Relazione Progetto: amazonAnalyzerKubernetes

Federico Canzonieri 1000024369

Luglio 2022

Contents

1 Introduzione

Questo progetto fa uso di diverse tecnologie per costruire una data pipeline che ha come sorgente dati **Amazon** (le recensioni di un prodotto) , al fine di eseguire della **sentiment analysis** per capire come variano le recensioni degli utenti.
In particolare le tecnologie usate sono **Kubernetes** e qualche servizio di **AWS** come **S3**, **Lambda**, **Amazon Opensearch Service** e **Amazon Managed Grafana** . Il repository è disponibile al seguente indirizzo: `https://github.com/federicocanzonieri/amazonAnalyzerKubernetes`.

Vediamo l'architettura dell'applicazione per capire il flusso dati.

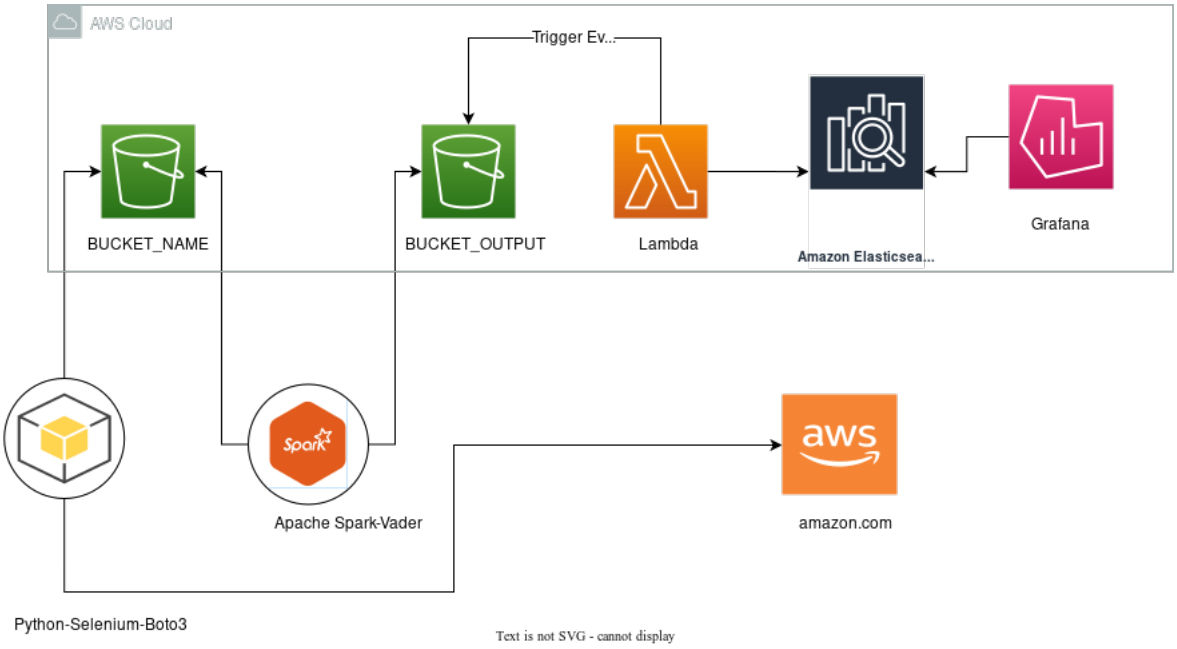

Figure 1: I Cerchi sono Pod di K8S

Come si può vedere dal grafico il flusso dei dati è il seguente: Python → S3 → Spark → S3 → Lambda → Amazon Opensearch Service/Grafana

2 Componenti

2.1 Python

Il primo componente in questa pipeline è il servizio di **scraping** dei dati da amazon. Tale servizio è costituito da una pod con al suo interno **Python** e **Selenium**, quest'ultimo è richiesto per eseguire il Javascript delle pagine web. Ci sono differenti parametri che si possono settare sul **python-deploment.yaml**

- CODE_PRODUCT: codice del prodotto da analizzare
- START_PAGE: numero pagina da cui iniziare
- END_PAGE: numero pagina finale
- MODE_REVIEWS: "recent" or "useful"
- DOMAIN_URL: "it","co.uk"

Talvolta può capitare che le richieste vengano bloccate tramite un **ReCaptcha**, per minimizzare la possibilità che ciò avvenga viene usata una libreria python **fake_useragent**, usata per arrichire le richieste e renderle più credibili.
Per il parsing delle pagine viene usata la libreria **BeautifulSoup**, per estrarre i dati interessanti:

- title, titolo della recensione
- rating, le stelle date al prodotto (1 a 5)
- body, la recensione
- date, data della recensione
- name, nome utente del utente
- verified_buy, se l'utente è verificato o meno
- helpful_vote, se la recensione ha avuto voti positivi
- country, nazione

Estrati i dati e trasformati in JSON, viene usato **boto3**, per inviare i dati sul bucket S3 [2].

```
        s3 = boto3.client('s3',aws_access_key_id=AWS_ACCESS_KEY_ID,
         aws_secret_access_key=AWS_SECRET_ACCESS_KEY )
        # print(dir(s3))
#       print(review)
        global counter
        s3.put_object(
            Body=json.dumps(review),
            Bucket=BUCKET_NAME,
            Key=NAME_FILES_S3+str(counter)
        )
```

Figure 2: Boto3

2.2 S3

I dati dal python vengono memorizzati su un bucket S3 [3].

2.3 Apache Spark

Il servizio di Apache Spark che gira su un altro pod, va a prendersi i dati dal bucket ed effettua delle operazioni su di esso tra cui il calcolo dello score/polarità della recensione, per capire se l'accezione è negativa, positiva o neutra, infine li va a mettere i dati processati su un altro bucket [4].

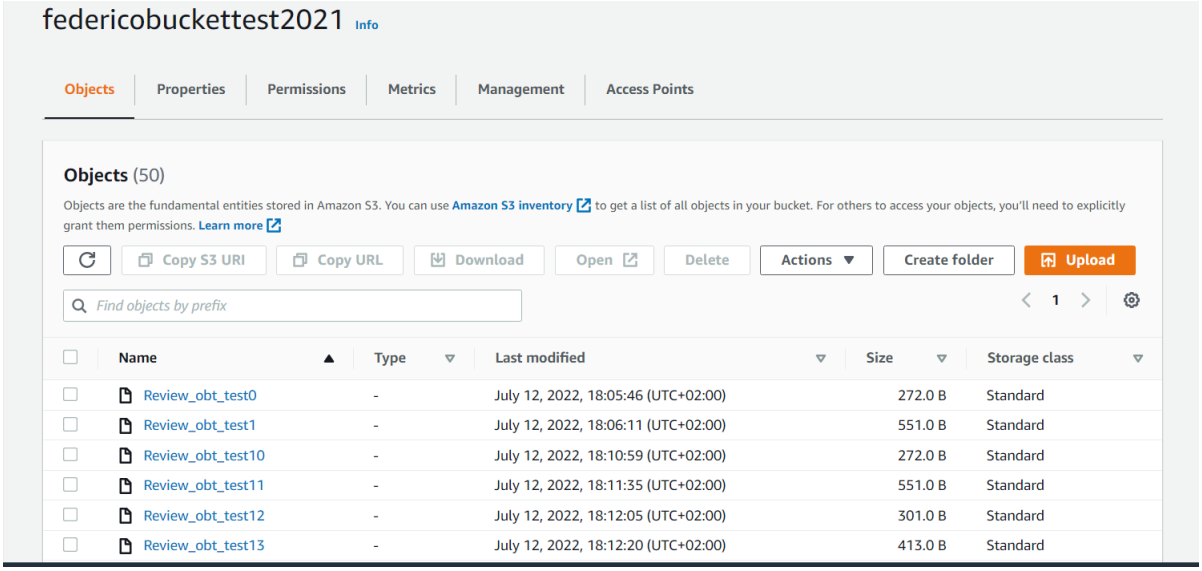

Figure 3: S3 contente i dati originali

```
sentimen=udf(get_sentiment,DoubleType())
splitt=udf(splitting,ArrayType(StringType()))
df=df.withColumn("sentiment",sentimen("title"))
df=df.withColumn("words",splitt("title"))


df.writeStream\
  .format("json")\
  .option("checkpointLocation", "./checkpoints") \
  .option("path", "s3a://"+BUCKET_OUTPUT +"/")\
  .start()\
  .awaitTermination()
```

Figure 4: Spark code

2.4 Lambda

Quando Spark mette i dati processati sull'altro bucket, viene **triggerata** una lambda function, che prende i dati e li mette su **opensearch** 5 . Bisogna configurare alcune variabili d'ambiente 7.

sample.py Execution results

```python
import boto3
import re
import requests
from requests_aws4auth import AWS4Auth
import os
import json


region = 'eu-west-3' # e.g. us-west-1
service = 'es'
# credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(os.getenv("aws_access_key_id"), os.getenv("aws_secret_access_key"), region, service)
host = 'https://search-test-opensearch-progetto-sis-gwmmirrvnf6xb33q5tmkdj7yqi.eu-west-3.es.amazonaws.com' # the OpenSearch Servi
index = 'reviews_amazon'
type = '_doc'
url = host + '/' + index + '/' + type
headers = { "Content-Type": "application/json" }
s3 = boto3.client('s3',aws_access_key_id=os.getenv("aws_access_key_id"),aws_secret_access_key=os.getenv("aws_secret_access_key"))
# Regular expressions used to parse some simple log lines
# Lambda execution starts here
def handler(event, context):
    print("PROVA\n")
    for record in event['Records']:

        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']

        # Get, read, and split the file into lines
        obj = s3.get_object(Bucket=bucket, Key=key)

        body=obj['Body'].read()
        lines=body.splitlines()
        for line in lines:
            r = requests.post(url, auth=awsauth, json=json.loads(line) , headers=headers)
```

Figure 5: Editor variabili lambda

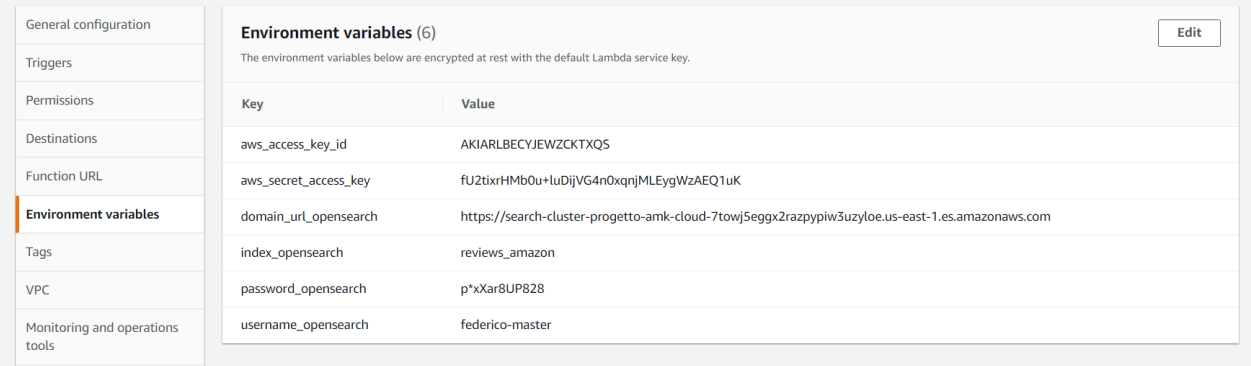

Figure 6: Editor variabili lambda

2.5 OpenSearch

L'ultimo componente è Opensearch di AWS, che permette di creare e visualizzare dashboard per fare delle analisi sui dati ad esempio delle **World Cloud**.

2.6 Grafana

Si possono visualizzare i dati anche tramite Grafana.

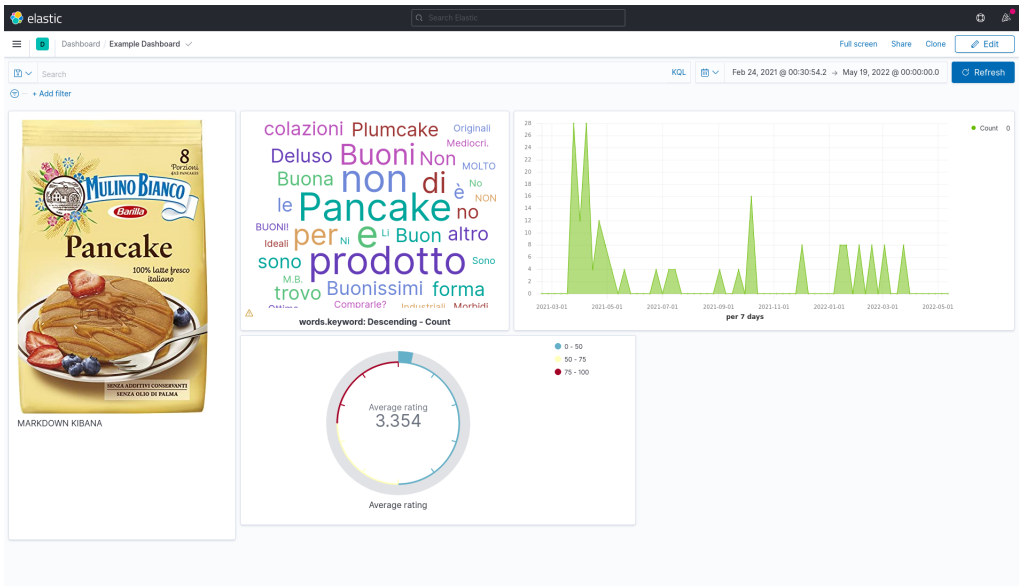

Figure 7: Dashboard