

**EKONOMICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA HOSPODÁRSKEJ INFORMATIKY**

**INŠTALÁCIA A NASTAVENIE ELASTIC V  
SYSTÉME LINUX**

**Zadanie z predmetu Big Data**

**2023**

**Bc. Ondrej Šima**

**Bc. Alena Stracenská**

# Obsah

<b>Úvod</b>	<b>4</b>
<b>1 Elasticsearch a ELK stack</b>	<b>5</b>
<b>2 Inštalácia kompletného ELK stacku</b>	<b>7</b>
2.1 Elasticsearch . . . . .	7
2.2 Kibana . . . . .	10
2.3 Logstash a Filebeat . . . . .	11
<b>3 Spracovanie Apache access logov pomocou Filebeat a Ingest Pipeline</b>	<b>11</b>
3.1 Elasticsearch Ingest Pipeline . . . . .	11
3.2 Konfigurácia Filebeatu . . . . .	15
3.3 Index Template . . . . .	17
<b>4 Parsovanie Fortigate logov pomocou Logstash</b>	<b>18</b>
4.1 Konfigurácia Logstashu . . . . .	18
4.2 Logstash Pipeline . . . . .	18
<b>5 Úvod do práce s dátami v Kibane</b>	<b>20</b>
5.1 Data views a indexy . . . . .	21
5.2 Vyhľadávanie pomocou Kibana Discover . . . . .	23
5.3 Vizualizácie . . . . .	26
5.3.1 Vizualizácia pomocou jednoduchšej agregácie . . . . .	27
5.3.2 Kombinovanie agregácií . . . . .	29
<b>Záver</b>	<b>31</b>
<b>Zoznam použitej literatúry</b>	<b>32</b>

## Zoznam obrázkov a tabuliek

Obrázok 1	Workflow ELK stack, zdroj: [vlastné spracovanie] . . . . .	6
Obrázok 2	Príklad správnej požiadavky na Elasticsearch pomocou nástroja <code>curl</code> , zdroj: [vlastné spracovanie] . . . . .	9
Obrázok 3	Ukážka nástroja Grok Debugger v prostredí Kibany, zdroj: [vlastné spracovanie] . . . . .	12
Obrázok 4	Príklad použitia Dev Tools Console pri vytváraní Ingest Pipeline, zdroj: [vlastné spracovanie] . . . . .	14
Obrázok 5	Príklad nastavenia Data View, zdroj: [vlastné spracovanie] . . . . .	22
Obrázok 6	UI prostredie Kibana Discover, zdroj: [vlastné spracovanie] . . . . .	24
Obrázok 7	Výsledok vyhľadávania. Namiesto predvoleného zobrazenia sme si zvolili len niekoľko, relevantných, polí, zdroj: [vlastné spracovanie] . . . . .	25
Obrázok 8	Stĺpcový graf zobrazujúci 25 najaktívnejších zdrojových IP adries, zdroj: [vlastné spracovanie] . . . . .	27
Obrázok 9	Stĺpcový graf s aplikovaným filtrom, zdroj: [vlastné spracovanie] . . . . .	28
Obrázok 10	Histogram všetkých HTTP požiadaviek, zdroj: [vlastné spracovanie] . . . . .	29
Obrázok 11	Vizualizácia zobrazujúca časový histogram HTTP požiadaviek rozdelených podľa HTTP metódy. Možno si všimnúť, že aj napriek tomu, že sme nastavili veľkosť bucketu na 1 minútu, body na grafe zobrazujú hodnotu za 5 minút, z dôvodu veľkého počtu dát na zobrazenie, zdroj: [vlastné spracovanie] . . . . .	30

# Úvod

Spracovanie Big Data alebo veľkých dát je v súčasnosti dá sa povedať až nevyhnutným krokom pre mnoho firiem a organizácií. S narastajúcim množstvom dát, ktoré sa každým dňom vytvárajú je nevyhnutné mať k dispozícii nástroje a technológie, ktoré umožnia rýchle a efektívne spracovanie a analýzu týchto dát.

Jedným z najefektívnejších a najpoužívanějších nástrojov pre správu, analýzu a vizualizáciu veľkých objemov dát je Elastic stack. Táto open-source sada nástrojov sa skladá z Elasticsearch, Logstash, Kibana a Beats, ktoré umožňujú jednoduché a rýchle spracovanie a analýzu veľkých objemov dát.

V tomto zadaní sa budeme zaoberať analýzou a spracovaním veľkých dát pomocou Elastic stacku. Opíšeme základné koncepty a princípy tejto technológie a ukážeme, ako sa dajú využiť na spracovanie, analýzu a vizualizáciu logov.

Cieľom tohto zadania je teda ukázať, ako sa dajú pomocou Elastic stacku spracovať a analyzovať veľké množstvá dát a umožniť čitateľovi porozumieť tomuto komplexnému a často využívanému nástroju.

# 1 Elasticsearch a ELK stack

Elasticsearch je populárny vyhľadávací nástroj založený na knižnici Apache Lucine. Ide o distribuované a RESTful vyhľadávacie riešenie s multitenantnou podporou, vytvorené v jazyku Java. Elasticsearch je možné využívať samostatne, ako komponent rozsiahlejšieho systému alebo vo forme ELK stacku. Samotný Elasticsearch je v mnohých komerčných softvéroch využívaný prakticky ako NoSQL databáza s rýchlym vyhľadávaním. Jednou z veľkých nevýhod Elasticsearchu je, že neumožňuje vytvárať väzby medzi jednotlivými indexami a neumožňuje vnorené dotazy. Elasticsearch podporuje okrem priameho vyhľadávania aj viacero možností agregácie dát.

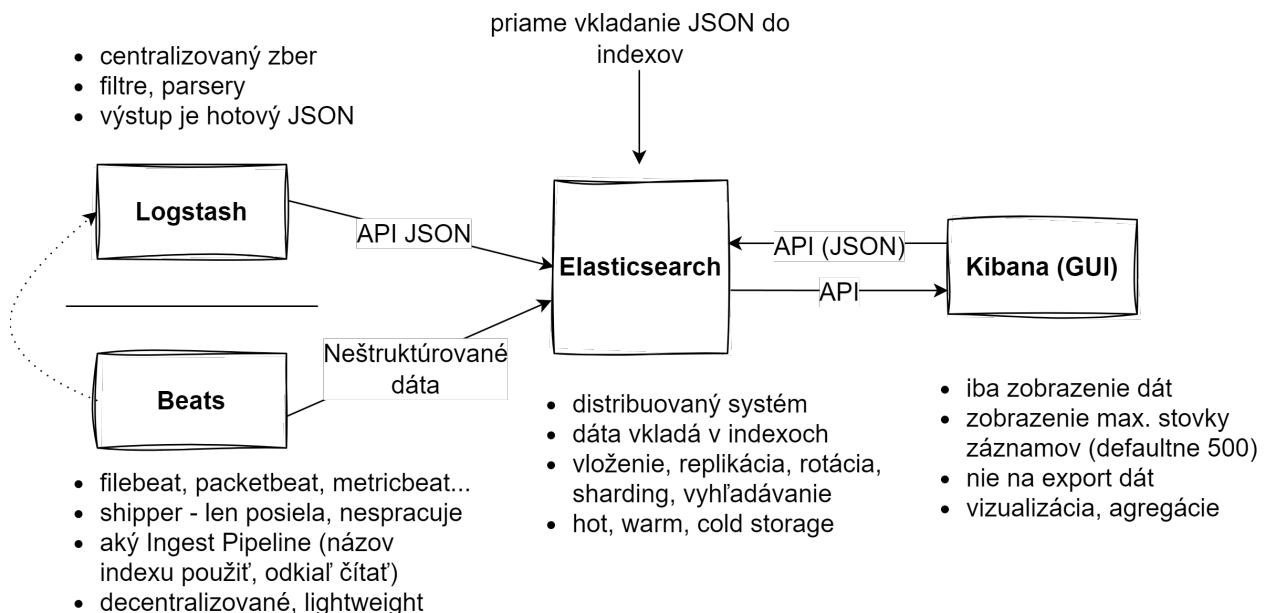
ELK stack je skratka pre softvérový stack zložený z komponentov Elasticsearch, Logstash a Kibana, ako alternatíva Logstashu sa v ELK stacku takisto používajú Beats. Tieto tri pôvodné komponenty ELK stack sa navzájom dopĺňajú funkcionalitou a spolu dotvárajú kompletné riešenie na zber, spracovanie, ukladanie, vyhľadávanie, agregovanie a vizualizáciu dát. Jednotlivé komponenty spĺňajú nasledujúce role:

- Elasticsearch - Ukladanie štrukturovaných dát vrátane manažmentu uložených dát, vyhľadávanie a agregácia dát, ingest dát z Beats modulov.
- Logstash - Centralizovaný zber štrukturovaných aj neštrukturovaných dát, parsovanie neštrukturovaných dát, spracovanie dát vrátane zmien v dátach alebo ich obohacovanie.
- Kibana - Vizualizácia dát ako text alebo ako objekt napríklad grafy, mapy atď. Vytváranie a generovanie JSON dopytov na Elasticsearch, konfigurácia a správa Elasticsearchu.
- Beats - decentralizovaný zber dát (nie spracovanie), odľahčená alternatíva ku Logstashu.

Beats je novšia platforma, ktorá dopĺňa klasický ELK stack o možnosť zberu dát priamo na cieľových systémoch (tzv. ang. "data shipper"), prakticky formou agentov. Beaty neslúžia na parsovanie dát, umožňujú len obmedzený processing, slúžia primárne na

zber a následné zasielanie dát. Z Beatu je možné neštruktúrované dáta po sieti poslať na spracovanie do Logstashu alebo na spracovanie priamo v Elasticsearchu pomocou Ingest Pipeline. V súčasnosti existuje 6 druhov Beatsov, podľa zdroja, typu čítaných dát a použitia sa delia na:

- Filebeat - Dáta číta z textových súborov.
- Packetbeat - Číta dáta posielané po sieti.
- Winlogbeat - Číta udalosti zo systému Windows.
- Auditbeat - Číta auditné logy a metriky na Linux systémoch.
- Metricbeat - Monitoruje systémové a aplikačné metriky, používa sa najmä pre aplikačný (APM) a systémový monitoring.
- Heartbeat - Monitoruje dostupnosť cieľových (zvolených) systémov alebo služieb.



**Obrázok 1:** Workflow ELK stack, zdroj: [vlastné spracovanie]

## 2 Inštalácia kompletného ELK stacku

ELK stack a všetky jeho komponenty nie sú závislé na konkrétnom operačnom systéme. Je možné ich nainštalovať a používať ako na systéme Windows, tak aj na rôznych distribúciach systému Linux alebo MacOS. K dispozícii sú samotné binárne súbory, balíky vo formáte `.deb` a `.rpm` a už predinštalovaný docker kontajner. Elastic ponúka aj možnosť plateného, cloudového riešenia.

Spomínané `.deb` a `.rpm` balíky majú všetky závislosti zahrnuté, vrátane správnej verzie OpenJDK. V prípade manuálnej inštalácie binárnych súborov alebo v prípade nemožnosti použiť pribalené OpenJDK je k dispozícii matica podporovaných verzií JDK pre danú verziu Elasticu, dostupná na [https://www.elastic.co/support/matrix#matrix\\_jvm](https://www.elastic.co/support/matrix#matrix_jvm).

V našom príklade budeme inštalovať ELK stack 8.6 na virtualizovaný systém Ubuntu 22.04 pomocou `.deb` archívov. Inštalácia a konfigurácia je jednoduchá a nemala byť zabráť viac ako niekoľko desiatok minút. Všetky komponenty budeme z dôvodov limitácií použitého hardvéru inštalovať na ten istý systém.

### 2.1 Elasticsearch

Elasticsearch balík nie je dostupný v štandardných Ubuntu repozitároch, preto si najskôr stiahneme verejný kľúč, ktorým sú balíky podpísané a pridáme si do APT Elastic repozitár. Na tento úkon je potrebné doinštalovať potrebný nástroj `apt-transport-https`, dostupný zo štandardných repozitárov. Tieto 3 kroky je nutné spraviť len raz, pre ostatné komponenty ELK stacku ich už nemusíme opakovať.

```
sudo apt-get install apt-transport-https
wget -qO - https://artifacts.elastic.co/GPG-KEY-Elasticsearch | sudo gpg
\ --dearmor -o /usr/share/keyrings/Elasticsearch-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/Elasticsearch-keyring.gpg]
\ https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee
\ /etc/apt/sources.list.d/elastic-8.x.list
```

Samotný balík následne nainštalujeme pomocou APT:

```
sudo apt-get update && sudo apt-get install Elasticsearch
```

Počas inštalácie si Elasticsearch vygeneruje heslo pre systémového užívateľa **elastic**. Tento užívateľ má práva ako superadmin, v produkčnom systéme by sa nemal používať mimo havarijného stavu. Vygenerované heslo si musíme uschovať, je nutné ho použiť na prvé prihlásenie do Kibany, kde potom môžeme vytvoriť nových užívateľov (v našej ukážke sme tohto užívateľa využívali aj na iné účely, napriek bezpečnostným hrozbám). Príklad vygenerovaného hesla:

```
-----Security autoconfiguration information-----

Authentication and authorization are enabled.
TLS for the transport and HTTP layers is enabled and configured.
The generated password for the elastic built-in superuser is : JKfYWMPmAe8QrMlsvZ0
If this node should join an existing cluster, you can reconfigure this with
'/usr/share/Elasticsearch/bin/Elasticsearch-reconfigure-node --enrollment-token
eyJ2ZXIiOiI4LjYuMiIsImFkciI6WyIxOTIuMTY4LjEwOS4xMjg6OTIwMCJdLCJmZ3IiOiJkNWMONzNmODgw
NDkxZTkWNTc2ZjdjYWRRkYzE5YjIyZGE0OTFlYmYyZjJkNTM2MGYxMmNlZjAzZTZkNmY4NmMwIiwia2V5Ijoic
zlRNkg0Y0JmdnRZOVZpbHNPOW86LXQ2djBrZ3dSSy1HUGtkM3c4NFg4ZyJ9'
after creating an enrollment token on your existing cluster.
You can complete the following actions at any time:
Reset the password of the elastic built-in superuser with
'/usr/share/Elasticsearch/bin/Elasticsearch-reset-password -u elastic'.
Generate an enrollment token for Kibana instances with
'/usr/share/Elasticsearch/bin/Elasticsearch-create-enrollment-token -s kibana'.
Generate an enrollment token for Elasticsearch nodes with
'/usr/share/Elasticsearch/bin/Elasticsearch-create-enrollment-token -s node'.
```

Službu Elasticsearch môžeme spustiť pomocou utility systemd:

```
sudo systemctl start Elasticsearch.service
```

V prípade potreby, napríklad na produkčnom systéme, možno v systemd nastaviť automatický štart služby po štarte operačného systému:

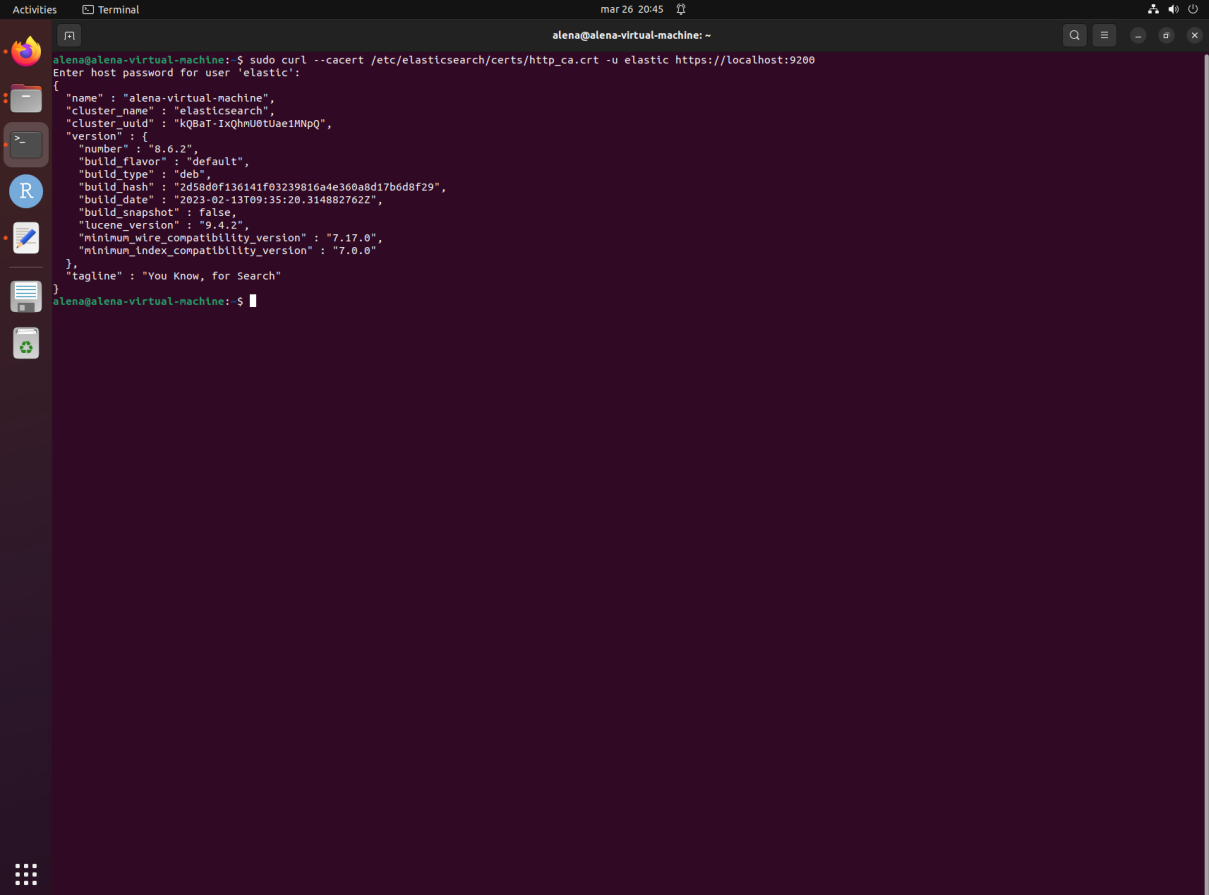
```
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable Elasticsearch.service
```

Správne nainštalovanie a dostupnosť Elasticsearch služby možno overiť HTTP(S) dopytom na localhost, port 9200 (ak nie je v konfigurácii zmenený port na ktorom Elasticsearch počúva) napríklad pomocou shellového browsera curl (v prípade že nie je na



našom systéme k dispozícii nainštalujeme ho pomocou `apt install curl`). V prípade neplatnej HTTP požiadavky nám Elastic vráti chybu, napríklad neexistujúce dopytované URI alebo skôr chybu pri autentifikácii. Správnu funkcionálnu môžeme vyskúšať platnou požiadavkou, v ktorej musíme zahrnúť aj prihlasovacie údaje a buď cestu ku certifikácii certifikačnej autority (CA certifikát), ktorým bol podpísaný TLS certifikát Elasticsearchu, alebo s príznakom ktorým povieme curlu, aby ignoroval neplatné (v našom prípade `selfsign`) TLS certifikáty. V našom prípade pošleme požiadavku na root URI a v prípade, že je inštalácia korektná nám Elasticsearch vráti JSON s informáciami o sebe:

```
curl --cacert /etc/Elasticsearch/certs/http_ca.crt -u elastic https://localhost:9200
```

A screenshot of a terminal window titled 'Terminal' with a dark background. The prompt is 'alena@alena-virtual-machine: ~'. The command entered is 'sudo curl --cacert /etc/elasticsearch/certs/http\_ca.crt -u elastic https://localhost:9200'. The output is a JSON object: {'name': 'alena-virtual-machine', 'cluster\_name': 'elasticsearch', 'cluster\_uuid': 'kQBaT-IxQhU0tUae1HmpQ', 'version': {'number': '8.6.2', 'build\_flavor': 'default', 'build\_type': 'deb', 'build\_hash': '2d58d0f136141f03239816a4e360a8d17b6d8f29', 'build\_date': '2023-02-13T09:35:20.314882762Z', 'build\_snapshot': false, 'lucene\_version': '9.4.2', 'minimum\_wire\_compatibility\_version': '7.17.0', 'minimum\_index\_compatibility\_version': '7.0.0'}, 'tagline': 'You Know, for Search'}. The prompt returns to 'alena@alena-virtual-machine: \$'.

```
alena@alena-virtual-machine: ~  
alena@alena-virtual-machine:~$ sudo curl --cacert /etc/elasticsearch/certs/http_ca.crt -u elastic https://localhost:9200  
Enter host password for user 'elastic':  
{  
  "name" : "alena-virtual-machine",  
  "cluster_name" : "elasticsearch",  
  "cluster_uuid" : "kQBaT-IxQhU0tUae1HmpQ",  
  "version" : {  
    "number" : "8.6.2",  
    "build_flavor" : "default",  
    "build_type" : "deb",  
    "build_hash" : "2d58d0f136141f03239816a4e360a8d17b6d8f29",  
    "build_date" : "2023-02-13T09:35:20.314882762Z",  
    "build_snapshot" : false,  
    "lucene_version" : "9.4.2",  
    "minimum_wire_compatibility_version" : "7.17.0",  
    "minimum_index_compatibility_version" : "7.0.0"  
  },  
  "tagline" : "You Know, for Search"  
}  
alena@alena-virtual-machine:~$
```

**Obrázok 2:** Príklad správnej požiadavky na Elasticsearch pomocou nástroja `curl`,  
zdroj: [vlastné spracovanie]

Konfiguračný súbor ku Elasticsearchu môžeme nájsť v súbore `/etc/Elasticsearch/E-`

lasticsearch.yml. Ten ponúka mnoho pokročilých nastavení, v našom prípade nám vyhovuje pôvodné nastavenie s jediným pridaným riadkom:

```
action.auto_create_index: log-*
```

## 2.2 Kibana

Inštalácia Kibany je podobne jednoduchá, pokiaľ Kibanu inštalujeme na rovnakom systéme netreba vykonať prvé kroky pomocou ktorých pridáme APT repozitár Elasticu. Samotnú Kibanu inštalujeme podobne ako Elasticsearch:

```
sudo apt-get install kibana
```

Ešte predtým, než spustíme kibana službu cez systemd, potrebujeme upraviť konfiguráciu Kibany a z Elasticsearchu vygenerovať token pre Kibanu - ten sa vygeneroval automaticky počas inštalácie Elasticsearchu. Ak ho nemáme môžeme ho vygenerovať na novo pomocou nástroja `Elasticsearch-create-enrollment-token`:

```
bin/Elasticsearch-create-enrollment-token -s kibana
```

Token si uchováme a pri prvom spustení ho vložíme priamo vo webovom GUI do príslušného poľa.

Konfiguračný súbor pre Kibanu je k dispozícii v priečinku `/etc/kibana`, súbor sa nazýva `kibana.yml`. V spomenutom súbore môžeme nastaviť sieťové adresy ku Elasticsearch nodom. V prípade ak sme inštalovali Kibanu na rovnakom systéme a používame len jeden Elasticsearch node, súbor by mal obsahovať riadok:

```
Elasticsearch.hosts: [ "http://localhost:9200" ]
```

Službu spustíme pomocou systemd, podobne ako u Elasticsearchu môžeme nastaviť automatické spúšťanie.

```
sudo systemctl start kibana.service
```

Na webové GUI Kibany pristúpime pomocou prehliadača na <https://localhost:5601>. Pred prvým prihlásením zadáme predtým vygenerovaný enrolment token a prihlásime sa pomocou užívateľa `elastic`.

## 2.3 Logstash a Filebeat

V prípade, že Logstash a Filebeat inštalujeme na rovnakom systéme, môžeme vynechať krok pridania repozitára. Logstash inštalujeme z balíka pomocou APT:

```
sudo apt-get install logstash
```

a Filebeat pomocou APT:

```
sudo apt-get install filebeat
```

Konfigurácii týchto programov sa budeme venovať v ďalších kapitolách. Konfiguračné súbory sú `/etc/logstash/logstash.yml`, respektívne `/etc/filebeat/filebeat.yml`. Logstash a Filebeat ovládame pomocou `systemd` podobne ako Elasticsearch alebo Kibanu.

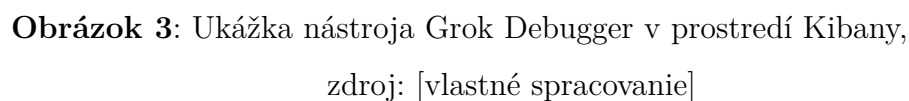
## 3 Spracovanie Apache access logov pomocou Filebeat a Ingest Pipeline

Do Elasticsearchu budeme parsovať a ukladať access logy zo súboru, podobne ako keby to robíme na produkčnom webserveri. K dispozícii máme logy z [kaggle.com](https://www.kaggle.com), súbor má veľkosť 3.2GB a obsahuje vyše 10 miliónov záznamov. Na čítanie tohto súboru sme zvolili komponent ELK stacku Filebeat a samotné dáta budeme parsovať pomocou Elasticsearch Ingest Pipeline. Pri ukladaní do indexu budeme používať vlastné názvy jednotlivých polí (nebudeme využívať Elastic Common Schema - ECS).

### 3.1 Elasticsearch Ingest Pipeline

Ingest pipeline je natívny ingestovací systém Elasticsearchu. Umožňuje okrem iného parsovanie, úpravu a obohacovanie dát. Ingest pipeline môžeme vytvoriť a nastaviť z GUI prostredia Kibany alebo pomocou API dopytu na Elasticsearch. Elastic takisto poskytuje

Pre parovanie nášho logu využijeme Grok. Grok je parovací jazyk, ktorý je na rozdiel od Regexu jednoduchý a priateľský pre používateľa. Pre overenie funkčnosti Grok parsera môžeme využiť Grok Debugger v Kibane.



```
[
  {
    "set": {
```

```

        "field": "event.ingested",
        "value": "{{_ingest.timestamp}}"
    }
},
{
    "rename": {
        "field": "message",
        "target_field": "event.original"
    }
},
{
    "grok": {
        "field": "event.original",
        "patterns": [
            "%{IPORHOST:source_ip} %{HTTPDUSER:ident} %{HTTPDUSER:auth} \\[%{HTTPDATE:apache_timestamp}\\] \\\"(?:%{WORD:http_method} %{NOTSPACE:http_request}(?: HTTP/%{NUMBER:http_version})?| %{DATA:rawrequest})\\\" %{NUMBER:http_response_code} (?:%{NUMBER:http_response_size}| -) \\\"%{DATA:http_referer}\\\" \\\"%{DATA:useragent}\\\""
        ],
        "ignore_missing": true
    }
},
{
    "date": {
        "field": "apache_timestamp",
        "formats": [
            "dd/MMM/yyyy:H:m:s Z"
        ],
        "target_field": "@timestamp",
        "ignore_failure": true
    }
},
{
    "remove": {
        "field": "apache.timestamp",
        "ignore_failure": true
    }
}
]

```

Možno si všimnúť že okrem samotného logu pracujeme aj s inými poliami, je to preto, lebo Filebeat po prečítaní riadku zo súbora tento riadok pošle v JSON formáte v poli message. Do tohto JSON potom pridá vlastné dáta: informácie o samotnom Filebeate a systéme z ktorého načítal dáta.

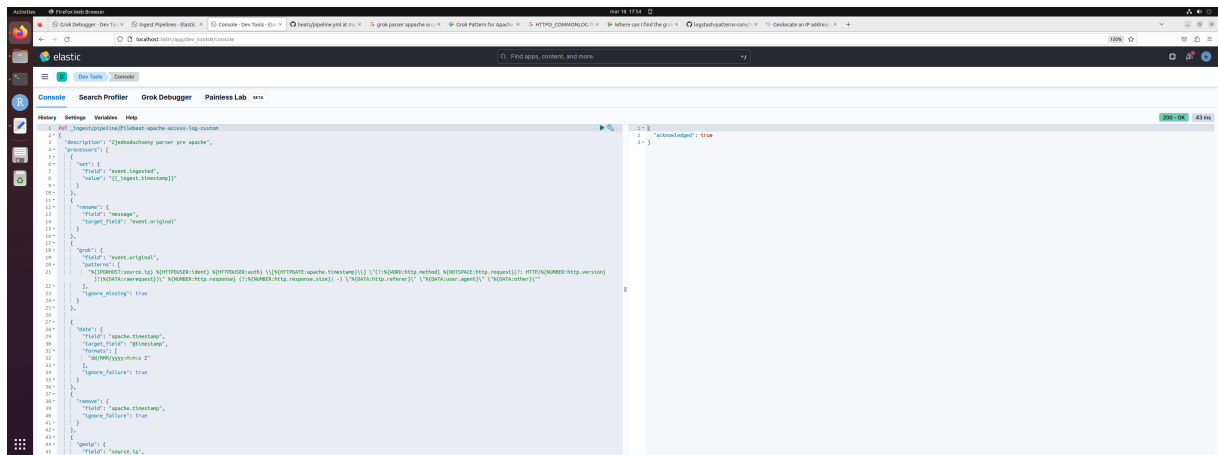
V našom nastavení Pipeline vykonáme nasledujúce akcie. Vytvoríme nové pole nazvané `event.ingested`, do ktorého nastavíme časovú pečat, kedy Ingest prijal konkrétny záznam. Premenujeme pole `message` na `event.original`. Pomocou Grok parsera vyparsojeme pôvodný log, názvy a typy polí sú uvedené v samotnom Grok scripture. Z vyparsovaného poľa `apache_timestamp` prečítame hodnotu časovej pečate v správnom formáte a uložíme ju do poľa `@timestamp`. `@timestamp` je špeciálne pole ktoré slúži na indexovanie dát. Následne vymažeme pole `apache_timestamp`.

Pri parsovaní tohto logu sme nepoužili Elastic Common Schema (ECS) ale vlastné názvy polí. ECS slúži na normalizáciu dát medzi rôznymi často používanými zdrojmi logov, čím umožňuje jednoduché vyhľadávanie medzi indexami.

Hotový JSON môžeme poslať Elasticsearchu pomocou API, cez Dev Tools v Kibane. Vytvoríme POST request na URI `_ingest/pipeline/nazov` s našim JSON:

POST `_ingest/pipeline/apache-custom`

```
{
  ..
}
```



**Obrázok 4:** Príklad použitia Dev Tools Console pri vytváraní Ingest Pipeline, zdroj: [vlastné spracovanie]

## 3.2 Konfigurácia Filebeatu

Konfiguračný súbor pre filebeat môžeme nájsť v `/etc/filebeat/filebeat.yml`, ktorý je od inštalácie predvyplnený základným nastavením a komentovanými príkladmi konfigurácie. Súbor, ako jeho prípona naznačuje, je vo formáte YAML a zaujímajú nás v ňom najmä 2 kľúče, a to `filebeat.inputs` a `output.Elasticsearch`. Ako ich názov prezrádza, nastavuje sa v nich vstup a výstup. Pri vstupe potrebujeme uviesť typ súboru, ktorý bude Filebeat čítať, cestu k nemu a v poslednom rade aktivovať tento vstup. Konfiguračný súbor otvoríme pomocou textového editora s root oprávneniami:

```
sudo nano /etc/filebeat/filebeat.yml
```

Nastavíme v ňom následné nastavenie pre náš vstupný súbor. V našom prípade budeme čítať akýkoľvek `*.log` súbor z cieľového priečinka (vrámci testovania a vykladovania môžeme načítavať menšiu vzorku súboru).

```
filebeat.inputs:
- type: log
  id: my-filestream-id
  enabled: true
  paths:
    /home/alena/access_log/*.log
```

Pre výstup nastavíme parametre pre Elasticsearch - adresu clusteru, prihlasovacie údaje (možno použiť aj autorizáciu pomocou privátneho kľúča), a keďže máme aktivované SSL šifrovanie buď príznak, ktorým vypneme overovanie certifikátov (keďže vrámci lokálneho nasadenia dôverujeme, že sa prihlasujeme na náš Elastic), alebo cestu ku CA certifikátu ktorý Elasticsearch vygeneroval pri inštalácii a podpísal s ním SSL/TLS certifikáty. Ďalej nastavíme parametre ktoré sa týkajú samotných dát a ako ich bude Elasticsearch spracovávať a ukladať:

- `index` - Nastavíme v ňom názov indexu do ktorého bude Elasticsearch vkladať dáta. Keďže spracovávame logy a v produkčnom systéme by sme mali kontinuálny stream dát, na konci názvu indexu zapíšeme aktuálny dátum. Týmto spôsobom docielime

to, že každý deň sa budú dáta z toho dňa zapisovať do nového indexu, čím vyriešime napríklad rotáciu indexov (v našom príklade rotáciu nepoužívame, máme statický vstup).

- pipeline - Zadáme názov Ingest Pipeline, ktorá bude spracovávať tento vstup.

```
output.Elasticsearch:
  hosts: ["https://localhost:9200"]
  ssl.verification_mode: none
  username: "elastic"
  password: "JKFYWMPMgAe8QrMlsvZ0"
  index: "log-filebeat-access_%{+yyyyMMdd}"
  pipeline: "apache-custom"
  bulk_max_size: 5000
```

V poslednom rade do samotného rootu YAML súboru dopíšeme názov a vzor šablóny indexu, tie musia mať rovnaký vzor ako vyššie uvedený názov indexu, keďže sa tento index bude vytvárať podľa šablóny:

```
setup.template.name: "log-filebeat-"
setup.template.pattern: "log-filebeat-*
```

V konfiguračnom súbore je takisto možné vybrať a nastaviť Filebeat modul, Elastic udržiava objemný repozitár prednastavených modulov pre spracovanie rôznych vstupov, predovšetkým logov z rôznych zariadení a systémov pomocou Filebeatu. Túto možnosť sme v tejto ukážke nevyužili.

Ďalšie nastavenia v tomto súbore sme nechali v pôvodnom stave (v prípade, že niektorý príznak nie je uvedený v konfiguračnom súbore, je braná jeho default hodnota). Bližšie informácie ku jednotlivým nastaveniam sú dostupné v dokumentácii.

Filebeat si zapisuje databázu súborov, ktoré prečítal a na ktorom riadku skončil, takže vie pokračovať, keď v súboroch pribudnú nové riadky a nevracia sa ku pôvodným súborom bez zmeny. Databáza sa nachádza v priečinku `/var/lib/filebeat/registry/filebeat` a pokiaľ by sme chceli znova načítať už raz načítaný súbor, môžeme tak spraviť zmazaním záznamu v tomto registri.

Pokiaľ máme všetko korektne nastavené môžeme spustiť Filebeat pomocou `systemd`:



```
sudo systemctl start filebeat.service
```

V našom príklade sme nevytvárali vzor (template) pre indexy, Elasticsearch nám vytvoril template sám. Pre účely tohto príkladu takto automaticky vytvorený template postačuje, no nevýhodou je množstvo zbytočných a nevyužívaných ECS polí ktoré nám v indexovom vzore vznikli. Index Template si popíšeme teoreticky a v skratke a je aplikovateľný aj na druhý príklad.

### 3.3 Index Template

Index template môžeme vytvoriť pomocou GUI rozhrania v Kibane alebo ho pomocou API volania vložiť priamo do Elasticsearchu pomocou JSONu. Index Template hovorí Elasticsearchu ako má ukladať dáta v indexoch, ako má mapovať polia (čiže aké typy majú jednotlivé polia, prípadne špeciálne nastavenia aplikovateľné na niektoré typy), aké polia má zobrazovať a podobne.

Index Template sa aplikuje na indexe pomocou Index Patternu, je preto vhodné navrhnúť si vhodnú schému názvov indexov, zahrnúť v nej napríklad typ zariadenia, ktoré loguje atď. Ak nastavíme v Template Index Pattern `log-*` bude táto šablóna aplikovaná na všetky indexy ktoré začínajú na `log-`. V prípade viacerých šablón, ktoré sú aplikovateľné na daný Index Pattern, je vybraná šablóna s najnižšou hodnotou priority. Automaticky vytvorené šablóny majú prioritu 160, je vhodné vytvárať šablóny s nižšou hodnotou.

Pri vytváraní vlastnej šablóny možno použiť predpripravené komponenty a následne z nich využiť prednastavené mapovania.

V nastavení indexu sa nastavujú veci spoločné s ukladáním dát ako napríklad počet replík, veľkosť shardov. Pre pokročilé nastavenie odporúčame pozrieť dokumentáciu.

Mapovanie hovorí najmä, aké typy majú jednotlivé polia. Názvy polí možno štrukturovať, napríklad do poľa `src` typu `Object` môžeme vložiť polia `src.ip` typu `IP` a `src.port` typu `Numeric`. Niektoré typy polí, napríklad `Text` nie sú agregovateľné, aby boli dáta agregovateľné aj pomocou takýchto polí je vhodné vytvoriť s rovnakým názvom a príponou `.keyword` aj pole typu `Keyword`. Indexovateľné sú iba niektoré typy, napríklad typ `Date`.

Elasticsearch podporuje dynamické mapovanie, čiže v prípade že mu prídu dáta s poľom, ktoré nepozná, automaticky vytvorí vhodné mapovanie. V prípade že je potrebné zmeniť mapovanie už existujúcich indexov, je nutné tieto indexy reindexovať. Viac v dokumentácii.

## 4 Parsovanie Fortigate logov pomocou Logstash

Pri tejto ukážke máme k dispozícii Python skript ktorý náhodne generuje logy sieťového firewallu Fortigate 501E a posiela ich pomocou protokolu Syslog po sieti. Na spracovanie týchto logov sme preto zvolili Logstash, ako simuláciu centralizovaného zberu logov z viacerých zariadení. Logy, ktoré máme k dispozícii sú vo formáte `klúč = "hodnota"`.

### 4.1 Konfigurácia Logstashu

V samotnom konfiguračnom súbore `/etc/logstash/logstash.yml` nebudeme v našom príklade nič nastavovať, keďže budeme využívať len 1 Logstash Pipeline v základom nastavení a bez ďalších funkcionalít. Pre pokročilé nastavenie je nápomocná dokumentácia.

### 4.2 Logstash Pipeline

Podobne ako pri Elasticsearch Ingest Pipeline, sa v Logstash spracovávajú dáta v rúrach (Pipeline). V konfigurácii môžeme rúram nastaviť výpočtové zdroje, počet workerov, veľkosti dávok a podobne. Pri spracovaní dát z viacerých zdrojov sú vhodné 2 prístupy: vytvorenie jednej rúry pre každý typ zdroja alebo vytvorenie jednej rúry pre všetky zdroje. Preposielanie dát medzi rúrami je možné, no neodporúča sa z dôvodu zhoršeného výkonu, najmä pri live spracovaní veľkého množstva dát. Každá rúra má pridelenú konfiguráciu, konfigurácie sa štandardne nachádzajú v priečinku `conf.d`.

Konfigurácia rúry je vo formáte pripomínajúcom JSON, klúče a hodnoty sú oddelené znakmi `=>`. Súbor je rozdelený na 3 časti: `input`, `filter` a `output`.

`input` opisuje odkiaľ získava rúra dáta. Môže napríklad čítať súbor, čítať štandardný vstup alebo výstup, počúvať na sieťovom porte, pripojiť sa na Kafka server alebo dostávať

prečítané logy z Beatsov. V našom prípade počúvame na porte 5164 Syslog protokol:

```
input {
  udp {
    port => 5164
    type => syslog
  }
}
```

filter slúži na samotné spracovanie dát. Možnosti spracovania sú podobné ako v Elasticsearch Ingest Pipeline. Najväčší rozdiel je v tom, že dáta sú spracované ešte mimo Elasticsearch, čo vo veľkom systéme, ktorý spracováva veľa dát môže mať výhody z pohľadu výkonu (Logstash na inom serveri ako Elasticsearch node). V našom prípade parsujeme logy ktoré boli doručené pomocou Syslog protokolu. Syslog pridal k samotnému logu RFC5424 hlavičku, ktorú Logstash vyparsoval automaticky a naše dáta sú v poli message. Prvá časť nášho logu ale nie je vo formáte klúč-hodnota, preto pomocou Grok parseru vyparsujeme z tejto časti časovú pečiatku, host.name - IP adresa zariadenia z ktorého log pochádza a zvyšok logu - ako event.original. Následne pomocou kv rozdelíme páry klúč-hodnota uložené v event.original.

```
filter {
  grok {
    match => [ "message", "(?:%{SYSLOGTIMESTAMP:syslog_timestamp}|%
    {TIMESTAMP_ISO8601:timestamp8601}) %{SYSLOGHOST:host.name}
    %{GREEDYDATA:event.original}" ]
  }
  kv {
    source => "event.original"
    field_split => " "
    value_split => "="
    remove_char_key => "<>\\[\\],\\;_"
    remove_char_value => "<>\\[\\],\\;"
    trim_key => "<>\\[\\],\\\""
    trim_value => "<>\\[\\]\\-\\\""
    include_brackets => false
  }
}
}
```

V prípade, ak by sme chceli robiť normalizáciu polí, môžeme tak urobiť pomocou

funkcie `mutate` alebo obohatenie dát, napríklad o geolokáciu IP adres, môžeme tak spraviť v časti `filter`.

Nakoniec, `output` nám určuje kam rúra pošle spracované dáta. Okrem možnosti poslať dáta na Elasticsearch môžeme dáta napríklad zapísať do súboru, poslať do inej rúry alebo vypísať v štandardnom výstupe atď. V našom prípade sme dáta posielali do nášho Elasticsearchu, nastavili sme pripojenie naň a takisto aj názov indexu, do ktorého bude Elasticsearch dáta zapisovať.

```
output {
  Elasticsearch {
    hosts => ["localhost:9200"]
    ssl => true
    cacert => "/etc/logstash/ca.crt"
    user => "elastic"
    password => "JKFYWMPMgAe8QrMlsvZ0"
    manage_template => false
    index => "log-logstash-forti-%{+yyyyMMdd}"
  }
}
```

Logstash zapneme podobne ako ostatné služby pomocou `systemd`. Chvíľu po jeho spustení začne počúvať na lokálnom porte 5164 a môžeme začať generovať logy pomocou skriptu `fortigate.py`. Keďže sme si nepripravili Index Template, podobne ako v predchádzajúcej ukážke sa nám vzor vytvorí sám a takisto sa automaticky nastaví mapovanie polí. Keď sa nám vygeneruje dostatok logov môžeme ukončiť generovací skript, približne za pol hodinu spusteného skriptu sa vygenerovalo tamer 12 miliónov záznamov, veľkosť Elasticsearch indexu bola takmer 10GB.

## 5 Úvod do práce s dátami v Kibane

Kibana je silný nástroj na prácu s veľkým počtom dát. Umožňuje vyhľadávanie v dátach, ich agregáciu a vizualizáciu, avšak nie je určená na export dát z Elasticsearchu. Používa sa skôr na prezentáciu agregácií z dát alebo na zobrazenie konkrétnych vyhľadávaných záznamov v počte, ktorý dokáže spracovať jej užívateľ, čo je obvykle maximálne niekoľko stoviek záznamov.

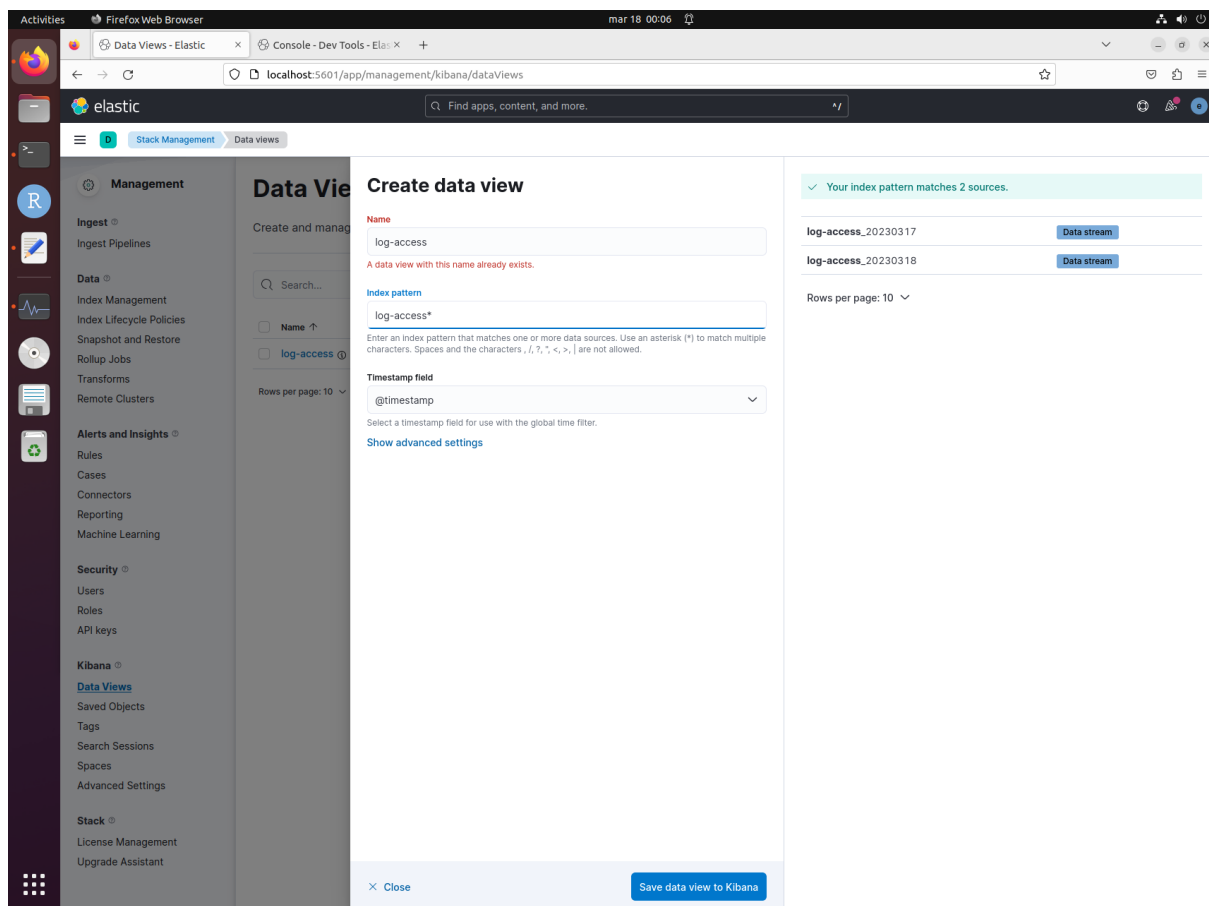
Kibana má okrem práce s dátami uložených v Elasticsearch aj niekoľko ďalších funkcií. Je to správa samotného Elasticsearchu, v tomto zadaní sme ju viackrát použili na nastavovanie samotného Elasticsearchu. Medzi ďalšie funkcionality, ktorým sa v tomto zadaní nevenujeme a niektoré z nich sú platené patria:

- Strojové učenie - Kibana spracováva dáta pomocou AI a vie z nich extrahovať rôzne vlastnosti alebo detekovať anomálie.
- Aplikačné vyhľadávanie - Rôzny komerčný softvér (Netgrif, Dynatrace, Fortisoar), ale aj webové aplikácie (Stack Overflow, Netflix, LinkedIn) používajú Elasticsearch ako NoSQL databázu s rýchlym vyhľadávaním. V prípade použitia Elasticsearchu môže slúžiť Kibana nie len ako pomocný nástroj pri vývoji, ale aj ako servisný nástroj pri samotnej prevádzke napr. webovej aplikácie.
- Aplikačný monitoring - Či už samotného ELK stacku alebo iných aplikácií, Kibana umožňuje vyhodnocovanie stavu a posielanie notifikácií a upozornení na základe logov, metrík alebo nainštalovaných APM agentov.
- Kybernetická bezpečnosť - Kibana poskytuje nadstavbu nad Elasticsearchom priamo optimalizovanú ako SIEM (Security information and event management), na koreláciu bezpečnostných udalostí a vytváraní bezpečnostných incidentov na základe pravidiel, vyhľadávanie indikátorov kompromitácie, pokročilého threat huntingu atď. Pre použitie ako SIEM existujú aj priamo zamerané distribúcie ELK stacku ako napríklad SOF-ELK alebo Wazuh SIEM.

## 5.1 Data views a indexy

Ešte predtým než nám Kibana umožní v dátach vyhľadávať musíme vytvoriť Data View. Jedná sa v podstate len o to, v akých indexoch nám Kibana umožní vyhľadávať, no pre nastavenie Data View je dôležité mať správnu schému názvov indexov. V našom prípade si vytvoríme 3 Data View, a to `log`, `log-access` a `log-forti`. Pre `log-access` a `log-forti` priradíme indexy `log-filebeat-access*` a `log-logstash-forti*` respektívne pre `log` priradíme index `log-*`. Týmto docielime, že máme samostatné Data View

pre jednotlivé zdroje logov a potom jeden spoločný Data View pre všetky logy. Použitie ECS na názvy polí umožňuje vyhľadávanie toho istého parametra, napríklad zdrojovej IP adresy vo všetkých logoch, z rôznych zariadení od rôzneho výrobcu naraz.



**Obrázok 5:** Príklad nastavenia Data View, zdroj: [vlastné spracovanie]

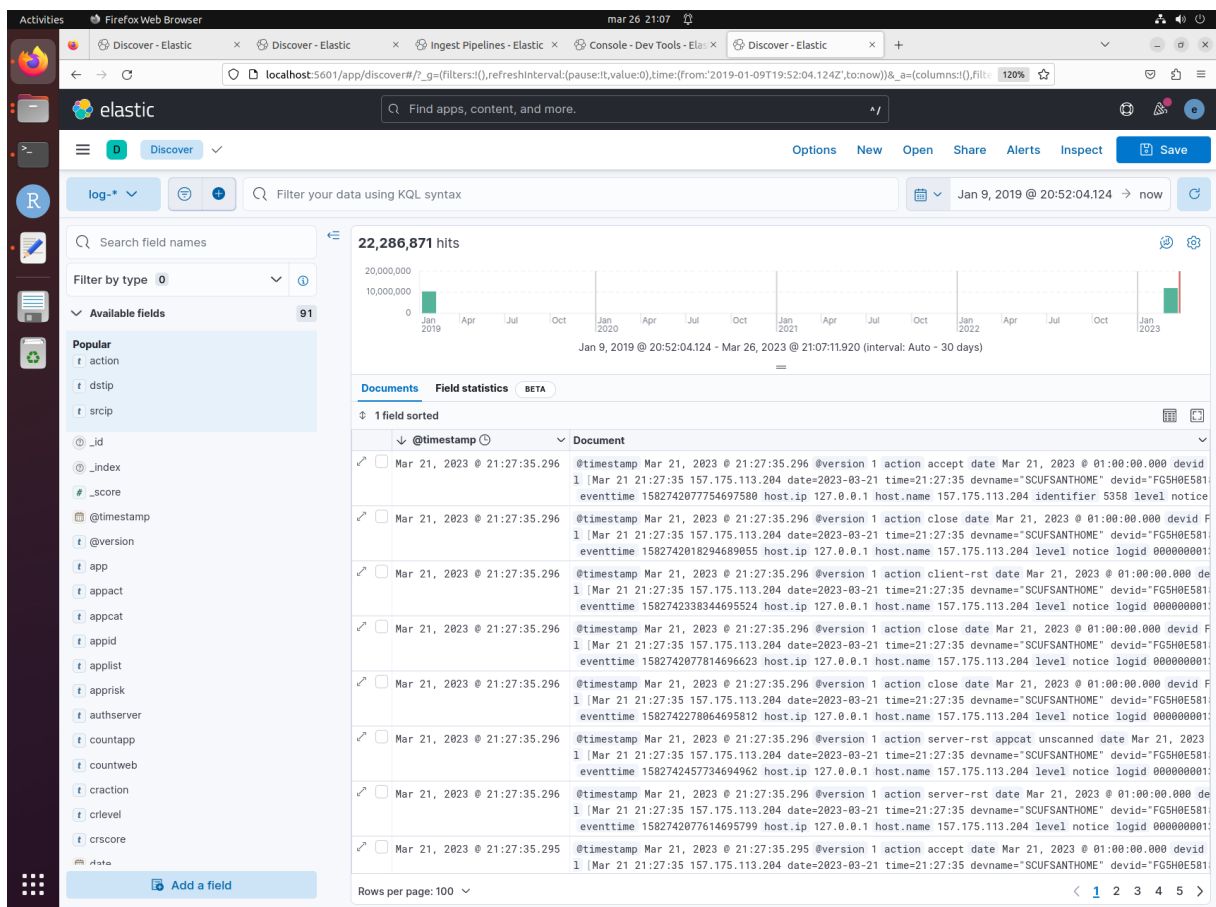
Samotné indexy môžeme takisto spravovať v Kibane, napríklad nastaviť im politiky životného cyklu, okrem rotácie starých indexov. V prípade, že máme k dispozícii viacej druhov úložiska (rýchle SSD + pomalé HDD), môžeme nastavovať aj úložné fázy pre naše dáta (napríklad, že najnovšie dáta v ktorých najviac vyhľadávame sa ukladajú na rýchle disky - Hot phase, potom prejdú na pomalé disky bez kompresie - Warm phase, a keď v nich vyhľadávame už len príležitostne prejdú do Cold phase, keď vyhľadávanie trvá dlhšie a dochádza ku kompresii). Indexy vieme takisto z Index Managementu mazať alebo si prezerať details. Tak ako celá funkcionality v Kibane, je možné všetky akcie robiť aj priamo, a to API volaním na Elasticsearch.

## 5.2 Vyhľadávanie pomocou Kibana Discover

Discover je asi najviac využívaná časť Kibany a slúži práve na vyhľadávanie v záznamoch uložených v Elasticsearchu. Samozrejme, vyhľadávať sa dá priamo v Elasticsearchovom Node pomocou API volania. Výhoda Kibany je, že nám toto API volanie sama vyskladá a vyhladané dáta zobrazí v čitateľnejšej forme.

Discover nám umožňuje vybranie Data View, v ktorom chceme vyhľadávať, zadanie query pomocou textového poľa vo forme zjednodušeného KQL (Kibana Query Language) alebo Lucene syntaxe. Prípadne pomocou tlačítka + naľavo od tohto textového poľa pomocou zjednodušeného grafického nástroja na vytváranie filtrov. Napravo od Query boxu nám ponúka výber časových údajov, od - do. Na výber sú 3 možnosti, a to absolútny čas, relatívny čas (napríklad posledných 16 minút, posledný deň atď.), potom možnosť nastaviť čas na **teraz** (dynamické nastavenie, stále sa aktualizuje podľa systémového času klienta).

Nad samotnými výsledkami vyhľadávania máme prehľadne časovú os s histogramom vyhladaných výsledkov a celkovým počtom výsledkov, tzv. **hitov**. Nakoniec naľavo od výsledkov je zoznam dostupných polí s možnosťou vyhľadávania polí a výberu polí na zobrazenie vo forme tabuľky.



Obrázok 6: UI prostredie Kibana Discover, zdroj: [vlastné spracovanie]

Vyhľadávanie pomocou KQL je jednoduché a prehľadné. V prípade, že chceme nájsť konkrétnu hodnotu napríklad v poli `http_method`, kde je metóda POST zadáme query:

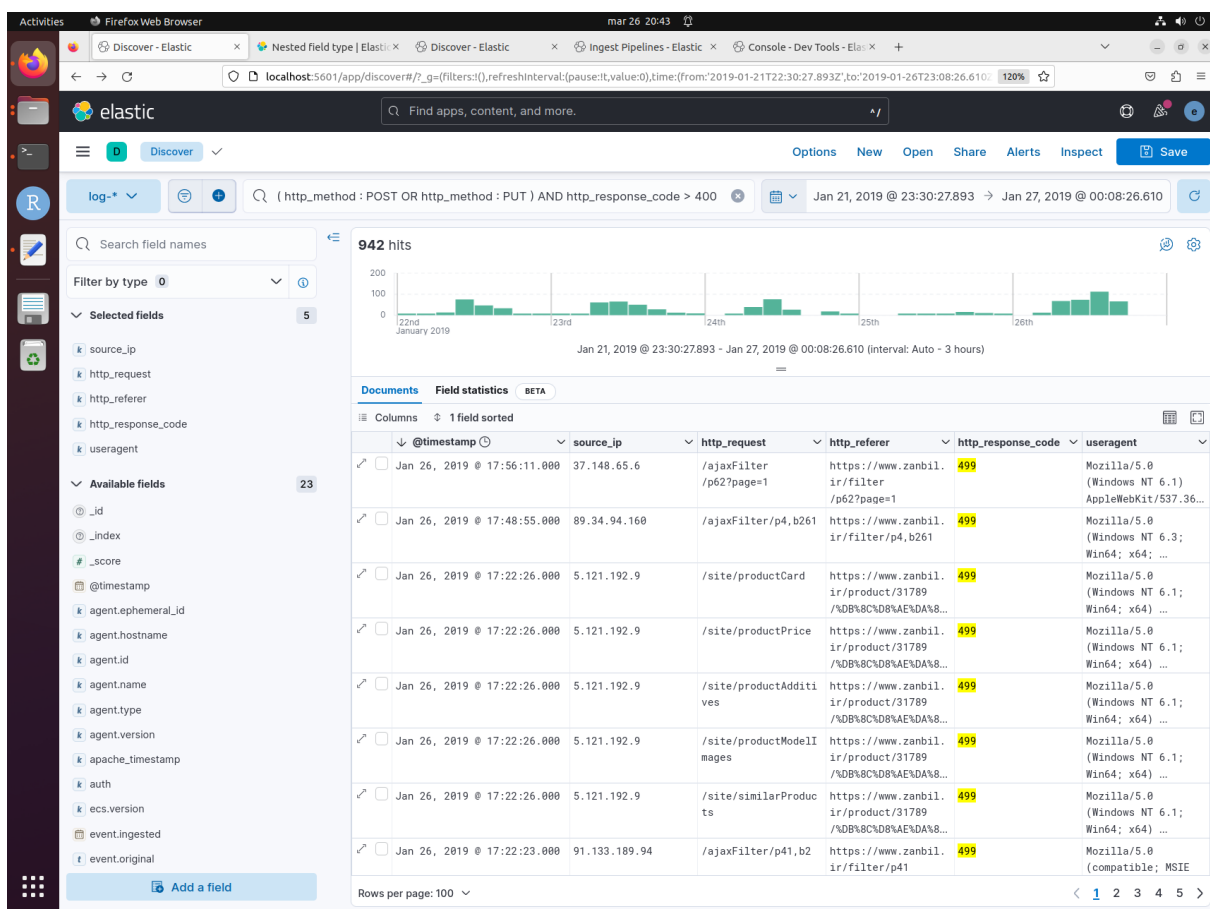
```
http_method : POST
}
```

KQL takisto umožňuje spájanie query pomocou logických operácií, napríklad:

```
( http_method : POST OR http_method : PUT ) AND http_response_code > 400
}
```

Toto vyhľadávanie nám našlo všetky záznamy, keď bola použitá HTTP metóda POST alebo PUT a zároveň server vrátil chybový kód 400 a vyšší.





**Obrázok 7:** Výsledok vyhľadávania. Namiesto predvoleného zobrazenia sme si zvolili len niekoľko, relevantných, polí, zdroj: [vlastné spracovanie]

Rovnaký dotaz môžeme zadať aj priamo do Elasticsearchu, s jeho vytvorením nám pomôže Kibana. Keďže Kibana s Elasticsearchom komunikuje pomocou API, čiže rovnakým spôsobom ako s Elasticsearchom budeme komunikovať my, môžeme si vzor dopytu vygenerovať pomocou Kibany. Tento prístup je vhodný aj keby sme neplánovali na prácu s Elasticsearchom využívať primárne Kibanu. Pomocou Kibany si môžeme vyskúšať a vyladiť vyhľadávania a agregácie dát a potom využívať už samotné API s predpripravenými volaniami. V časti Discover alebo v jednotlivých vizualizáciách je na hornej lište vpravo dostupný Inspector. V ňom je dostupný JSON obsah API volania a aj odpovede Elasticsearchu. Ten vieme pomocou tlačítka **Open in Console** otvoriť v Kibana Dev Tools a ďalej ladiť a následne z neho priamo vygenerovať curl príkaz so správnymi escaped znakmi v JSON štruktúre. Tento spôsob je vhodný aj pri exportovaní väčšieho množstva dát z

Elasticsearchu, obmedzením je maximálny počet záznamov ktorý vie Elasticsearch vrátiť v rámci jedného dopytu. V prípade, že celkový počet výsledkov presiahne 10 000 záznamov, Elasticsearch vráti aj token vďaka ktorému je možné reťaziť dopyty. Pre takéto použitie je vhodné použiť napríklad Python API. Elasticsearch takisto ponúka možnosť použiť SQL-like dopyt namiesto JSON štruktúry, viac informácií je dostupných v dokumentácii.

### 5.3 Vizualizácie

Hlavným nástrojom, ktorým Kibana umožňuje vytvárať agregácie nad dátami uloženými v Elasticsearch sú vizualizácie. Umožňuje tak prehľadné zobrazenie, pričom aj pri grafickom zobrazení je možné agregované dáta exportovať ako text. Špeciálny prípad vizualizácií sú mapy, fungujú na podobnom princípe ako klasické vizualizácie, keďže naše dáta neobsahujú a neboli obohatené o geolokácie v správnom formáte, problematike máp sa venovať nebudeme.

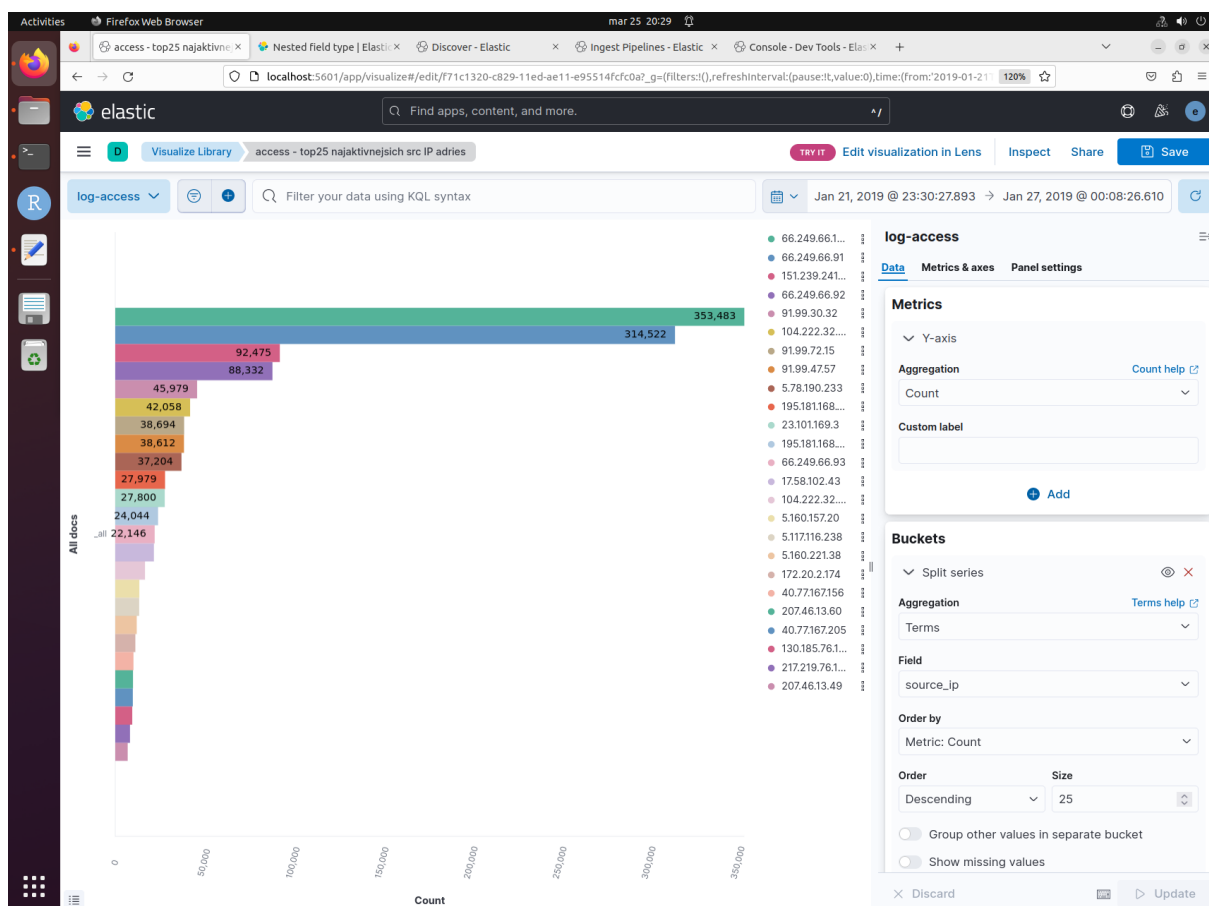
Kibana umožňuje tvorbu niekoľkých typov vizualizácií, ich hlavným rozdielom je spôsob prezentácie. Vizualizácie môžeme tvoriť z menu knižnice vizualizácií alebo priamo z dashboardov, vizualizácie sa vytvárajú nad Data View, resp. nad indexami. Vizualizácie môžu byť viacerých typov:

- **Lens** je novší, drag-and-drop, systém vytvorený pre zjednodušenie procesu tvorby vizualizácií. V podstate umožňuje rovnaké možnosti ako tvorba starších typov vizualizácií, jediný rozdiel je novšie UI a možnosť meniť typ vizualizácie za behu. V praxi, keď vieme akú vizualizáciu (napr. aký typ grafu) chceme vytvoriť, je prehľadnejšie použiť starý systém.
- **Agregation based** je starší, klasický, systém tvorby vizualizácií. Užívateľ musí najskôr zvoliť aký typ grafu chce použiť, zvoliť Data View alebo Index a následne vytvára vizualizáciu už zo zvolených dát a pomocou zvoleného grafu. Tento systém si popíšeme na príklade.
- **Mapy** sú vizualizácie u ktorých môžeme za pomoci GPS súradníc zobrazovať na mape rôzne agregácie nad dátami.

- **Custom Visualisation** je prostredie v ktorom je možné vytvárať pokročilé vizualizácie pomocou jazyka Vega.
- **TSVB** je nástroj na rýchle zobrazenie časových radov.

### 5.3.1 Vizualizácia pomocou jednoduchej agregácie

Vrámcí ukážky sme si vytvorili 2 vizualizácie, ktoré používajú jednoduchú **Terms** agregáciu, keďže sa jedná o rovnaký princíp opíšeme len jeden príklad. Vytvorili sme si stĺpcový graf zobrazujúci počet výskytov 25 najaktívnejších IP adries v dostupnom Apache access logu.



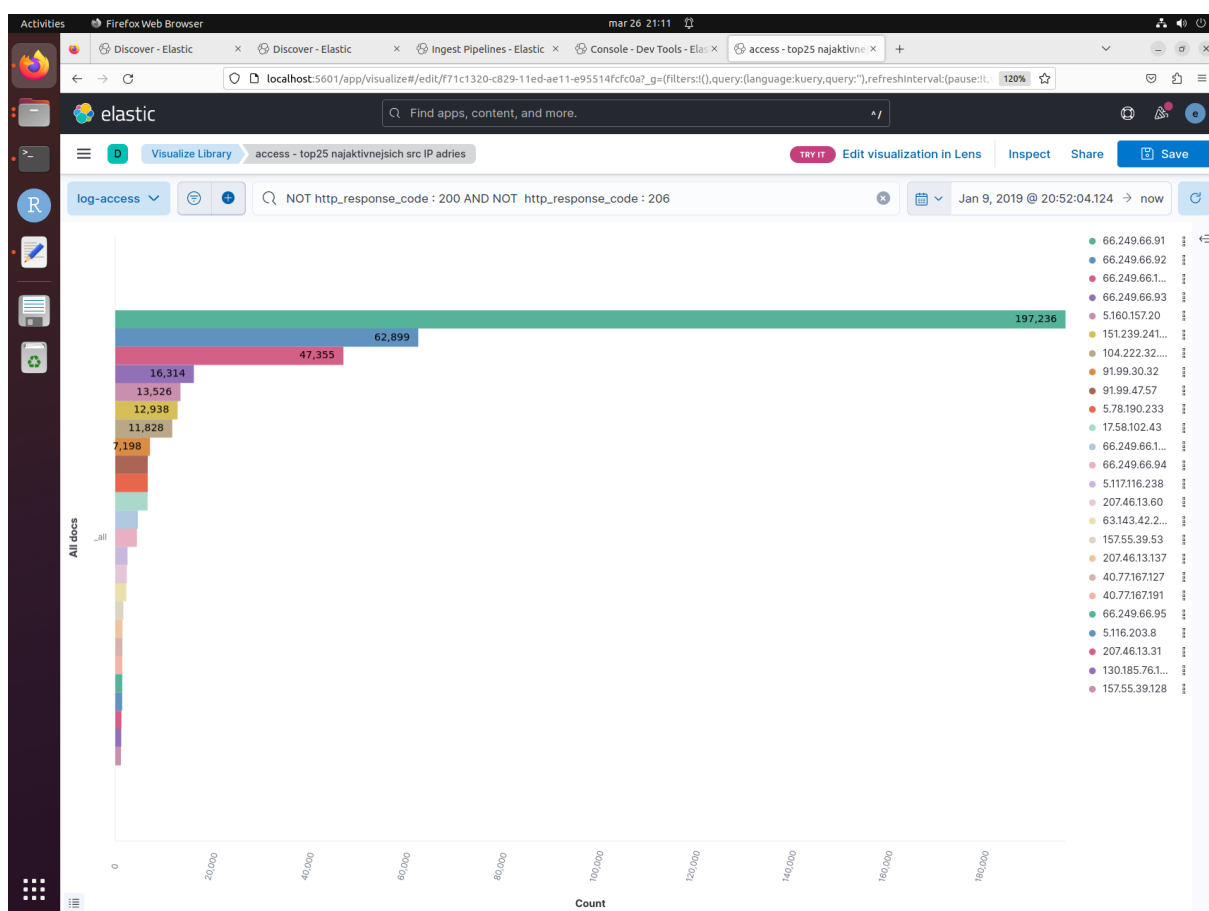
**Obrázok 8:** Stĺpcový graf zobrazujúci 25 najaktívnejších zdrojových IP adries,  
zdroj: [vlastné spracovanie]

Na y-osi osi sme ponechali ako agregáciu celkový počet (Count), v sekcii **Buckets**

sme vybrali rozdeliť sériu (Split Series), vybrali agregáciu **Terms**, aplikovali ju na pole `source_ip`, zvolili sme počet 25. V ďalšej záložke je možné nastaviť metriky a dáta, napríklad mód samotného stĺpcového grafu alebo nastavenia ôs. V poslednej záložke nastavujeme bočný panel s legendou (veľkosť, zalomenie textu atď.), farebnú schému grafu a podobne. Postup vytvárania je pri iných typoch grafov prakticky zhodný.

Pomocou Inspectoru môžeme exportovať hodnoty zobrazené v grafe v textovom formáte.

Vo vizualizáciách je takisto možné používať filtre. Filtre aplikované vo vizualizácii sa uložia s vizualizáciou, napr. v použití v dashboarde je daný filter aplikovaný len na konkrétnu vizualizáciu, nie celý dashboard.

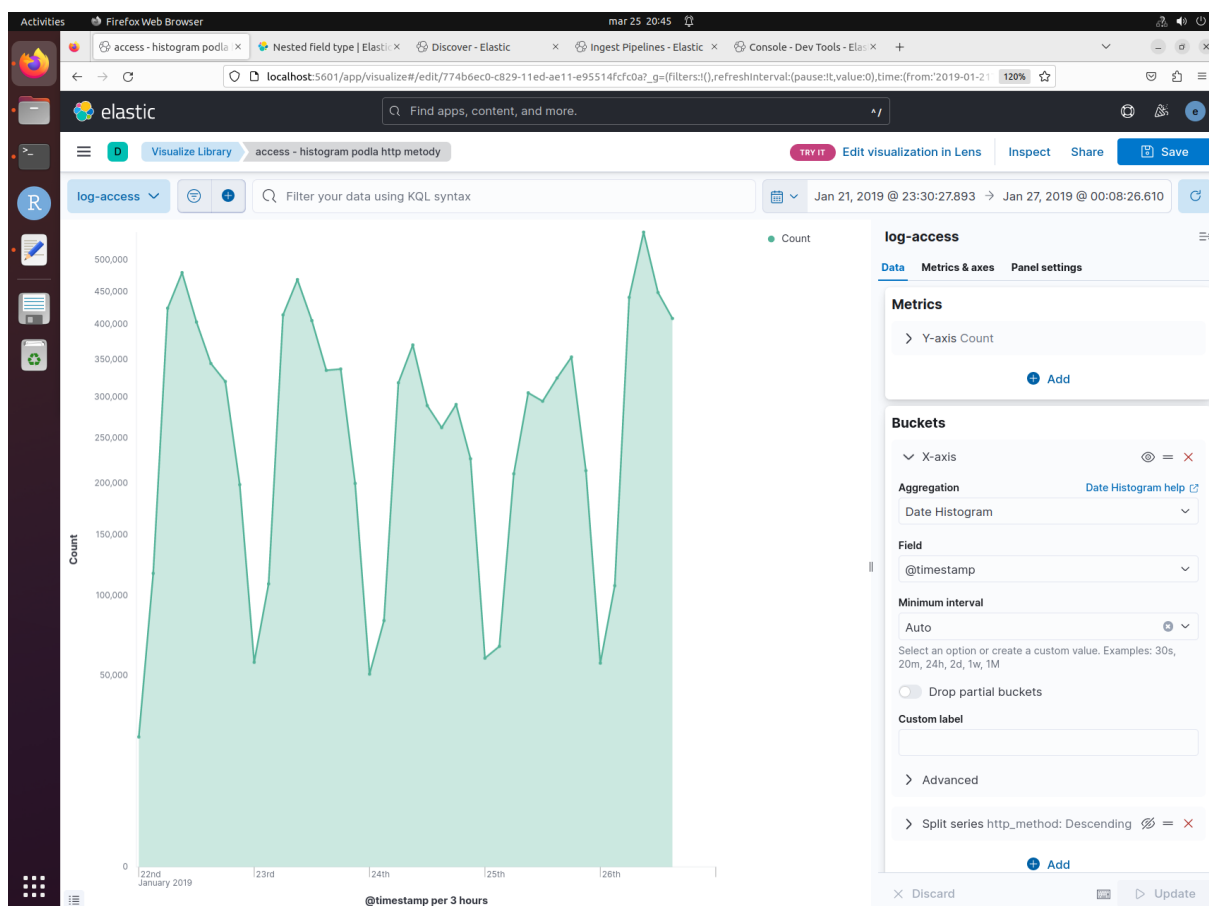


Obrázok 9: Stĺpcový graf s aplikovaným filtrom, zdroj: [vlastné spracovanie]

### 5.3.2 Kombinovanie agregácií

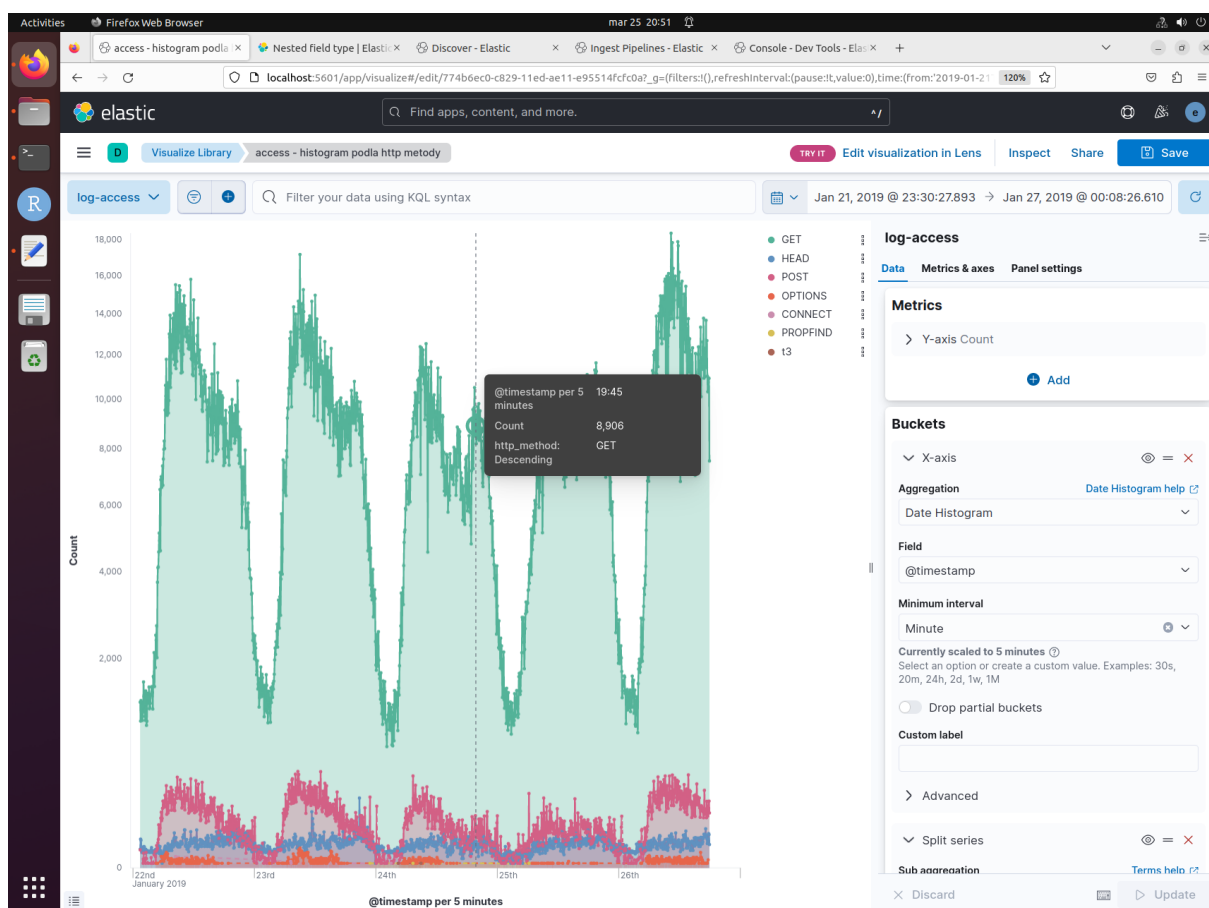
Agregácie je možné vo vizualizácií takisto kombinovať. V tejto ukážke vytvoríme časový histogram HTTP požiadaviek a rozdelíme ho podľa typu HTTP požiadavky. Použijeme tie isté dáta v podobe Apache access logov.

Na vizualizáciu použijeme **Area** graf, ako agregáciu na y-osi ponecháme celkový počet. Na X os vyberieme agregáciu **Date Histogram** podľa poľa **@timestamp**. Minimálny interval môžeme ponechať na auto, Kibana potom prispôbí veľkosť bucketov podľa množstva zobrazovaných dát. V prípade ak by naša vizualizácia vyžadovala presnejšie zobrazenie, napríklad za každú minútu, môžeme toto vo vizualizácii zohľadniť. Ak by sme v takom prípade zobrazovali viac ako niekoľko hodín dát, Kibana automaticky zväčší interval bucketov z dôvodu obmedzenia maximálneho počtu bodov ktoré môže na grafe zobrazíť.



Obrázok 10: Histogram všetkých HTTP požiadaviek, zdroj: [vlastné spracovanie]

Na grafe sa nám zobrazil časový histogram požiadaviek, ďalšie agregácie do neho môžeme pridávať pomocou `split series`, v našom prípade použijeme znova agregáciu `Terms` a ako pole vyberieme `http_method`. Z dôvodu prehľadnosti grafu nastavíme mierku y-osi ako Square root, ak by sme ponechali lineárnu mierku osi, málo zastúpené HTTP metódy by neboli na grafe dobre viditeľné. [1][2][3][4]



**Obrázok 11:** Vizualizácia zobrazujúca časový histogram HTTP požiadaviek rozdelených podľa HTTP metódy. Možno si všimnúť, že aj napriek tomu, že sme nastavili veľkosť bucketu na 1 minútu, body na grafe zobrazujú hodnotu za 5 minút, z dôvodu veľkého počtu dát na zobrazenie, zdroj: [vlastné spracovanie]

## Záver

Cieľom tohto zadania bolo opísať, nainštalovať Elastic stack aj jeho časti a ukázať spracovanie, analýzu a vizualizáciu logov zo zdrojového súboru, ako aj z live forti generovaných logov prostredníctvom skriptu fortigate.py. Cieľ, ktorý sme si zadali bol splnený.

Na základe reálnych skúseností či už praxe alebo analýzou rôznych iných nástrojov môžeme povedať že Elastic stack je vynikajúci nástroj pre spracovávanie logov. Vďaka jeho schopnosti zbierať, spracovávať a analyzovať obrovské množstvá logových dát z rôznych zdrojov. Elastic Stack umožňuje rýchle a efektívne odhalenie a riešenie problémov, ktoré by inak mohli zostať neodhalené.

Môžeme povedať, že použitie Elastic stacku na spracovanie logov je výhodné a účinné riešenie pre firmy a organizácie, ktoré sa zaoberajú veľkými objemami tohto typu dát.

Veríme, toto zadanie prinesie čitateľovi nie len obohatenie teoretických, ale aj praktických vedomostí o danej technológii.

## Zoznam použitej literatúry

- [1] ELASTICSEARCH B.V. 2023. *Elasticsearch Guide [8.6]* In elastic.co [online]. 2023. [citované dňa 27.03.2023]. Dostupné na internete: <https://www.elastic.co/guide/en/elasticsearch/reference/current/release-highlights.html>.
- [2] ELASTICSEARCH B.V. 2023. *Filebeat Guide [8.6]* In elastic.co [online]. 2023. [citované dňa 27.03.2023]. Dostupné na internete: <https://www.elastic.co/guide/en/beats/filebeat/current/index.html>.
- [3] ELASTICSEARCH B.V. 2023. *Kibana Guide [8.6]* In elastic.co [online]. 2023. [citované dňa 27.03.2023]. Dostupné na internete: <https://www.elastic.co/guide/en/kibana/current/deb.html>.
- [4] ELASTICSEARCH B.V. 2023. *Logstash Guide [8.6]* In elastic.co [online]. 2023. [citované dňa 27.03.2023]. Dostupné na internete: <https://www.elastic.co/guide/en/logstash/8.6/index.html>.