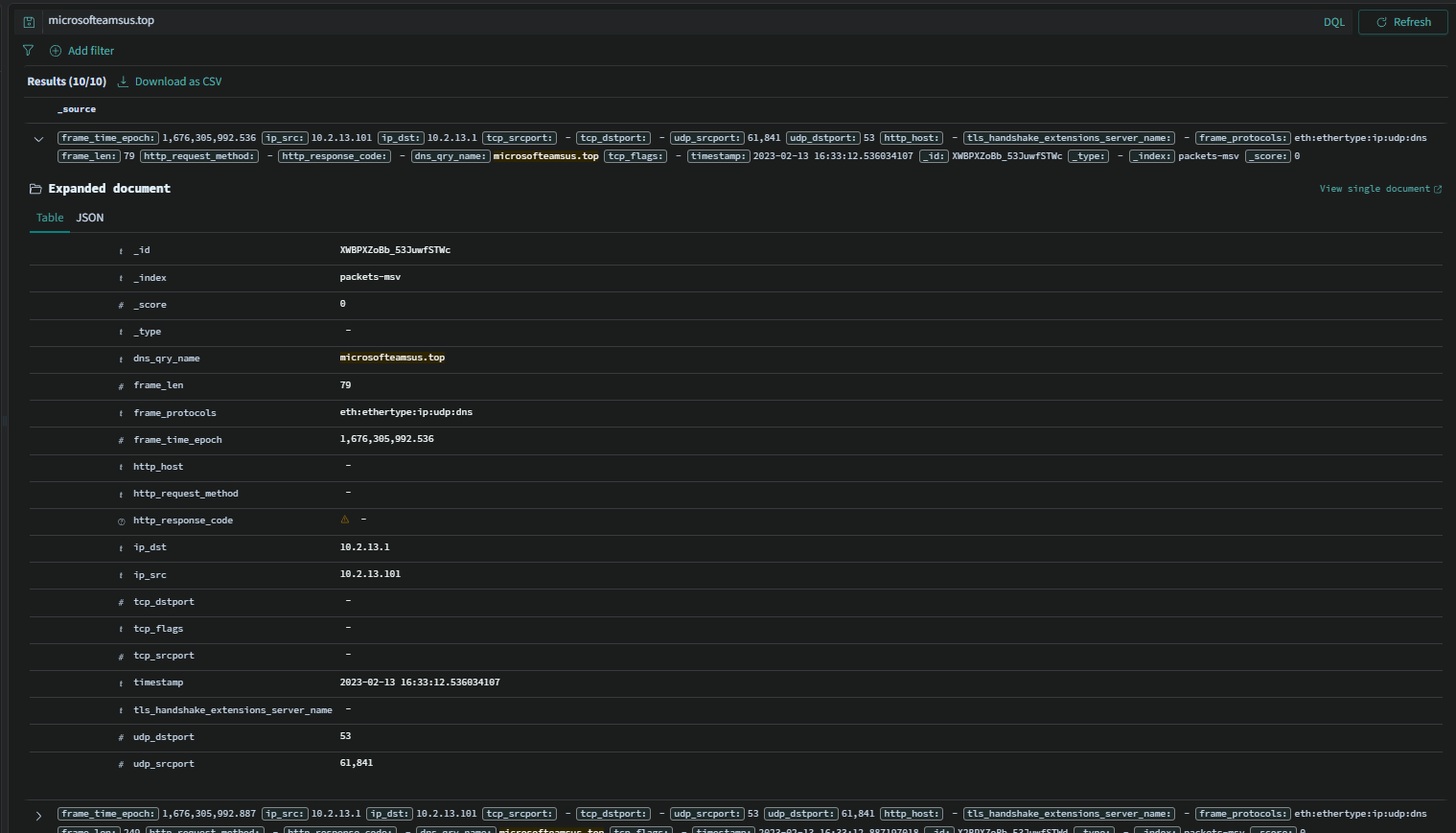


# Запуск панели, подготовка OpenSearchDashboards.

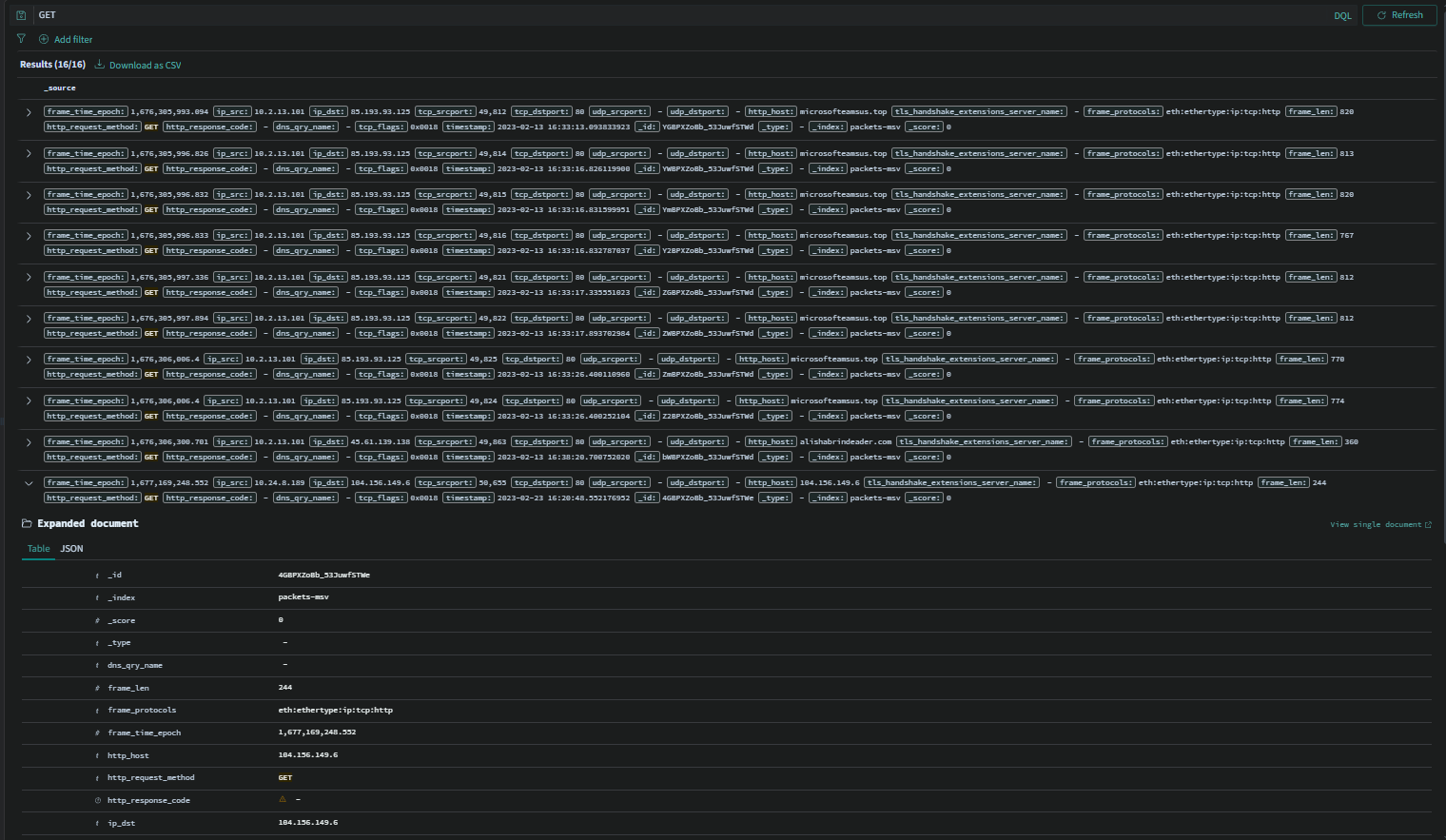


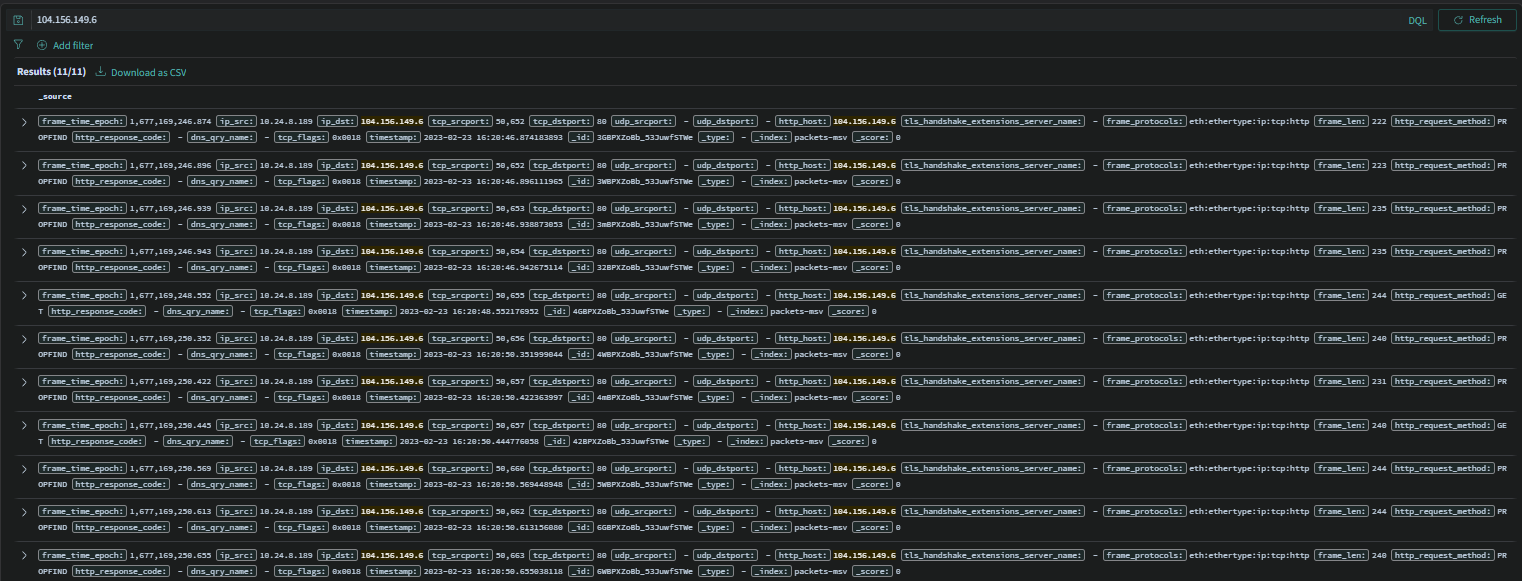


# Выполним поиск в разделе Discover.



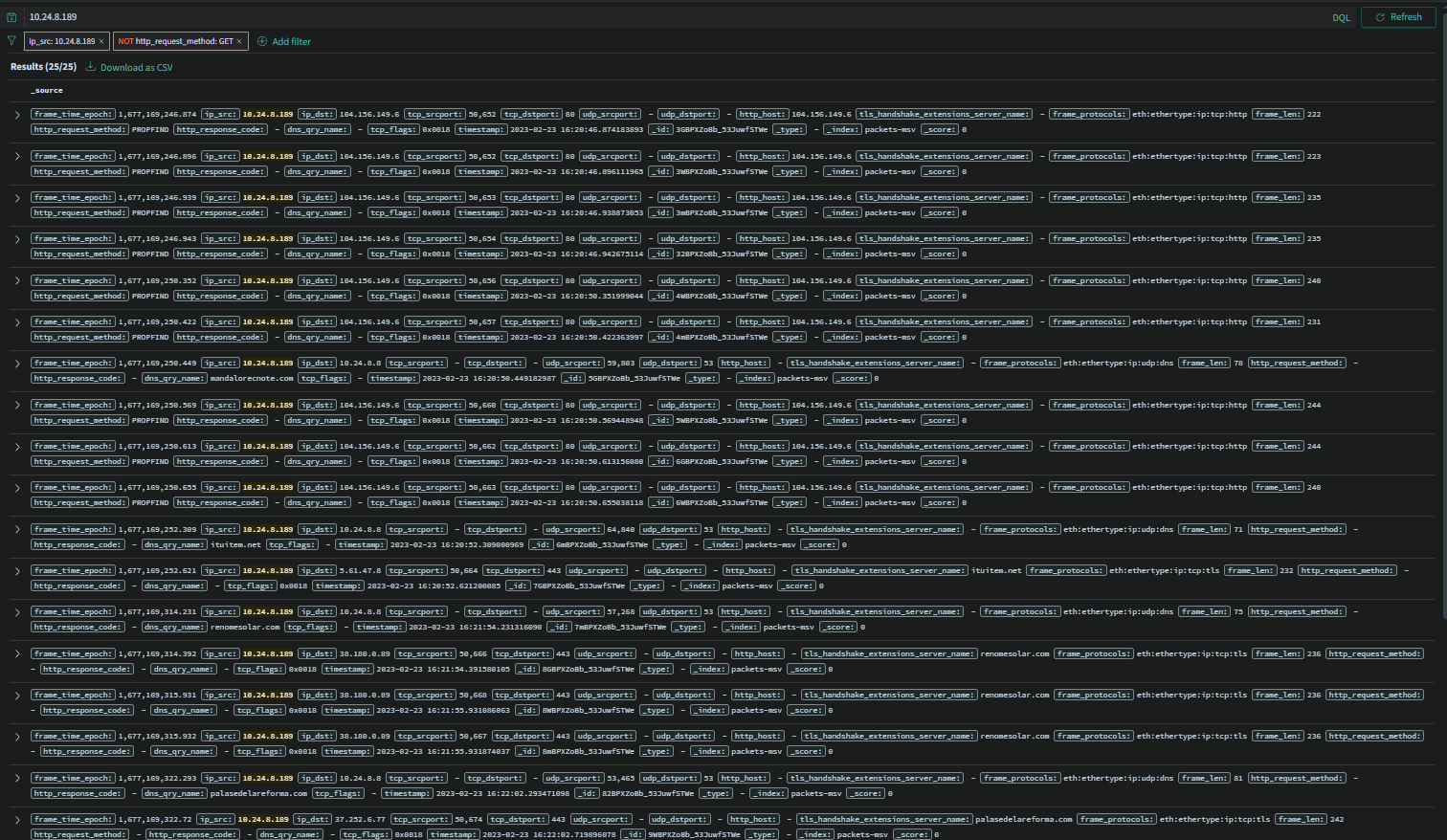
Попробовал функцию поиска, выполнил поисковый запрос по одному из найденных доменов microsofteamsus.top. Запросы используют метод GET.



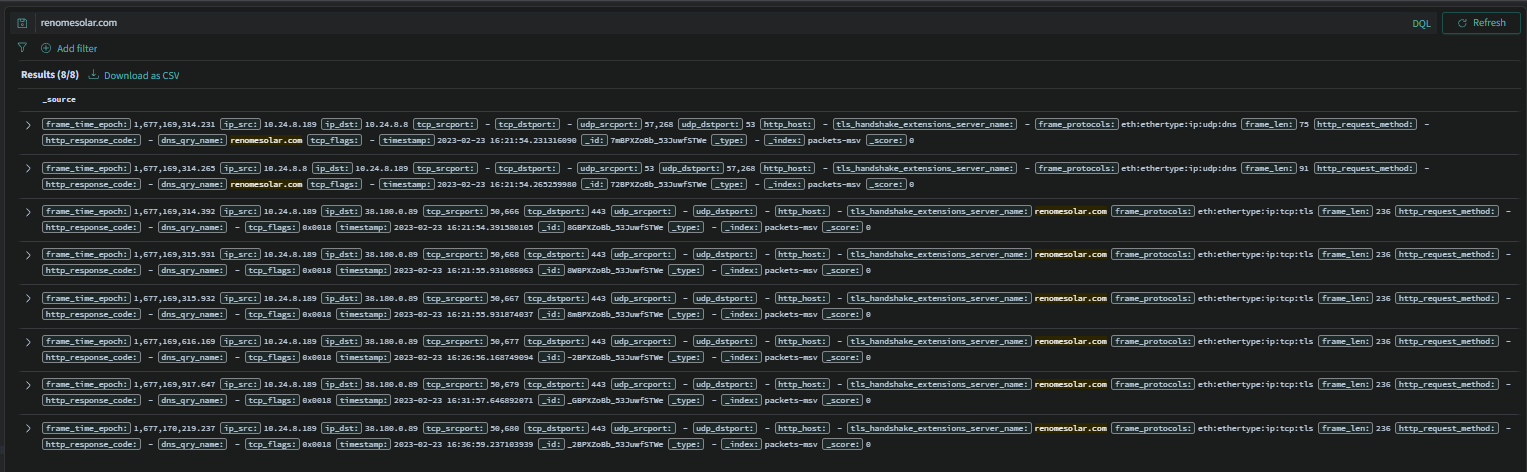


Решил изучить все GET запросы. Нашел запросы от http хоста 104.156.149.6. Изучив вывод поиска по хосту заметил, что все выполненные им запросы практически идентичны и используют метод GET, предполагаю, что это скрипт.

Выполню поиск по ip-источнику этих запросов(10.24.8.189) исключив GET запросы:



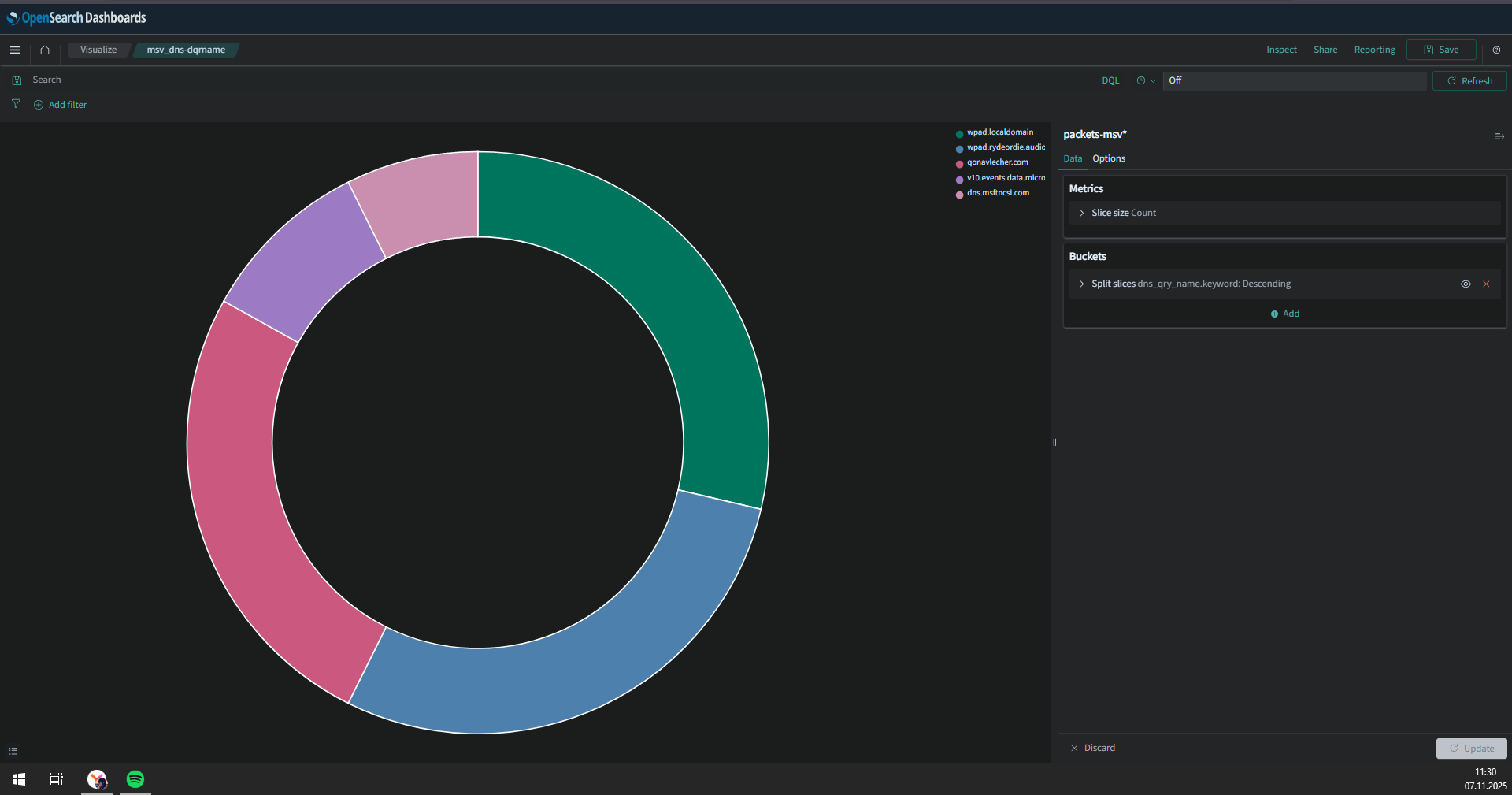
Здесь в основном запросы на порты 443 и 53. Предполагаю, что это хаб или другое устройство организующее устройства в сети, возможно устрйоства на котором расположен dns сервер.



Нашел запросы к 38.180.0.89 по 443, связанные с renomesolar.com. Первые два запроса являются dns запросами. Сначала 10.24.8.189 отправил запрос 10.24.8.8 на 53 порт, затем получил ответ от 10.24.8.8 и начал отправлять запросы 38.180.0.89 на 443 порт.  
Хороший пример того как работает dns сервер. Можем сделать вывод, что 10.24.8.189 является одним из устройств в сети, а 10.24.8.8 является локлаьным dns сервером этой сети.

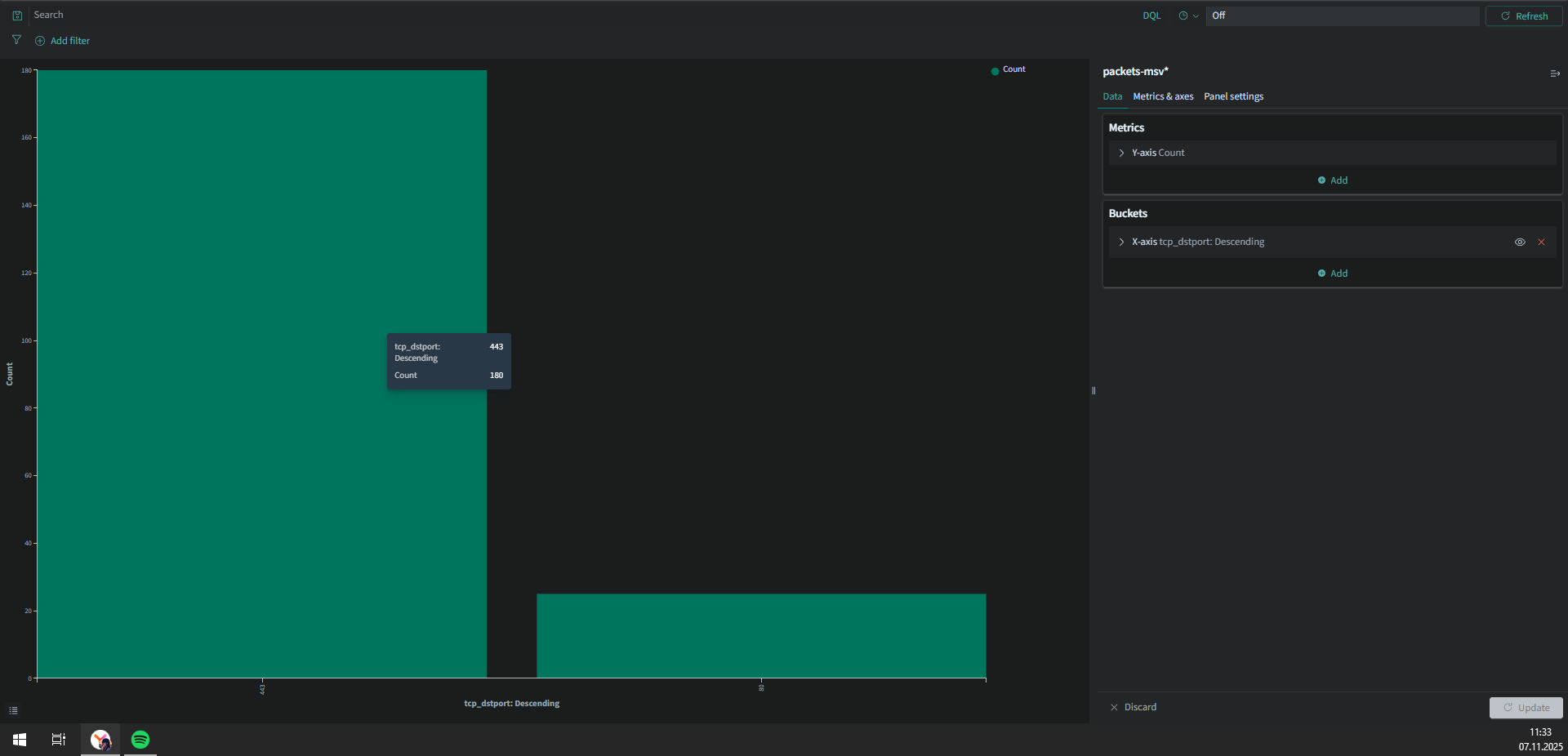
# Создадим несколько визуализаций.

* 1. По полю dns\_qry\_name.keyword



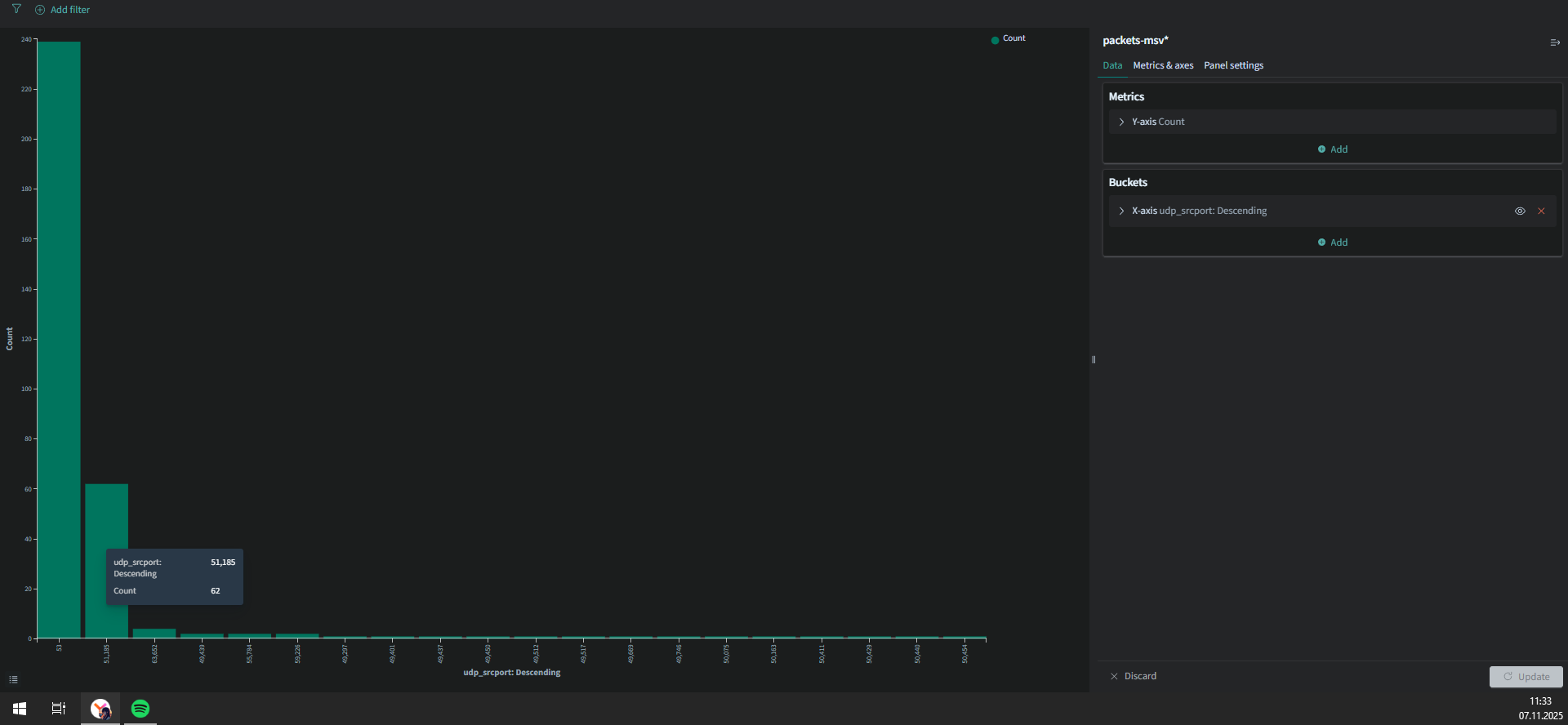
Чаще всего запросы шли к dns wpad.localdomain и wpab.rydeordie.audio – по 78 запросов.  
 Чуть чаще к qonavlecher.com – по 70, и совсем редко к двум от компании Microsoft.

* 1. По полю tcp\_dstport



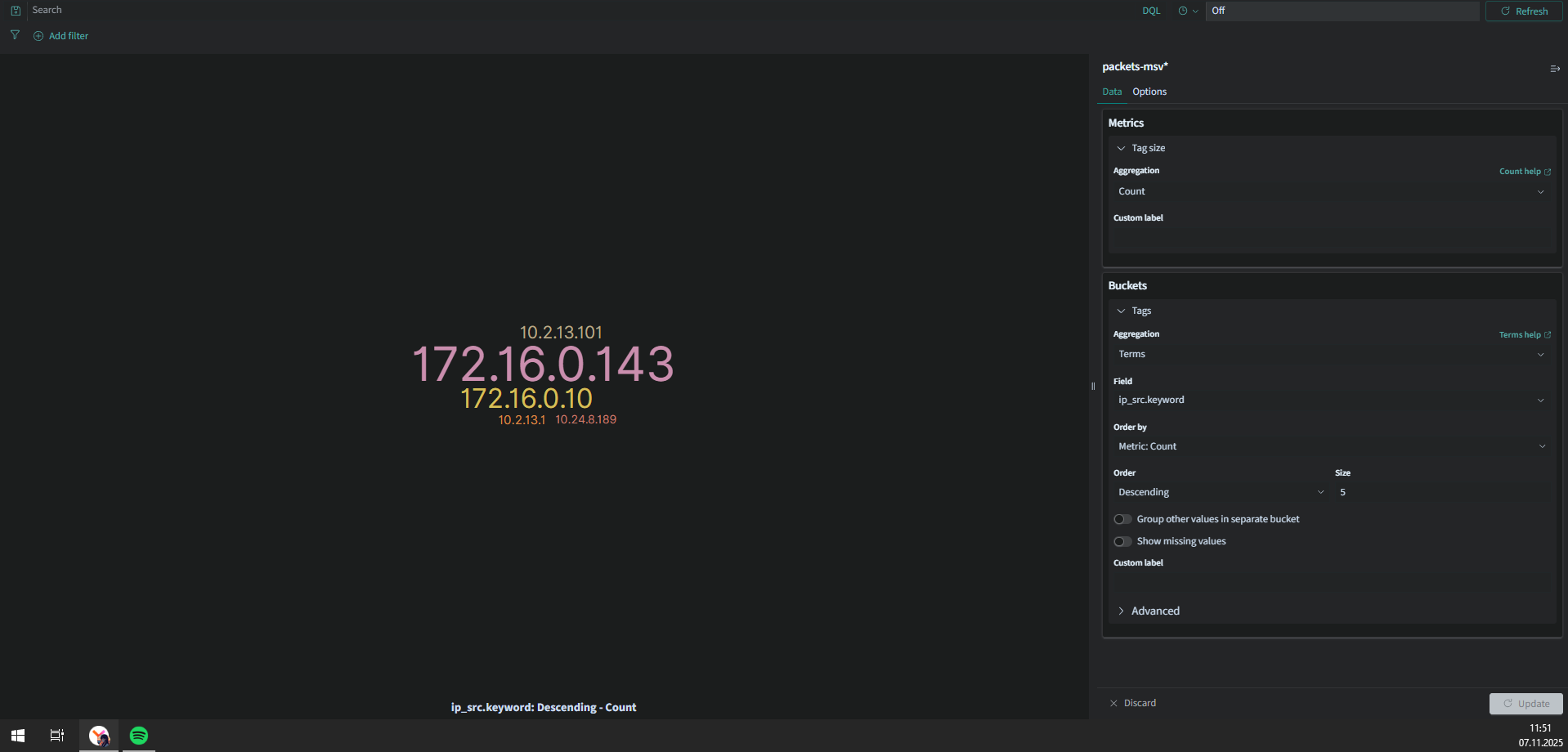
На данной диаграмме видно, что наиболее часто используемым dst портом является 443(180 запросов) из чего можно сделать вывод что в основном все запросы были по защищенному протоколу, лишь 25 запросов относятся к 80 порту.

* 1. По полю udp\_srcport



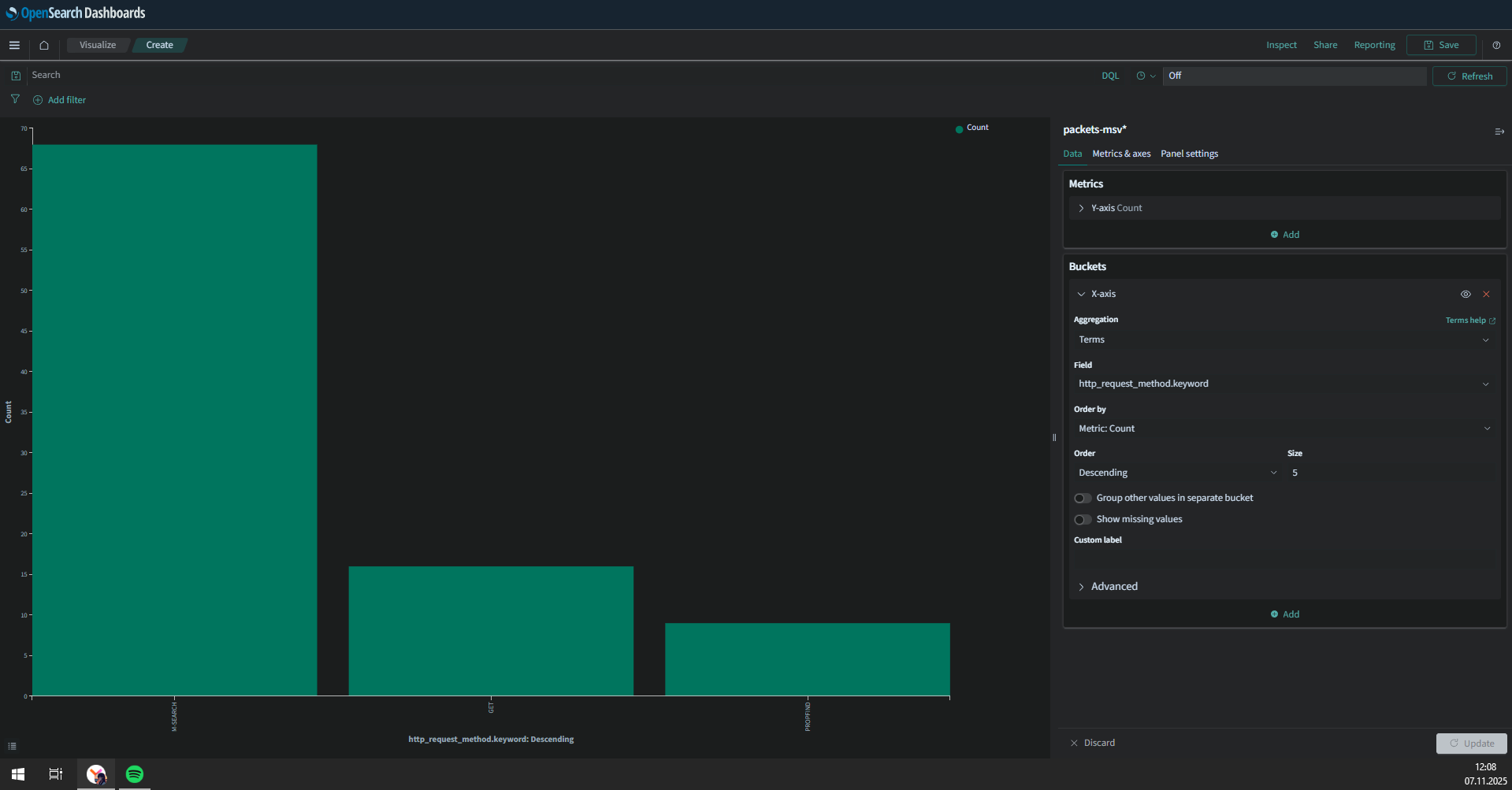
На данной диаграмме можно заметить, что самым часто используемым портом является 53 - порт который чаще всего используется для dns серверов(239 запросов), на втором месте 51185 порт у которого всего 62 запроса. Следом уже идут другие по 4, 2 и 1 запросам.  
(По исследованным диаграммам можно сделать предположение, что на устройстве или в локальной сети устройства был запущен локальный DNS-сервер. Вероятно для отслеживания запросов.)

* 1. По полю ip\_src



На это диаграмме мы видим вывод с сортировкой по частоте использования ip адреса источников запросов – локальные адреса с которых выполняются запросы. Наиболее часто используемым является 172.16.0.143 – который вероятно является одним из основных адресов устройства в его локальной сети.

* 1. По полю http\_rewuest\_method

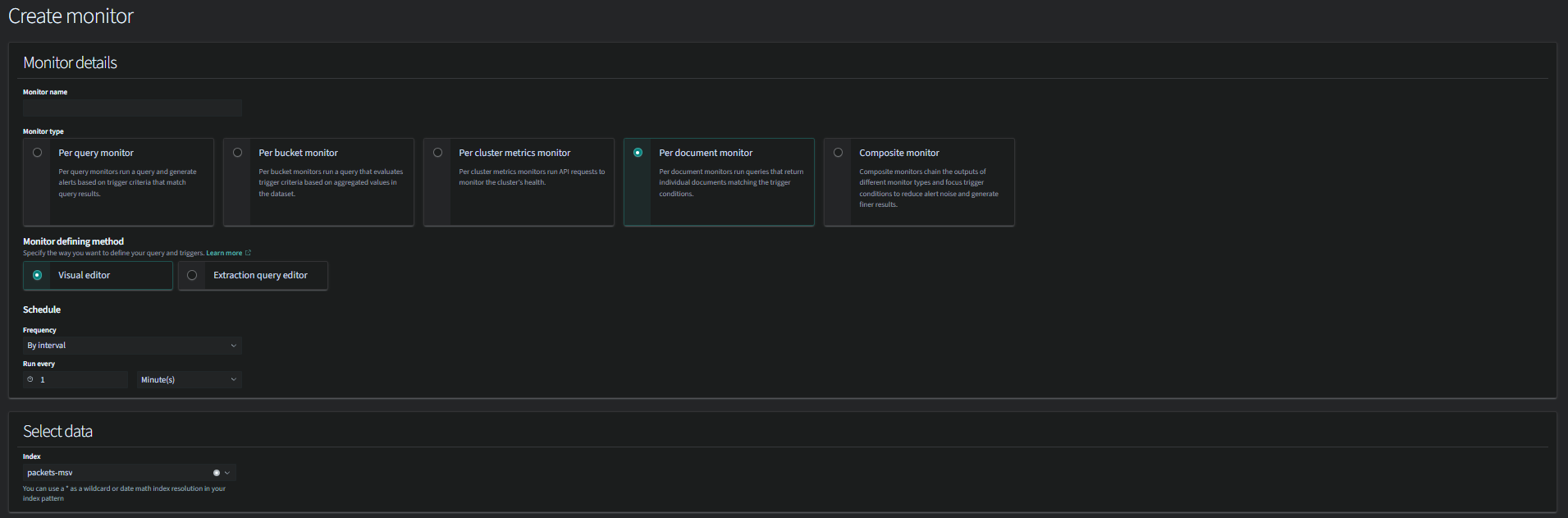
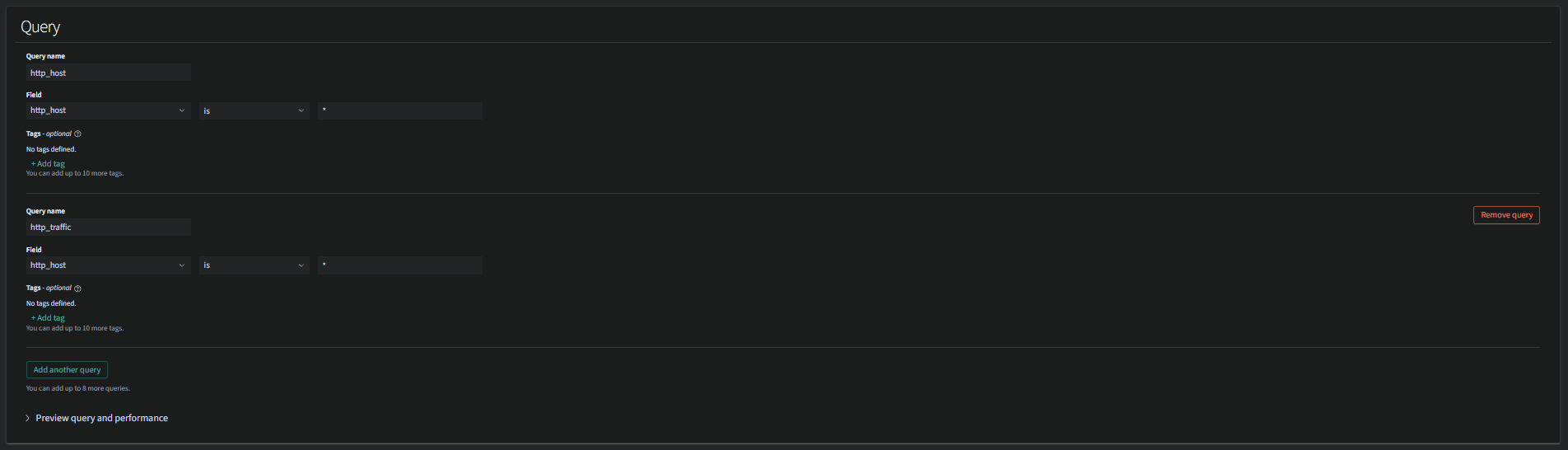


Здесь мы видим диаграмму по используемым методам http запросов. Наиболее часто используемым является метод M-SEARCH , который используется устройствами (например, Smart TV, медиаплеерами) для поиска других сервисов по сети – 68 запросов, Значительно меньшее количество стандартных запросов GET и совсем немного запросов PROPFIND (метод для работы с веб-ресурсами, как WebDAV) говорят о том, что обычный веб-трафик и управление файлами являются вторичной активностью в данном срезе данных. Следовательно можно сделать вывод, что в основном сетевая активность связана с автоматическим поиском и объявлением сетевых устройств.

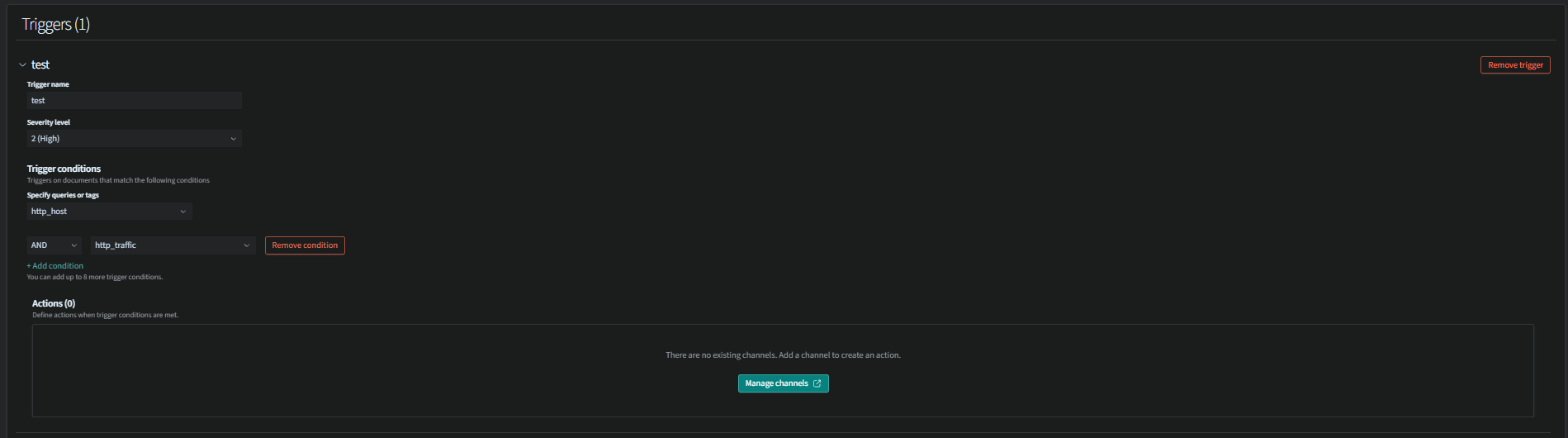
# Изучим OpenSearch Alerting.

Из-за некоторых ограничений нашей версии контейнера, я попытался настроить простой монитор который будет срабатывать если есть ЛЮБОЕ количество документов

Начальная настройка:

  
  
Создаем Query. Из-за ограничений платформы для нашего сценария создадим два Query с одинаковыми условиями:  
  


Создаем тригер:  
AND http\_traffic = условие "ЕСЛИ есть документы, соответствующие запросу http\_traffic", что эквивалентно "Сработать, если обнаружен HTTP трафик".



Таким образом наш тригер будет срабатывать когда какой-либо http трафик появится. Однако из-за ограничений api в нашей реализации контейнера большинство функционала недоступно, и по большей части из-за выключенной безопасности.

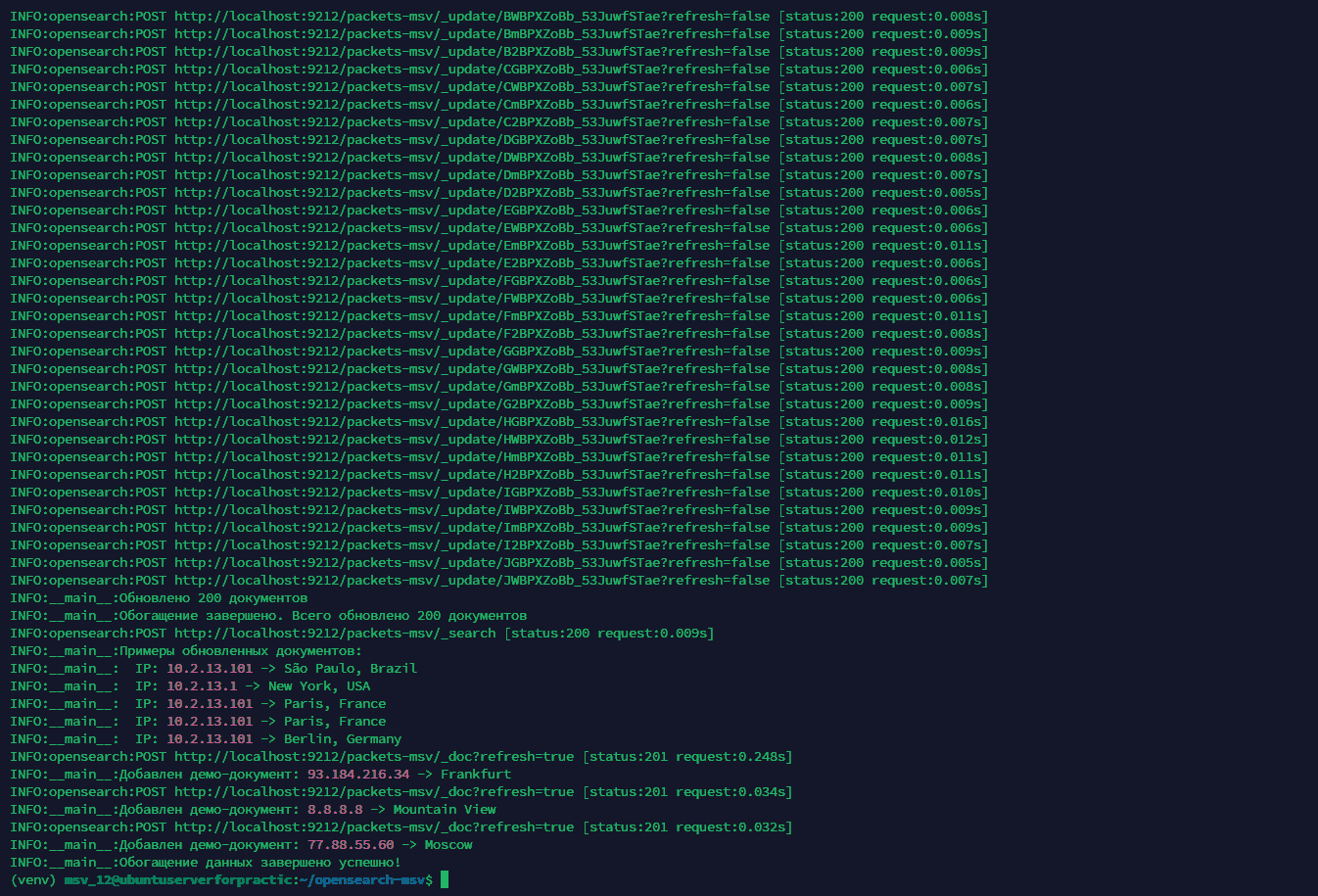
# Изучим возможности OpenSearch Dashboards по анализу гео-данных

В наших данных нет геометрической информации, поэтому для работы с гео-данными потребуется обогащение исходных данных. Для этого обычно используются сервисы по типу MaxMind GeoIP2 и IP2Location. Однако в целях изучения мы обогатим наши исходные данные вручную “выдуманными” данными, для чего будем использовать Python скрипт: <https://github.com/l1ratch/WC_BISO/blob/main/5_sem/ТХвСК/practic/pr_5/ewgd.py>

Для обогащения данных был разработан Python-скрипт, который выполняет следующие операции:

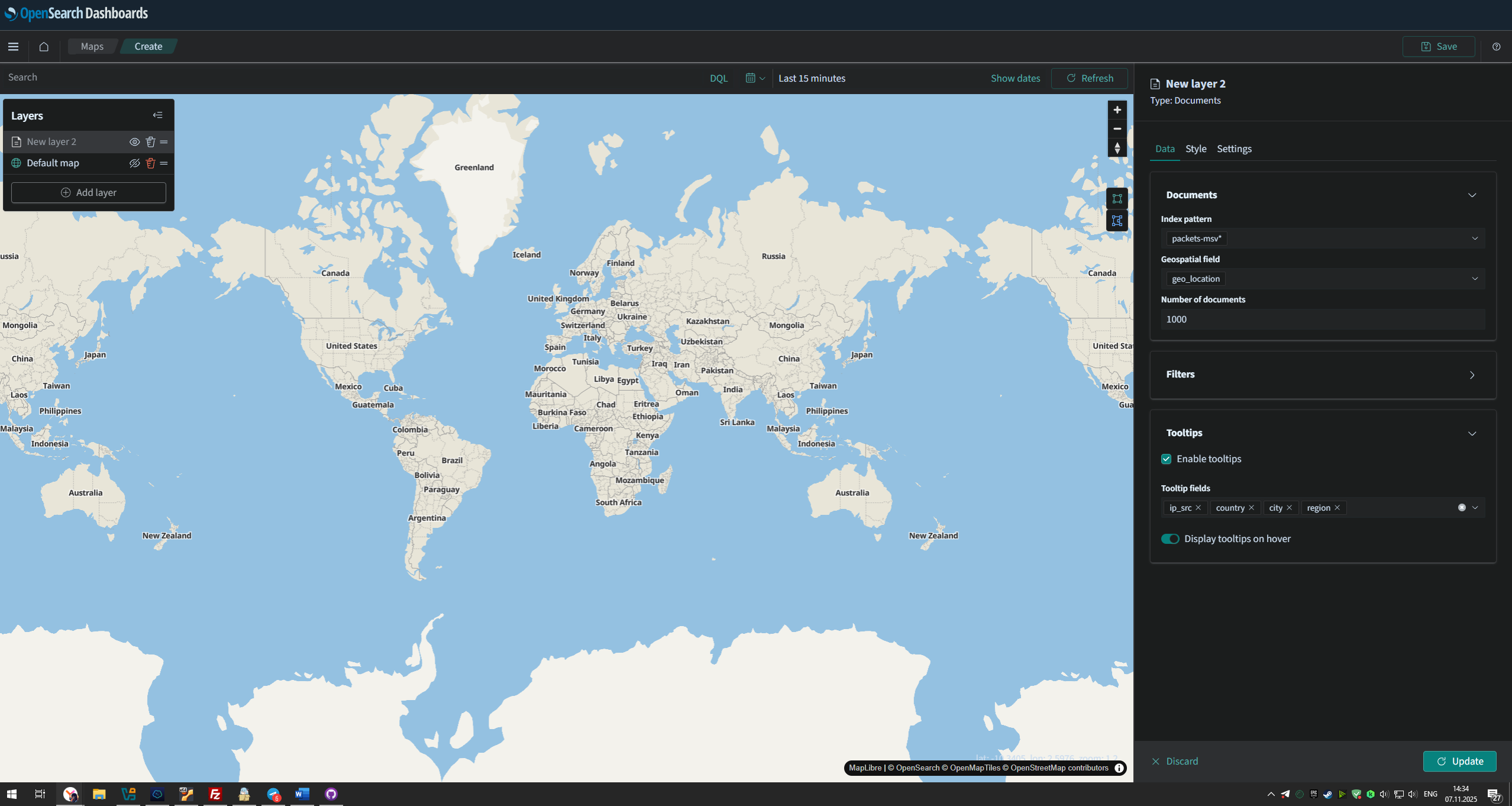
1. Добавление полей в маппинг индекса OpenSearch для хранения геоданных:  
   geo\_location (тип geo\_point) - координаты широты/долготы;  
   country - страна расположения;  
   city – город;  
   region - географический регион.
2. Генерация демонстрационных данных на основе предопределенного списка городов мира, включая:  
   Столицы и крупные города (Нью-Йорк, Лондон, Токио и др.);  
   Различные географические регионы (Северная Америка, Европа, Азия и т.д.);  
   Реалистичные координаты для каждого местоположения.
3. Обновление документов в индексе packets-msv с присвоением случайных географических меток существующим записям сетевого трафика.

Приступим к обогащению:

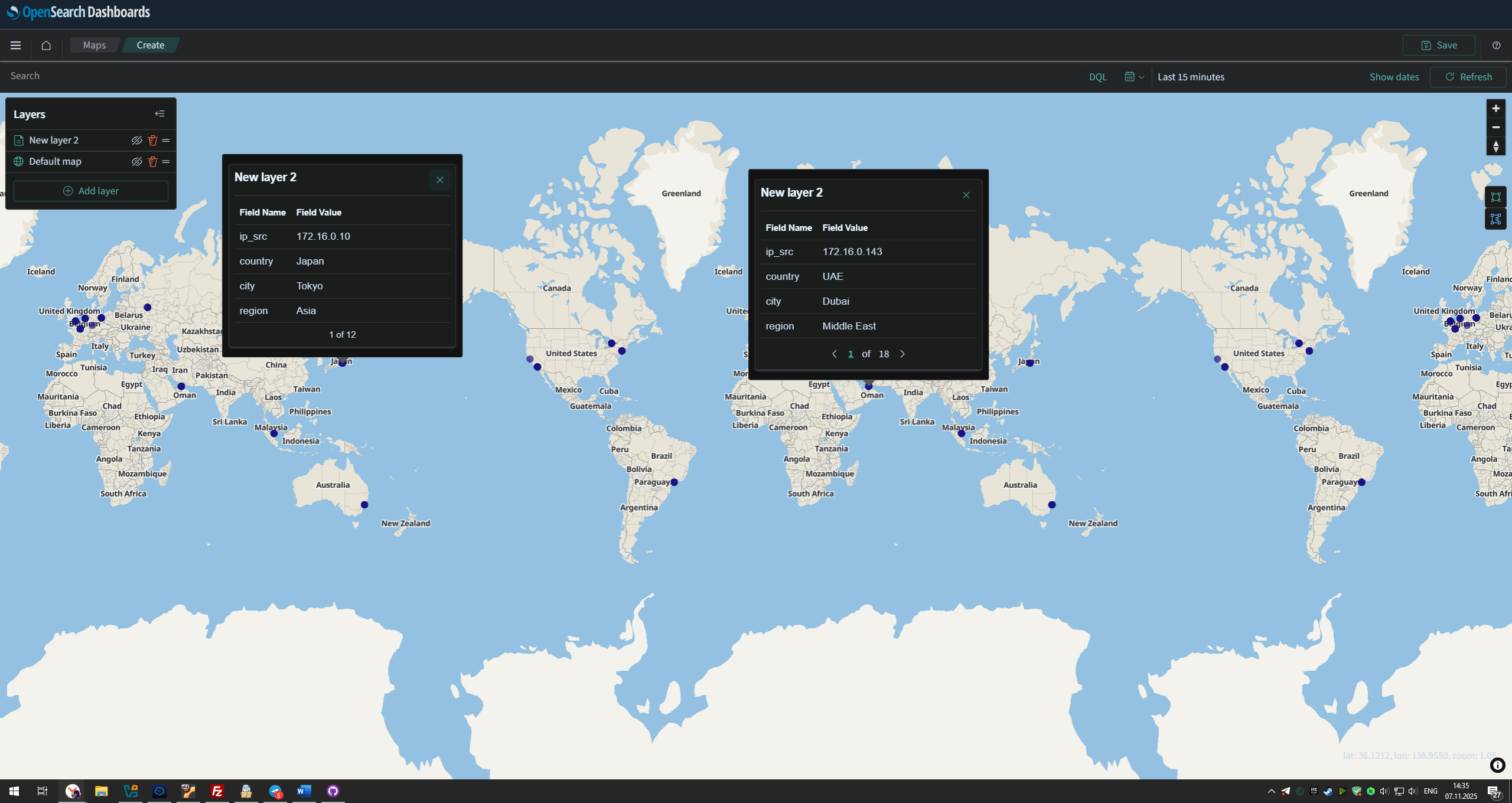


Проверим добавление данных:



Открываем в визуализациях карту и добавляем новый слой:  
  


Получаем карту с точками отображающие географические локации добавленные нами и связанные с данными из исходных файлов:



В этом пункте мы обновили документы сетевого трафика путем добавления географических меток. Для обогащения данных использовался Python-скрипт, который присвоил случайные географические локации из предопределенного списка, включающего 14 городов из различных регионов мира.

# Заключение

В ходе выполнения практической работы были получены навыки развертывания и администрирования контейнеризированных приложений с использованием Docker и Docker Compose. Успешно развернута система OpenSearch и панель OpenSearch Dashboards, произведена загрузка и анализ сетевого трафика, а также визуализация результатов.

С помощью инструмента TShark выполнена предварительная обработка PCAP-файлов, после чего данные были загружены в OpenSearch для дальнейшего анализа. Исследование логов показало характерные особенности сетевой активности, включая использование протоколов HTTP, DNS и защищённых соединений HTTPS. Были выявлены возможные признаки автоматизированных запросов и работы локального DNS-сервера.

Кроме того, реализовано обогащение данных географическими метками, что позволило расширить возможности аналитики и визуализации. Также изучены базовые принципы настройки мониторинга и оповещений (Alerting) в OpenSearch, что является важным элементом систем информационной безопасности.

Таким образом, в результате проделанной работы достигнута цель — освоены инструменты для работы с частично структурированными данными и получения аналитической информации на основе сетевого трафика с использованием современных технологий контейнеризации и анализа данных.