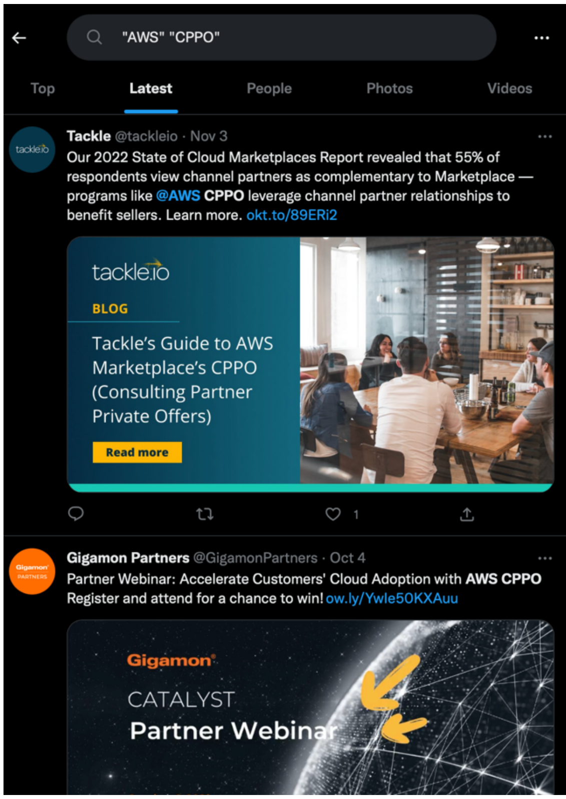
Harvard University Extension School
"Principles of Big Data Processing"
CSCI E-88, Fall 2022
Final Project
by Michael Charara

Project Goal and Problem Statement

This project aims to study real-time tweets focused around the Cloud Marketplaces for Amazon Web Services, Google Cloud Platform, and Microsoft Azure. I will demonstrate how to build a system that collects twitter data which mentions this information, index them into ElasticSearch for further analytics, and visualize with Kibana.

Terms:
"AWS Marketplace"
"GCP Marketplace"
"Azure Marketplace"


"AWS" "CPPO"
Top
Latest
People
Photos
Videos

Tackle @tackleio · Nov 3
Our 2022 State of Cloud Marketplaces Report revealed that 55% of respondents view channel partners as complementary to Marketplace — programs like @AWS **CPPO** leverage channel partner relationships to benefit sellers. Learn more. okt.to/89ERi2

tackle.io
BLOG
Tackle's Guide to AWS Marketplace's CPPO (Consulting Partner Private Offers)
Read more
1

Gigamon Partners @GigamonPartners · Oct 4
Partner Webinar: Accelerate Customers' Cloud Adoption with **AWS CPPO**
Register and attend for a chance to win! ow.ly/Ywle50KXAuu

GitHub

https://github.com/netdevmike/CSCIE-88-Final-Project

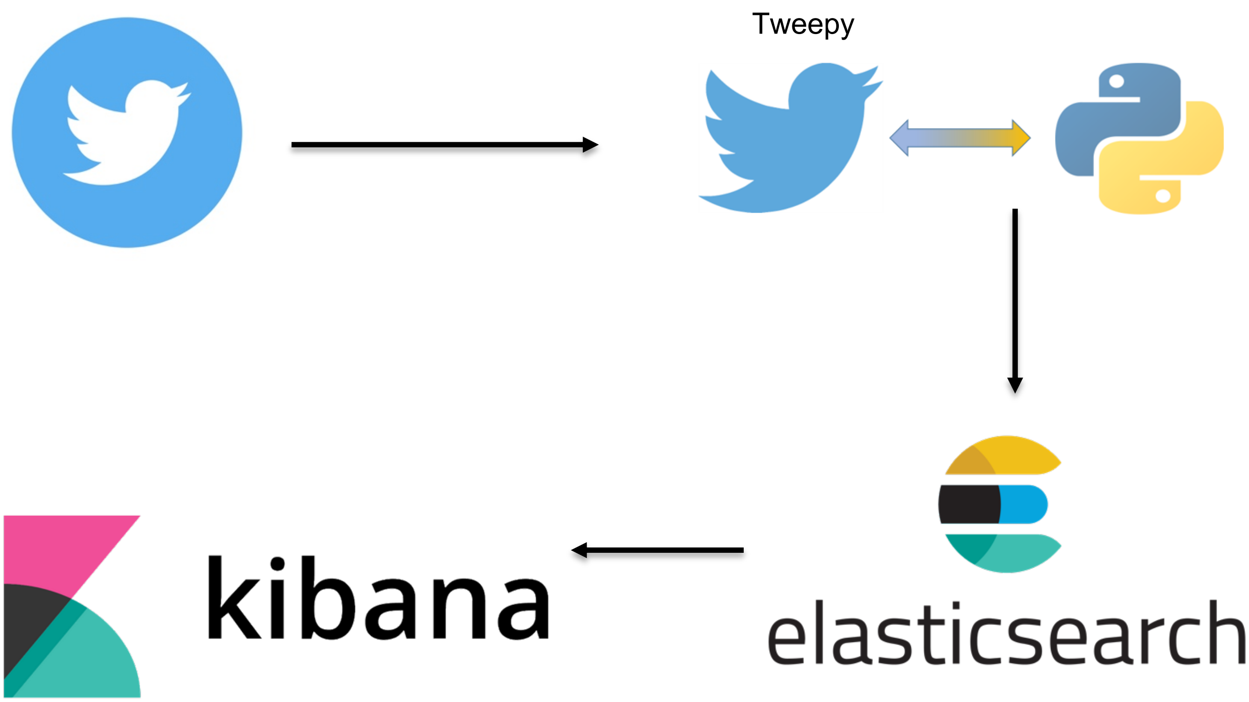
Big Data Source

Twitter APIv2
For twitter I used the Twitter API v2 data dictionary to pull the 'root-level' attributes.
https://developer.twitter.com/en/docs/twitter-api/data-dictionary/introduction

Expected results

Expect to find which marketplace, and more specifically, which partner marketplace program, has the most mentions worldwide.

Processing Pipeline



Pipeline Overview and Technologies used

- Collect data using Twitter API v2 using Python/Tweepy.
  - Tweepy is an open source Python package that gives you a very convenient way to access the Twitter API with Python. Tweepy includes a set of classes and methods that represent Twitter's models and API endpoints, and it transparently handles various implementation details, such as: Data encoding and decoding.
    - Data is exported to a CSV file
    - https://www.tweepy.org
- Messaging/Stream Processing Tier: Push data to Elasticsearch using Python
  - Eland and Pandas are used to push CSV file to Elasticsearch
    - Eland is a Python client and toolkit for DataFrames and machine learning in Elasticsearch.
      - https://www.elastic.co/guide/en/elasticsearch/client/eland/current/overview.html
    - Pandas is a open source data analysis and manipulation tool built on top of the Python programming language
      - https://pandas.pydata.org

- Visualization Tier: Kibana with ElasticSearch to visualize received data and discover which cloud marketplaces are the most popular
    - Kibana is a free and open frontend application that sits on top of the Elastic Stack, providing search and data visualization capabilities for data indexed
        - https://www.elastic.co/what-is/kibana
    - Elasticsearch is a distributed, free and open search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured.
        - https://www.elastic.co/what-is/elasticsearch

Implementation

First I created a config.py file to store my credentials

```
API_KEY = "XXX"
API_SECRET_KEY = "XXX"
ACCESS_TOKEN = "XXX"
ACCESS_TOKEN_SECRET = "XXX"
BEARER_TOKEN = "XXX"

CLOUD_ID = "Cloud-Marketplace-Twitter-Data:XXX
APIKEY_ID = "XXX"
APIKEY_KEY = "XXX"
```

I then created a connection.py file to store my connections to both tweepy and elasticsearch

```
import tweepy
from elasticsearch import Elasticsearch

import config

index = "cloud_marketplace_data_20221210"

es_client = Elasticsearch(
    cloud_id=config.CLOUD_ID,
    api_key=(config.APIKEY_ID, config.APIKEY_KEY)
)

client = tweepy.Client(bearer_token=config.BEARER_TOKEN)

print(es_client.info())
```

Created a getTweets.py file which is used to collect the tweets. Note that I separated AWS GCP and Azure into separate queries. This way, based on what was found in the tweet, the added data would append the additional column with either AWS, Azure, or GCP, allowing the data to be visualized based on Cloud Marketplace.

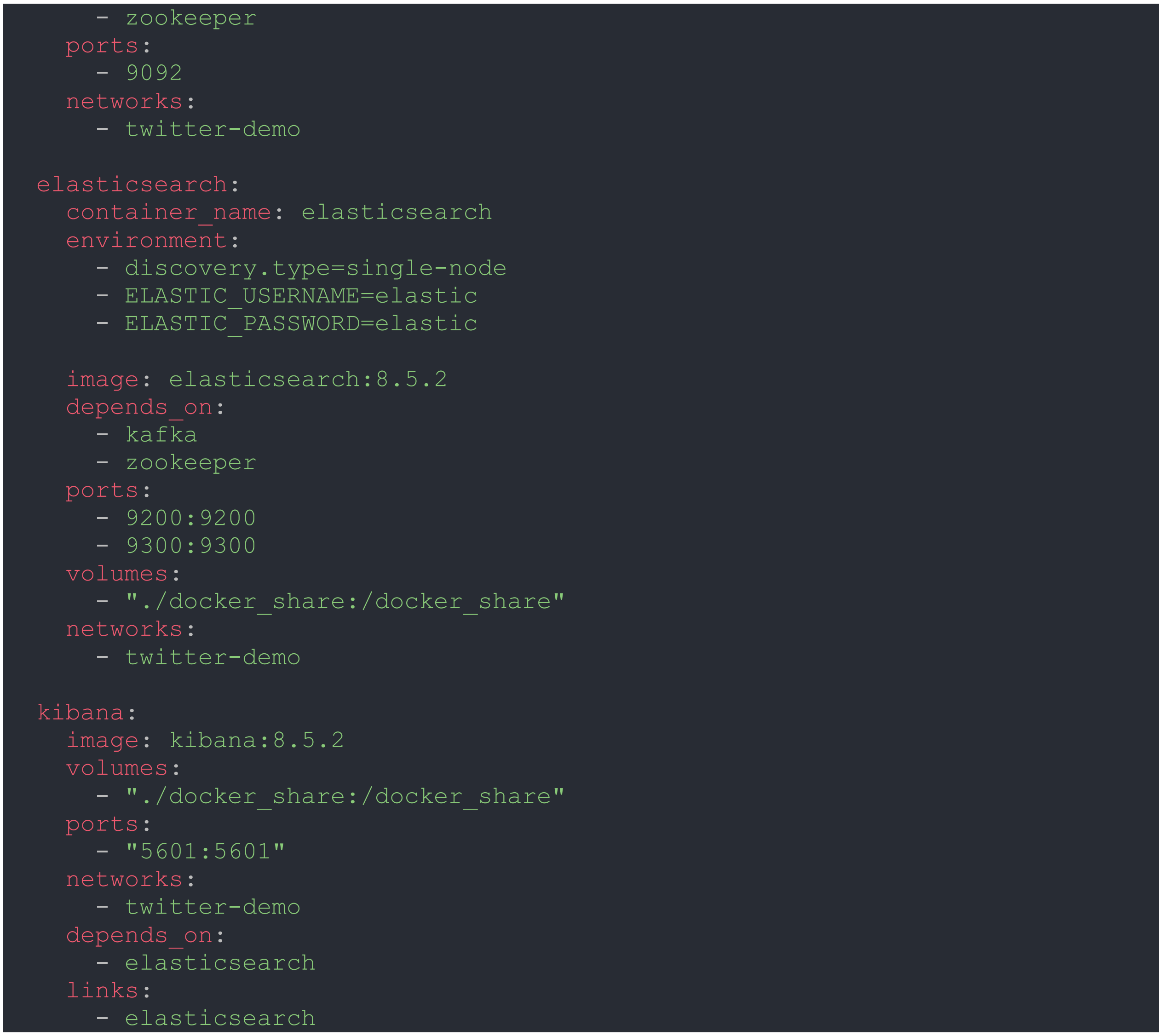
```
import tweepy
import config
import pandas as pd
from elasticsearch import Elasticsearch

# index = "Cloud_Marketplace_Data"
es = Elasticsearch(
    cloud_id=config.CLOUD_ID,
    api_key=(config.APIKEY_ID, config.APIKEY_KEY),
)

client = tweepy.Client(bearer_token=config.BEARER_TOKEN)

AWS = "AWS Marketplace -is:retweet"
GCP = "GCP Marketplace -is:retweet"
Azure = "Azure Marketplace -is:retweet"

columns = ['User', 'Tweet', 'Cloud']
data = []

for tweet in tweepy.Paginator(client.search_recent_tweets, query=AWS,
max_results=100).flatten(limit=1000):
    data.append([tweet.id, tweet.text, 'AWS'])

for tweet in tweepy.Paginator(client.search_recent_tweets, query=GCP,
max_results=100).flatten(limit=1000):
    data.append([tweet.id, tweet.text, 'GCP'])

for tweet in tweepy.Paginator(client.search_recent_tweets, query=Azure,
max_results=100).flatten(limit=1000):
    data.append([tweet.id, tweet.text, 'Azure'])

df = pd.DataFrame(data, columns=columns)

print(df)

df.to_csv('tweets.csv')
```

Used eland and pandas to send the data to to elasticsearch

```
import eland as ed
import pandas as pd
from connection import es_client, index


def main():
    df = pd.read_csv(
        'tweets.csv',
    )

    ed.pandas_to_eland(
        pd_df=df,
        es_client=es_client,
        es_dest_index=index,
        es_if_exists="replace",
    )


if __name__ == '__main__':
    main()
```

I also ran this locally in docker and on the elastic cloud. Below is the Docker file I used while implementing the local version. Do note that I could not run the elk container on my M1 mac due to it not being compatible with ARM. I had to set up Elastic, logstash, Kafka, and Kibana individually.

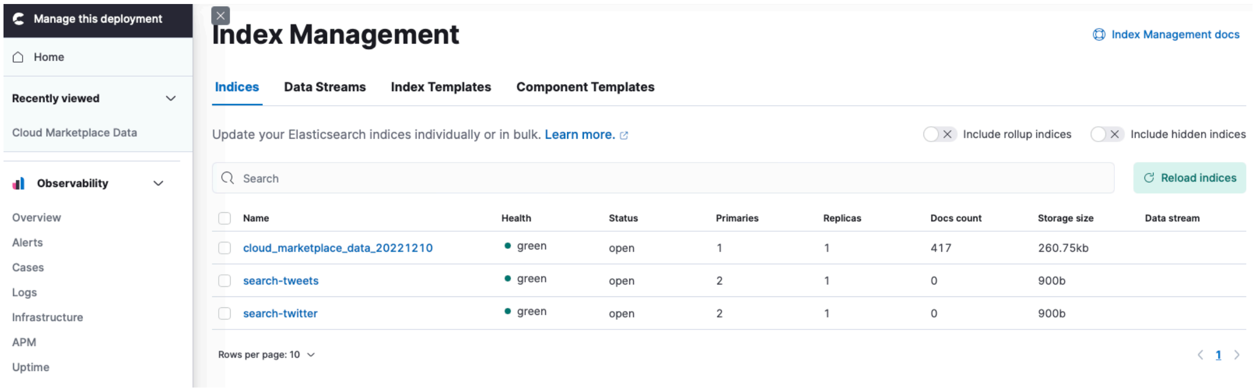
```
        version: '3.5'

networks:
  twitter-demo:
    name: twitter-demo-net
    driver: bridge

services:
  zookeeper:
    image: zookeeper
    ports:
      - "2181:2181"
    networks:
      - twitter-demo

  kafka:
    image: wurstmeister/kafka
    environment:
      # KAFKA_BROKER_ID: 1
      # (Hack for Mac)use this if you want to have docker host node to be
used as broadcast ip
      HOSTNAME_COMMAND: "/sbin/ip route|awk '/src/ { print $$NF }'"
      # Use below for Linux
      # HOSTNAME_COMMAND: "ip route get 1.2.3.4 | awk '{print $$7}'"
      KAFKA_ADVERTISED_PORT: 9092
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENERS: PLAINTEXT://kafka:9092
      # KAFKA_CREATE_TOPICS: "varnish_raw_logs:10:1"
    depends_on:
```



```yaml
      - zookeeper
    ports:
      - 9092
    networks:
      - twitter-demo

  elasticsearch:
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - ELASTIC_USERNAME=elastic
      - ELASTIC_PASSWORD=elastic

    image: elasticsearch:8.5.2
    depends_on:
      - kafka
      - zookeeper
    ports:
      - 9200:9200
      - 9300:9300
    volumes:
      - "./docker_share:/docker_share"
    networks:
      - twitter-demo

  kibana:
    image: kibana:8.5.2
    volumes:
      - "./docker_share:/docker_share"
    ports:
      - "5601:5601"
    networks:
      - twitter-demo
    depends_on:
      - elasticsearch
    links:
      - elasticsearch
```

Results

Image below shows the index which was created in the python script

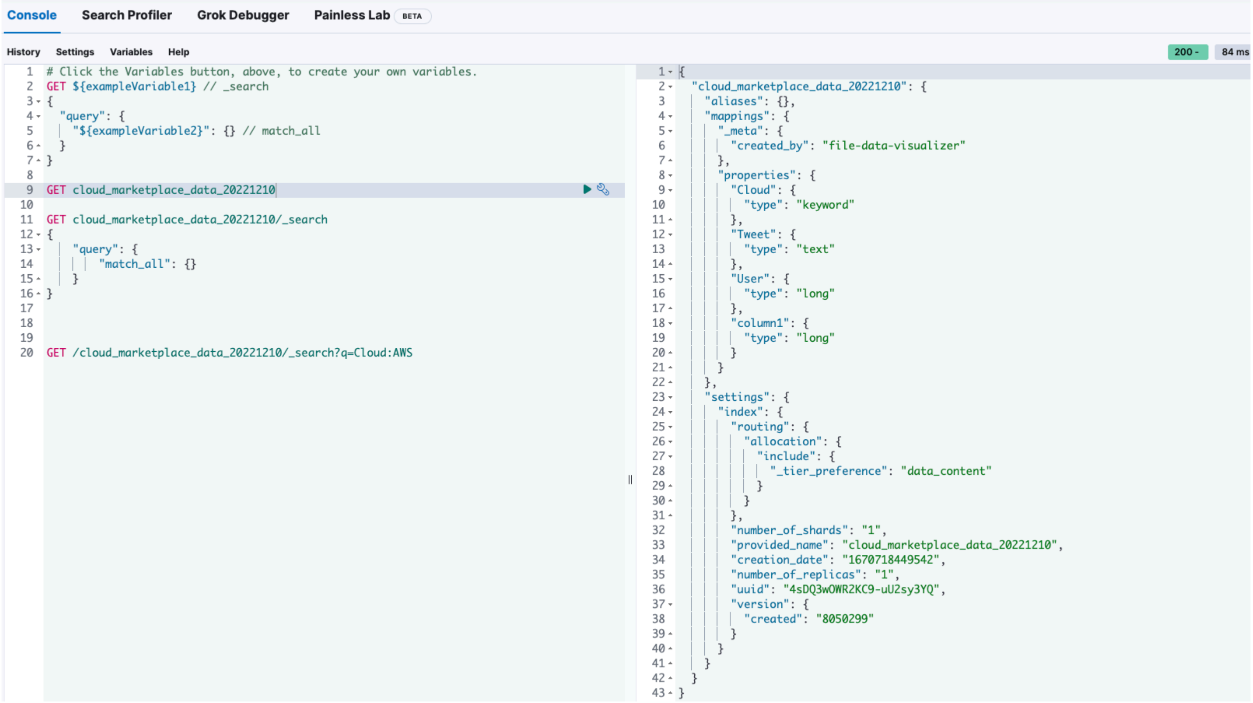


Index Management

Update your Elasticsearch indices individually or in bulk. Learn more.

Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Data stream
cloud_marketplace_data_20221210	● green	open	1	1	417	260.75kb	
search-tweets	● green	open	2	1	0	900b	
search-twitter	● green	open	2	1	0	900b	

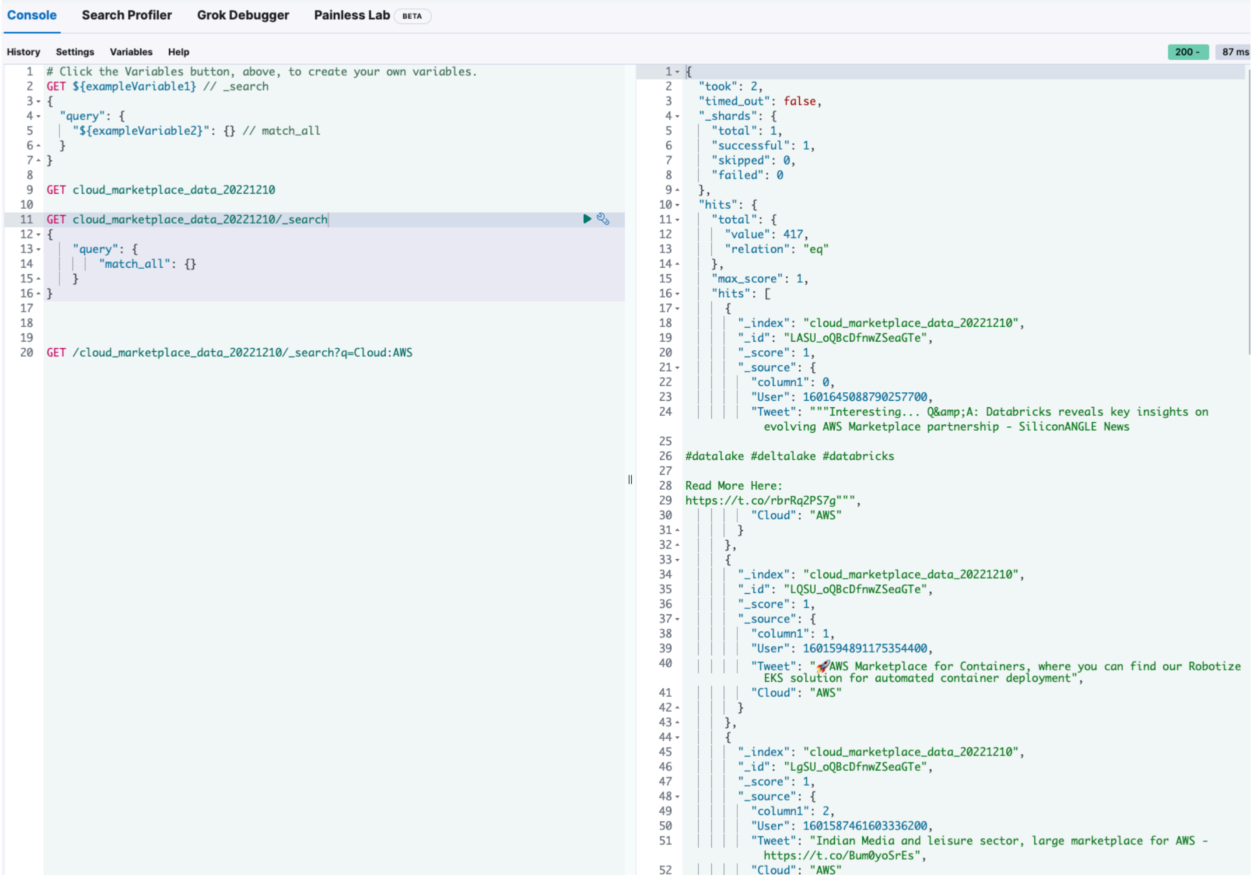
The index



```
# Click the Variables button, above, to create your own variables.
GET ${exampleVariable1} // _search
{
  "query": {
    "${exampleVariable2}": {} // match_all
  }
}

GET cloud_marketplace_data_20221210

GET cloud_marketplace_data_20221210/_search
{
    "query": {
        "match_all": {}
    }
}



GET /cloud_marketplace_data_20221210/_search?q=Cloud:AWS
```

```
{
  "cloud_marketplace_data_20221210": {
    "aliases": {},
    "mappings": {
      "_meta": {
        "created_by": "file-data-visualizer"
      },
      "properties": {
        "Cloud": {
          "type": "keyword"
        },
        "Tweet": {
          "type": "text"
        },
        "User": {
          "type": "long"
        },
        "column1": {
          "type": "long"
        }
      }
    },
    "settings": {
      "index": {
        "routing": {
          "allocation": {
            "include": {
              "_tier_preference": "data_content"
            }
          }
        },
        "number_of_shards": "1",
        "provided_name": "cloud_marketplace_data_20221210",
        "creation_date": "1670718449542",
        "number_of_replicas": "1",
        "uuid": "4sDQ3wOWR2KC9-uU2sy3YQ",
        "version": {
          "created": "8050299"
        }
      }
    }
  }
}
```

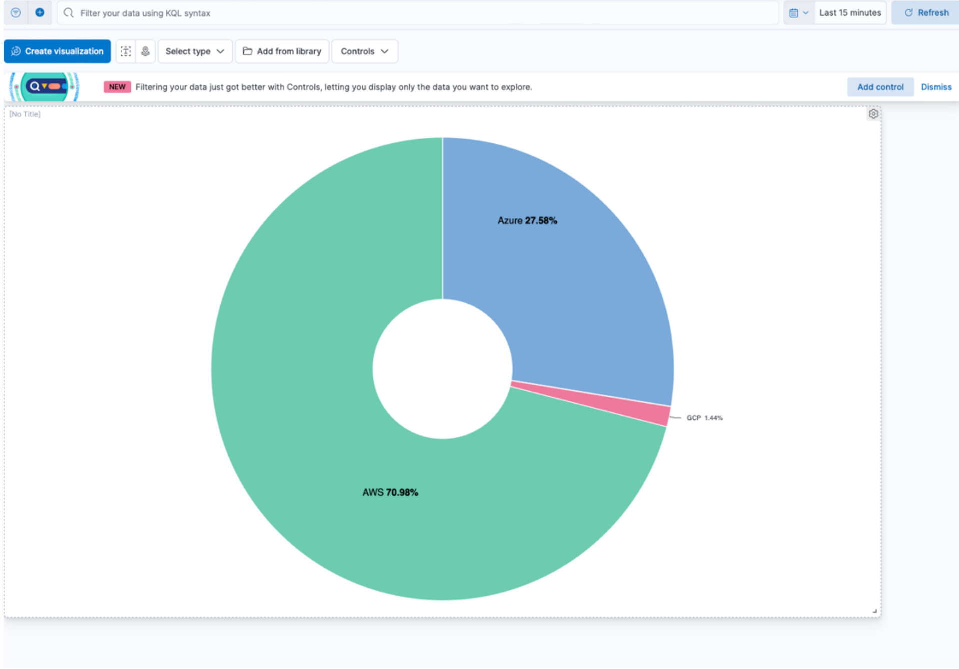
Data in the Index



```
# Click the Variables button, above, to create your own variables.
GET ${exampleVariable1} // _search
{
  "query": {
    "${exampleVariable2}": {} // match_all
  }
}

GET cloud_marketplace_data_20221210

GET cloud_marketplace_data_20221210/_search
{
    "query": {
        "match_all": {}
    }
}



GET /cloud_marketplace_data_20221210/_search?q=Cloud:AWS
```

```
{
  "took": 2,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 417,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "cloud_marketplace_data_20221210",
        "_id": "LASU_oQBcDfnwZSeaGTe",
        "_score": 1,
        "_source": {
          "column1": 0,
          "User": 1601645088790257700,
          "Tweet": """Interesting... Q&amp;A: Databricks reveals key insights on
            evolving AWS Marketplace partnership - SiliconANGLE News

#datalake #deltalake #databricks

Read More Here:
https://t.co/rbrRq2PS7g""",
          "Cloud": "AWS"
        }
      },
      {
        "_index": "cloud_marketplace_data_20221210",
        "_id": "LQSU_oQBcDfnwZSeaGTe",
        "_score": 1,
        "_source": {
          "column1": 1,
          "User": 1601594891175354400,
          "Tweet": "🚀AWS Marketplace for Containers, where you can find our Robotize
            EKS solution for automated container deployment",
          "Cloud": "AWS"
        }
      },
      {
        "_index": "cloud_marketplace_data_20221210",
        "_id": "LgSU_oQBcDfnwZSeaGTe",
        "_score": 1,
        "_source": {
          "column1": 2,
          "User": 1601587461603336200,
          "Tweet": "Indian Media and leisure sector, large marketplace for AWS -
            https://t.co/Bum0yoSrEs",
          "Cloud": "AWS"
```

Data being visualized



NEW Filtering your data just got better with Controls, letting you display only the data you want to explore.

Conclusions and Lesson Learned

Describe what you have learned during this project:

- What issues did you have?
  - Issues with running ELK stack locally on M1 mac
  - Issues authentication with elastic from python script
  - Issues authenticating with Twitter APIv2
- What limitations, if any, did you run into with the technologies used?
  - The main limitation I ran into was trying to run everything locally on my M1 mac due to the arm chip not being compatible.
  - Another limitation was that I would need to run the script to maintain the pipeline because I was ingesting with python.
- What would you do differently next time?
  - One thing I wanted to do was measure tweet sentiment (positivity/negativity). Although AWS showed a larger number of tweets, perhaps Azure or GCP had a higher ratio of positive tweets.
  - This is functionality is actually included functionality in the twitter API
    - https://developer.twitter.com/en/blog/community/2020/how-to-analyze-the-sentiment-of-your-own-tweets
  - Unfortunately, I ran into so many issues trying to host everything locally on my mac I did not have time to implement this functionality.
- What alternatives to technologies you used you might consider?
  - If I could go back, I would have begun with elastic cloud from the beginning or hosted the ELK stack in an EC2. I wanted to host everything locally on my mac, but I wasted a lot of time debugging and trying to get the stack to work that I ran out of time and could not upgrade my program's functionality.
- Where would you take your project next?
  - I will expand on the project. I want to run the python script in Google Clouds Cloud Run. I want to set the code up daily and include the date in the index. This would allow me to collect data daily and visualize the data in Kibana over time. I would also want to add the sentiment functionality. Regarding Elastic, I would continue to use the cloud version over hosting this locally. Using elastic cloud with GCP Cloud run would allow for full automation.