UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Corso di Laurea Magistrale
in Ingegneria Informatica

Progetto di Sistemi Distribuiti e Cloud:

Sistema distribuito per la visualizzazione di tweet geolocalizzati tramite utilizzo della dashboard interattiva OpenSearch

Giuseppe Centraco
Matricola: 227591

Anno accademico 2022/2023

Sommario

1. Introduzione

Oggi è possibile generare dati da qualsiasi dispositivo ed in qualsiasi momento, per questo motivo la quantità dei dati prodotti e raccolti è cresciuta in modo esponenziale fino ad introdurre un nuovo fenomeno da studiare, il concetto di Big Data, i quali hanno aperto un nuovo mondo di problematiche ed opportunità. Tutto ciò è agevolato anche dalla possibilità di impossessarsi di memorie di grandi dimensioni in modo economico.

I Big Data hanno un grande valore economico perchè offrono la possibilità alle aziende di estrarre informazioni, anticipare delle tendenze e dei pattern ricorrenti, prendere decisioni mirate, ottimizzare processi, individuare opportunità di business, interpretare meglio i comportamenti umani, introdurre prodotti innovativi sul mercato, ottenere vantaggi sui competitor e tanto altro.

Esistono diversi dispositivi e piattaforme che producono e raccolgono dati, tra questi i social network giocano un ruolo cruciale. In particolare, Twitter è una fonte di dati estremamente ricca e utile per l'analisi dei Big Data perché si tratta di una piattaforma in cui gli utenti condividono costantemente pensieri, opinioni, informazioni e contenuti. La piattaforma conta attualmente più di 354 milioni di utenti attivi, tra i quali 60 milioni accedono e la utilizzano regolarmente ogni giorno.

Su Twitter ogni giorno vengono condivisi miliardi di tweet, ovvero brevi messaggi testuali correlati di link o immagini che spaziano dagli argomenti di attualità alle opinioni personali, dalle tendenze popolari alle notizie di ultima ora.

Lo scopo di questo progetto è quello di realizzare e implementare un sistema distribuito per la visualizzazione su mappa di grandi moli di tweet geolocalizzati, con la possibilità per l'utente di impostare delle query per il filtraggio dei dati. I tweet (estratti tramite API Twitter), che riguardano prevalentemente argomenti di natura politica, sono forniti e il dataset di riferimento è disponibile al seguente indirizzo:

https://drive.google.com/drive/folders/1vVNFRQnpxP8EhWcp0JdPAdDRr2wZewUh

2. Analisi dei requisiti

Una delle principali fasi per la realizzazione di un sistema distribuito è l'analisi dei requisiti perché fornisce una base solida per la successiva progettazione, sviluppo e test. Consiste nel definire e stabilire in modo efficace quelle che sono le esigenze, le aspettative e i vincoli del sistema da realizzare. Una corretta analisi dei requisiti permette di garantire che un progetto software soddisfi le esigenze dei sui stakeholders e aiuta a evitare errori costosi.

Gli stakeholder del sistema in questione non sono ben definiti in quanto si tratta di un'applicazione grossomodo generica, ma possono rientrare nella categoria:

- Professionisti operanti nel marketing;
- Consulenti politici;
- Ricercatori;

- Analisti di dati;
- Responsabili della sicurezza;
- Partner commerciali.

In questo contesto, quindi, sono stati definiti gli obiettivi del sistema: requisiti funzionali e requisiti non funzionali.

2.1 Requisiti funzionali

Per quanto riguarda i requisiti funzionali, ovvero le funzionalità che il sistema deve svolgere per soddisfare le esigenze degli utenti e raggiungere gli obiettivi prefissati, dalla traccia sono stati identificati i seguenti:

- ***Visualizzazione su mappa***: i tweet geolocalizzati devono essere visualizzabili su una mappa interattiva e consultabili facilmente dall'utente finale;
- ***Filtraggio dei dati***: l'utente deve avere la possibilità di filtrare i tweet tramite un pannello di controllo per analizzare solo quelli di interesse in base a parametri personalizzati, come ad esempio hashtag, tipo di post, utente, data e altro;
- ***Visualizzazione in formato tabellare***: i tweet con tutte le informazioni di contesto devono essere disponibili agli utenti finali anche in formato tabellare per un consulto più dettagliato;
- ***Ricerca dei Tweet***: l'utente deve essere in grado di trovare e analizzare determinati tweet tramite una ricerca;
- ***Query preimpostate***: il sistema dovrebbe includere una serie di query predefinite che gli utenti possono selezionare per ottenere risultati immediati e rilevanti;
- ***Widget Dashboard***: la presenza di vari widget personalizzabili sulla dashboard permette agli utenti di visualizzare informazioni specifiche o statistiche sui tweet, tramite grafici a barre o diagrammi a torta;
- ***Esportazione dei dati***: possibilità per gli utenti di esportare i dati dei tweet in formato CSV, per ulteriori analisi o elaborazioni al di fuori del sistema;
- ***Creazione di report***: possibilità di scaricare un report (in formato pdf o PNG) dell'analisi appena effettuata e disponibile per la condivisione.

2.2 Requisiti non funzionali

A differenza dei requisiti funzionali, quelli non funzionali si concentrano sulle qualità globali del sistema. Sono fondamentali tanto quanto gli altri, perché essi contribuiscono a definire l'esperienza complessiva dell'utente.

- ***Scalabilità***: considerato che il progetto deve essere implementato come sistema distribuito, a maggior ragione bisogna tenere d'occhio questo requisito. Infatti, il sistema deve essere in grado di adattarsi a carichi di lavoro variabili;
- ***Disponibilità***: il sistema deve essere disponibile e accessibile agli utenti per la maggior parte del tempo, evitando interruzioni o momenti di inattività;
- ***Affidabilità***: si intende la capacità di un sistema di svolgere le sue funzioni in modo coerente e prevedibile nel corso del tempo, senza guasti o interruzioni. Il sistema deve essere tollerante ai guasti;

- ***Usabilità***: è un aspetto critico, influenzato molto dall'interfaccia grafica, affinché si possa garantire un'esperienza positiva agli utenti, in particolare si riferisce alla facilità con cui gli utenti possono interagire con il sistema e sfruttare le sue funzionalità in modo intuitivo ed efficiente;
- ***Compatibilità***: gli utenti devono avere la possibilità di accedere al sistema tramite i principali browser web e dispositivi più usati;
- ***Prestazioni***: i tempi di risposta del sistema devono essere brevi anche quando si tratta di gestire un grande volume di dati;
- ***Sicurezza***: bisogna prevedere un opportuno sistema di autenticazione e autorizzazioni adeguate.

3. Progettazione

Una volta individuati i requisiti, si è passati alla fase di progettazione dove, in funzione delle caratteristiche richieste, vengono definite le linee principali della struttura del sistema insieme alle scelte architetturali. Si spiega come il sistema è stato suddiviso in componenti, quali tecnologie sono state utilizzate e come le varie parti del sistema interagiscono tra loro.

3.1 AWS

Come richiesto dalla traccia, la tecnologia di riferimento per lo sviluppo del progetto è stata AWS (Amazon Web Services). Mentre il negozio web è cresciuto nel corso degli anni, la necessità di un'infrastruttura scalabile è diventata ogni giorno più urgente.

Questo ha portato alla creazione della propria infrastruttura orientata ai servizi. In particolare, AWS offre servizi di cloud computing, elaborazione e distribuzione di contenuti e molto altro, ideali per creare applicazioni sofisticate in modo flessibile, scalabile e affidabile.

Tra le caratteristiche principali di AWS vi sono:

- Servizi di elaborazione, quali:
    - Amazon Elastic Compute Cloud (EC2), il quale permette di creare, configurare e gestire istanze virtuali di server. In particolare, è possibile selezionare la dimensione delle istanze, il sistema operativo, le configurazioni di rete e le opzioni di storage in base alle proprie necessità. Utilizzato per carichi di lavoro persistenti;
    - AWS Lambda, invece, è un servizio che consente di eseguire del codice senza la necessità di gestire dei server. Non viene creata e mantenuta un'istanza di server, ma Lambda esegue il codice solo quando viene richiesto, per questo motivo si dice che sia una piattaforma guidata dagli eventi.
- Servizi di storage, come Amazon Simple Storage Service (S3) per avere uno spazio di archiviazione (tramite bucket) scalabile, sicuro e duraturo per i dati;
- Servizi di database, come Amazon Relational Database Service (RDS), utili per semplificare la configurazione, l'utilizzo e la scalabilità dei database relazionali nel cloud;
- Servizi di sicurezza, come AWS Identity and Access Management (IAM), per gestire in modo centralizzato le identità e le autorizzazioni per accedere alle risorse dei servizi AWS;

- Servizi di rete, come Amazon Virtual Private Cloud (VPC) che fornisce un'infrastruttura di rete sicura;
- Servizi di Content Delivery Network (CDN), come Amazon CloudFront che offre una consegna rapida e sicura dei contenuti, tra cui pagine web, immagini, video e altri file statici;
- Servizi di messaggistica completamente gestiti, come Amazon Simple Queue Service (SQS) che permette di inviare, archiviare e ricevere messaggi tra diverse parti di un'applicazione.

In particolare, i due servizi utili ai fini dello sviluppo del progetto sono stati: Amazon OpenSearch Service e Amazon OpenSearch Dashboard.

3.1 Amazon OpenSearch Service

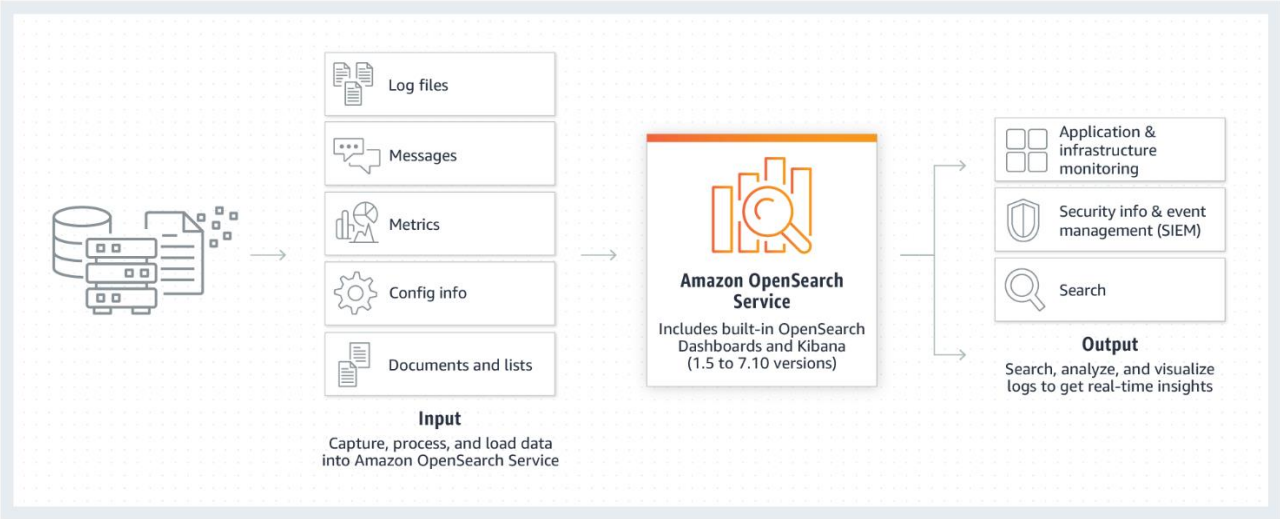

Figura 1 - Amazon OpenSearch Service (https://aws.amazon.com/it/opensearch-service/)

Amazon OpenSearch Service è una suite di ricerca e analisi dei dati distribuita derivata da ElasticSearch. Si tratta di un progetto open source supportato da diverse organizzazioni tra cui AWS. Ad oggi ci sono decine di migliaia di clienti attivi con centinaia di migliaia di cluster in gestione che elaborano centinaia di trilioni di richieste al mese.

Il servizio OpenSearch di Amazon, offre le versioni più recenti di OpenSearch, supportando 19 versioni di ElasticSearch e funzioni di visualizzazione basate su OpenSearch Dashboard e Kibana. Tra le principali caratteristiche mantenute da ElasticSearch ci sono ad esempio la potente ricerca full-text (viene considerata non solo la corrispondenza esatta, ma anche il significato e il contesto delle parole), l'indicizzazione veloce dei dati, l'aggregazione dei dati, le query complesse, la scalabilità e la ridondanza dei dati.

Inoltre, Amazon OpenSearch Service può essere utilizzato in diversi contesti, come l'indicizzazione e la ricerca di grandi quantità di dati strutturati e non strutturati, la creazione di motori di ricerca per siti web e applicazioni, l'analisi dei log di sistema, l'analisi dei dati di monitoraggio e molto altro ancora.

I costi del servizio vengono addebitati solo per le risorse utilizzate, senza alcun requisito di utilizzo minimo. In particolare, vi sono tre fattori che determinano il costo:

- Ore istanza, ovvero il numero totale di ore in cui un'istanza è disponibile;
- Quantità di archiviazione;
- Dati trasferiti all'interno o all'esterno del servizio OpenSearch.

La scelta di utilizzare questo servizio gestito è stata dettata dal fatto che è possibile creare e configurare un'istanza OpenSearch senza doversi preoccupare della gestione, del monitoraggio e della manutenzione dell'infrastruttura. Creare un dominio OpenSearch vuol dire configurare il tipo e il numero di nodi da utilizzare per i proprio cluster. Ogni dominio è raggiungibile tramite un end-point.

Per il progetto in questione, è stato sfruttato il piano gratuito di 12 mesi con il quale AWS permette di utilizzare il servizio OpenSearch con:

- 750 ore al mese di un'istanza single-AZ t3.small.search, offre risorse di calcolo e memoria adeguate per gestire carichi di lavoro di ricerca su piccola scala. Ha una capacità di calcolo di livello base con una singola unità di CPU virtuale (vCPU) e 2 GB di memoria. È importante notare che le specifiche di un'istanza di questo tipo possono variare a seconda delle configurazioni e delle esigenze specifiche del progetto;
- 10GB al mese di archiviazione EBS (Elastic Block Store) facoltativa, fornisce volumi di archiviazione altamente affidabili e scalabili. I volumi EBS possono essere usati per una vasta gamma di applicazioni, come database, applicazioni aziendali, repository di file, server web e altro ancora.

3.1.1 Configurazione: Creazione account

Figura 2 - Modulo registrazione AWS root

La creazione dell'account root, con le autorizzazioni per tutte le risorse e i servizi all'interno di AWS, è particolarmente intuitiva e semplice, occorre:

- Inserire le informazioni personali come nome, indirizzo email e numero di telefono;
- Selezionare il tipo di account che si desidera;
- Fornire le informazioni di pagamento;
- Iniziare ad utilizzare i servizi.

3.1.2 Configurazione: Creazione account IAM

A differenza di un account root, un account IAM (Identify and Access Management) è un'entità separata (può essere un utente, un servizio o una risorsa), creata all'interno dell'ambiente AWS per consentire la gestione delle identità, degli accessi ai servizi AWS e per consentire temporaneamente autorizzazioni specifiche.

Un account IAM offre inoltre, funzionalità di autenticazione e controllo degli accessi avanzati, consentendo di implementare politiche di sicurezza personalizzate come l'autenticazione a più fattori. Inoltre ogni utente IAM deve essere associato ad un gruppo con delle policy particolari in base alle necessità del caso.

A tal riguardo, per il progetto è stato creato il seguente utente IAM per bloccare l'accesso alle informazioni di fatturazione e negare altre autorizzazioni rischiose.

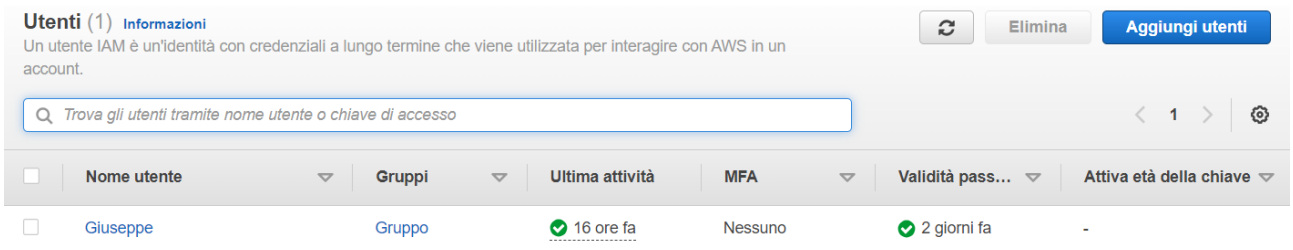

Figura 3 - Utente IAM Giuseppe

Associato al seguente gruppo, con la policy "AmazonOpenSearchServiceFullAccess". Questa policy fornisce accesso completo a tutti i servizi e le risorse di OpenSearch Service. È una policy permissiva che consente al gruppo di creare, gestire, modificare ed eliminare completamente i domini di OpenSearch.

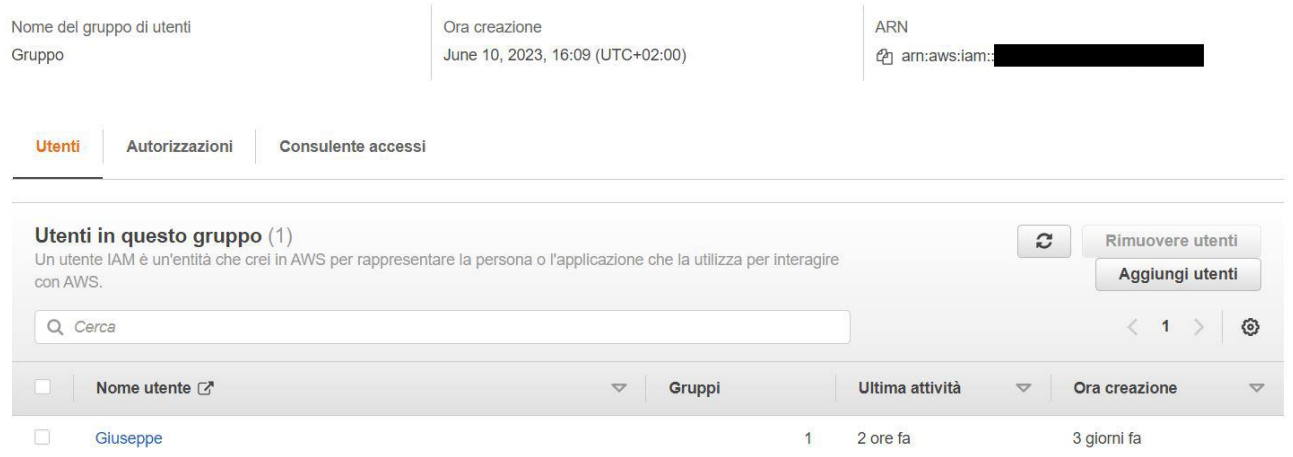

Figura 4 - Utente IAM Giuseppe

3.1.3 Configurazione: Creazione dominio Amazon OpenSearch Service

Un dominio OpenSearch Service è costituito da un insieme di nodi che collaborano per creare un cluster di ricerca. I nodi possono essere distribuiti su più macchine virtuali o istanze di server fisiche

all'interno dell'infrastruttura di AWS. Ogni dominio è raggiungibile tramite un indirizzo che prende il nome di end-point.

Essendo Amazon OpenSearch Service un servizio gestito, AWS si occupa dell'installazione, del provisioning delle risorse e della manutenzione del cluster di nodi. Questo consente di concentrarsi su altri aspetti altrettanto fondamentali del progetto.

Per il progetto è stata definita la seguente configurazione di dominio:

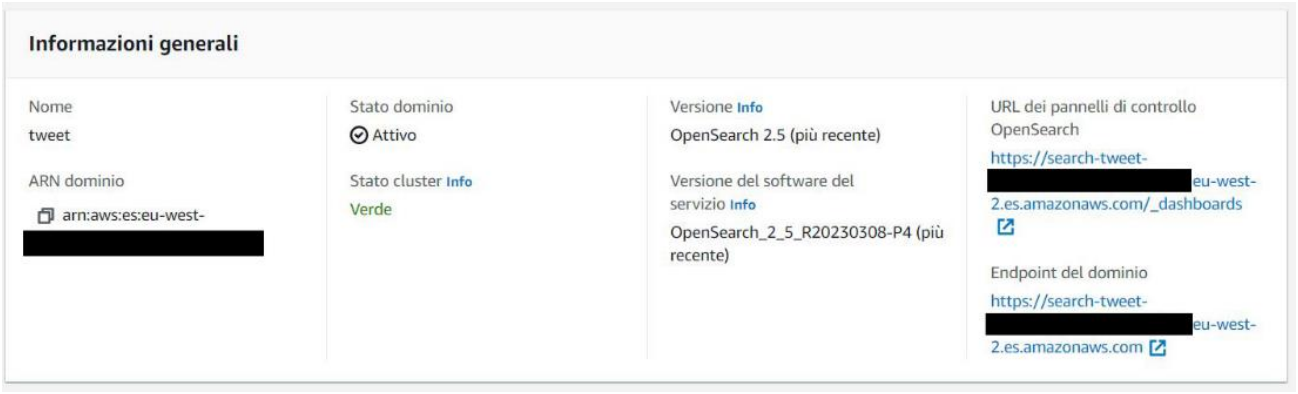

Figura 5 - Configurazione dominio

- Regione AWS: "***Europa (Londra) eu-west-2***", si tratta di una località geografica fisica in cui AWS ha costruito e posizionato i propri data center. A scopo didattico, non si è a conoscenza di della nazionalità degli stakehoder, per tanto è stata scelta quest'area solo per prossimità all'Italia consentendo di ridurre al minimo la latenza e migliorare le prestazioni;
- Nome del dominio: "***tweet***";
- Metodo di creazione: "***Creazione personalizzata***", la quale consente di configurare manualmente le diverse opzioni incluse quelle per disponibilità, sicurezza, backup e manutenzione. Questo approccio permette di avere maggiore flessibilità e controllo sulle caratteristiche e sul comportamento del dominio;
- Modelli: "***Produzione***", per mantenere le impostazioni predefinite e le best practice per mantenere una disponibilità elevata e prestazioni costanti in produzione;
- Deployment options:
    - "***Dominio senza stand by***", si tratta di un'architettura in cui non sono presenti nodi standby nel dominio, quindi tutti i nodi del dominio sono nodi attivi e partecipano attivamente al processo di indicizzazione e di ricerca dei dati. L'assenza di nodi standby, però, comporta che non ci sia un backup immediatamente disponibile nel caso in cui ci fosse un fallimento e il tempo di ripristino potrebbe lievitare (il failover non è automatico). Tuttavia, anche se questa scelta non è stata del tutto vantaggiosa, è stata l'unica soluzione percorribile per avere la possibilità di utilizzare una della istanze gratuite messe a disposizione dal piano di prova;
    - Zone di disponibilità: "***2-AZ***", ovvero la zona di disponibilità in cui le risorse sono distribuite. In questo caso i nodi vengono distribuite in due zone di disponibilità all'interno della regione AWS scelta. La distribuzione dei nodi in più zone di disponibilità consente di proteggere l'applicazione da eventuali guasti o interruzioni.

- Opzioni del motore: "***2.5***", si tratta della versione più recente e di conseguenza quella più aggiornata;
- Nodi di dati:
    - Tipo di istanza: "***t3.small.search***", si tratta di un tipo di istanza adatta per scopi di test e sviluppo, quindi congeniale allo sviluppo del progetto. Viene offerta gratuitamente fino a 750 ore al mese di utilizzo e fino a 20 GiB di archiviazione EBS per scopo generico;
    - Numero di nodi: "***4***", ovviamente lavorando in 2 zone di disponibilità è stato necessario scegliere un numero di nodi multiplo di 2, distribuendo, quindi, 2 nodi in una zona e 2 in un'altra;

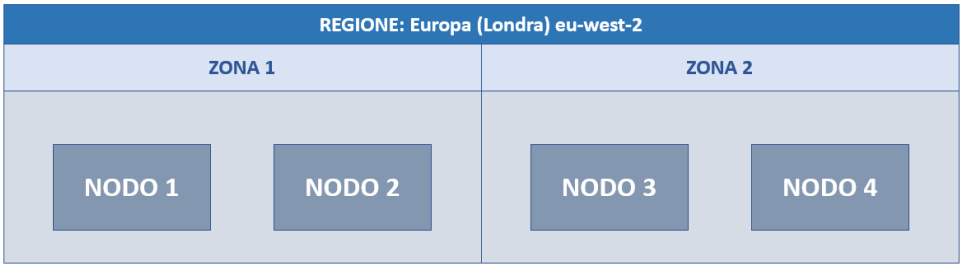

Figura 6 - Distribuzione dei nodi in zone

    - Tipo di storage: "**EBS**", unica alternativa di storage per OpenSearch Service. I volumi EBS consentono di dimensionare le risorse di archiviazione del dominio in modo indipendente dalle risorse di calcolo;
    - Tipo di volume EBS: "***Uso generico (SSD) – gp3***", si tratta di una classe di storage progettata per offrire prestazioni elevate e bilanciare il costo;
    - Dimensione di storage EBS per nodo: "***10 GiB***", considerando la presenza di 4 nodi si raggiungono 40GiB che sono abbastanza per lo scopo didattico del progetto.
- Nodi di master dedicati, si tratta di nodi specifici che vengono utilizzati per gestire le attività di coordinamento e amministrazione del cluster. Sono nodi che non partecipano all'elaborazione delle richieste di ricerca o all'indicizzazione dei dati, ma sono responsabili di attività come la gestione della topologia del cluster, la sincronizzazione dei metadati, la gestione degli indici e la distribuzione delle operazioni di scrittura. Essi coordinano anche il processo di elezione del master nel caso in cui il nodo master primario dovesse fallire. La scelte attuate sono le seguenti:
    - Tipo di istanza: "***t3.small.search***";
    - Numero di nodi master: "***3***". in un'altra;

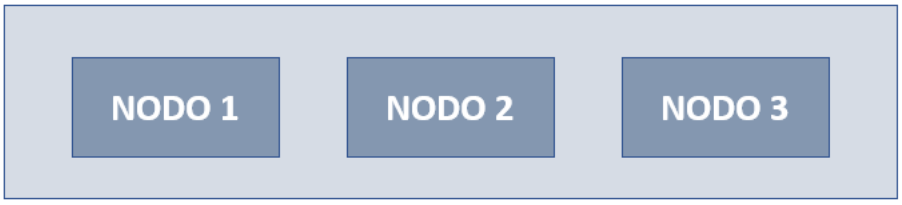

Figura 7 - Nodi master nel cluster

- Rete:
    - "***Accesso pubblico***";
- Controllo granulare degli accessi, è stato "***abilitato***" per definire e gestire in modo dettagliato i privilegi e le autorizzazioni degli utenti all'interno del sistema. L'obiettivo del controllo granulare degli accessi è quello di garantire la sicurezza e la protezione dei dati sensibili, limitando l'accesso solo alle persone autorizzate e prevenendo accessi non autorizzati o attività dannose. Inoltre è stato "***creato un utente master***" con un proprio nome utente e password;
- Policy d'accesso: "**Usa solo il controllo granulare degli accessi**". In questo modo viene consentito l'accesso libero al dominio, ovviamente per accedervi sarà necessario fare uso delle informazioni definite in precedenza nella sezione controllo granulare degli accessi;
- Crittografia: utilizzando il controllo granulare degli accessi, che richiede l'abilitazione di HTTPS, si ha una crittografia da nodo a nodo e una crittografia dei dati inattivi. Per crittografare e decifrare le risorse si utilizza la "***chiave di proprietà di AWS"***.

3.2 Amazon OpenSearch Dashboard

Il servizio OpenSearch di Amazon è anche fornito in bundle con uno strumento di visualizzazione del pannello di controllo, OpenSearch Dashboard.

OpenSearch Dashboard è un'applicazione web basata su Kibana che fornisce un'interfaccia intuitiva e potente per esplorare, analizzare e visualizzare i dati indicizzati in un cluster di OpenSearch. È possibile creare widget grafici, come grafici a barre, grafici a torta, grafici a dispersione e altro ancora, per rappresentare i dati in modo visivo e comprensibile. È possibile configurare query di ricerca per recuperare dati specifici dal cluster e applicare filtri per raffinare i risultati.

L'interfaccia di OpenSearch Dashboard permette di trascinare e rilasciare i widget sulla dashboard, configurare le opzioni di visualizzazione, personalizzare i colori e le etichette e salvare le configurazioni per un rapido accesso futuro. È anche possibile creare dashboard interattive con filtri dinamici, tabelle di dati e altre funzionalità per esplorare i dati in modo approfondito.

Le varie dashboard realizzate si possono condividere con altri utenti. Inoltre, una soluzione ai fini reportistici è quella di esportare le dashboard in diversi formati, come immagini o file PDF.

4. Implementazione

Questa sezione riguarda l'effettiva realizzazione del sistema. Dopo avere introdotto gli obiettivi del progetto e il tipo di tecnologie utilizzato, di seguito sono riportate le varie fasi affrontate per portare a termine il progetto.

4.1 Pre-processing

In seguito ad un'analisi eseguita sui file json costituenti il dataset si è notata fin da subito una criticità, non banale considerando che l'obiettivo principale del progetto è la visualizzazione di punti geolocalizzati, ovvero la presenza di valori null in corrispondenza dell'attributo "coordinates" nella maggior parte dei tweet forniti.

A scopo didattico, l'ostacolo è stato raggirato tramite una valorizzazione dei valori. Infatti, tramite un opportuno script Python per ogni tweet è stato possibile generare in maniera casuale delle coordinate scegliendo a caso dei punti ricadenti nel territorio degli USA (formati da latitudine e longitudine) da assegnare all'attributo "coordinates" ove mancante.

Lo script Python per valorizzare le coordinate (facente uso delle librerie random, json, Polygon e Point) è costituito dai seguenti metodi:

- ***updateCoordinates(file, file_path, destination_path)***:
  - riceve un file da valorizzare;
  - memorizza ogni riga del file in una lista di stringhe;
  - per ogni stringa nella lista:
    - converte la stringa da json a dizionario;
    - modifica la chiave "coordinates" con il metodo get;
    - ricostruisce l'oggetto json e lo scrive sul file di destinazione.

```
def updateCoordinates(file, file_path, destination_path):

    source_file = open(file_path, 'r')
    lines = source_file.readlines()
    destination_file = open(destination_path + "/[formattato]" + file, 'w')

    for i in range(3):
        j=json.loads(lines[i][:-1])
        j["coordinates"] = get()
        tmp=json.dumps(j)
        destination_file.write(tmp)
        destination_file.write(",\n")
```

Figura 8 - Metodo updateCoordinates

- ***get()***, inizializza un poligono che racchiude tutte le coordinate ricadenti nel territorio degli Stati Uniti;

```
def get():

    poly = Polygon([(48.15166836057548, -124.58903083449438), (48.65216753389164, -95.25936768004513),
                    (45.23353305229313, -89.50253174254513), (41.39867609503304, -83.30624268004513),
                    (44.7362220690925, -70.21053955504513), (30.880317812873436, -81.64807662791249),
                    (26.311212460510912, -80.12504589057635 ), (25.163254748225288, -80.56449901557635 ),
                    (30.782162613334748, -83.68605587893336 ), (30.360896401726905, -104.12206586729037),
                    (32.4609814623837, -108.91210492979037  ), (33.12592490726583, -116.99804242979037 ),
                    (39.357545183181486, -123.67772992979037)])

    p = randomPointGenerator(poly)
    return pointToDictionary(p)
```

Figura 9 - Metodo get

- ***randomPointGenerator(poly)***, genera in modo random in un punto interno alle coordinate del poligono ricevuto come parametro;

```python
def randomPointGenerator (poly):

    min_x, min_y, max_x, max_y = poly.bounds

    flag = True
    point = Point([0,0])

    while flag:
        point = Point([random.uniform(min_x, max_x), random.uniform(min_y, max_y)])
        if point.within(poly):
            flag = False

    return point
```

Figura 10 - Metodo randomPointGenerator

- ***pointToDictionary(point)***, metodo di utilità che viene utilizzato per trasformare un punto in un dizionario Python;

```python
def pointToDictionary(point):

    geo_point = {
        "lat": point.x,
        "lon": point.y
    }

    return geo_point
```

Figura 11 - Metodo pointToDictionary

- ***main()***, richiede da input il path della cartella contenente i file da valorizzare e quello della cartella in cui memorizzare i json rielaborati. Successivamente per ogni file presente nella cartella di input, richiama il metodo updateCoordinates che si occupa di tutto il resto.

```python
def main():

    folder_source = input("Inserisci il path della cartella con i json dei quali modificare le coordinate: ")
    folder_destination = input("Inserisci il path della cartella in cui memorizzare i file modificati: ")

    file_list = os.listdir(folder_source)

    for file in file_list:
        file_path = folder_source + "/" + file
        updateCoordinates(file, file_path, folder_destination)
        print("Coordinate " + file + " modificate")
```

Figura 12 - Metodo main

4.2 Creazione indice

Per la corretta mappatura dei dati, Amazon OpenSearch Service deve avere a disposizione un indice in cui viene descritto ogni attributo dei file json associato al relativo tipo di dato. Anche se vi è la presenza del mapping dinamico, alcuni tipi di attributi come le coordinate geospaziali e la data di creazione dei tweet non vengono mappati correttamente. Per tale motivo è stato realizzato un indice ad hoc consultabile all'interno del metodo ***createIndex()***, il quale:

- Definisce un indice con i soli campi "coordinates" e "created_at";

- Tramite una richiesta PUT invia l'indice all'endpoint del dominio creato in precedenza.

```
def createIndex(endpoint_url, index_name, username, password):

  index_url = endpoint_url + index_name

  data = {
    "settings": {
      "index.mapping.total_fields.limit":3000
    },
    "mappings": {
      "properties": {
        "coordinates": {
          "type": "geo_point",
          "ignore_malformed": True
        },
        "created_at" : {
          "type" : "date",
          "format" : "EEE MMM dd HH:mm:ss Z YYYY"
        }
      }
    }
  }

  auth = (username, password)

  response = requests.put(index_url, json=data, auth=auth)
  print("Risultato creazione indice " + response.text)
```

Figura 13 - Metodo createIndex

4.3 Caricamento tweet

Una volta realizzato e caricato con successo l'indice, si è passato al caricamento dei tweet in precedenza valorizzati. Per questioni legate ai vincoli sulla dimensione dello storage gratuito non sono stati caricati tutti i tweet. I tweet sono stati inviati all'endpoint tramite delle richieste POST.

```
def uploadJSON(endpoint_url, index_name, file, username, password):

  file_name = "_doc"
  url = endpoint_url + index_name + "/" + file_name

  headers = {
    "Content-Type": "application/json"
  }

  with open(file, "r") as file:
    data = file.readlines()

  auth = (username, password)
  successo = 0
  fallimento = 0

  for i in range(102,499):
   response = requests.post(url, headers=headers, data=data[i].rstrip(",\n"), auth=auth)
```

Figura 14 - Metodo uploadJSON

4.4 Creazione index pattern

Per la visualizzazione dei dati su OpenSearch Dashboard è stato creato un Index Pattern (chiamato "***indice****"), il quale è necessario ad OpenSearch Dashboard per capire come interpretare e organizzare gli indici dei dati all'interno di un cluster. Si tratta di uno strumento importante per organizzare, filtrare e interrogare i dati in modo efficace all'interno di un cluster. Dopo la creazione dell'index pattern è stato effettuato un check per controllare la validità della mappatura dei vari campi.

4.5 Creazione Dashboard

Si è proseguito con la creazione della dashboard, ovvero l'obiettivo principale del progetto. Una DashBoard in OpenSearch non è altro che la raccolta di più widget e visualizzazioni per visualizzare e interagire con i dati permettendo di eseguire delle ricerche, impostare filtri, esplorare i dettagli dei dati, ecc. Su OpenSearch le visualizzazioni consentono di rappresentare i dati in modo intuitivo e semplice per gli utenti, dopo la creazione e il salvataggio possono essere integrate e raggruppate insieme in una pannello per riassumere l'analisi dei dati.

Nella dashboard, denominata "***Dashboard Tweet Geolocalizzati***" sono stati inseriti le seguenti visualizzazioni:

- Coordinate Map, mappa per la visualizzazione dei tweet geolocalizzati. Per il corretto funzionamento della mappa, nelle opzioni di configurazione è stato inserito l'index pattern creato in precedenza. I punti sulla mappa sono disponibili in 4 colori distinti in base all'intervallo in cui ricade il tweet. I tweet vengono visualizzati secondo la seguente aggregazione: per ogni coordinata (quindi stessa latitudine e stessa longitudine) i tweet vengono ordinati in modo decrescente per data di creazione e viene visualizzato il più recente. Nel tooltip visibile al passaggio del mouse vengono esibiti: id del tweet, latitudine e longitudine. Sarebbe stato preferibile aggiungere ulteriori informazioni nel tooltip, come ad esempio il testo del tweet ma OpenSearch Dashboard non lo consente tra le opzioni native.

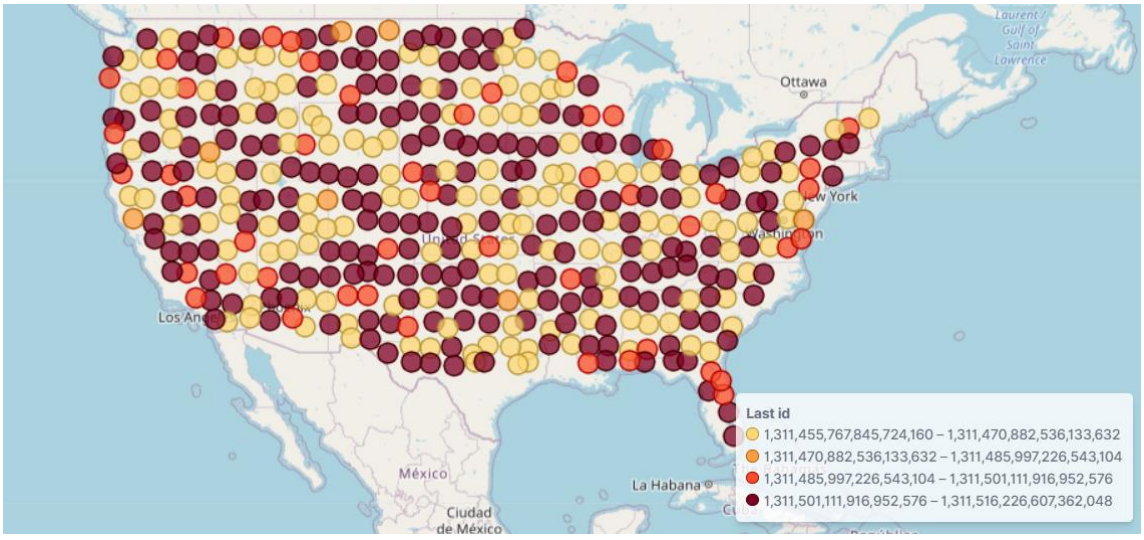

Figura 15 - Mappa tweet

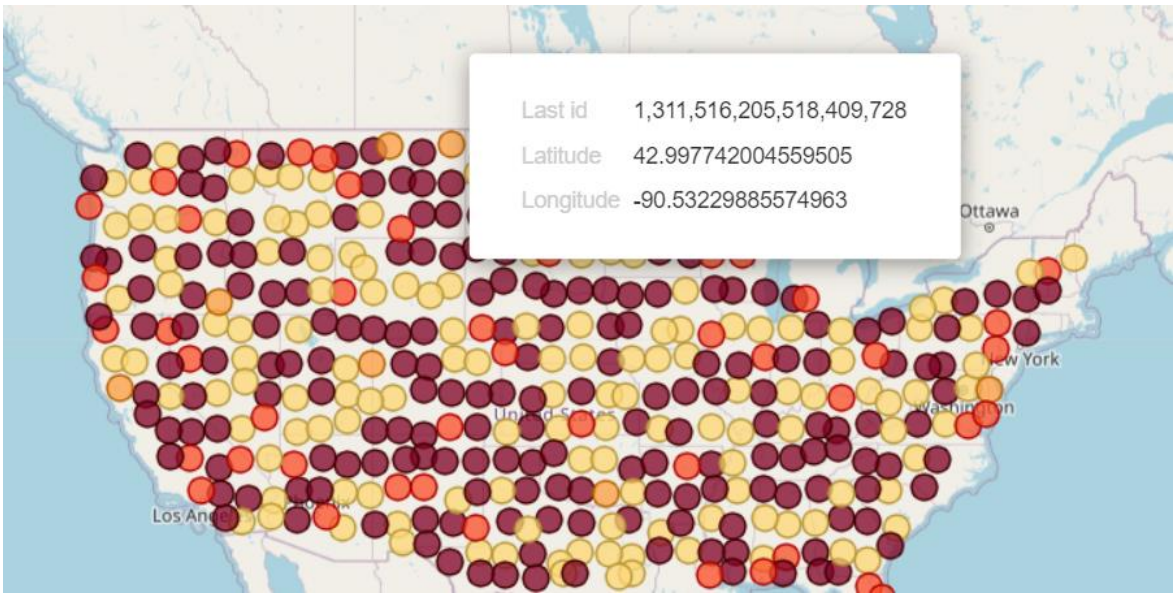

Figura 16 - Tooltip mappa tweet

- Widget markdown, per descrivere il funzionamento della dashboard;

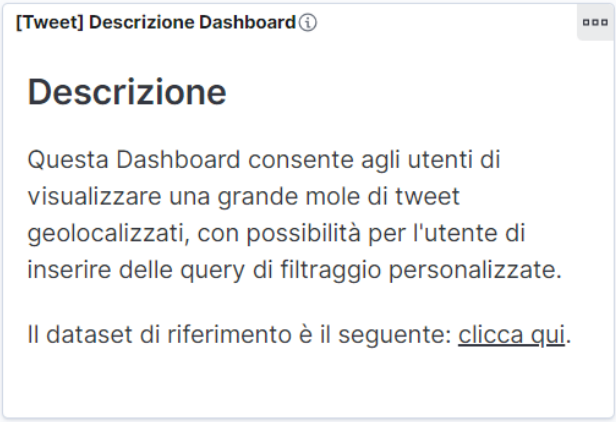

Figura 17 - Descrizione Dashboard

- Widget, per filtrare i tweet con i seguenti parametri selezionabili dall'utente:
  - Id utente;
  - Nome utente;
  - URL utente;
  - Num. follower utente;
  - Utente verificato;
  - Num. amici utente;
  - Id post;
  - Data;
  - Num. volte ricondivisione;
  - Num. like;
  - Ricerca Full Text;
  - Hashtag;
  - Post citazione;

- Post sensibile;
- Ricondiviso;
- Sorgente.

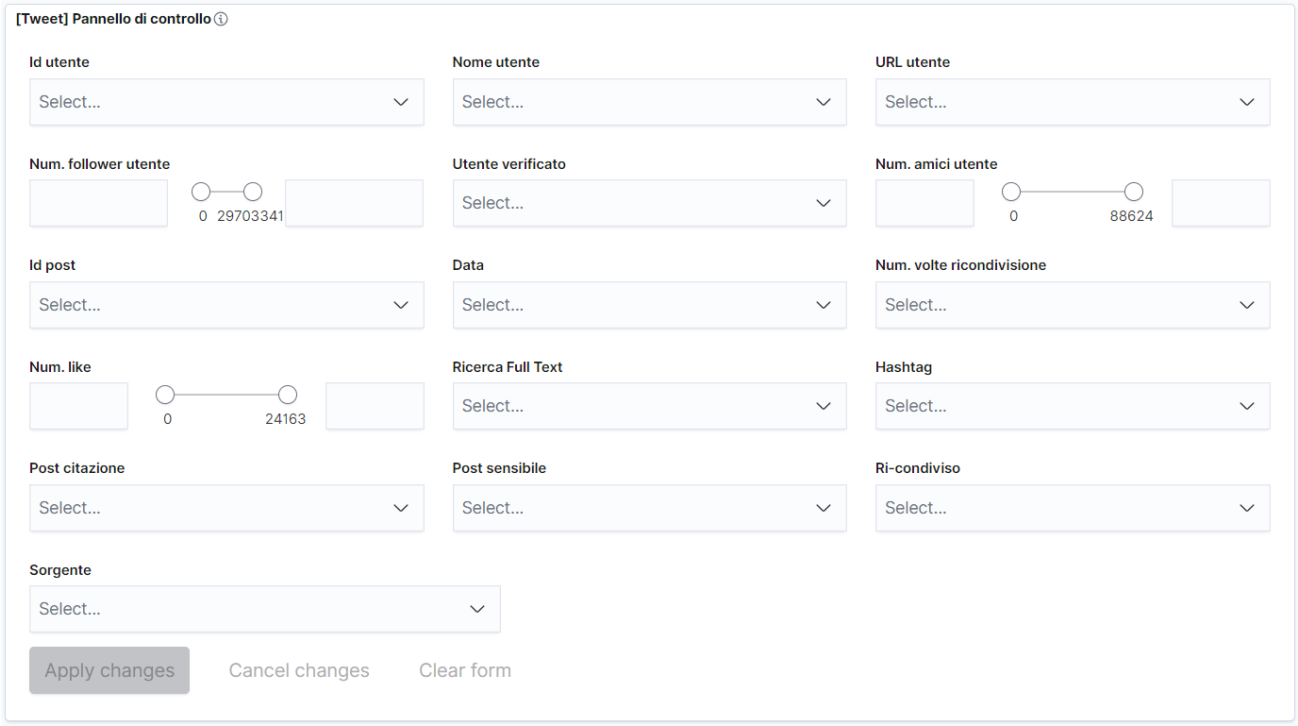

Figura 18 - Widget filtri

- Tabella, per rappresentare in formato tabellare i tweet con i seguenti attributi:
  - Time;
  - Coordinates;
  - favorite_count;
  - lang;
  - retweet_count;
  - user.id.

Figura 19 - Tabella tweet

- Metric, per visualizzare il totale di diversi tipi di tweet:
  - Tweet generici totali;
  - Tweet citazioni totali;
  - Tweet ricondivisi totali;

- Tweet sensibili totali.

Figura 20 - Contatori tweet

- Area, per rappresentare il numero di tweet per giorno. Sono presenti due visualizzazioni con arco temporale di 15 giorni, una ordinata in senso crescente e l'altra in senso decrescente (per mettere in evidenza i giorni in cui si posta di più e quelli in cui si posta di meno);

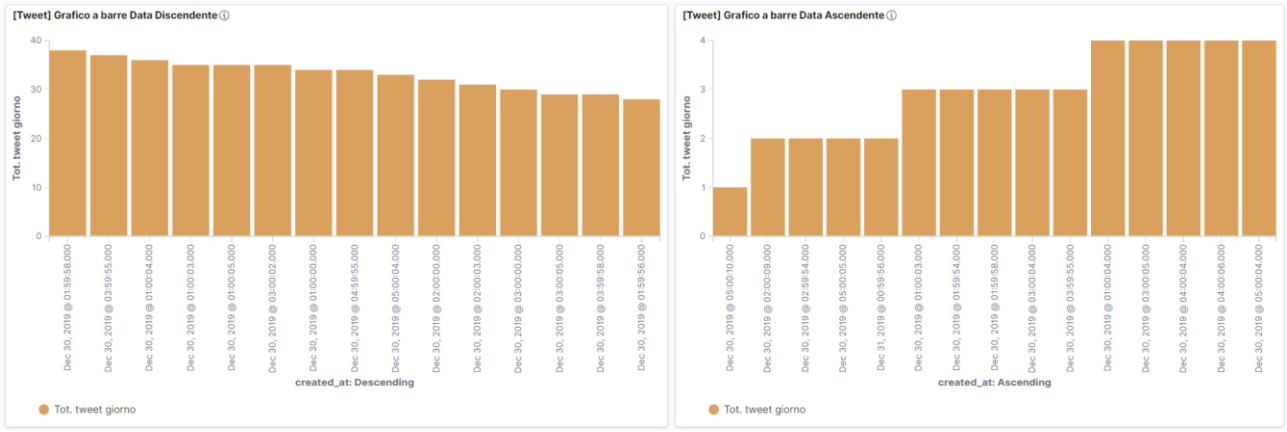

Figura 21 - Numero di tweet per giorno

- Area, per rappresentare il numero di tweet per utente;

Figura 22 - Tweet per utente

- Tag cloud, per la visualizzazione grafica dei dati testuali mettendo in evidenza le parole chiave più frequenti;

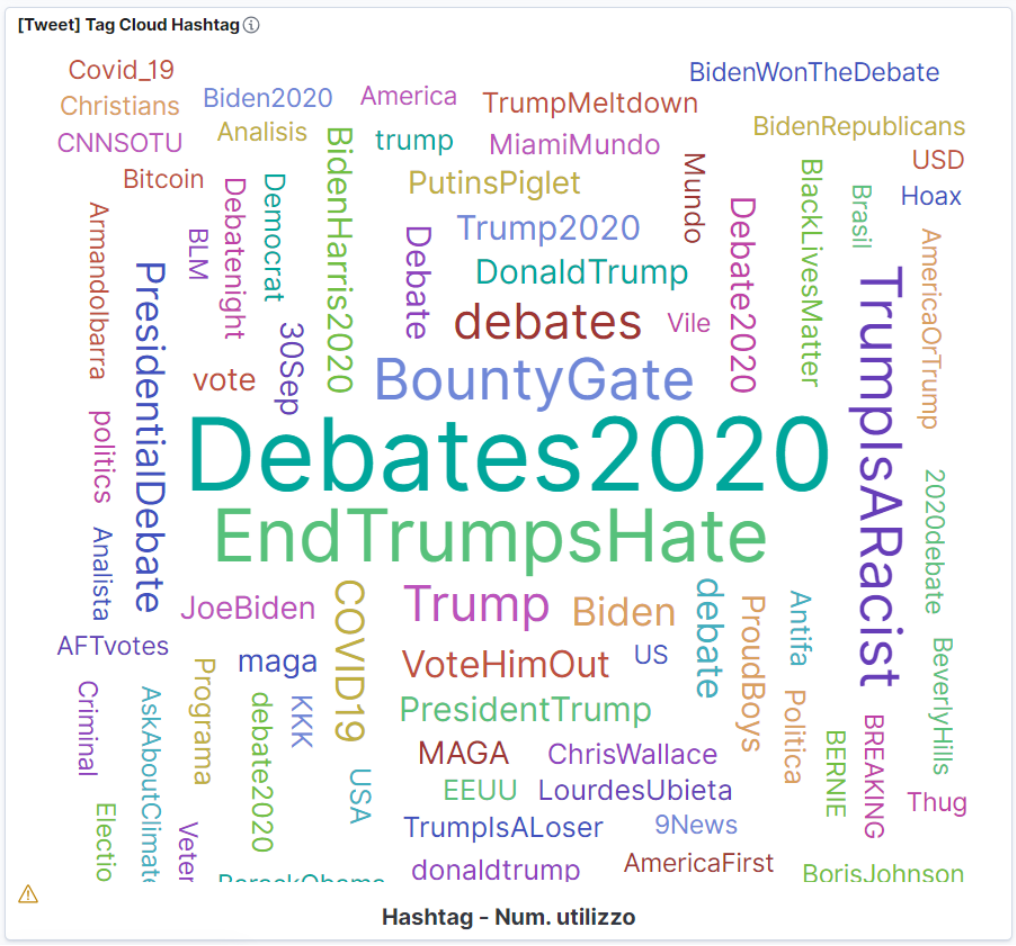

Figura 23 - Tag cloud per hashtag

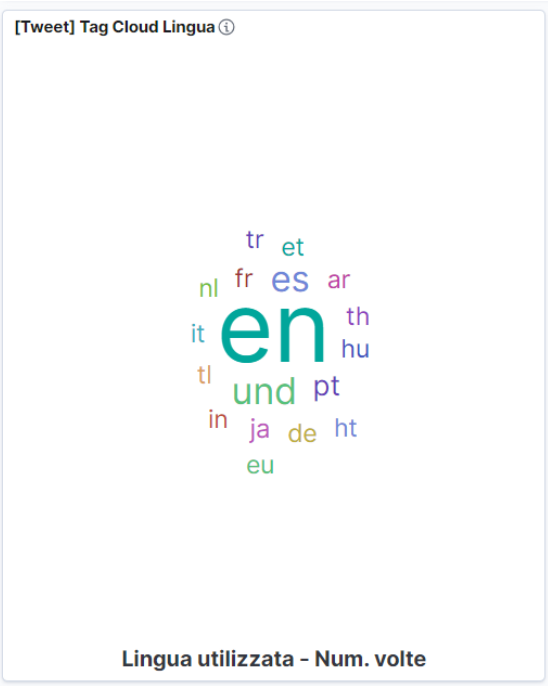

Figura 24 - Tag cloud per lingua del tweet

- Pie, per la rappresentazione dei 5 hashtag più utilizzati e delle 5 sorgenti più utilizzate in base ai tweet ricercati dall'utente;

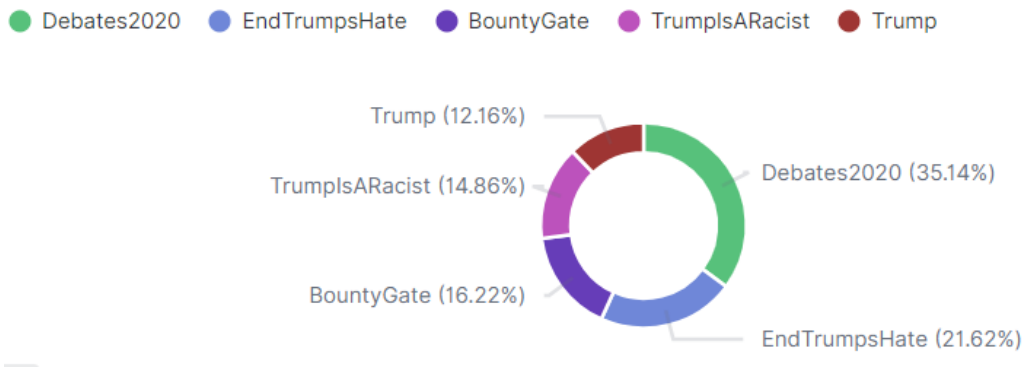

Figura 25 - Esempio dei 5 hashtag più usati

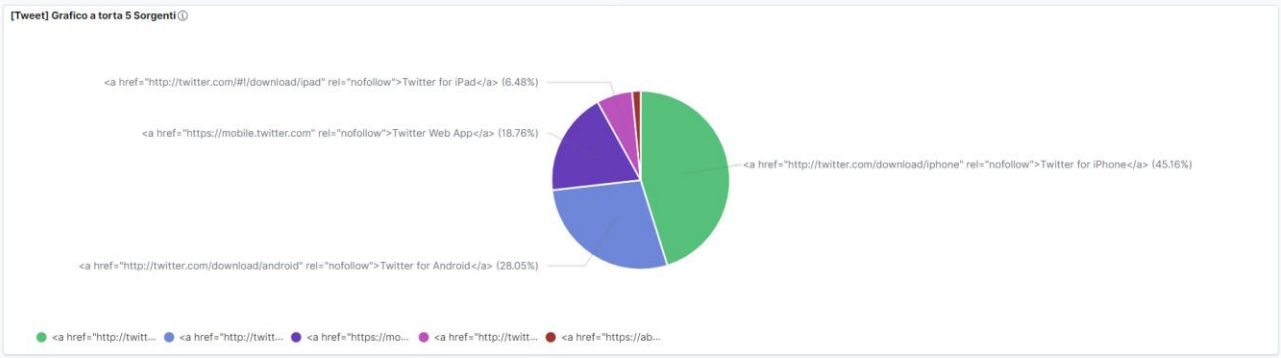

Figura 26 - Esempio delle 5 sorgenti più utilizzate

- Line, per rappresentare tramite grafico a linee il numero di like per tweet con ordine decrescente.

Figura 27 - Esempio di num. like per tweet

5. Test e risultati

Nel contesto del progetto realizzato, è fondamentale valutare l'efficacia e l'affidabilità del sistema. A tal fine è stata dedicata una fase di test approfondita per valutare le prestazioni in base agli obiettivi prefissati.

Sono state verificate che le funzionalità di filtraggio, visualizzazione e ricerca dei tweet geolocalizzati fossero implementate correttamente, inoltre è stata valutata la capacità del sistema di elaborare grandi quantità di dati in modo scalabile e senza compromettere le prestazioni complessive.

Grazie alla funzionalità di creazione report offerta dallo stesso OpenSearch, di seguito sono riportati i link di tre esempi di test/ricerche effettuati e disponibili nella repository ufficiale del progetto:

- **Link Report 1** (ricerca senza alcun filtro effettuata su tuti i tweet);
- **Link Report 2** (ricerca filtrando per tweet citazione, utente verificato, numero di like, ri-condivisione);
- **Link Report 3** (ricerca filtrando per hashtag);
- **Link Report 4** (ricerca filtrando per utente verificato e amici dell'utente);
- **Link Report 5** (ricerca filtrando per utente).

In alcuni dei report precedenti ci sono delle visualizzazioni che riportano la dicitura "No results found", in quanto i filtri selezionati potrebbero essere troppo restrittivi.

Inoltre, sempre a scopo di test sono state effettuate delle semplici query DSL (linguaggio di interrogazione utilizzato in OpenSearch), della seguente forma:

```
{
  "query": {
    "match": {
      "campo": "termine"
    }
  }
}
```

Figura 28 - Struttura query DSL utilizzate

Complessivamente, i test hanno confermato l'affidabilità del sistema e la sua capacità di soddisfare gli obiettivi prefissati.

6. Considerazioni finali

Le varie fasi descritte hanno portato alla realizzazione di un sistema distribuito per la visualizzazione su mappa di grandi moli di tweet geolocalizzati. Partendo dall'analisi dei requisiti sono stati definiti gli obiettivi del sistema, sia le funzioni da implementare e sia i vincoli da rispettare.

Successivamente è stato affrontato uno studio approfondito della tecnologia AWS e dei servizi offerti, in quanto non era mai stata utilizzata in passato. Solo in questo modo è stata chiara la linea

progettuale e implementativa da seguire, ovvero sfruttare Amazon OpenSearch Dashboard insieme al rispettivo Amazon OpenSearch Service.

Nella fase di test sono stati fatti dei check per confermare il raggiungimento degli obiettivi prefissati per il sistema, da questa è risultato che tutti i requisiti sono stati raggiunti con successo.

Per quanto riguarda le considerazioni finali, si possono illustrare alcuni aspetti che avrebbero potuto agevolare e/o rendere migliore questo progetto didattico:

- Un dataset qualitativamente migliore. Infatti, come illustrato nella fase di pre-processing l'attributo cruciale per lo sviluppo del progetto era mancante nella maggior parte dei record. Sicuramente, utilizzare delle coordinate random significa rendere non veritieri i risultati delle varie analisi. Anche la data di creazione dei tweet ristretta a pochi giorni di distanza non offre la possibilità all'utente finale di godere di alcuni grafici meglio comprensibili;
- Istanze AWS più performanti. La realizzazione del progetto e alcune delle sue caratteristiche sono state limitate dal tipo di istanza utilizzata (perché gratuita e con delle restrizioni). Nonostante in questo modo siano stati raggiunti tutti gli obiettivi del progetto, con l'utilizzo di altri tipi di istanze sarebbe stato possibile sfruttare alcune delle seguenti caratteristiche, ovvero:
    - Ulteriori opzioni riguardanti la sicurezza;
    - Archiviazione dati a caldo e a freddo, si riferisce alla tipologia di indice creato. Gli indici a caldo mantengono i dati più recenti e più spesso interrogati, i quali vengono mantenuti in una fase di archiviazione primaria per un consulto più immediato. In una fase di archiviazione a lungo termine, invece, vengono memorizzati i dati degli indici a freddo. Questo tipo di configurazione si può impostare in fase di creazione dell'indice tenendo in considerazione il fattore temporale;
    - Regolazione automatica, è l'opzione che permette al sistema di adattarsi in modo dinamico alle richieste di carico di lavoro e per ottimizzarne le prestazioni. Questo permette al sistema di aumentare la capacità di calcolo oppure il numero di nodi allocati in automatico, in basa al carico di lavoro.

Per quanto riguarda, invece, gli sviluppi futuri ci sono ancora altre opportunità che potrebbero rendere ulteriormente completo il sistema, come:

- Integrazione con altri dataset provenienti da altre piattaforme social o dispositivi, offrendo la possibilità agli utenti di visualizzare e filtrare dati da diverse fonti in un'unica interfaccia. Questo consentirebbe di incrociare i dati e scoprire altre novità interessanti;
- Aggiunta di librerie di sentiment analysis, le quali consentirebbero un'analisi ancora di più approfondita sui tweet rilasciati dagli utenti scoprendo altri pattern nascosti e informazioni significative;
- Aggiunta di librerie di machine learning, le quali potrebbero essere utilizzate per la generazione di previsioni o altri tipi di suggerimenti sui dati in input;

- Analisi di dati giornalieri per un consulto real time. Infatti, tramite le API di Twitter è possibile acquisire ed estrarre tweet in tempo reale da inviare direttamente ad un cluster OpenSearch per una rappresentazione visiva tramite dashboard.