Universidade de Brasília
Faculdade UnB Gama

Programação de Sistemas Paralelos e Distribuídos

Projeto de Pesquisa Final

Programação de Streaming em clusters Spark/Kafka

Micaella Lorraine Gouveia de Lima - 17/0111288

Renan Welz Schadt - 16/0143403

Professor Fernando William Cruz

Introdução

A Programação Distribuída é quando a execução do sistema ocorre em vários ambientes que se tornam interligados por uma rede de internet ou intranet. Uma rede de computadores é definida como um sistema constituído pela conexão de dois ou mais computadores e seus periféricos com o objetivo de comunicação, compartilhamento e troca de dados. Já a programação paralela é uma forma de computação em que vários cálculos são realizados ao mesmo tempo, operando sob o princípio de que grandes problemas geralmente podem ser divididos em problemas menores, que então são resolvidos concorrentemente. Neste trabalho foi abordado o Apache Spark e Apache Kafka, além do banco ElasticSearch e o plugin de visualização Kibana.

Apache Spark

O Spark é um framework de código aberto para computação distribuída. Provê uma interface para programação de clusters com paralelismo e tolerância a falhas. Ele estende o modelo de programação MapReduce popularizado pelo Apache Hadoop, facilitando bastante o desenvolvimento de aplicações de processamento de grandes volumes de dados. Além do modelo de programação estendida, os componentes funcionam integrados na própria ferramenta, como o Spark Streaming, o Spark SQL e o GraphX.

O Spark possui dois tipos de processamento de dados: em batch e em streaming. O batch é um lote de pontos de dados que foram agrupados em um intervalo de tempo específico. Outro termo frequentemente usado para isso é uma janela de dados. Já o processamento de dados em stream lida com dados contínuos e é essencial para transformar de grandes a rápidos.

Apache Kafka

Já o Apache Kafka é uma plataforma open-source de processamento de streams. O projeto tem como objetivo fornecer uma plataforma unificada, de alta capacidade e baixa latência para tratamento de dados em tempo real. Sua camada de armazenamento é, essencialmente, uma fila de mensagens que utilizam o paradigma Publisher/Subscriber, maciçamente escalável projetada como um log de transações distribuídas, tornando-o altamente valioso para infra-estruturas corporativas que processam transmissão de dados.

ElasticSearch

O Elasticsearch é um mecanismo de busca e análise de dados distribuído, gratuito e aberto para todos os tipos de dados, incluindo textuais, numéricos, geoespaciais, estruturados e não estruturados. O Elasticsearch é desenvolvido sobre o Apache Lucene. Comumente chamado de ELK Stack (pelas iniciais de Elasticsearch, Logstash e Kibana), o Elastic Stack agora inclui uma rica coleção de agentes lightweight conhecidos como Beats para enviar dados ao Elasticsearch.

Kibana

O Kibana é um plugin de visualização de dados de fonte aberta para o Elasticsearch. Ele fornece recursos de visualização em cima do conteúdo indexado em um cluster Elasticsearch. Os usuários podem criar gráficos de barra, linha e dispersão, ou gráficos e mapas de torta em cima de grandes volumes de dados.

Metodologia

A dupla se organizou da seguinte forma:

1. Pareamentos remotos para alinhamento do que foi feito, das dificuldades encontradas, das possíveis soluções e definição de novas tarefas.
2. Desenvolvimento em conjunto de todas as partes do projeto.

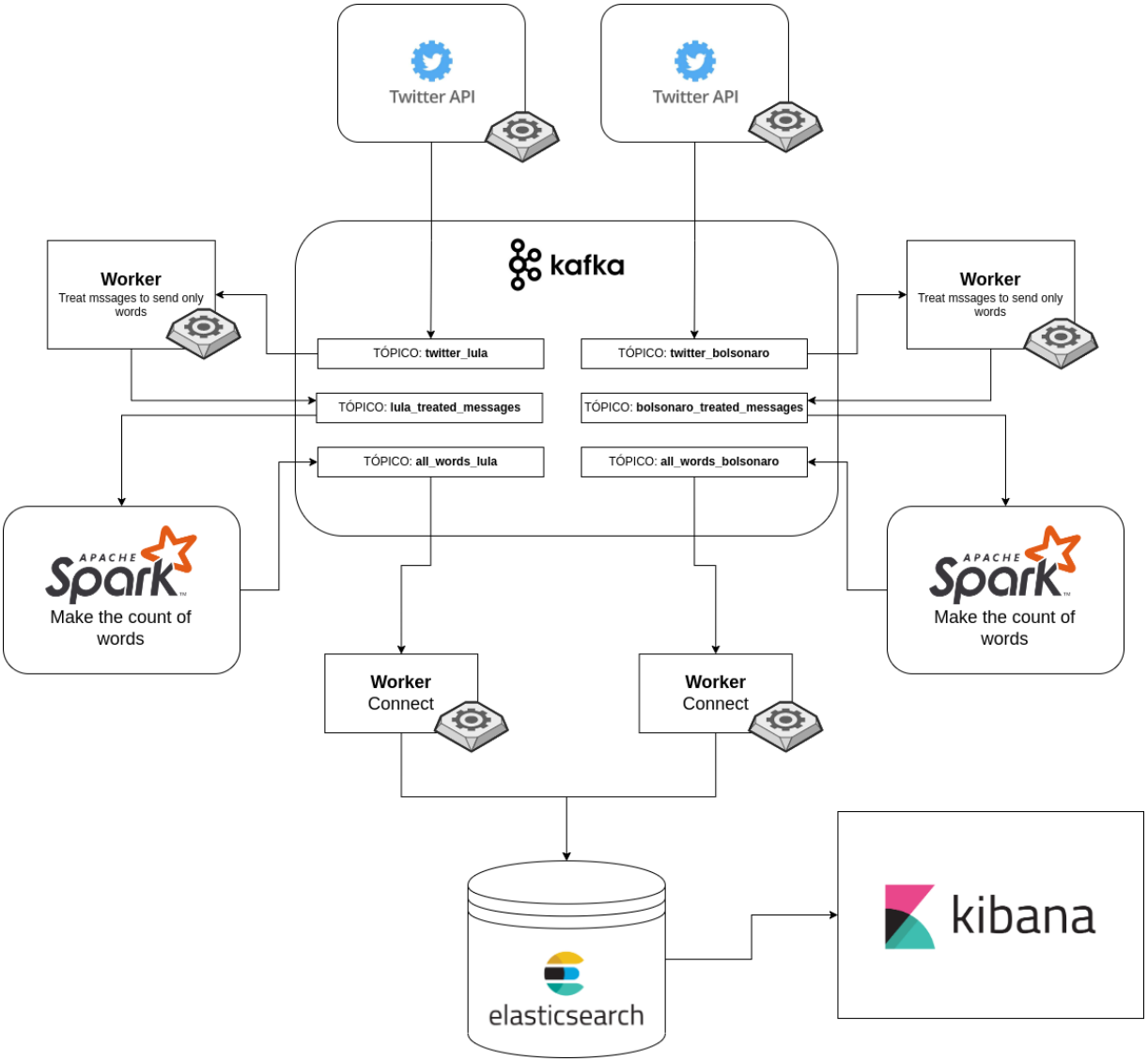
Objetivo

Faltando menos de um mês para as eleições de 2022, o objetivo do projeto é verificar as palavras mais utilizadas quando se fala dos principais candidatos à presidência, Luiz Inácio Lula da Silva e Jair Messias Bolsonaro dentro da rede social Twitter.

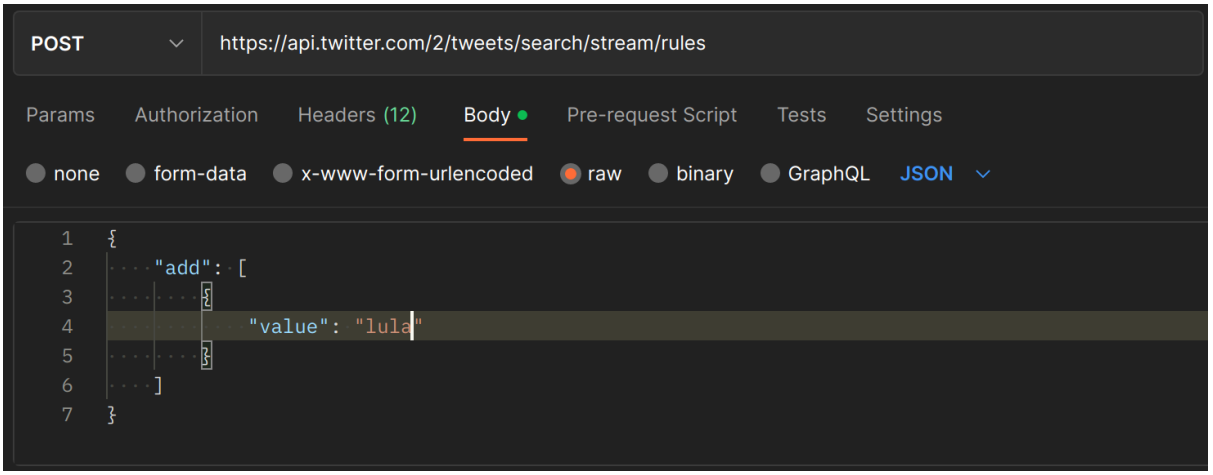
Twitter é uma rede social e um serviço de microblog, que permite aos usuários enviar e receber atualizações pessoais de outros contatos. Cada usuário pode publicar pequenas mensagens relatando suas posições sobre os mais diversos assuntos.

Implementação

Neste trabalho foi construído um serviço de análise de dados advindos das mensagens enviadas no Twitter. Para isso a seguinte arquitetura foi planejada:



Primeiramente, foram criadas duas contas de desenvolvedor na plataforma Twitter, sendo possível então consumir as mensagens enviadas de forma de stream. Uma das contas ficou responsável em produzir mensagens relacionadas ao candidato Lula e o outro ao candidato Bolsonaro. Para selecionar as palavras chaves a serem filtradas, foram criadas regras da seguinte forma:



Foram criados dois produtores de mensagens que enviam os dados brutos advindos da API do Twitter para um tópico específico do Kafka, sendo eles o twitter_lula e twitter_bolsonaro. Esses tópicos estão conectados com workers de tratamento, responsáveis em receber esses dados brutos, tratá-los, e enviar cada palavra tratada para novos tópicos específicos, sendo eles o lula_treated_messages e bolsonaro_treated_messages. O tratamento das mensagens e envio para o novo tópico foram feitos da seguinte maneira (worker.py):

```python
if __name__ == '__main__':

    exclude_words = ["de", "em", "a", "per", "por", "o", "do", "no", "ao", "pelo", "os", "dos",
        "nos", "aos", "pelos", "a", "da", "na", "à", "pela", "as", "das", "nas", "às", "pelas", "um", "dum",
        "num", "uns", "duns", "nuns", "uma", "duma", "numa", "umas", "dumas", "numas"]

    consumer = get_consumer()

    producer_lula = KafkaProducer(
        bootstrap_servers=KAFKA_SERVERS,
        value_serializer=lambda v: json.dumps(v).encode('utf-8'),
        key_serializer=lambda v: json.dumps(v).encode('utf-8'),
        acks='all',
        retries=3,
        request_timeout_ms=1000
    )

    while True:
        try:
            for event in consumer:
                print(f"{event.topic}: {event.value}" )
                message = json.loads(event.value)['data']['text']

                words = message.split()

                for word in words:
                    word = word.lower()
                    word = re.sub(r'[^\w\s]', '', word)

                    if not word.startswith("@") and word not in exclude_words and len(word) > 2:
                        producer_lula.send(
                            topic='lula_treated_messages',
                            value=word,
                            key=word,
                        )

    except Exception as exc:
```

Após o tratamento, duas aplicações Spark ficam recebendo essas palavras já tratadas e ficam responsáveis em fazer a contagem de cada palavra recebida. O Spark manda as palavras no modo update, ou seja, somente as linhas novas ou que foram atualizadas são mandadas ao tópico Kafka.

```
spark = SparkSession \
    .builder \
    .appName("SparkKafkaWordCount") \
    .getOrCreate()

spark.sparkContext.setLogLevel("ERROR")

# Subscribe to 1 topic
df = spark \
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "localhost:9092") \
  .option("subscribe", "lula_treated_messages") \
  .load()

# Split the lines into words
words = df.select(
   explode(
       split(df.value, " ")
   ).alias("value")
)

# Generate running word count
words = words.groupBy("value").count()
wordCounts = words.withColumnRenamed('count', 'key')

# Start running the query that prints the running counts to the console
# query = wordCounts \
#     .writeStream \
#     .format("console") \
#     .start() \
#     .awaitTermination()

finalStream = wordCounts.selectExpr("CAST(key AS STRING)","CAST(value AS STRING)").writeStream \
    .outputMode("update") \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("checkpointLocation", "/tmp") \
    .option("failOnDataLoss", "false") \
    .option("topic", "all_words_lula") \
    .start() \
```

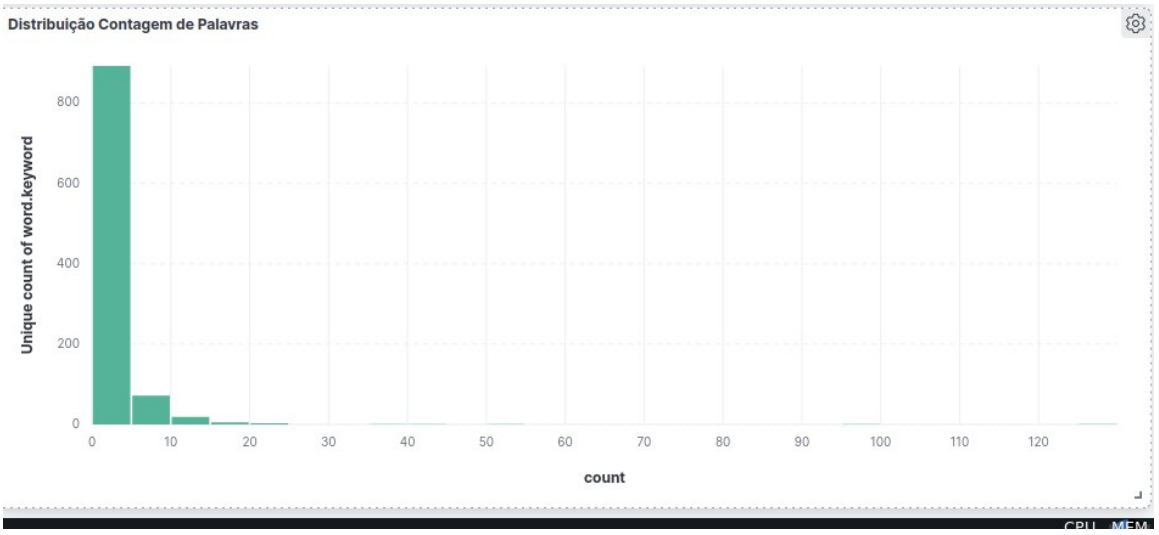
Após a contabilização, o Spark envia os resultados para novos tópicos do Kafka, sendo eles all_words_lula e all_words_bolsonaro. Foram criados dois scripts responsáveis em

receber esses resultados e enviar para o banco ElasticSearch.

```python
from kafka import KafkaConsumer
import json
from elasticsearch import Elasticsearch

# Password for the 'elastic' user generated by Elasticsearch
ELASTIC_PASSWORD = "9YPkwXkgfc65dmj9+LXC"

# # Create the client instance
client = Elasticsearch(
    "https://localhost:9200",
    ca_certs="/home/elasticsearch-8.4.1/config/certs/http_ca.crt",
    basic_auth=("elastic", ELASTIC_PASSWORD)
)

# Successful response!
print(client.info())

consumer = KafkaConsumer('all_words_lula')


for message in consumer:
    if message != None:
        print (f'{json.loads(message.value.decode("utf-8"))} - {json.loads(message.key.decode("utf-8"))}')

        doc = {
            'word': json.loads(message.value.decode("utf-8")),
            'count': json.loads(message.key.decode("utf-8"))
        }

        resp = client.index(index="words_lula", id=json.loads(message.value.decode("utf-8")), document=doc)

        print(resp['result'])
```

Os scripts realizam a conexão com o Elastic e colocam as palavras em seus respectivos índices, cada um relativo a um candidato, é salvo um json que contém a palavra e sua respectiva contagem, o id desse json é a própria palavra, ou seja, quando o Spark envia uma atualização as palavras atualizadas são automaticamente substituídas no Elastic.

Após a inserção dos dados no ElasticSearch, o plugin com o Kibana fica responsável em gerar gráficos relevantes para fazer uma análise sucinta dos dados coletados. O primeiro gráfico mostra a distribuição das palavras em relação ao número de ocorrência delas, a maioria das palavras tem seu contado entre 0 e 5:



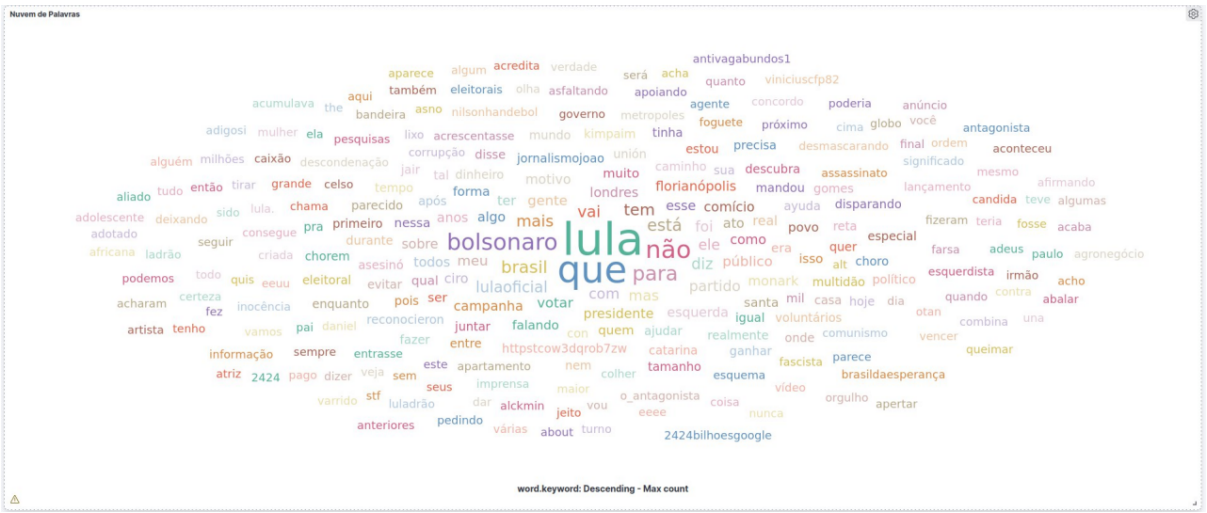
Com 10 minutos de coleta de dados, as 15 palavras mais utilizadas em relação ao candidato Lula são:

Lista de Palavras Capturadas

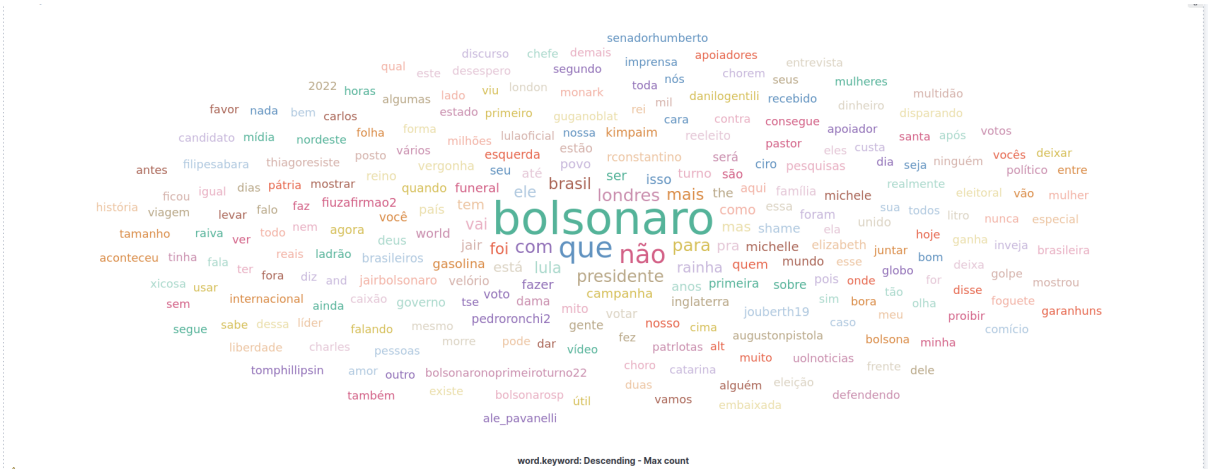
Top 15 values of word.keyword	Median of count
lula	127
que	98
não	51
bolsonaro	44
para	36
brasil	24
está	22
mais	20
tem	19

Já o segundo gráfico mostra uma nuvem de palavras para cada índice do ElasticSearch, podendo então verificar quais são as palavras mais faladas quando se fala de cada candidato.

Com 20 minutos de coleta de dados foi possível gerar a seguinte nuvem de palavras relacionadas ao candidato Lula:



Com os mesmos 20 minutos de coleta foi possível gerar a seguinte nuvem de palavras relacionadas ao candidato Bolsonaro:

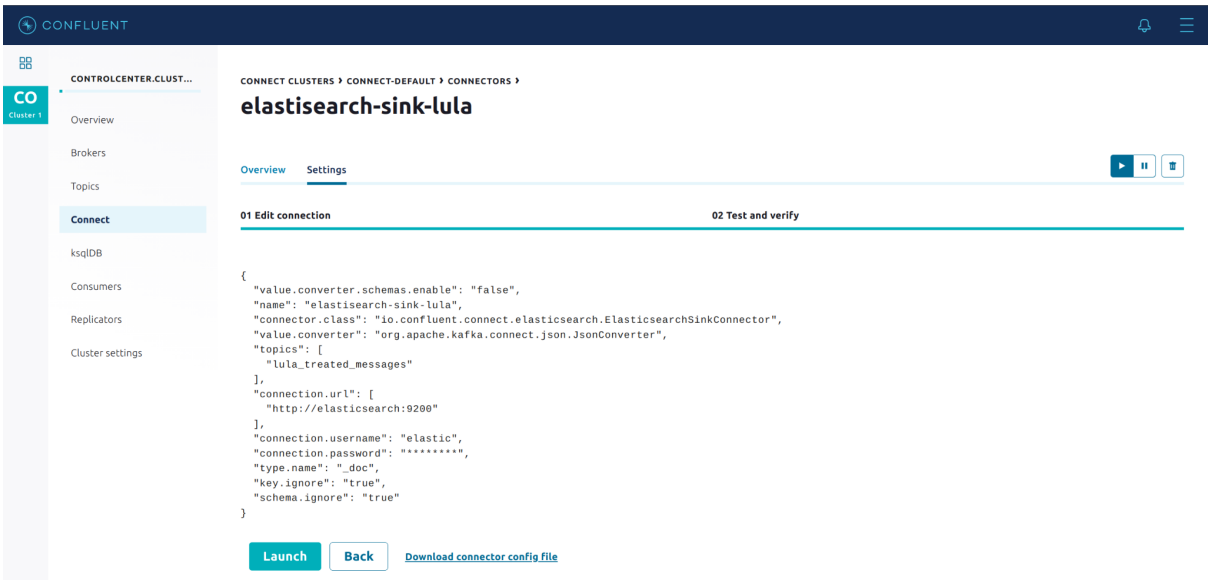


Uso do Kafka Connect

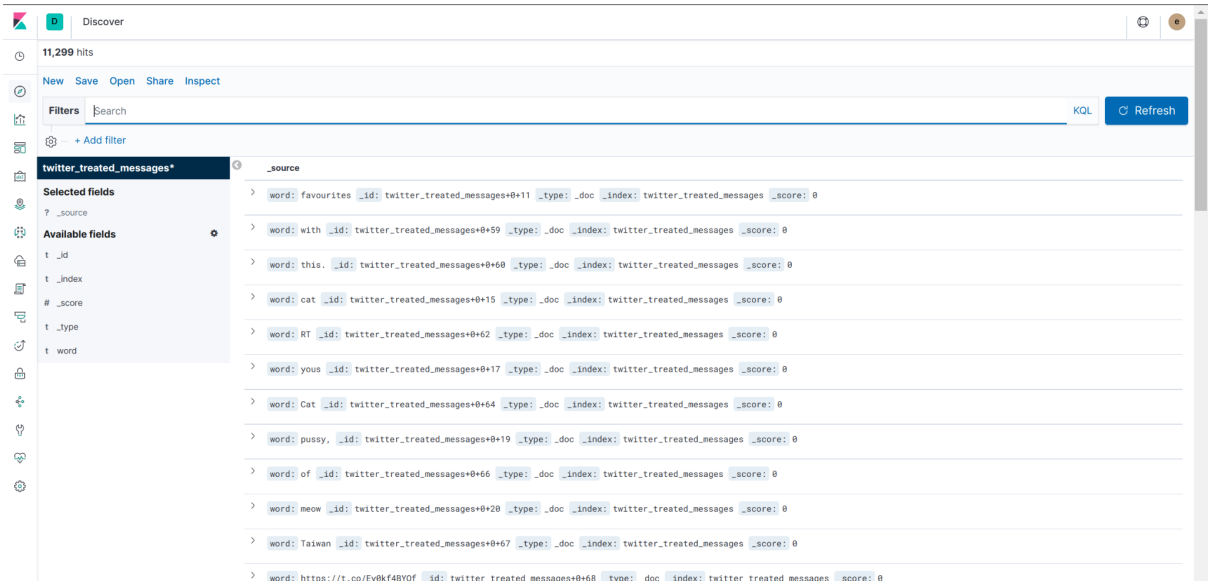
O Kafka Connect, componente de código aberto do Apache Kafka, é um framework para conectar o Apache Kafka a sistemas externos, como bancos de dados, armazenamentos de chave-valor, índices de pesquisa e sistemas de arquivos.

Para a solução principal não foi usado o Kafka Connect, mas para efeito de aprendizado, fizemos um projeto menor para entender o funcionamento do Kafka Connect. Para isso foi criada uma aplicação dockerizada que faz toda a parte de produção de mensagens do Twitter, tratamento do texto e envio das palavras para o Kafka. Nesta parte, as mensagens já eram mandadas para o ElasticSearch por meio do Kafka Connect.

A conexão do Kafka com o ElasticSearch usando o Kafka Connect foi feita usando a interface gráfica Confluent Control Center fazendo a seguinte configuração:



É possível verificar pelo Kibana o recebimento das palavras no ElasticSearch:



Conclusão

O experimento foi um ótimo aprendizado, dando oportunidade de conhecer e aprofundar em novas tecnologias que lidam com programação paralela e distribuída. Com as experiências passadas conseguimos adquirir novas habilidades e conseguimos desenvolver de uma forma mais eficiente. Nossa maior dificuldade foi o processamento de todas as ferramentas juntas. Elas demandam uma grande quantidade de memória RAM e muitas vezes o computador travava ou demorava muito a responder, também tivemos algumas dificuldades com a API do Twitter, que parava de executar as requisições após um certo limite diário.

- **Relato (Micaella Gouveia)**

O experimento me trouxe novos desafios, principalmente por lidar com streaming de dados em multi nós, e trazer uma nova tecnologia que não tinha nenhum contato anteriormente, que é o ElasticSearch e Kibana. Eu particularmente fiquei mais responsável pela parte do stream de dados vindos da API do Twitter e Kafka em geral, além estar a frente no teste do Kafka Connect. Pude aprofundar meus conhecimentos em Kafka e aprender bastante sobre seu funcionamento. Mesmo com as complicações do final de semestre, acredito que eu e minha dupla nos empenhamos e entregamos um trabalho interessante.

- **Relato (Renan Schadt)**

O experimento trouxe grande aprendizado, foi muito interessante pensar no fluxo como um todo, envolvendo diversos tópicos Kafka, Spark, Elastic e Kibana. A nuvem de

palavras geradas no final ficou bem interessante e dá pra tirar algumas informações bem valiosas a partir de outros gráficos que experimentamos no Kibana. Tanto o Elastic como o Kibana são ferramentas enormes, acredito que com mais tempo poderíamos ter explorado outras visualizações.