

REAL-TIME DASHBOARDING USING BIGDATA TOOLS

**Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering**

by

SANJAY R (Reg. No. 40111132)

DHEERAJ PULAKANDAM (Reg. No. 40110319)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119**

APRIL – 2024

REAL-TIME DASHBOARDING USING BIGDATA TOOLS

**Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering**

by

SANJAY R (Reg. No. 40111132)

DHEERAJ PULAKANDAM (Reg. No. 40110319)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119**

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Sanjay R (40111132), Dheeraj Pulakandam (40110319)** who carried out the project entitled **"REAL-TIME DASHBOARDING USING BIGDATA TOOLS"** under my supervision from November 2023 to April 2024.

V. Nirmalrani
10/04/2024

Internal Guide

Dr. V. NIRMALRANI, M.TECH., Ph.D.

[Signature]
Head of the Department
HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Dr. L. K. SHANAN, M.E., Ph.D.
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeppiaur Nagar, Kotturambakkam, Chennai - 600 119.

Submitted for Viva voce Examination held on 13/04/2024

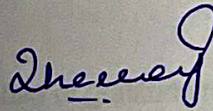
[Signature]
13/4/24
Internal Examiner

[Signature]
13/4/24
External Examiner

DECLARATION

I, **DHEERAJ PULAKAND (40110319)** hereby declare that the Project Report entitled "**REAL-TIME DASHBOARDING USING BIGDATA TOOLS**" done by me under the guidance of **Dr. V. NIRMALRANI M.TECH., Ph.D.** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 10/4/2024



PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean, School of Computing, and Dr. L. Lakshmanan M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. V. Nirmalrani, M.TECH., Ph.D.**, for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

The movie analytics project aims to leverage big data tools such as Kafka, Apache Spark, and Apache Airflow to process and analyze large volumes of movie-related data. The project also incorporates the use of DVC (Data Version Control) for efficient data management and versioning, while deploying the solution on the AWS (Amazon Web Services) platform. Furthermore, the project involves creating a real-time dashboard that visualizes the analyzed data, providing valuable insights into various movie-related metrics. The project workflow begins with data ingestion using Kafka, which allows for real-time streaming of movie-related data from diverse sources such as social media, review websites, and ticketing platforms. Apache Spark, a powerful distributed computing framework, is utilized for scalable data processing and analysis. Leveraging Spark's capabilities, the project performs data transformations, feature engineering, and statistical analysis to derive meaningful insights from the collected data. Apache Airflow is employed for task scheduling, automation, and workflow management. It orchestrates the different stages of the data pipeline, ensuring the smooth execution of data processing tasks, from data ingestion to transformation and analysis. DVC is integrated into the project for efficient data versioning, reproducibility, and collaboration. It allows for tracking changes in the data and code, enabling easy rollback and sharing of project artifacts among team members. The project leverages DVC to ensure consistency and traceability in the data processing pipeline. The project is deployed on the AWS cloud platform, utilizing its scalable infrastructure and services such as EC2 (Elastic Compute Cloud) instances, S3 (Simple Storage Service), and EMR (Elastic MapReduce). This allows for efficient resource utilization, elasticity, and high availability of the movie analytics solution. The dashboard presents key insights and visualizations derived from the analyzed movie data, providing stakeholders with an intuitive interface to explore and monitor various movie-related metrics in real-time. In summary, the movie analytics project combines big data tools like Kafka, Apache Spark, and Apache Airflow, along with DVC for data versioning, and deploys the solution on AWS.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF ABBREVIATIONS	viii
	LIST OF FIGURES	ix
1	INTRODUCTION	1
	1.1 Three V's of Big Data	1
	1.2 Insights from Big Data Analytics	2
2	LITERATURE SURVEY	3
	2.1 Inferences from Literature Survey	3
	2.2 Open problems in Existing System	5
	2.3 Challenges of Big Data Analytics and Real Time Systems	6
3	AIM AND SCOPE OF PRESENT INVESTIGATION	8
4	DESCRIPTION OF PROPOSED SYSTEM	11
	4.1 Selected Methodology or process model	12
	4.2 Architecture of the Proposed System	13
	4.2.1 High-Level System Architecture	13
	4.2.2 Data Flow	15
	4.2.3 User Interface and Real-Time Dashboard	15
	4.2.4 CI/CD Workflow of Instances GitHub Actions	15
	4.2.5 Software And Services Required	16
	4.3 Data Set Meta Data Overview	19
5	RESULTS AND DISCUSSION	21
	5.1 Output	21
	5.2 Resource Rationalization	22
	5.3 Comparative Analysis	23
6	SUMMARY AND CONCLUSION	24
	REFERENCES	25

APPENDIX	27
A SOURCE CODE	27
B SCREEN SHOTS	35
C RESEARCH PAPER	39

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ACL	Access Control List
AWS	Amazon Web Service
CD	Continuous Deployment
CI	Continuous Integration
DVC	Data Version Control
IAM	Identity Access Management
IGW	Internet Gate Way
IoT	Internet of Things
JSON	Java Script Object Notation
EC2	Elastic Cloud Computing
ECR	Elastic Container Registry
EMR	Elastic Map Reduce
MLOps	Machine Learning Operations
NAT	Network Address Translation
S3	Simple Storage Service
VPC	Virtual Private Cloud

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
4.1	High-Level Architecture of the Processed Model	12
4.2	End-to-End Architectural Flow with Big Data Components	13
4.3	CI / CD Pipeline using GitHub Actions	15
4.4	Real-Time Insight Dashboard in QuickSight.	21
4.5	Instance using Public Subnet	22
B.1.1	Producer and Consumer Instances on EC2	36
B.1.2	Starting Producer Container	36
B.1.3	Starting Consumer Container	37
B.1.4	Real-Time Streaming-1	37
B.1.5	Real-Time Streaming-2	38
B.1.6	QuickSight Dashboard (Direct Query)	38

CHAPTER – 1

INTRODUCTION

The term Big Data as coined today, is used to describe the exponential growth and availability of structured and unstructured data. However, Big Data is not just about lots of data, it is a concept providing an opportunity to find new insight into existing data as well guidelines to capture and analyze future data. The movie analytics project is a real-time dashboarding system that uses big data technologies to provide stakeholders in the movie industry with actionable insights into key metrics. The project leverages Kafka for real-time data streaming, Apache Spark for scalable data processing and analysis, Apache Airflow for workflow orchestration, and DVC for data management and version control. The project is deployed on AWS to take advantage of its scalability, reliability, and cost-effectiveness. The movie analytics project has the potential to revolutionize the way that the movie industry operates by providing stakeholders with the insights they need to make data- driven decisions about everything from movie production and marketing to distribution and release scheduling.

1.1 Three V's of Bigdata

Volume: Several factors contribute to increasing the volume of Data. Information is becoming a tangible resource and is not being discarded. Thus, Transaction-based data stored through the years, unstructured data streaming in from social media, sensors and machine to machine data being collected etc., contribute to the increasing volume of data which is handled by purchasing additional online storage. Other methods like implementing tiered storage systems, outsourcing data management, profiling data sources are being adopted. In the past the storage of data was the main concern, but with decreasing storage costs other issues emerge such as how to determine relevance within large data volumes and how to use analytics to create value from relevant data.

Velocity: Data is streamed in at unprecedented speed and must be dealt with in a timely manner. Reacting quickly enough to deal with data velocity is a challenge for

most organizations. News channels, radios and social media have changed how fast we receive the news. The data movement is now almost real time, and the update window has reduced to fractions of the seconds.

Variety: Data today comes in all types of formats. Structured numeric data in traditional databases, Information created from line-of-business applications etc. and unstructured text documents, email, video, audio, stock ticker data and financial transactions. Managing, merging, and governing different varieties of data is something many organizations still grapple with. More recently, additional Vs have been proposed for addition to the model, including variability -- the increase in the range of values typical of a large data set -- and value, which addresses the need for valuation of enterprise data.

1.2 Insights from Bigdata analytics

Here are some specific examples of how the movie analytics project can be used to derive actionable insights from movie-related data:

- **Analyze box office performance:** The project can be used to analyze box office performance data in real time to identify trends and patterns. For example, the project could be used to identify the movies that are performing well on opening day or the movies that are generating the most buzz on social media. This information could be used to inform marketing and distribution decisions.
- **Understand audience engagement:** The project can be used to analyze audience engagement data from social media, review sites, and other sources in real time. This data could be used to identify the most popular movies, the most talked-about actors and actresses, and the most popular genres. This information could be used to inform marketing and production decisions.
- **Track social media trends:** The project can be used to track social media trends related to movies in real time. This data could be used to identify emerging trends, such as new genres or actors that are gaining popularity. This information could be used to develop new movies and TV shows that are likely to appeal to audiences.

CHAPTER – 2

LITERATURE SURVEY

2.1 Inference from Literature Survey

Many efforts have been made to transform raw data into insightful real-time dashboards. Below are some real-world examples and studies like our project.

- Specifically, real-time dashboards have been crafted to observe the disruptive occurrences in the real estate sector, notably amid economic difficulties like the COVID-19 pandemic. This execution was carried out by Balamurugan Soundararaj, Christopher Pettit, and Oliver Lock in the year 2022 (e.g. [3]).
- An interactive dashboard was developed for real-time analytics, offering comprehensive insights into the evolving landscape of the COVID-19 outbreak in India. The adept execution of this initiative can be credited to the collaborative efforts of Arun Mitra, Biju Soman, and Gurpreet Singh in July 2021 (e.g. [4]).
- Munawar et al. (2020) proposed a vision wherein the real estate market, shaped by the influence of big data (e.g. [7], [6]), undergoes a transition into a 'smart real estate' paradigm comparable to the concept of 'smart cities. Within this framework, the advanced information and analytical capabilities inherent in smart real estate are anticipated to benefit stakeholders significantly. These advantages encompass tailored services for customers, precise property valuation for lenders, streamlined management and transaction processes for owners, enhanced fraud detection and taxation efficiency for government entities, and increased flexibility and risk mitigation for investors.
- Anas Bodor, Meriem Hnida, and Najima Daoudi proposed a model for monitoring and addressing the critical issue of lacking effective monitoring in machine learning projects, posing risks to model quality and sustainability in 2023. It gives a detailed explanation about the measurements and tools used in MLOps, focusing on the need to keep track of things constantly while creating and running machine learning models. For instance, Dominik Kreuzberger, Niklas Kuhl, and Sebastian Hirschl proposed a similar methodology for MLOps, as seen in references [1] and [5].

- Velmurugan A, Albert Mayan J, Mohana Prasad R, Yovan Felix A(2019), “Implementing Health Care Center using Hadoop for Analysis and Prediction of Diseases” this proposed system uses BigData tools like Hadoop to analyze and process a large amount of data to predict the diseases [6].
- Mohammad Hasnine, Ho tan Nguyen, Huyen T. T. Bui, Go`khan Akc,apınar, and Hiroshi Ueda banded on developing MOEMO (Motion and Emotion) [9], a real- time learning analytics system. MOEMO uses automated emotion detection from lecture videos, captured by built-in webcams, to measure online learner’s affective states of engagement and concentration.
- Fatih Gu`rcan and Muhammet Berigel [10], in collaboration, published a paper titled “Real-Time Processing of Big Data Streams: Lifecycle, Tools, Tasks, and Challenges” in IEEE. The paper defines a lifecycle for real-time big data processing, and associated tools, and frameworks with lifecycle phases. It addresses challenges such as volume, variety, scalability, and privacy in real- time big data processing. The study covers tools like Flume, Kafka, Nifi, Spark Streaming, Hbase, and more, offering valuable insights into the field.

In summary, the compilation of real-world examples and research studies showcased in the related works section underscores the breadth and depth of real-time dashboarding and data analytics applications across various domains. From addressing economic disruptions in real estate to monitoring the dynamic landscape of a global pandemic, these endeavors exemplify the versatility and significance of real-time analytics in today’s data-driven world. Furthermore, proposals for transitioning traditional markets into “smart” paradigms and models for enhancing machine learning project sustain- ability underscore the transformative potential of advanced data analytics techniques. Additionally, insights into healthcare analysis, emotion detection in online learning, and the lifecycle of real-time big data processing offer valuable perspectives on the evolving landscape of data-driven decision-making. Collectively, these contributions illuminate the myriad opportunities and challenges in real-time data analytics, shaping future research and innovation trajectory in the field.

2.2 Open Problems in Existing System

Existing systems that don't utilize big data tools may encounter various limitations and challenges when it comes to processing and analyzing large volumes of data. Some of the common problems in such systems include:

- **Scalability Issues:** Traditional systems may struggle to scale when the amount of data grows significantly. This can result in performance degradation and the inability to handle large datasets efficiently.
- **Slow Data Processing:** Without big data tools like Apache Spark, processing and analyzing large datasets can be time-consuming and may not meet real-time or near- real-time processing requirements.
- **Data Variety:** Movie-related data can come in various formats, such as text, images, videos, and structured data. Traditional systems may not handle diverse data types effectively, making it challenging to extract insights from them.
- **Data Integration:** In movie analytics, data often comes from diverse sources, including social media, streaming platforms, ticket sales, and more. Integrating data from these sources can be complex and error-prone in traditional systems.
- **Data Quality:** Data quality is a significant concern, as inaccuracies, inconsistencies, and missing data can affect the accuracy of analysis. Big data tools often include data cleaning and transformation capabilities, which are lacking in traditional systems.
- **Lack of Automation:** Traditional systems may rely heavily on manual processes for data ingestion, processing, and analysis, leading to inefficiencies and delays in delivering insights. Automated workflows provided by big data tools streamline these processes and enable faster decision-making for various organizations.
- **Performance Bottlenecks:** Traditional databases may experience performance bottlenecks when querying large datasets. Indexing and optimizing queries for complex analytical tasks can be challenging.
- **Cost Inefficiency:** Scaling traditional systems may require significant hardware and infrastructure investments, leading to cost inefficiency compared to cloud-based or big data platforms that allow for on-demand scaling.
- **Lack of Real-time Insights:** Traditional systems are often not well-suited for real-time analytics and monitoring. Big data tools with stream processing capabilities can

provide insights as data flows in.

- **Resource Constraints:** Traditional systems may be limited by hardware resources, such as memory and processing power, making it challenging to perform complex analyses on large datasets.
- **Version Control and Data Management:** Efficient version control and data management, as provided by tools like DVC, are often missing in traditional systems, which can lead to difficulties in tracking changes and reproducing results.
- **Data Security and Compliance:** Traditional systems may not have the same level of security and compliance features as modern big data platforms. Handling sensitive movie-related data and ensuring compliance with regulations may be more challenging.

2.3 Challenges of Big Data Analytics and Real Time Systems

In the expansive landscape of Big Data analytics and Real-Time systems, our proposed system encounters a myriad of challenges, spanning technical intricacies and organizational hurdles alike. Among these challenges, one of the foremost concerns revolves around the considerable cost implications inherent in implementing and sustaining real-time technology. The financial commitment required to develop and maintain a sophisticated infrastructure capable of processing vast volumes of data in real time is substantial, necessitating careful consideration and allocation of resources. Furthermore, architecting a complex real-time system that seamlessly integrates various components while ensuring scalability and efficiency poses a formidable task. This entails not only designing robust data pipelines and processing mechanisms but also orchestrating the interaction between different subsystems to achieve optimal performance. Moreover, the dynamic nature of data in real-time environments presents a significant management challenge. As data continuously streams in from various sources, maintaining its integrity, relevance, and accessibility becomes increasingly complex. This necessitates the implementation of robust data management strategies, including efficient storage mechanisms, effective data governance frameworks, and agile data processing pipelines. Additionally, in the fast-paced realm of digital marketing and customer engagement, accurately targeting the right audience with relevant campaigns is paramount. However, the ever-changing

preferences of subscribers, coupled with intense market competition, make this task inherently challenging. Operators must navigate through a sea of data to identify meaningful insights and trends, all while ensuring compliance with regulatory requirements and ethical considerations. Old-fashioned tools for understanding and storing data aren't very good at dealing with the huge amount and different types of data we have today. Things like logs from machines, posts on social media, and readings from sensors are growing fast. These tools have trouble making sense of all this information, which means we might miss important things and make not-so-great decisions. To make the most of all this data, organizations need to use new and smarter tools that can handle it better and help them make better decisions for their business.

CHAPTER – 3

AIM AND SCOPE OF PRESENTATION INVESTIGATION

AIM:

The aim of this research is to develop and implement a real-time dashboarding system that efficiently handles data from IoT devices, leveraging modern technologies and cloud-based solutions. The system aims to provide actionable insights in real-time, enabling organizations to make informed decisions and optimize operational processes. By focusing on the design, development, and deployment of such a system, this research aims to contribute to the advancement of real-time data analytics and IoT applications, addressing the growing need for scalable and efficient solutions in today's data-driven world.

SCOPE:

The scope of this research encompasses the development and implementation of a comprehensive real-time dashboarding system, designed to address specific challenges and achieve defined project goals. Each component of the system plays a crucial role in ensuring seamless data handling, processing, and visualization, thereby contributing to the overall success of the project. The scope includes the following key aspects:

i) Data Ingestion:

The system is designed to securely ingest data from IoT devices, storing it in Amazon Simple Storage Service (S3) using designated buckets. This process involves configuring read-only access policies, IAM roles, and EC2 instances for retrieving and processing raw data. The systematic workflow ensures secure data handling and efficient transition between storage and processing platforms.

ii) Networking:

Setting up the network infrastructure involves creating a customized Virtual Private Cloud (VPC) to provide a private space for cloud services. Security Groups and Access Control Lists (ACLs) are configured to control traffic and ensure secure communication within the VPC. The VPC is connected to an Internet Gateway (IGW) to enable public traffic, while instances are placed

within subnets for added security.

iii) Data Streaming:

The system employs Confluent Cloud for storing and processing real-time data streams, ensuring high availability and reliability. EC2 instances are responsible for handling data retrieval and processing, with dockerized producer and consumer codes for consistent execution. Continuous updates and automatic deployments are facilitated through GitHub Actions, while robust security measures ensure safe communication channels with Confluent Cloud.

iv) Data Processing:

Streaming data is processed using AWS Glue and Amazon EMR, with Apache PySpark employed for data processing and analysis. The processed data is stored in S3 buckets and cataloged using AWS Glue, enabling easy querying through Amazon Athena. This phase marks the final step in the data processing journey, preparing the data for further analysis and visualization.

v) Finding Patterns and Dashboarding:

Processed data is visualized and analyzed in real-time using Amazon QuickSight, which offers a variety of charting options for insightful data representation. The flexibility of QuickSight allows for easy sharing of real-time dashboards with stakeholders, enhancing collaboration and decision-making processes within organizations.

INVESTIGATION:

The investigation phase of our research project unfolds as a comprehensive journey into the intricate landscape of developing and implementing a real-time dashboarding system. This pivotal phase constitutes a meticulous exploration and analysis of the multifaceted components and cutting-edge technologies indispensable for the seamless deployment and operation of the envisioned system. Our investigative endeavors traverse a broad spectrum of domains, commencing with the pivotal endeavor of securely ingesting data from a diverse array of Internet of Things (IoT) devices into the resilient confines of Amazon Simple Storage Service (S3). Here in lies the crux of our exploration: understanding and optimizing the

mechanisms that govern the storage and retrieval of data, while meticulously configuring Identity Access Management (IAM) roles and Elastic Compute Cloud (EC2) instances to ensure the smooth flow of raw data into the repository. As our investigation unfolds, we delve deeper into the intricate intricacies of networking, venturing into the realm of customized Virtual Private Clouds (VPCs) to establish a robust and resilient network infrastructure. Herein, our focus shifts to the meticulous configuration of Security Groups, Access Control Lists (ACLs), and subnets, each serving as a cornerstone in fortifying the digital bastions that safeguard our cloud-based operations. Furthermore, we embark on a journey of discovery, exploring the nuances of data streaming facilitated by Confluent Cloud, an endeavor characterized by its seamless integration of high-availability data storage and processing capabilities. Our exploration extends to the realm of security, as we meticulously implement stringent measures such as SASL and SSL protocols to fortify communication channels and shield our operations from potential threats. Venturing further into the realm of data processing, our investigation unfolds within the expansive confines of Amazon Web Services (AWS) Glue and Amazon Elastic MapReduce (EMR), where we harness the immense processing power of Apache PySpark to unlock the latent insights concealed within our data troves. Here, our efforts are dedicated to optimizing processing workflows, ensuring efficient resource utilization, and orchestrating seamless data transitions between storage and processing tiers. As our investigation unfolds, we chart a course towards data visualization and analysis, where the dynamic capabilities of Amazon QuickSight come to the fore. Our exploration encompasses a diverse array of visualization techniques and analytical methodologies, aimed at distilling raw data into actionable insights that drive informed decision-making and strategic initiatives. Additionally, we explore datasets like MovieLens, which are important for learning, research, and business. We carefully study these datasets to understand how they fit with our project goals and appreciate the people who made them. Overall, our investigation covers a wide range of topics, all aiming to create a real-time dashboarding system that helps businesses make better decisions using data.

CHAPTER – 4

DESCRIPTION OF PROPOSED SYSTEM

The proposed Advanced Movie Analytics Platform represents a cutting-edge system poised to revolutionize the movie analytics landscape. Building upon the foundation of robust big data technologies like Apache Kafka, Spark, and Airflow, it takes data streaming, processing, and workflow automation to new heights. Integrated seamlessly with Data Version Control (DVC) for streamlined data management, hosted on the flexible and scalable AWS infrastructure, and sporting a feature-rich real-time dashboard, the platform is tailored to provide stakeholders in the movie industry with in-depth and actionable insights. Notably, it addresses data quality challenges, enhances data security and compliance measures, and equips users with advanced analytical capabilities, including predictive analytics and recommendation systems. In a fast-paced and ever-evolving movie industry, this proposed platform serves as an indispensable tool for informed, data-driven decision-making.

Real Time Systems:

Real-time is a broad concept involving many types of processes, tools, and users. It does not have to be fast: the deadline may be days or weeks. For instance, the time could be as short as that for a weather forecasting system or if 48 hours at the Texas Instruments European Program Information Centre (epic) for literature requests. Real-time software solutions monitor and analyze business activities to give a wide range of users the real-time visibility they need to see a problem or opportunity, make a fully informed decision, and then act accordingly. Many use cases for real-time technology are being deployed today, in many industries to enhance customer behavior, evaluate sales performance, identify a new social media sentiment etc. and one such major application is in the context of Big Data.

4.1 Selected Methodology or Process Model

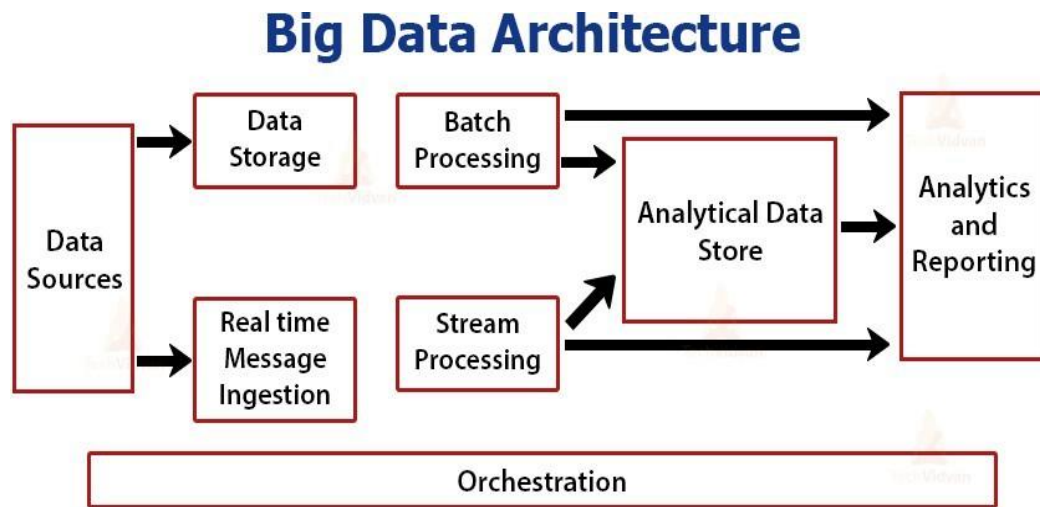


Fig: 4.1: High-Level Architecture of the Proposed Model

1. Data Extraction:

Data extraction involves collecting information from various sources, including databases, logs, APIs, and streaming platforms, in real-time or near-real-time.

2. Data Transformation:

Data transformation refers to the process of converting and structuring incoming data to match the desired format and quality, making it suitable for analysis and storage.

3. Change Data Capture (CDC):

CDC is a mechanism to identify and capture incremental changes in data as they occur, ensuring that only the modified data is processed in real-time ETL.

4. Data Streaming:

Data streaming is the continuous flow of data from source to destination, allowing for real-time processing and analysis, often using technologies like Apache Kafka or Apache Flink.

5. Monitoring and Alerting:

Real-time monitoring involves the continuous tracking of ETL processes and data quality, with alerts set up to notify of any issues or anomalies for prompt resolution.

4.2 Architecture of the Proposed System

The architecture and overall design of a proposed system play a pivotal role in realizing the goals of a project. In the context of your movie analytics project with a real-time dashboard, the architecture and design might look like this:

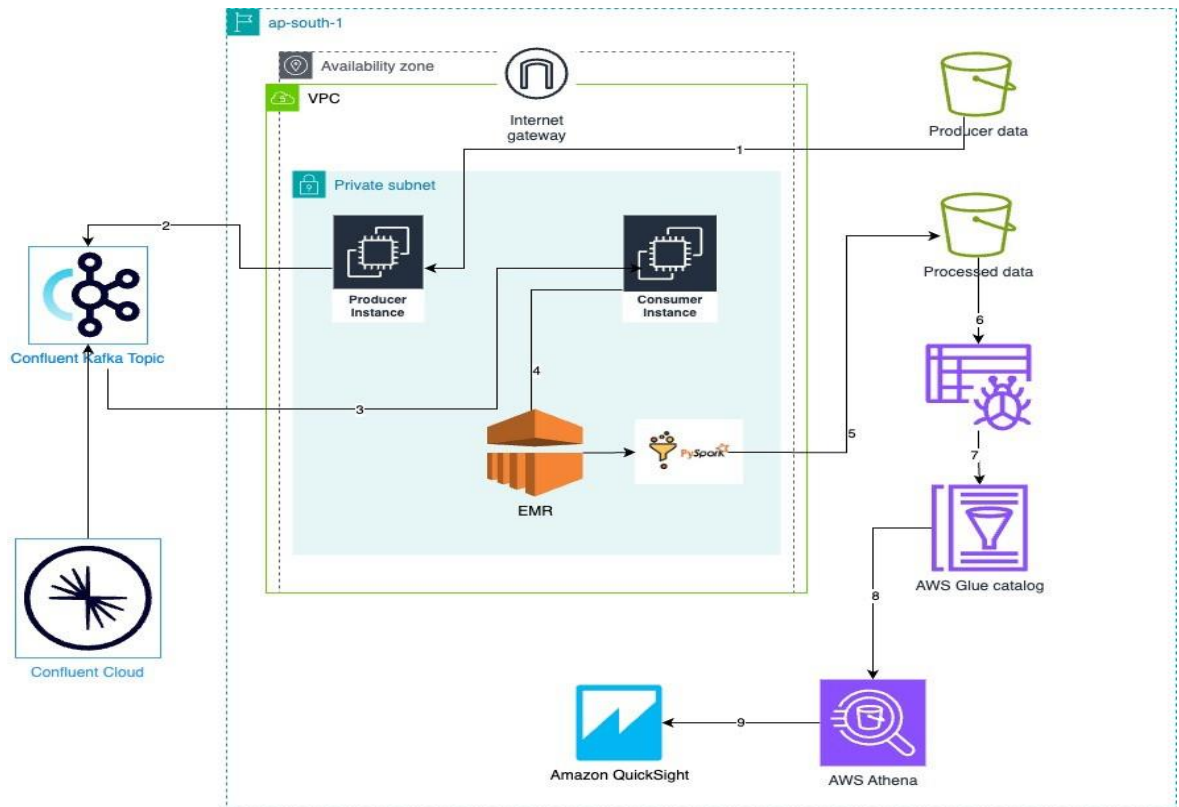


Fig: 4.2: End-to-End Architectural Flow with Big Data Components

4.2.1 High-Level System Architecture

The high-level architecture of the project unfolds as a systematic process encompassing various components and interactions:

1. Data Generation:

Originating from real IoT devices or simulated sources, batches of raw data are generated to simulate user interactions or events.

2. Amazon S3 Storage:

The raw data undergoes secure storage within Amazon Simple Storage Service (S3) buckets, offering robust durability and scalable data storage capabilities

3. *Confluent Cloud:*

Within a private subnet, dedicated producer and consumer instances establish connections with Confluent Cloud, a fully managed Kafka service. The producer transmits data to a Kafka topic, strategically partitioned into six segments.

4. *Security Protocols:*

Ensuring the integrity of data transmission, the system employs Security Assertion Markup Language (SASL) and Secure Sockets Layer (SSL) protocols, upholding secure communication channels between the producer, consumer, and Confluent Cloud.

5. *Data Consumption:*

The consumer instance adeptly retrieves real-time data from the Kafka topic, facilitating seamless data processing.

6. *Amazon EMR Processing:*

The consumed data undergoes further processing using Apache PySpark on Amazon Elastic Map Reduce (EMR), unlocking the potential for advanced analytics.

7. *S3 Storage (EMR Output):*

Processed data finds its way back into Amazon S3, establishing an organized storage repository and maintaining a comprehensive catalog for efficient future querying.

8. *AWS Glue and Athena:*

AWS Glue Crawler undertakes a meticulous analysis of the data schema, resulting in the creation of a comprehensive catalog. This catalog enables users to query the data effortlessly through Amazon Athena, leveraging a familiar SQL-like interface.

9. *Data Visualization:*

The culmination of the process involves leveraging Amazon QuickSight for visually representing data insights. This step provides a variety of charting options, empowering descriptive and real-time analyses.

This plan makes sure we have a safe, adaptable, and effective system for creating real-time dashboards with big data. By using Confluent Kafka, we make things easier and more flexible. Plus, we follow strict security rules and regularly update the system, making it strong and up-to-date with industry standards.

4.2.2 Data Flow

1. Data is ingested through various sources, including streaming platforms, social media, and movie databases.
2. Apache Kafka processes and streams this data to microservices for real-time analysis and transformation.
3. Microservices perform data cleansing, feature extraction, and apply machine learning models to generate insights.
4. Insights are stored in a database and simultaneously presented on the real-time dashboard for user interaction.

4.2.3 User Interface and Real-Time Dashboard

The user interface provides a dynamic and intuitive real-time dashboard for stakeholders. Key features of the dashboard include:

- Real-time data visualizations, including charts, graphs, and tables.
- Interactive features for exploring data and drilling down into specific metrics.
- Options for generating custom reports and exporting data.
- User management and authentication for access control.
- User-friendly design for ease of use.

4.2.4. CI/CD Workflow of Instances using GitHub Actions

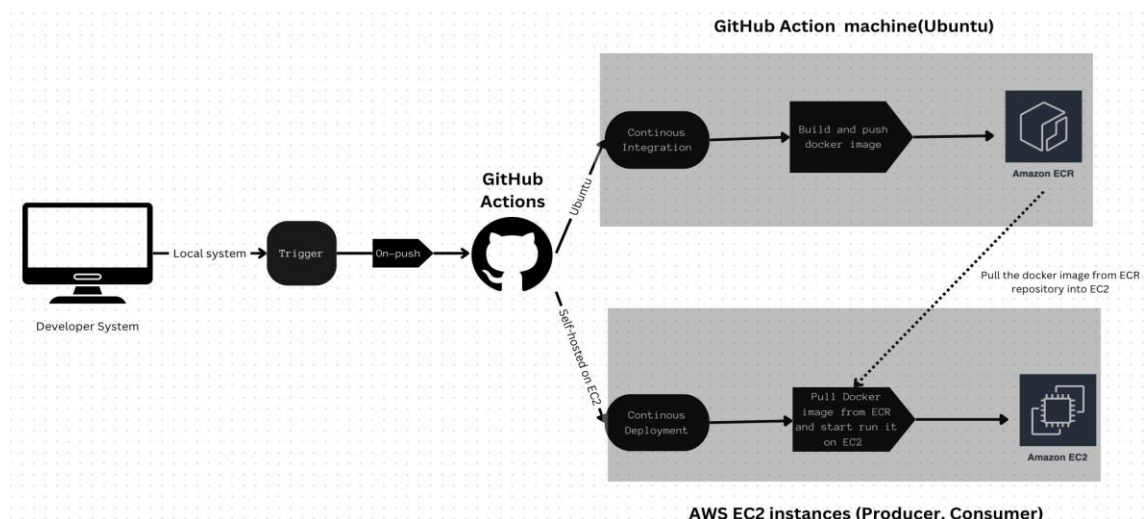


Fig: 4.3: CI / CD pipeline using GitHub Actions.

For the seamless integration and deployment of our producer and consumer components, we rely on GitHub Actions to facilitate continuous integration and continuous deployment (CI/CD) workflows. This process is orchestrated with precision, utilizing resources both from GitHub and AWS to ensure efficiency and reliability. Within the CI phase, a robust Ubuntu machine provided by GitHub Actions serves as the foundation for executing the necessary tasks. Concurrently, the CD process unfolds on dedicated AWS EC2 instances, meticulously designated for hosting our producer and consumer functionalities. Figure 4.3 vividly illustrates the orchestration of our CI/CD workflow, underscoring the interconnectedness of various components and stages within the deployment pipeline. During the CI phase, a Docker image is meticulously crafted, encapsulating the requisite dependencies and configurations for our components. Subsequently, this image is securely pushed to a private Amazon Elastic Container Registry (ECR) repository, ensuring a centralized and controlled storage environment for our artifacts. Transitioning to the CD phase, the Docker image residing within the private ECR repository is seamlessly pulled onto the designated EC2 instances, meticulously prepared to host our containerized environments. Within this controlled ecosystem, the Docker image is executed, enabling our producer and consumer components to operate seamlessly within their respective environments. This streamlined CI/CD pipeline serves as a cornerstone of our development process, ensuring that updates and enhancements to our components are rigorously tested, efficiently built, and seamlessly deployed, thus maintaining the integrity and reliability of our real-time dashboarding system.

4.2.5 Software and Services Required

Apache Kafka (Confluent Cloud):

At the core of our software implementation lies Apache Kafka, a distributed streaming platform renowned for its exceptional throughput, fault tolerance, and scalability. Serving as the backbone for ingesting, storing, and processing real-time data streams, Kafka plays an instrumental role in facilitating seamless data flow across the ecosystem. Leveraging Confluent Cloud, a serverless Kafka service, further augments the capabilities of our real-time dashboarding system by simplifying infrastructure management and enabling hassle-free scalability

Apache Spark:

Complementing Kafka and ZooKeeper is Apache Spark, a powerful open-source framework for real-time data processing and analytics. Spark offers a rich set of APIs and libraries for distributed stream processing, enabling complex data transformations, aggregations, and machine learning tasks in real-time. With its ability to handle large-scale data processing and fault tolerance, Spark enhances the capabilities of our real-time dashboarding system, facilitating advanced analytics and insights generation.

AWS Services:

In addition to Apache Kafka and Spark, our software implementation leverages a suite of AWS services to enhance scalability, reliability, and efficiency in data analytics. Amazon S3 (Simple Storage Service) serves as a scalable and durable storage solution for storing and archiving data, providing high availability and low latency access to data assets. Amazon Athena, a serverless query service, enables ad-hoc querying of data stored in S3 using standard SQL, eliminating the need for complex infrastructure provisioning and management.

GitHub:

GitHub is a web-based platform for version control and collaboration. It allows developers to host and review code, manage projects, and build software together. GitHub provides features like pull requests, code reviews, issue tracking, and continuous integration/continuous deployment (CI/CD) pipelines, enabling teams to work efficiently and collaborate seamlessly.

Amazon S3:

Amazon Simple Storage Service (S3) is an object storage service that offers scalable, secure, and durable storage for data in the cloud. S3 provides features like data encryption, versioning, lifecycle management, and access control, making it suitable for a wide range of use cases including backup and restore, data archiving, content distribution, and data lake storage.

Amazon EC2:

Amazon Elastic Compute Cloud (EC2) is a web service that provides resizable compute capacity in the cloud. EC2 allows users to launch virtual servers, known as instances, on-demand and scale capacity up or down as needed. EC2 instances come in various sizes and configurations, enabling users to choose the right instance type for their applications based on performance, memory, storage, and networking requirements.

Amazon Athena:

Amazon Athena is an interactive query service that allows users to analyze data stored in Amazon S3 using standard SQL queries. Athena is serverless and requires no infrastructure setup, allowing users to quickly query large-scale datasets without managing complex data warehouses or clusters. It supports various data formats including CSV, JSON, Parquet, and ORC.

Amazon EMR:

Amazon Elastic MapReduce (EMR) is a managed big data platform that simplifies the deployment and operation of Apache Hadoop, Apache Spark, Apache Hive, and other big data frameworks on AWS. EMR allows users to process and analyze large-scale datasets using distributed computing frameworks, without the need to provision and manage clusters manually.

AWS Glue:

AWS Glue, a fully managed extract, transform, and load (ETL) service, automates the process of preparing and transforming data for analysis. With Glue, organizations can easily discover, catalog, and clean datasets, accelerating the data preparation phase and enabling faster insights generation. By automating ETL tasks and providing a serverless architecture, Glue streamlines the data pipeline and enhances the efficiency of our real-time dashboarding system.

Docker:

Docker is a platform for developing, shipping, and running applications in containers. Containers allow developers to package applications and their dependencies into a

standardized unit, ensuring consistency across different environments. Docker simplifies building, managing, and deploying applications using containers.

Amazon QuickSight:

For visualization and dashboarding capabilities, our software implementation incorporates Amazon QuickSight, a cloud-based business intelligence service. QuickSight enables the creation of interactive dashboards and visualizations, empowering users to explore and analyze data insights in real-time. With features such as built-in machine learning-driven analytics and seamless integration with AWS data sources, QuickSight facilitates data-driven decision-making.

4.3 Dataset Metadata Overview

This project utilizes three DAT file format datasets, comprising 1,000,209 ratings from about 3,900 movies by 6,040 users on MovieLens in 2000. Widely employed in education, research, and industry, MovieLens datasets have significant usage in programming books and courses. Originating from the MovieLens movie recommendation system in 1997, these datasets offer insights into user activity. The article provides a concise history of MovieLens, emphasizing lessons learned and best practices, and offering valuable guidance for effective utilization in new research. Cite the dataset using the referenced paper (e.g. [11]). Special thanks to Shyong Lam and Jon Herlocker for their work on the dataset. The GroupLens Research Project at the University of Minnesota's Department of Computer Science and Engineering, led by Professors John Riedl and Joseph Konstan, explores various aspects of information filtering, collaborative filtering, and recommender systems. Commencing in 1992, the project gained recognition in 1996 for a global trial of an automated collaborative filtering system for Usenet news. Over time, it has broadened its focus to encompass overall information filtering solutions, integrating content-based methods, and enhancing collaborative filtering technology.

i) Rating data file

All the ratings are in the "rating.dat" file containing three fields. Each user has at least 20 ratings.

- UserID range between 1 and 6040
- MovieID range between 1 and 3952
- Ratings are made on a 5-star scale

ii) Movie data file

The “movies.dat” file contains movie information in the following format.

- MovieID is a unique identifier for each movie.
- Title is the movie title, matching those provided by IMDB, including the release year.
- Genres of the movie, pipe-separated, selected from a predefined set, including Action, Adventure, Animation, Children’s, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, and Western.

iii) User data file

The “users.dat” file contains user information in the for- mat: UserID, Gender, Age, Occupation, and Zip-code. Demographic information is voluntarily provided by users and is not verified for accuracy. Only users who have provided demographic information are included in this dataset.

- Gender is indicated by “M” for male and “F” for female.
- Age falls within predefined ranges, including “Under 18”, “18-24”, “25-34”, “35-44”, “45-49”, “50-55”, and “56+”.
- Occupation field is represented by integers corresponding to specific occupations, with over 20 different occupations included in this field.

This project utilizes three datasets in DAT file format from MovieLens, containing a total of 1,000,209 ratings for about 3,900 movies by 6,040 users in the year 2000. These datasets are widely used in education, research, and industry, particularly in programming books and courses. Originating from the MovieLens movie recommendation system in 1997, these datasets provide insights into user activity. The datasets include rating data, movie data, and user data files. The rating data file (“rating.dat”) includes UserID, MovieID, and ratings on a 5-star scale. The movie data file (“movies.dat”) contains MovieID, title, and genres separated by pipes. The user data file (“users.dat”) includes UserID, gender, age, occupation, and zip code, with demographic information voluntarily provided by users. The project acknowledges the contributions of Shyong Lam and Jon Herlocker to the dataset and highlights the GroupLens Research Project at the University of Minnesota, led by Professors John Riedl and Joseph Konstan, for their exploration of information filtering and collaborative filtering technologies since 1992.

CHAPTER – 5

RESULTS AND DISCUSSION

5.1 Output

These are the basic queries that are performed by Amazon QuickSight. By using direct queries, the user or the data analytics team can visualize near real-time insights and make critical business decisions on top of that. This kind of information system contributes significantly to the business group and acts as a decision support system for various fields like finance, sales, marketing, and more. QuickSight enables you to effortlessly analyze sales data, uncover trends, and strategize for improved outcomes. In Amazon QuickSight, we can give permissions more granularly, allowing us to set which group of people can access the real-time dashboards. QuickSight is integrated with Amazon Athena, which is used to query S3 bucket data. By leveraging the Amazon Glue catalog and Athena, QuickSight can utilize these services to create interactive dashboards. In our case, the output from PySpark on Amazon EMR is stored in S3. QuickSight is then used to develop dynamic real-time dashboards. The images below showcase the insights from the queries in QuickSight.

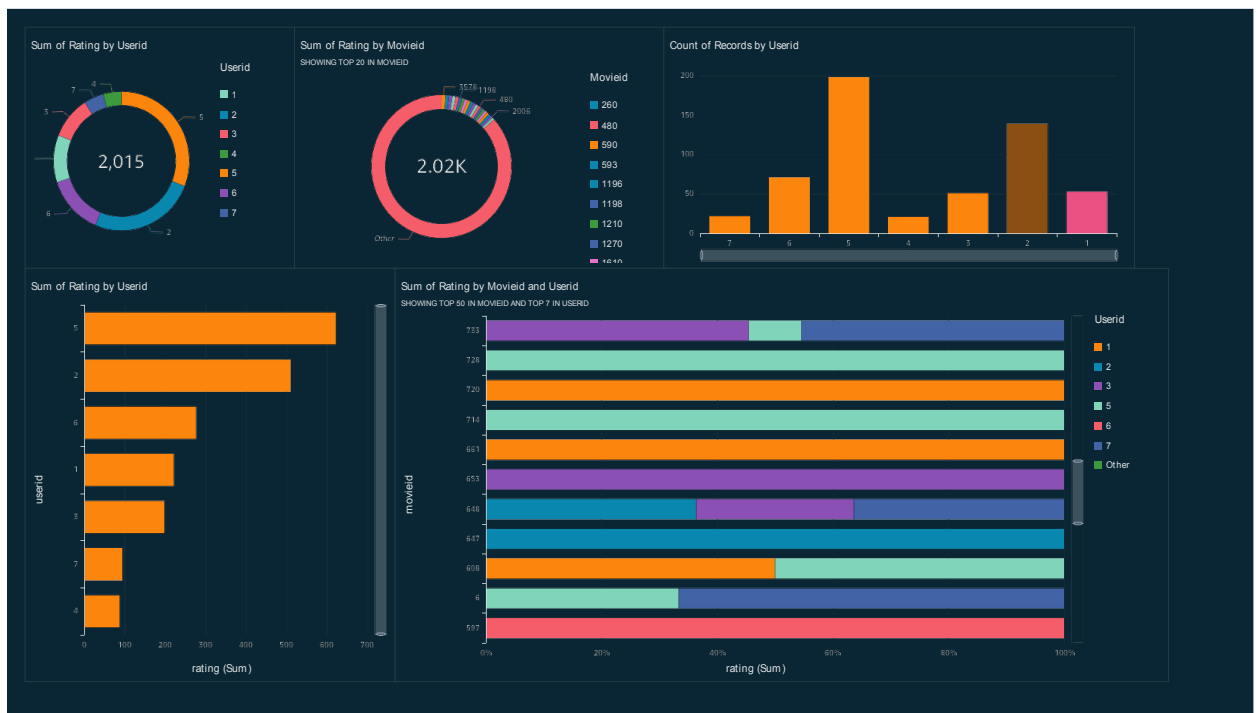


Fig: 4.4: Real-Time Insight Dashboard in QuickSight.

When prioritizing cost efficiency, it becomes paramount to explore transitioning the private subnet to a public subnet configuration. This strategic shift empowers instances, including the Producer and Consumer, to directly access an internet gateway, thereby enabling the provision of a public IP address accessible to a wider audience. By embracing this transition, organizations stand to benefit from the elimination of the necessity for a NAT gateway, consequently mitigating associated expenses. However, this adjustment isn't without its considerations. While it promises cost savings, it concurrently introduces heightened security implications owing to the expanded exposure of the system to the public domain. Therefore, while the pursuit of cost reduction measures holds undeniable importance, it's equally crucial to meticulously weigh the potential trade-offs and security considerations inherent in such architectural adjustments. Striking a delicate balance between cost optimization and security posture is imperative to ensure the integrity and resilience of the system in the face of evolving threats and challenges. By adopting a thoughtful and comprehensive approach, organizations can effectively navigate the complexities of architectural decisions, ultimately driving sustainable cost efficiencies without compromising on security and reliability.



5.3 Comparative Analysis

Cloud systems offer unparalleled adaptability, effortless scalability, and cost-effective operation, all efficiently managed. By leveraging Confluent Kafka, organizations gain access to a multitude of benefits, including a serverless utility model, robust data governance, comprehensive monitoring capabilities with a user-friendly interface, and freedom from the constraints of native Kafka languages like Java. Confluent Kafka empowers developers to innovate using a diverse range of programming languages, such as C++ and Python, democratizing the development process and fostering agility in solution creation. With Confluent Kafka leading the way, organizations can confidently navigate the complexities of modern data architectures, driving unparalleled efficiency and innovation. In contrast, traditional setups face limitations in scalability, requiring manual efforts and involving higher upfront costs. Leveraging distributed storage, cloud systems outperform traditional solutions, providing global accessibility. Traditional systems often encounter challenges accessing real-time data, relying instead on data warehouses for information. For example, when conducting monthly sales analyses, this reliance can lead to delays in accessing critical insights, potentially compromising decision-making, information processing, and monitoring capabilities. The suggested system harnesses dashboarding tools provided within the same cloud platform, such as Amazon QuickSight, enabling direct querying with minimal latency. In contrast, traditional systems may rely on third-party tools like PowerBI and Tableau, which could introduce compatibility challenges and additional complexities. Unlike old-fashioned computer setups that require extensive upkeep and physical space, cloud systems operate differently. They prioritize security from the start and offer flexibility for quick changes and new ideas without delay. In AWS (Amazon Web Services), the Shared Responsibility Model (SRM) ensures a collaborative approach to security, where AWS manages infrastructure and data centers, while customers take responsibility for securing their data and applications. Cloud computing is highly efficient as it eliminates the need for numerous physical machines to be maintained on-premises. It facilitates seamless scalability, allowing businesses to easily expand their online presence as they grow. Moreover, cloud systems enable seamless collaboration from anywhere in the world, fostering productivity and innovation. Overall, cloud systems offer superior safety, flexibility, and usability compared to traditional setups, making them the preferred choice for modern organizations.

CHAPTER – 6

SUMMARY AND CONCLUSION

In summary, our real-time dashboard system, powered by big data and Confluent Kafka, ensures efficiency, security, and scalability, seamlessly integrating with AWS services. Integration with Confluent Cloud offers serverless flexibility, extended data retention, comprehensive monitoring tools, versatile data connectors, self-balancing clusters, and a user-friendly cloud UI. Committed to continuous integration, our system stands as a cost-effective choice for real-time analytics, boasting elastic scaling and simplified management without Zookeeper. However, real-time dashboards come with challenges such as network latencies and the limitation of seeing real-time data only upon page refresh when using direct queries. Moreover, our proposed system doesn't directly utilize real-time data from diverse sensors; instead, it simulates this data by generating our Confluent Kafka producer. Looking ahead, we envision the development of a standalone web application to address these constraints. This application will serve as a platform for visualizing real-time data, updating every second, like candle plots used in trading insights of the stock market. These future initiatives represent our commitment to overcoming existing limitations and enhancing the capabilities of real-time analytics.

REFERENCES

- [1] Machine Learning Operations (MLOps): Overview, Definition, and Architecture proposed by Dominik Kreuzberger, Niklas Kuhl, and Sebastian Hirschl date of publication 27 March 2023.
- [2] Big Data and Its Applications in Smart Real Estate and the Disaster Management Life Cycle: A Systematic Analysis proposed by Hafiz Suliman Munawar, Siddra Qayyum, Fahim Ullah. Published: 26 March 2020.
- [3] Real-Time Dashboards to Monitor the Impact of Disruptive Events on the Real Estate Market. The case of the COVID-19 Pandemic in Australia was proposed by Balamurugan Soundararaj, Christopher Pettit, and Oliver Lock. Published on 10 June 2022 Volume 2, article number 14, (2022).
- [4] An interactive dashboard for real-time analytics and monitoring of COVID-19 outbreak in India: a proof of concept. Proposed by Arun Mitra, Biju Soman, and Gurpreet Singh in July 2021. Conference: IFIP 9.4 2021, 1st virtual conference. conference theme: resilient ICT4D.
- [5] Machine Learning Models Monitoring in MLOps Metrics and Tools. Proposed by Anas Bodor, Meriem Hnida and Daoudi Najima. December 2023 International Journal of Interactive Mobile Technologies (iJIM) 17(23):125-139.
- [6] Velmurugan A, Albert Mayan J, Mohana Prasad R, Yovan Felix A(2019), "Implementing Health Care Center using Hadoop for Analysis and Prediction of Diseases", International Journal of Engineering and Advanced Technology (IJEAT), Vol.8, No.6, pp.3456-3459. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetism Japan, p. 301, 1982].
- [7] Big Data Analytics of Movie Rating Predictive System. Proposed by Neeta Choudhry, Jianyun Xie, and Xiaoling Xia. The 5th Annual International Conference on Information Systems and Artificial Intelligence [ISAI2020] took place on May 22-23, 2020, in Zhejiang, China, as published in Volume 1575 of the Journal of Physics: Conference Series.
- [8] Data visualization techniques for real-time information A custom and dynamic dashboard for analyzing survey results. Proposed by Renato Mauricio Toasa G, Marisa Maximiano, Catarina I. Reis, David Guevara. Published in 2018 on the 13th (CISTI).

- [9] A Real-Time Learning Analytics Dashboard for Automatic Detection of Online Learners' Affective States. Mohammad Nehal Hasnine. Published on Research Center for Computing and Multimedia Studies, Hosei University at Tokyo 102-8160, Japan.
- [10] Real-Time Processing of Big Data Streams: Lifecycle, Tools, Tasks, and Challenges. Proposed by Fatih Gurcan and Muhammet Berigel. October 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT).
- [11] The MovieLens Datasets: History and Context. Proposed by F. Maxwell Harper and Joseph A. Konstan. Volume 5, Issue 4 of ACM Transactions on Interactive Intelligent Systems.
- [12] Albert Mayan. J, Yovan Felix. A, Chintalapudi M, Balaji R, and Vigneswaran R (2015) presented a paper titled "Fault-Open Minded Resource Allocation Rescheduling Algorithm and Expense Minimization for Cloud Systems" in the Global Journal of Pure and Applied Mathematics, Volume 11, Issue 6, pages 4111-4117.
- [13] Hu and Liu (2004) presented a study titled "Extraction and Condensation of Customer Feedback" at the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, held in Seattle, WA, USA, in August 2004.
- [14] Alsaqer and Sasi (2017) conducted research titled "Summarization and Sentiment Analysis of Movie Reviews Using RapidMiner" at the 2017 International Conference on Networks and Advances in Computational Technologies (NetACT), held in Thiruvananthapuram, India, in July 2017.
- [15] Liu, C.-Y., Chen, M.-S., and Tseng, C.-Y. (Publisher: IEEE). IncreSTS: Towards Real-Time Incremental Short Text Summarization on Comment Streams from Social Network Services.

APPENDIX

A. Source code

The source code comprises two distinct modules, each dedicated to the functionality of producer and consumer instances, ensuring clear separation of concerns and modular development.

1. Entity Module

This file contains a collection of versatile utility functions, including conversions between dictionaries and instances for JSON serialization, retrieval of S3 bucket objects, and schema extraction for serialization purposes. It serves as a shared module for both the producer and consumer modules, facilitating consistency and reuse of essential functionalities

i. generic.py

```
import logging
import os, sys
import json
import boto3
import pandas as pd
import numpy as np
from src.constants import *
from src.kafka_logger import logging

class Generic:
    def __init__(self, record:dict):
        for k,v in record.items():
            setattr(self, k, v)

    def to_dict(self):
        return self.__dict__

    @classmethod
    def s3_obj(cls):
        s3 = boto3.resource(
            service_name="s3",
            region_name=os.getenv('AWS_REGION'),
            aws_access_key_id=os.getenv('AWS_ACCESS_KEY_ID'),
            aws_secret_access_key=os.getenv('AWS_SECRET_ACCESS_KEY')
        )
```

```

s3client = s3.Bucket(BUCKET_NAME).Object(DATA_PATH).get()
return s3client

@classmethod
def dict_to_instance(cls,dict_1:dict,ctx):
    print(dict_1, ctx)
    return cls(dict_1)

@classmethod
def get_object(cls, data_path:str, obj):
    try:
        columns = Generic.get_columns(SCHEMA_FILE_PATH)
        for line in obj["Body"].iter_lines():
            if line:
                line = line.decode('utf-8').strip().split("::")
                generic = Generic(dict(zip(columns, line)))
                yield generic
    except Exception as e:
        logging.info(e)

@classmethod
def get_schema(cls,schema_file_path:str):
    try:
        schema=dict()
        schema.update({
            "$id": "http://example.com/myURI.schema.json",
            "$schema": "http://json-schema.org/draft-07/schema#",
            "additionalProperties": False,
            "description": "Sample schema to help you get started.",
            "properties": dict(),
            "title": "SampleRecord",
            "type": "object"})
        columns = Generic.get_columns(schema_file_path)
        for column in columns:
            schema["properties"].update(
                {
                    f'{column}' :
                    {
                        "discription" : fgeneric {column}',
                        "type" : "string"
                    }
                }
            )

```

```

    )
    schema = json.dumps(schema)
    return schema
except Exception as e:
    logging.info(e)
@staticmethod
def get_columns(file_path:str):
    with open(file_path, 'r') as f:
        columns = f.readline().strip().split(',')
    return columns
def instance_to_dict(instance: Generic, ctx):
    return instance.to_dict()

```

2. Consumer code of Confluent Kafka

The consumer code for Confluent Kafka is tasked with retrieving data from the Apache Kafka broker hosted on Confluent Cloud. It performs deserialization of the serialized data generated by the producer instance, which is then subscribed to and consumed by the consumer instance operating on Amazon EC2. Additionally, it handles the Kafka configuration file for SSL and SASL protocols, along with setting up API keys, secret keys, and schema registry keys.

i. Kafka_Config module

```

import os
API_KEY = os.getenv('API_KEY', None)
ENDPOINT_SCHEMA_URL = os.getenv('ENDPOINT_SCHEMA_URL', None)
API_SECRET_KEY = os.getenv('API_SECRET_KEY', None)
BOOTSTRAP_SERVER = os.getenv('BOOTSTRAP_SERVER', None)
SCHEMA_REGISTRY_API_KEY = os.getenv('SCHEMA_REGISTRY_API_KEY', None)
SCHEMA_REGISTRY_API_SECRET = os.getenv('SCHEMA_REGISTRY_API_SECRET', None)
SECURITY_PROTOCOL = "SASL_SSL"
SSL_MECHANISM = "PLAIN"
def sasl_config():
    sasl_conf = {
        'sasl.mechanism': SSL_MECHANISM,
        'bootstrap.servers': BOOTSTRAP_SERVER,
        'security.protocol': SECURITY_PROTOCOL,
        'sasl.username': API_KEY,
        'sasl.password': API_SECRET_KEY }

```

```

print(os.getenv('ENDPOINT_SCHEMA_URL', None))
print(sasl_conf)
return sasl_conf
def schema_config():
    return {
        'url' : ENDPOINT_SCHEMA_URL,
        'basic.auth.user.info' : f'{SCHEMA_REGISTRY_API_KEY}:{SCHEMA_REGISTRY_API_SECRET}'
    }

```

ii. *json_consumer.py*

```

import os, sys
import json
from confluent_kafka import Consumer
from confluent_kafka.serialization import SerializationContext, MessageField
from confluent_kafka.schema_registry.json_schema import JSONDeserializer
from src.kafka_logger import logging
from src.constants import *
from src.kafka_config import sasl_config
from src.entity.generic import Generic
def consume_data(topic:str, schema_path:str):
    schema_str = Generic.get_schema(schema_path)
    json_deserializer = JSONDeserializer(schema_str, from_dict=Generic.dict_to_instance)
    consumer_conf = sasl_config()
    consumer_conf.update({
        'group.id' : 'group1',
        'auto.offset.reset' : 'earliest'})
    consumer = Consumer(consumer_conf)
    consumer.subscribe([topic])
    en = 0
    s3 = Generic.s3_obj()
    while True:
        try:
            msg = consumer.poll(1.0)
            if msg is None:continue
            en = en+1
            record: Generic = json_deserializer(msg.value(), SerializationContext(msg.topic(),
MessageField.VALUE))
            s3object = s3.Object(BUCKET_NAME, f'rating-data-{en}.json')
            s3object.put(
                Body=(bytes(json.dumps(record.to_dict()).encode('UTF-8'))))

```

```

    )
    print(record.to_dict())
except KeyboardInterrupt:
    break
consumer.close()

```

3. Producer code for Confluent Kafka and the configuration for them are same as the consumer modules code.

i. *json_producer.py*

```

import os, sys
import time
import pandas as pd
import numpy as np
from typing import List
from uuid import uuid4
from src.kafka_config import sasl_config, schema_config
from src.entity.generic import Generic, instance_to_dict
from src.kafka_logger import logging
from confluent_kafka import Producer
from confluent_kafka.serialization import StringSerializer, SerializationContext, MessageField
from confluent_kafka.schema_registry import SchemaRegistryClient
from confluent_kafka.schema_registry.json_schema import JSONSerializer
def delivery_report(err, msg):
    if err is not None:
        logging.info("Delivery failed for User record {}: {}".format(msg.key(), err))
        return
    logging.info('User record {} successfully produced to {} [{}] at offset {}'.format(
        msg.key(), msg.topic(), msg.partition(), msg.offset()))
def produce_data(topic:str, file_path:str,schema_path:str):
    logging.info(f"Topic: {topic} file_path:{file_path}")
    schema_str = Generic.get_schema(schema_path)
    schema_registry_conf = schema_config()
    schema_registry_client = SchemaRegistryClient(schema_registry_conf)
    string_serializer = StringSerializer("utf-8")
    json_serializer = JSONSerializer(schema_str, schema_registry_client, instance_to_dict)
    producer = Producer(sasl_config())
    print(f"Producing user data for Topic-{topic}")
    producer.poll(1.0)

try:

```

```

c = 0
s3Client = Generic.s3_obj()
for instance in Generic.get_object(file_path,s3Client):
    logging.info(f"Topic: {topic} file_path:{instance.to_dict()}")
    producer.produce(topic=topic,
                     key=string_serializer(str(uuid4())), instance.to_dict()),
                     value=json_serializer(instance, SerializationContext(topic, MessageField.VALUE)),
                     on_delivery=delivery_report
                    )
    if c % 10 == 0:
        print(instance.to_dict())
        print("Producer Flushing...")
        producer.flush()
        time.sleep(2)

except KeyboardInterrupt as e:
    logging.info(e)

```

4. The Dockerfile and Docker Compose file are essential components utilized for containerization, enabling the storage of container images within the Amazon Elastic Container Registry (ECR) as part of our AWS infrastructure

i. Dockerfile

```

FROM python:3.8
RUN mkdir consumer
COPY . ./consumer/
WORKDIR /consumer
RUN pip install --upgrade pip
RUN pip install -r requirements.txt
RUN apt update -y && apt install awscli -y
CMD ["python", "consumer_main.py"]

```

ii. docker_compose.yml

```

version: '1'
services:
  consumer:
    image: ${IMAGE_NAME}
    container_name: consumer_container
    environment:
      - API_KEY=${API_KEY}

```

- API_SECRET_KEY=\${API_SECRET_KEY}
- ENDPOINT_SCHEMA_URL=\${ENDPOINT_SCHEMA_URL}
- BOOTSTRAP_SERVER=\${BOOTSTRAP_SERVER}
- SCHEMA_REGISTRY_API_KEY=\${SCHEMA_REGISTRY_API_KEY}
- SCHEMA_REGISTRY_API_SECRET=\${SCHEMA_REGISTRY_API_SECRET}
- SECURITY_PROTOCOL=\${SECURITY_PROTOCOL}
- SSL_MECHANISM=\${SSL_MECHANISM}
- AWS_ACCESS_KEY_ID=\${AWS_ACCESS_KEY_ID}
- AWS_SECRET_ACCESS_KEY=\${AWS_SECRET_ACCESS_KEY}

5. The GitHub Actions workflow file orchestrates the Continuous Integration and Continuous Deployment (CI/CD) workflow. During Continuous Integration, the consumer and producer components are converted into Docker images as specified in the YAML file and then pushed to AWS ECR. In Continuous Deployment, the latest image from ECR is pulled into the respective instances, initiating the container runtime

i. main.yaml

```
name: workflow
on:
  push:
    branches:
      - main
permissions:
  id-token: write
  contents: read
jobs:
  build-push-image-to-ecr:
    name: Continous Delivery
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v3
      - name: Install Utilizes
        run: |
          sudo apt-get update
          sudo apt-get install -y jq unzip

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v1
```

```

with:
  aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
  aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
  aws-region: ${ secrets.AWS_REGION }
- name: Login to AMAZON ECR
  id: login-ecr
  uses: aws-actions/amazon-ecr-login@v1
- name: Build, tag and push image to Amazon ECR
  id: build-image
  env:
    ECR_REGISTRY: ${ steps.login-ecr.outputs.registry }
    ECR_REPOSITORY: ${ secrets.ECR_REPOSITORY_NAME }
    IMAGE_TAG: latest
  run: |
    docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
    docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
    echo "::set-output name=image::$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG"

```

Continuous-Deployment:

```

needs: build-push-image-to-ecr
runs-on: self-hosted
steps:
- name: Checkout
  uses: actions/checkout@v3
- name: Configure AWS credentials
  uses: aws-actions/configure-aws-credentials@v1
  with:
    aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
    aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
    aws-region: ${ secrets.AWS_REGION }
- name: Install Utilizes
  run: |
    sudo apt-get update
    sudo apt-get install -y jq unzip
- name: Login to AMAZON ECR
  id: login-ecr
  uses: aws-actions/amazon-ecr-login@v1

- name: Pull docker image
  run: |
    docker rmi ${ secrets.AWS_ECR_LOGIN_URI }/${ secrets.ECR_REPOSITORY_NAME }:latest

```

```

-f docker rm consumer_container

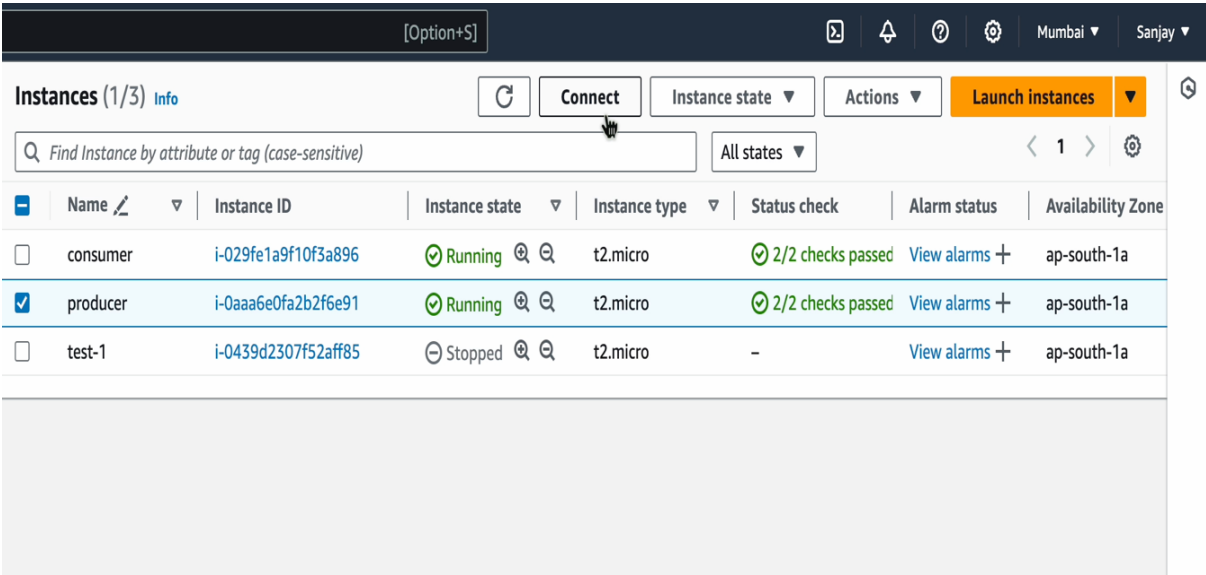
docker pull ${ secrets.AWS_ECR_LOGIN_URI }}/${ secrets.ECR_REPOSITORY_NAME }}:latest
- name: Run docker image to serve users
run: |
    export IMAGE_NAME="${ secrets.AWS_ECR_LOGIN_URI }}/${ secrets.ECR_REPOSITORY_NAME }}:latest"
    AWS_ACCESS_KEY_ID="${ secrets.AWS_ACCESS_KEY_ID }}"
    AWS_SECRET_ACCESS_KEY="${ secrets.AWS_SECRET_ACCESS_KEY }}"
    AWS_REGION="${ secrets.AWS_REGION }}"
    export API_KEY="${ secrets.API_KEY }}"
    API_SECRET_KEY="${ secrets.API_SECRET_KEY }}"
    ENDPOINT_SCHEMA_URL="${ secrets.ENDPOINT_SCHEMA_URL }}"
    export BOOTSTRAP_SERVER="${ secrets.BOOTSTRAP_SERVER }}"
    SCHEMA_REGISTRY_API_KEY="${ secrets.SCHEMA_REGISTRY_API_KEY }}"
    export SCHEMA_REGISTRY_API_SECRET="${ secrets.SCHEMA_REGISTRY_API_SECRET }}"
    SECURITY_PROTOCOL="${ secrets.SECURITY_PROTOCOL }}"
    export SSL_MECHANISM="${ secrets.SSL_MECHANISM }}"
    docker compose up -d

```

B. Screenshots

The screenshots shown below are from AWS management console which showcases various services like S3, EC2 instances and the real time streaming of data from producer to consumer.

- i. The following image illustrates the producer and consumer instances within the AWS console, situated within a private subnet.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☐	consumer	i-029fe1a9f10f3a896	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a
☒	producer	i-0aaa6e0fa2b2f6e91	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a
☐	test-1	i-0439d2307f52aff85	Stopped	t2.micro	-	View alarms +	ap-south-1a

Fig: B.1.1: Producer and Consumer Instances on EC2

- ii. Starting the container which is for producer module will start flushing the data from the S3 to the Confluent Kafka server every 100th record it reads.

```
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Thu Apr  4 14:25:25 2024 from 103.197.115.178
ubuntu@ip-172-31-47-247:~$ docker ps
CONTAINER ID   IMAGE                                     COMMAND                  CREATED        STATUS
PORTS         NAMES
7e92f236f1c1   .....dkr.ecr.ap-south-1.amazonaws.com/rds_project_images:latest "python producer_mai..." 8 weeks ago   Up 9 min
es
producer_container
ubuntu@ip-172-31-47-247:~$ docker logs producer_container
Producing user data for Topic-RDS_MOVIE_TOPIC
{'UserID': '1', 'MovieID': '1193', 'Rating': '5'}
Producer Flushing...
{'UserID': '1', 'MovieID': '661', 'Rating': '3'}
Producer Flushing...
{'UserID': '1', 'MovieID': '914', 'Rating': '3'}
Producer Flushing...
{'UserID': '1', 'MovieID': '3408', 'Rating': '4'}
Producer Flushing...
{'UserID': '1', 'MovieID': '2355', 'Rating': '5'}
Producer Flushing...
{'UserID': '1', 'MovieID': '1197', 'Rating': '3'}
Producer Flushing...
{'UserID': '1', 'MovieID': '1287', 'Rating': '5'}
Producer Flushing...
{'UserID': '1', 'MovieID': '2804', 'Rating': '5'}
Producer Flushing...
```

Fig.B.1.2: Starting Producer container

- iii. Starting the consumer container inside the consumer instance.

```
System information as of Thu Apr  4 14:26:27 UTC 2024

System load:                0.0
Usage of /:                  20.3% of 28.89GB
Memory usage:                24%
Swap usage:                  0%
Processes:                   99
Users logged in:             0
IPv4 address for br-c2e67b016fe9: 172.18.0.1
IPv4 address for docker0:     172.17.0.1
IPv4 address for eth0:        172.31.46.191

Expanded Security Maintenance for Applications is not enabled.

35 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sat Mar 16 04:35:38 2024 from 13.233.177.5
ubuntu@ip-172-31-46-191:~$ docker ps -a
CONTAINER ID   IMAGE                                     COMMAND                  CREATED        STATUS
PORTS         NAMES
c730c2af31e6   .....dkr.ecr.ap-south-1.amazonaws.com/consumer:latest "python consumer_mai..." 7 weeks ago   Exited (137) 24 minu
es ago
consumer_container
ubuntu@ip-172-31-46-191:~$ docker start consumer_container
consumer_container
ubuntu@ip-172-31-46-191:~$
```

Fig.B.1.3: Starting Consumer Container

- iv. From the consumer instance, the incoming data, initially in object form, undergoes deserialization into JSON dictionary format. Subsequently, each record is enumerated and stored in real-time within the S3 bucket, thereby enhancing data management and accessibility.
- Initial JSON file count S3

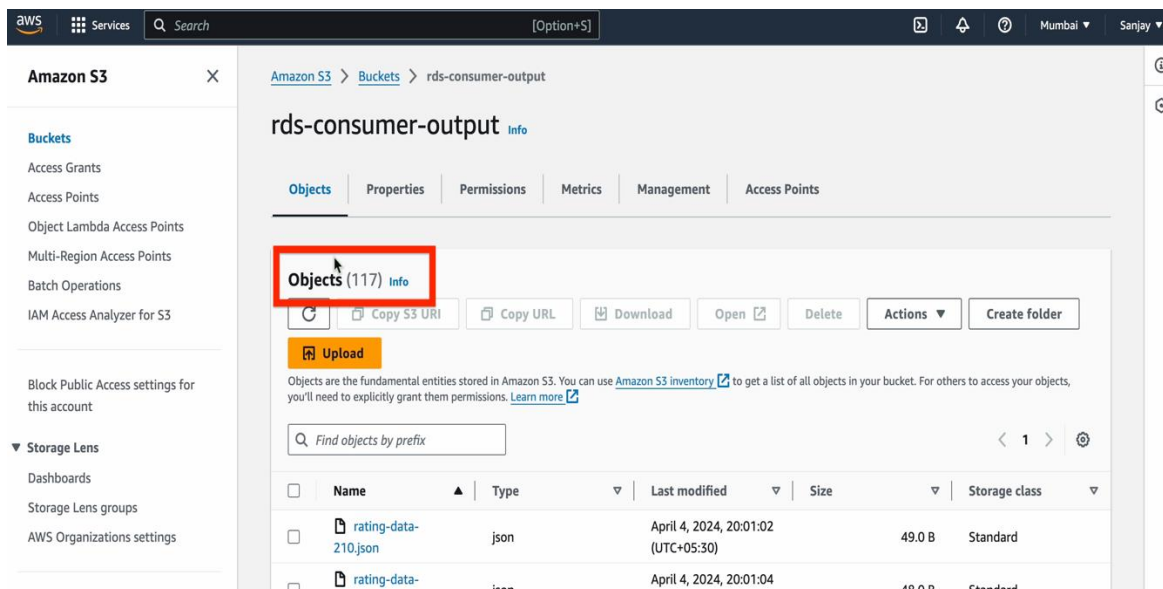


Fig: B.1.4: Real-Time Streaming-1

- Real-time updation of JSON files from consumer EC2 instances.

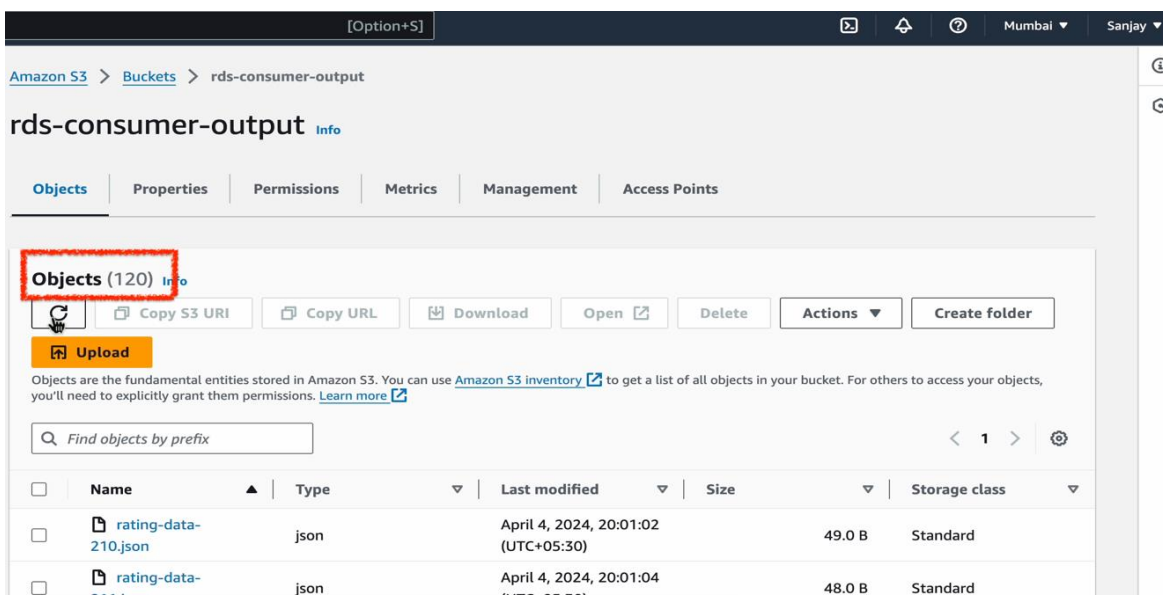


Fig: B.1.5: Real-Time Streaming-2

- v. Utilizing the S3 bucket 'rds-consumer-output', an AWS Glue Crawler is employed to interpret the specified schema and ingest the data. This data is then connected to AWS Athena, enabling seamless querying. Subsequently, AWS QuickSight is utilized to create dynamic dashboards, capable of displaying various visualizations such as pie charts, bar charts, line plots, and candle plots

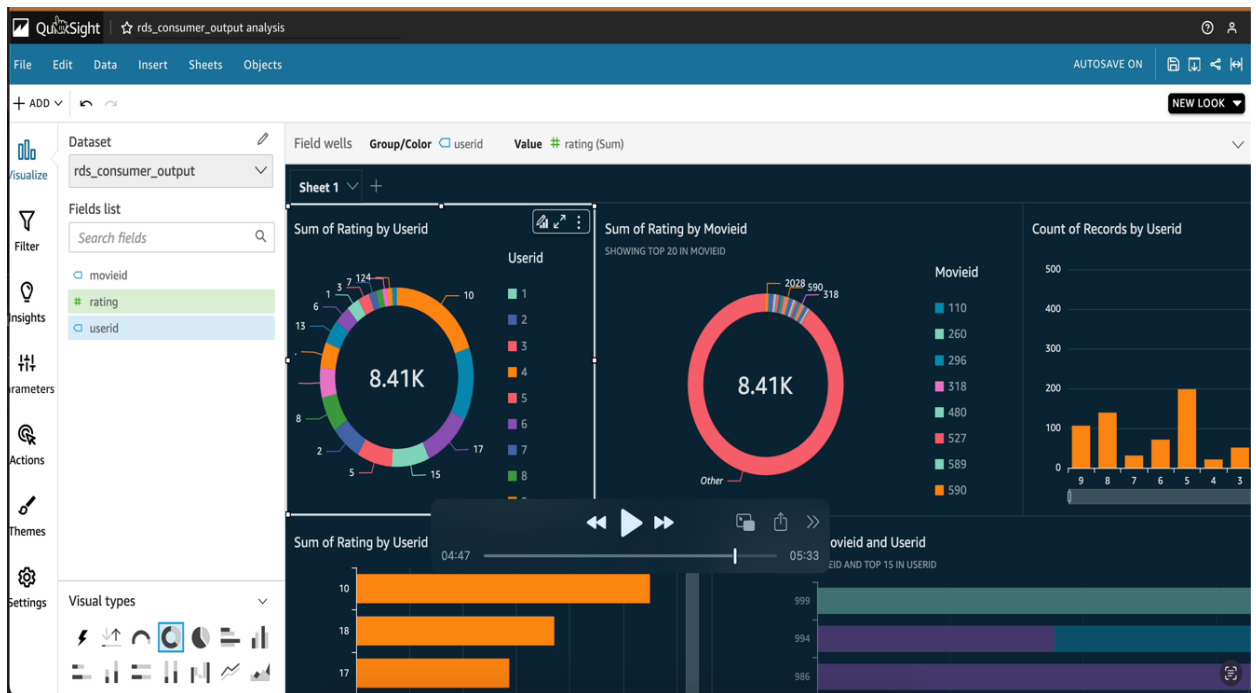


Fig: B.1.6: QuickSight Dashboard (Direct query)

Real-time Dashboarding using Bigdata tools

1st Sanjay R

dept. Computer Science and
Engineering Sathyabama Institute of
Science and Technology
Chennai, India
sanjaysanjay1270@gmail.c
om

2nd Dheeraj Pulakhandam

dept. Computer Science and
Engineering Sathyabama Institute of
Science and Technology
Chennai, India
dhirajpulakhandam@gmail.c
om

3rd Dr. V. Nirmalrani(Professor)

dept. Computer Science and
Engineering Sathyabama Institute of
Science and Technology
Chennai, India
nirmalrani.it@sathyabama.ac
.in

Abstract—Real-time dashboarding software empowers organizations to harness the burgeoning IoT data stream, transforming raw sensor readings into actionable insights through a secure and scalable architecture. Data from connected devices seamlessly flows via lightweight protocols like MQTT and is managed by a robust streaming platform like Kafka. Big data tools like Spark then extract meaning from this high-velocity data, preparing it for visualization on dynamic dashboards hosted by cloud platforms. These user-centric dashboards track key metrics, identify trends, and drive informed decision-making, but tackling challenges like scalability, security, and data quality is crucial to unlock the full potential of this transformative technology.

Keywords—BigData, Real-time processing, Producer, Consumer, Apache Kafka, Amazon Web Services (AWS), Elastic cloud compute(EC2), Data Security, Data Visualization, Docker, Containerization, Continuous Integration and Continuous Deployment workflow(CI/CD), GitHub Actions.

I. INTRODUCTION

As the usage of the Internet of Things (IoT) increases, a new kind of technology arises where vast amounts of real-time data are continuously generated across diverse networks. Because of this large amount of data, the complexity of handling the data also increases, making the traditional analytical tools more inefficient. As organizations use IoT to improve efficiency and gain insights, handling data becomes important. This includes integrating, processing, and visualizing data effectively. But there are challenges. Managing systems like Kafka, ZooKeeper, and Spark involves tasks like maintenance, monitoring, scaling, and optimizing. Configuring these systems to meet our needs is also crucial. And keeping data secure is a big concern too, especially with cyber threats out there. This paper introduces a comprehensive Real-Time Dashboarding System designed to address these challenges, providing a seamless solution for aggregating, analyzing, and visualizing data from a variety of IoT devices within a specific network. Various IoT systems employ the Message Queuing Telemetry Transport (MQTT) protocol, a lightweight and efficient communication protocol, to facilitate data exchange between IoT devices and a central server. The data generated by these devices is transmitted in real time and

ingested by robust streaming platforms like Apache Kafka. This paper delves into the architecture of a real-time dashboarding system that not only efficiently handles the flow of data from IoT devices but also ensures scalability, security, and data quality. Producer and consumer are the instances that contain the API to act as producers and consumers of the confluent Kafka streaming platform, which will be hosted on Elastic Cloud Compute (EC2) as separate virtual machines. These producer and consumer instances will be dockerized and run as containers inside these cloud-hosted virtual machines. Containerization provides a lightweight and efficient way to encapsulate applications and their dependencies, ensuring consistent deployment across different environments. Docker, as a containerization platform, allows the producer and consumer components to run in isolated environments, avoiding conflicts with the underlying infrastructure and simplifying the deployment process. The deployment and continuous integration/continuous deployment (CI/CD) processes in the proposed architecture are facilitated by GitHub Actions by setting up triggers. The streaming data passes through the Kafka streaming platform, it is further processed by leveraging Amazon EMR, and then results can be visualized using Amazon QuickSight with the help of AWS Athena.

II. RELATED WORKS

Many efforts have been made to transform raw data into insightful real-time dashboards. Below are some real-world examples and studies similar to our project.

- Specifically, real-time dashboards have been crafted to observe the disruptive occurrences in the real estate sector, notably amid economic difficulties like the COVID-19 pandemic. This execution was carried out by Balamurugan Soundararaj, Christopher Pettit, and Oliver Lock in the year 2022 (e.g. [3]).
- An interactive dashboard was developed for real-time analytics, offering comprehensive insights into the evolving landscape of the COVID-19 outbreak in India. The adept execution of this initiative can be credited to the collaborative efforts of Arun Mitra, Biju Soman, and Gurpreet Singh in July 2021 (e.g. [4]).

Munawar et al. (2020) proposed a vision wherein the real estate market, shaped by the influence of big data

(e.g.[7], [6]), undergoes a transition into a 'smart real estate' paradigm comparable to the concept of 'smart cities'. Within this framework, the advanced information and analytical capabilities inherent in smart real estate are anticipated to benefit stakeholders significantly. These advantages encompass tailored services for customers, precise property valuation for lenders, streamlined management and transaction processes for owners, enhanced fraud detection and taxation efficiency for government entities, and increased flexibility and risk mitigation for investors.

- Anas Bodor, Meriem Hnida, and Najima Daoudi proposed a model for monitoring and addressing the critical issue of lacking effective monitoring in machine learning projects, posing risks to model quality and sustainability in 2023. It provides a comprehensive guide on metrics and tools within the MLOps framework, emphasizing the importance of continuous monitoring during both the development and production phases [5]. Also, similar work on MLOps was proposed by Dominik Kreuzberger, Niklas Ku"hl, and Sebastian Hirschl (e.g. [1], [5]) implements the best approach and metrics to define the pipeline of the MLOps.
- Velmurugan A, Albert Mayan J, Mohana Prasad R, Yovan Felix A(2019), "Implementing Health Care Center using Hadoop for Analysis and Prediction of Diseases" this proposed system uses BigData tools like Hadoop to analyze and process a large amount of data to predict the diseases [6].
- Mohammad Hasnine, Ho tan Nguyen, Huyen T. T. Bui, Go"kan Akc,apinar, and Hiroshi Ueda banded on developing MOEMO (Motion and Emotion) [9], a real- time learning analytics system. MOEMO uses automated emotion detection from lecture videos, captured by built-in webcams, to measure online learner's affective states of engagement and concentration.
- Fatih Gu"rcan and Muhammet Berigel [10], in collaboration, published a paper titled "Real-Time Processing of Big Data Streams: Lifecycle, Tools, Tasks, and Challenges" in IEEE. The paper defines a lifecycle for real-time big data processing, and associated tools, and frameworks with lifecycle phases. It addresses challenges such as volume, variety, scalability, and privacy in real- time big data processing. The study covers tools like Flume, Kafka, Nifi, Spark Streaming, Hbase, and more, offering valuable insights into the field.

In summary, the compilation of real-world examples and research studies showcased in the related works section underscores the breadth and depth of real-time dashboarding and data analytics applications across various domains. From addressing economic disruptions in real estate to monitoring the dynamic landscape of a global pandemic, these endeavors exemplify the versatility and significance of real-time analytics in today's data-driven world. Furthermore, proposals for transitioning traditional

markets into "smart" paradigms and models for enhancing machine learning project sustain- ability underscore the transformative potential of advanced data analytics techniques.

III. PROPOSED SYSTEM

This part introduces our proposed system, which is carefully designed to tackle the challenges we discussed earlier and achieve our project goals. The system consists of a well-thought-out set of steps, each with its specific role which are mentioned below.

A. Data Ingestion

The initial data is securely stored in Amazon Simple Storage Service (S3) using designated buckets. These buckets are configured with a read-only access policy linked to a specific Identity Access Management (IAM) role. This IAM role is then associated with an Elastic Compute Cloud (EC2) instance responsible for retrieving the raw data from the S3 bucket. The S3 bucket policy explicitly specifies the Amazon Resource Name (ARN) and ID of the user role, ensuring restricted and more granular access. Subsequently, the processed data is sent to Confluent Cloud, where it is retained for over a week. Another EC2 instance is assigned to consume this data and seamlessly provide it to an Amazon Elastic MapReduce cluster (EMR) for further advanced processing. This systematic work- flow ensures secure data handling, efficient processing, and seamless transition between storage and processing platforms.

B. Networking

To set up the network for running these cloud services, we use a customized Virtual Private Cloud (VPC). This is like creating a private space in the cloud where our services can operate. Think of it as building your secure digital playground. Within this VPC, we configure Security Groups, which act like special fences around each instance (like the producer and consumer). These "fences" decide which traffic is allowed in and out. There's also something called an Access Control List (ACL), which is a bit like a list of traffic rules, but it doesn't remember what happened before. On the other hand, Security Groups are more "smart" and remember past interactions, making them more secure. Now, we place our instances and services inside a subnet. It's like dividing our playground into different sections. This can be a default configuration or something we customize based on our needs. Finally, the VPC is connected to an Internet Gateway (IGW), which allows the public traffic to inside the VPC. ACL acts as a gatekeeper to control traffic within the VPC and the subnet. By running the instance inside a private subnet these instances cannot accessed by the public through IGW. This connection can also be set as default, but for added security tailored to user needs, it's better to configure it accordingly. To enable inbound rules in private subnets we need to set up a NAT gateway. In simpler terms, it's like making sure only the right people can get into different parts of our digital playground, ensuring a safe and secure environment for our cloud services.

C. Data Streaming

The partnership between the EC2 instance and the IAM user role ensures the smooth handling of data from diverse IoT devices. To replicate real-time data processing without incurring high costs, we emulate the data retrieval process from S3. The amassed data finds a secure home within Confluent Cloud, automatically retained for a week. Subsequently, another EC2 instance takes charge, meticulously organizing the data for in-depth analysis in Amazon EMR and the creation of insightful dashboards in Amazon QuickSight. This approach optimizes the data-handling process by aggregating it into smaller, manageable batches. Each passing second witnesses the creation and processing of a set of records by the designated instance, achieving a balance between computational capacity and real-time processing requirements. Also, both the producer and consumer code undergo dockerization, ensuring their consistent execution as in the test environment. Using GitHub Actions, continuous updates and automatic deployments are seamlessly orchestrated, enhancing the system's efficiency to meet industry standards. To establish a secure connection with Confluent Cloud, robust security measures, including SASL and SSL protocols, are implemented, ensuring safe and encrypted communication channels between the producer, consumer, and the cloud infrastructure.

D. Data Processing

The EC2-based consumer instance transfers the streaming data into an S3 bucket where each record is stored as a JSON file and the file names are enumerated and we use AWS GLUE to process that data through Amazon Elastic MapReduce (EMR). Within EMR, the data undergoes processing through Apache PySpark, which involves mapping it to relevant datasets to uncover valuable insights. More details about these datasets can be found in the next section of this paper. After the data processing in EMR, each record is uniquely identified by enumerating its name and stored in S3. Following this, the AWS Glue service is connected to S3, offering a crawler that examines the data's structure and creates a catalog for S3. With AWS Glue, users can easily query the data within Amazon Athena using a language similar to Structured Query Language (SQL). This marks the final step in the data processing journey.

E. Finding Patterns and Dashboarding

After processing in Amazon Athena, the data is brought to life in Amazon QuickSight. It's like telling a story with various charts, such as pie charts, line charts, bar charts, and different plots. These charts help in describing and analyzing the data in real-time. The exciting part is that this real-time dashboard can be shared with everyone or limited to a specific group within a company. This sharing flexibility makes their work more efficient as it provides a dashboard showcasing insights from the live raw data, making it easier for everyone to understand and work with

the analytics team to get much more inference from the insights.

IV. DATASET METADATA OVERVIEW

This project utilizes three DAT file format datasets, comprising 1,000,209 ratings from about 3,900 movies by 6,040 users on MovieLens in 2000. Widely employed in education, research, and industry, MovieLens datasets have significant usage in programming books and courses. Originating from the MovieLens movie recommendation system in 1997, these datasets offer insights into user activity. The article provides a concise history of MovieLens, emphasizing lessons learned and best practices, and offering valuable guidance for effective utilization in new research. Cite the dataset using the referenced paper (e.g. [11]). Special thanks to Shyong Lam and Jon Herlocker for their work on the dataset. The GroupLens Research Project at the University of Minnesota's Department of Computer Science and Engineering, led by Professors John Riedl and Joseph Konstan, explores various aspects of information filtering, collaborative filtering, and recommender systems. Commencing in 1992, the project gained recognition in 1996 for a global trial of an automated collaborative filtering system for Usenet news.

A. Rating data file

All the ratings are in the "rating.dat" file containing three fields. Each user has at least 20 ratings.

- **UserID** range between 1 and 6040
- **MovieID** range between 1 and 3952
- **Ratings** are made on a 5-star scale

B. Movie data file

The "movies.dat" file contains movie information in the following format.

- **MovieID** is a unique identifier for each movie.
- **Title** is the movie title, matching those provided by IMDB, including the release year.
- **Genres** of the movie, pipe-separated, selected from a predefined set, including Action, Adventure, Animation, Crime, Documentary, Film-Noir, Horror, Musical, Mystery, Romance, War, and Western.

C. User data file

The "users.dat" file contains user information in the format: UserID, Gender, Age, Occupation, and Zip-code. Demographic information is voluntarily provided by users and is not verified for accuracy. Only users who have provided demographic information are included in this dataset.

- **Gender** is indicated by "M" for male and "F" for female.
- **Age** falls within predefined ranges, including "Under 18", "18-24", "25-34", "35-44", "45-49", "50-55", and "56+".
- **Occupation** field is represented by integers corresponding to specific occupations, with over 20 different occupations included in this field.

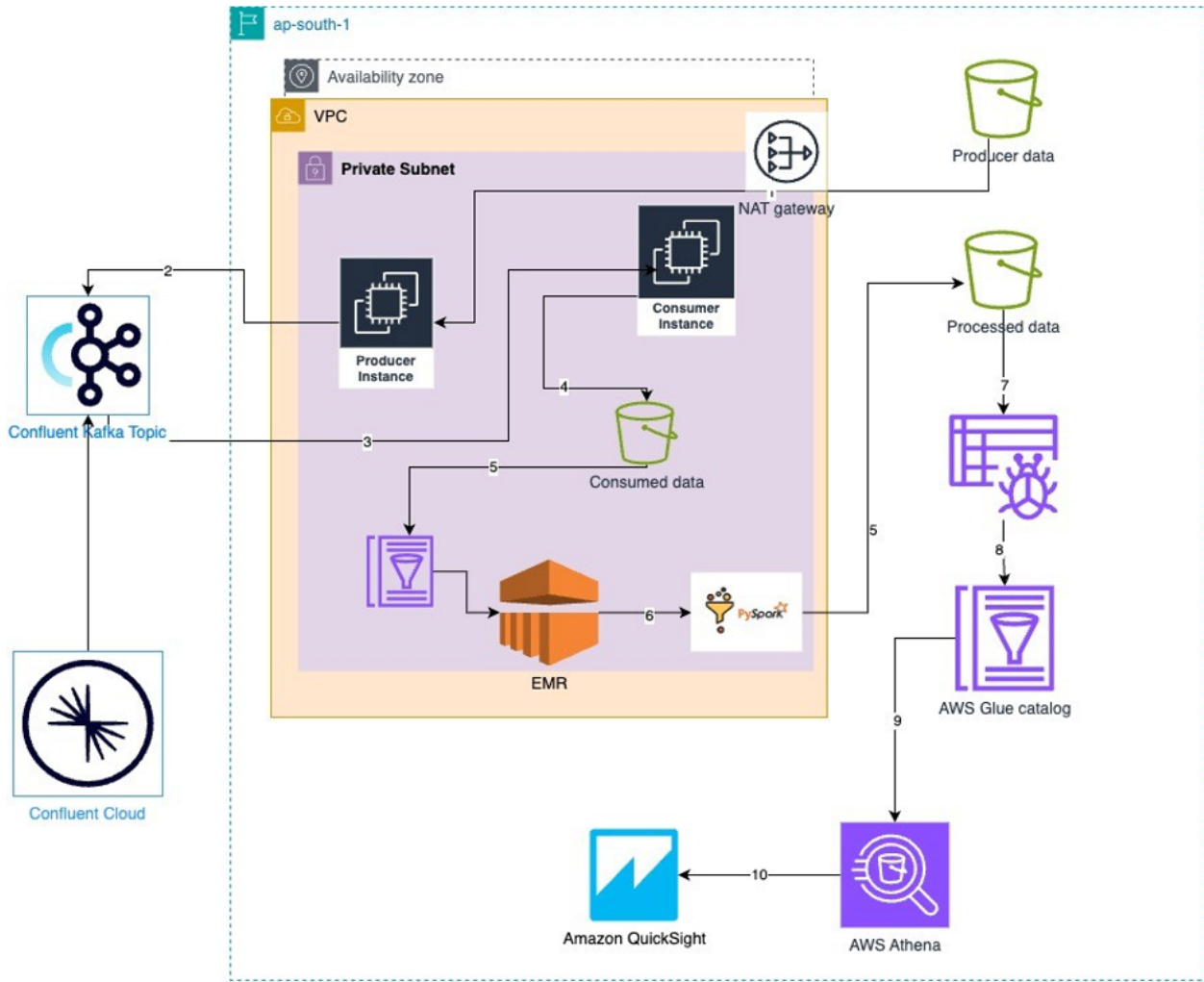


Fig. 1. Architecture and workflow of the real-time dashboarding system.

V. ARCHITECTURE OF PROPOSED SYSTEM

The system's architecture diagram (Fig. 1) shows how different parts work together. It gives a clear picture of how each piece contributes to the overall operation, ensuring things. This diagram helps us understand how the system is structured and how data and processes flow through it. It shows how everything is organized and how each part has a specific role in making the system work. In the AWS Pacific region (ap-south-1), a Virtual Private Cloud (VPC) has been set up with a dedicated private subnet to enhance the security of the application. Both the producer and consumer instances within this subnet operate without public IP addresses. To enable external API connections, a NAT Gateway is employed. However, for cost optimization, the option of switching to a public subnet is available, bearing in mind the trade-offs between security and expenses.

Next, the producer instance efficiently utilizes the NAT gateway to establish a secure connection with Confluent Cloud. It effectively transfers data from the S3 bucket to a specified topic inside the Confluent Cloud which has 6 partitions, enhancing the organization's retrieval process. On the consumer end, data consumption from the specified topic in Confluent Cloud is orchestrated with precision. The communication is safeguarded through the implementation of SASL, and SSL protocols and follows a plain SSL mechanism, ensuring data integrity and security. The structured schema verification adds an extra layer of reliability to the entire data processing pipeline. Following data consumption, the retrieved information undergoes additional processing within Amazon EMR and is subsequently stored in an S3 bucket. To facilitate seamless querying, an Amazon Glue Crawler is employed to create a comprehensive catalog.

VI. CI/CD WORKFLOW

For the continuous integration and deployment of the producer and consumer components, we rely on GitHub Actions for continuous integration and continuous deployment (CI/CD). The CI process runs on an Ubuntu machine provided by GitHub Actions, while the CD process is handled on self-hosted AWS EC2 instances designated for the producer and consumer. Figure 2 illustrates the CI/CD workflow using

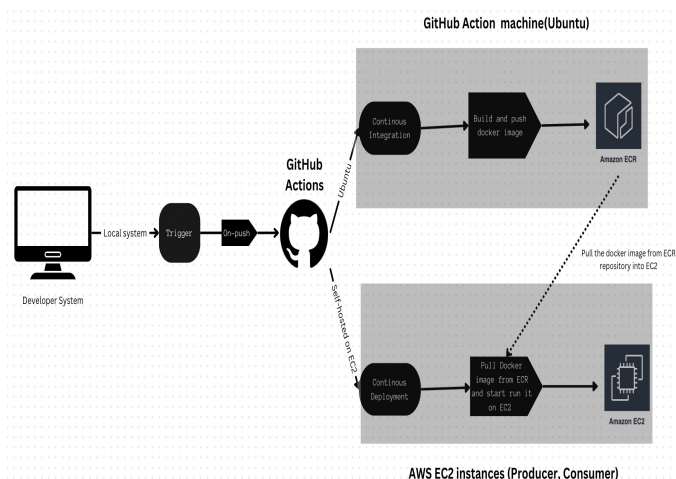


Fig. 2. CI/CD pipeline using GitHub Actions

GitHub Actions. During the continuous integration phase, a Docker image is built and subsequently pushed to a private Amazon Elastic Container Registry (ECR) repository. In the subsequent continuous deployment phase, the Docker image stored in the private ECR repository is pulled onto the designated EC2 instances, where it is then executed within a containerized environment. This streamlined CI/CD pipeline ensures that updates to the producer and consumer components are efficiently tested, built, and deployed in a controlled manner. The visual representation of the CI/CD workflow is depicted below for clarity.

VII. RESOURCE RATIONALIZATION

If prioritizing cost efficiency emerges as a significant concern, it becomes imperative to consider transitioning the private subnet to a public subnet configuration. This transition enables the instances (Producer and Consumer) to obtain direct access to an internet gateway, thereby facilitating the provision of a public IP address that is accessible to a broader audience. While this adjustment offers the advantage of eliminating the necessity for a NAT gateway, consequently reducing associated expenses, it concurrently introduces heightened security implications due to the expanded exposure of the system to the public domain. Therefore, while pursuing cost reduction

measures is undoubtedly important, it is equally crucial to carefully weigh the potential trade-offs and security considerations associated with such architectural adjustments.

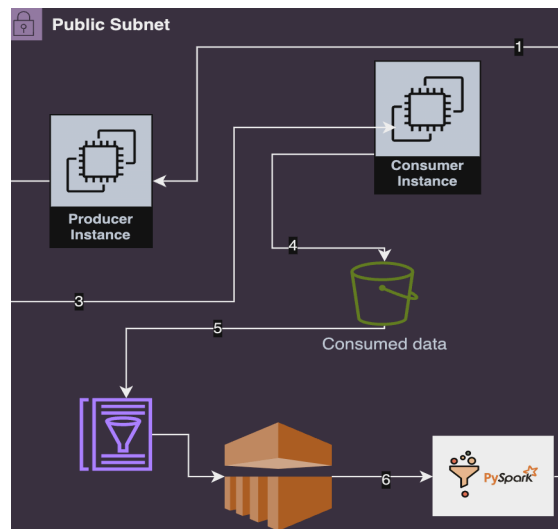


Fig. 3. Instances inside Public Subnet

VIII. COMPARATIVE ANALYSIS

Cloud systems provide unmatched adaptability, effortless scalability, and cost-effective operation, all efficiently managed. By using Confluent Kafka, organizations can unlock a realm of possibilities, including the attainment of a serverless utility model, robust data governance, comprehensive monitoring capabilities with user interface, and liberation from the constraints of native Kafka languages such as Java. Confluent Kafka empowers developers to build innovative solutions using a wide array of programming languages, including C++, Python, and more, thus democratizing the development process and fostering agility in solution creation. With Confluent Kafka at the forefront, organizations can navigate the complexities of modern data architectures with confidence, driving unparalleled efficiency and innovation. In contrast, traditional setups face limitations in scalability, entail manual efforts, and involve higher upfront costs. Leveraging distributed storage, cloud systems outperform traditional solutions, providing global accessibility. Traditional systems often face challenges accessing real-time data, relying instead on data warehouses for information. For instance, when conducting monthly sales analyses, leading to delays in accessing critical insights, potentially compromising decision-making, information processing, and monitoring capabilities. The suggested system leverages dashboarding tools provided within the same cloud platform, such as Amazon QuickSight, enabling direct querying with minimal latency. However, traditional systems may rely on third-party tools like PowerBi and Tableau, which could introduce compatibility challenges and additional complexities. Unlike old-fashioned computer setups that need lots of upkeep and

physical space, cloud systems work differently. They're super safe because they're designed with security in mind right from the start. Plus, they're flexible, so you can quickly change things and make new stuff without waiting around. This means businesses can keep up with changes and come up with new ideas faster. In AWS (Amazon Web Services), they follow something called the Shared Responsibility Model (SRM). This means that while AWS takes care of certain aspects of security, like the infrastructure and data centers, customers also have a responsibility to secure their data and applications. It's like a team effort to keep everything safe. Cloud computing is highly efficient as it eliminates the need for numerous physical machines to be maintained on-premises. And if your business grows, it's easy to get more space online. Also, everyone can work together from anywhere in the world, thanks to the cloud. Overall, cloud systems are way better than the old ones because they're safer, more flexible, and easier to use.

IX. OUTPUT

These are the basic queries that are performed by Amazon QuickSight. By using direct queries, the user or the data analytics team can visualize near real-time insights and make critical business decisions on top of that. This kind of information system contributes significantly to the business group and acts as a decision support system for various fields like finance, sales, marketing, and more. QuickSight enables you to effortlessly analyze sales data, uncover trends, and strategize for improved outcomes. In Amazon QuickSight, we can give permissions more granularly, allowing us to set which group of people can access the real-time dashboards. QuickSight is integrated with Amazon Athena, which is used to query S3 bucket data. By leveraging the Amazon Glue catalog and Athena, QuickSight can utilize these services to create interactive dashboards. In our case, the output from PySpark on Amazon EMR is stored in S3. QuickSight is then used to develop dynamic real-time dashboards. The images below showcase the insights from the queries in QuickSight.

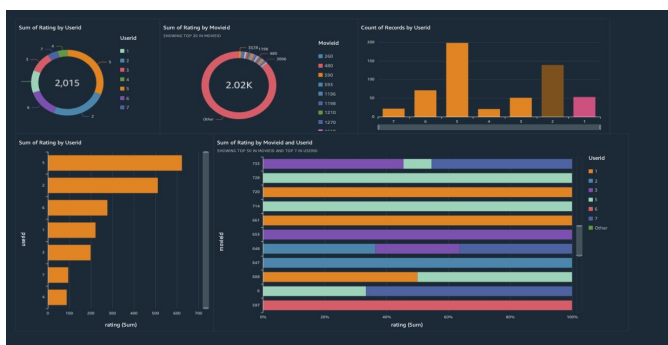


Fig. 4. Real-time insights dashboard on QuickSight

X. CONCLUSION

In summary, our real-time dashboard system, driven by big data and Confluent Kafka, ensures efficiency, security,

and scalability, seamlessly integrating with AWS services. Confluent Cloud integration provides serverless flexibility, extended retention, monitoring tools, data connectors, self-balancing clusters, and a user-friendly cloud UI. Committed to continuous integration, our system is a cost-effective choice for real-time analytics, featuring elastic scaling and simplified management without Zookeeper. However, real-time dashboards present challenges like network latencies and other issues like using direct queries leads to see the real-time data only when the page is refreshed. Additionally, the proposed system doesn't directly utilize real-time data from diverse sensors; instead, it simulates this data by generating our own Confluent Kafka producer. Looking ahead, considering the constraints associated with direct queries, we envision developing a standalone web application. This application will serve as a platform for visualizing real-time data, with the capability to update every second, akin to candle plots used in trading insights of the stock market. These forthcoming initiatives represent the future scope and methodology to overcome existing limitations, thereby augmenting the capabilities of real-time analytics.

XI. ACKNOWLEDGMENT

We extend our gratitude to all those who contributed to this research. We sincerely appreciate the guidance and support provided by our supervisors throughout this study. I want to thank our teacher, Dr. V. Nirmalrani (Professor), our professor, for her invaluable guidance and support throughout this research project. Additionally, we would like to thank our colleagues and peers for their valuable insights and discussions that helped us to enrich this work. Their support has been instrumental in our journey.

REFERENCES

- [1] Machine Learning Operations (MLOps): Overview, Definition, and Architecture proposed by Dominik Kreuzberger, Niklas Ku'hl, and Sebastian Hirschl date of publication 27 March 2023.
- [2] Big Data and Its Applications in Smart Real Estate and the Disaster Management Life Cycle: A Systematic Analysis proposed by Hafiz Suliman Munawar, Siddra Qayyum, Fahim Ullah. Published: 26 March 2020.
- [3] Real-Time Dashboards to Monitor the Impact of Disruptive Events on the Real Estate Market. The case of the COVID-19 Pandemic in Australia was proposed by Balamurugan Soundararaj, Christopher Pettit, and Oliver Lock. Published on 10 June 2022 Volume 2, article number 14, (2022).
- [4] An interactive dashboard for real-time analytics and monitoring of COVID-19 outbreak in India: a proof of concept. Proposed by Arun Mitra, Biju Soman, and Gurpreet Singh in July 2021. Conference: IFIP 9.4 2021, 1st virtual conference. conference theme: resilient ICT4D.
- [5] Machine Learning Models Monitoring in MLOps Metrics and Tools. Proposed by Anas Bodor, Meriem Hnida and Daoudi Najima. December 2023 International Journal of Interactive Mobile Technologies (IJIM) 17(23):125-139.
- [6] Velmurugan A, Albert Mayan J, Mohana Prasad R, Yovan Felix A(2019), "Implementing Health Care Center using Hadoop for Analysis and Prediction of Diseases", International Journal of Engineering and Advanced Technology (IJEAT), Vol.8, No.6, pp.3456-3459. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetism Japan, p. 301, 1982].
- [7] Big Data Analytics of Movie Rating Predictive System. Proposed by Neeta Choudhry, Jianyun Xie, and Xiaoling Xia. The 5th Annual

Inter- national Conference on Information Systems and Artificial Intelligence [ISAI2020] took place on May 22-23, 2020, in Zhejiang, China, as published in Volume 1575 of the Journal of Physics: Conference Series.

- [8] Data visualization techniques for real-time information A custom and dynamic dashboard for analyzing survey results. Proposed by Renato Mauricio Toasa G, Marisa Maximiano, Catarina I. Reis, David Guevara. Published in 2018 on the 13th (CISTI).
- [9] A Real-Time Learning Analytics Dashboard for Automatic Detection of Online Learners' Affective States. Mohammad Nehal Hasnine. Published on Research Center for Computing and Multimedia Studies, Hosei University at Tokyo 102-8160, Japan.
- [10] Real-Time Processing of Big Data Streams: Lifecycle, Tools, Tasks, and Challenges. Proposed by Fatih Gurcan and Muhammet Berigel. October 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT).
- [11] The MovieLens Datasets: History and Context. Proposed by F. Maxwell Harper and Joseph A. Konstan. Volume 5, Issue 4 of ACM Transactions on Interactive Intelligent Systems.
- [12] Albert Mayan, J, Yovan Felix, A, Chintalapudi M, Balaji R, and Vigneswaran R (2015) presented a paper titled "Fault-Open Minded Resource Allocation Rescheduling Algorithm and Expense Minimization for Cloud Systems" in the Global Journal of Pure and Applied Mathematics, Volume 11, Issue 6, pages 4111-4117.
- [13] Hu and Liu (2004) presented a study titled "Extraction and Condensation of Customer Feedback" at the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, held in Seattle, WA, USA, in August 2004.
- [14] Alsager and Sasi (2017) conducted research titled "Summarization and Sentiment Analysis of Movie Reviews Using RapidMiner" at the 2017 International Conference on Networks and Advances in Computational Technologies (NetACT), held in Thiruvananthapuram, India, in July 2017.
- [15] Liu, C.-Y., Chen, M.-S., and Tseng, C.-Y. (Publisher: IEEE). IncreSTS: Towards Real-Time Incremental Short Text Summarization on Comment Streams from Social Network Services



7th International Conference on
Inventive Computation Technologies
ICICT 2024
24-26, April 2024 ~ Lalitpur, Nepal

icict.conf@gmail.com
www.icictb.com/2024

Letter of Acceptance

Author Name: Sanjay R, Dheeraj Pulakhandam, V. Nirmalrani

Affiliation Details: Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India

Dear Author

It is with great pleasure that we extend our warmest congratulations to you on the acceptance of the paper titled "**Real-time Dashboarding using Bigdata tools**"- **PAPER ID: ICICT-440** for presentation at the 7th International Conference on Inventive Computation Technologies, scheduled to be held in Lalitpur, Nepal, from April 24th to April 26th, 2024.

Your submission was subjected to a rigorous review process, and the result that your paper has been selected for inclusion in our conference program. We believe that your contribution will greatly enrich the discussions and knowledge exchange at our event.

Your participation will undoubtedly contribute to the success of the 7th International Conference on Inventive Computation Technologies.

Once again, congratulations on your acceptance, and we anticipate your valuable contribution to our conference.




Prof. Dr. Subarna Shakya
Conference Chair - ICICT 2024

Proceedings by


Acceptance Letter from ICICT 2024

CONFYY
GSTIN - 33AALFC6457C1ZL

Invoicing and payments
powered by

Payment Receipt Transaction Reference: pay_Nib7t7C8R2IomE

This is a payment receipt for your transaction on ICICT

AMOUNT PAID ₹ 9,750.00

ISSUED TO
sanjaysanjay1270@gmail.com
+919566044840

PAID ON
5 Mar 2024

DESCRIPTION	UNIT PRICE	QTY	AMOUNT
Nepal Indian Authors	₹ 9,750.00	1	₹ 9,750.00
	Total		₹ 9,750.00
	Amount Paid		₹ 9,750.00

Payment Receipt

SESSION - 8

PARALLEL SESSION - 3

Date: 26-04-2024

Time: 9.15 AM to 11.30 AM

Paper ID	Paper Title	Author(s)	Time Slot
ICICT 544	Discover Image Restoration: Analysis and Novel Architecture Proposal	Poonam Pawar, Bharati Ainapure	09:15 am to 09:35 am
ICICT 440	Real-time Dashboarding using Bigdata tools	Sanjay R, Dheeraj Pulakhandam, V. Nirmalrani	09:35 am to 09:55 am
ICICT 561	Unveiling the Potential: Homomorphic Encryption and its Applications in Multi-Cloud Security	Aahash Kamble, Moses Makuei Jiet, Chetan Puri	09:55 am to 10:15 am
		Sridevi Chitti, Sreeja Reddy Nandikonda	10:15 am

Conference Schedule