

Project Report
on
“Real Time Data Streaming Application”

CPSC 531-03 22470

Advance Database Management

Fall, 2022

Under Guidance Of

Prof. Tseng-Ching James Shen

Department of Computer Science



California State University

Fullerton CA – 92831

December 2022

Prepared By:-

Sambhaji Ippar(885865899) - sambhaji@csu.fullerton.edu

Onkar Muttemwar(885199950) - onkar.muttemwar@csu.fullerton.edu

Aman Rathore(885186841) - aman.r@csu.fullerton.edu

Contents

Introduction	3
Functionalities	3
Architecture Overview	4
Technologies and tools used	5-6
Project skills needed but not limited to	6
Dataset	6-7
GitHub Location of Code	7
Deployment Instructions	8-15
Steps to Run the Application	16
Test Results	16-17
References	18

Introduction

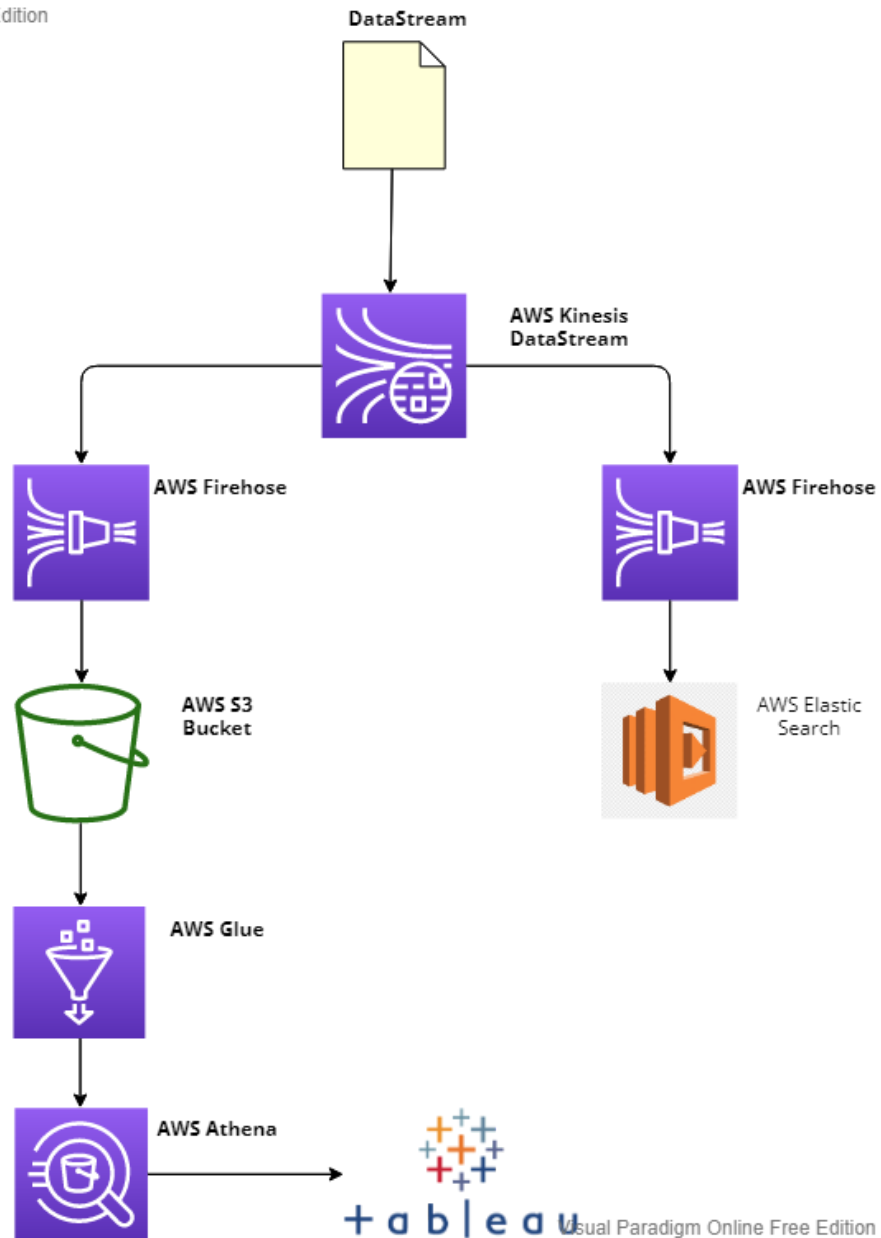
- Real-time Data streaming is a process by which big volumes of data are processed as soon as they are generated as continuous streams.
- According to reports, more than a quarter of the data created would be real-time. Many data sources create each type of data, including IoT sensors, smart devices, and gaming applications producing data at high volumes and velocities.
- So there is a need to process this data in real-time as there is some business where real-time processing and analytics are crucial to getting an edge over competitors. Also, it enables faster decision-making along with various other advantages.
- Detecting fraud in real-time, ride-share, and e-commerce apps are important examples of real-time processing.

Functionalities

- Producers rapidly simulate the data stream, which AWS Kinesis ingests in real time.
- The ingested data in AWS Kinesis is partitioned using shards and sent to Firehose.
- The two connected AWS Firehose ingested the data from Kinesis. One continuously loads the real-time data to Elasticsearch or OpenSearch, while the other loads the raw data into our S3 bucket.
- The AWS Elasticsearch or OpenSearch monitors our data and produces visual insights using Kibana.
- The raw data within the S3 bucket is transformed by glue crawlers and moved to Athena.
- The AWS Athena is then querying the data and producing the visual insights in Tableau, which is connected to Athena server.

Architecture Overview:

Visual Paradigm Online Free Edition



- The data Stream is a python program simulating the data in real time.
- The output data stream is ingested by DataStream, which KinesisFirehose then processes.
- OpenSearch then uses it for real-time analysis using Kibana

- S3 uses the other Firehose for Batch processing which is then used to add data in glue tables and analysis using Athena and Tableau

Technologies and tools used:

1. Python



2. Tableau



3. AWS Kinesis



4. Amazon Kinesis



5. AWS S3



6. AWS Glue



7. AWS Athena



8. AWS ElasticSearch



Project skills needed but not limited to:

To work on the project, one must have the following skills but not limited to.

- Having a basic understanding of cloud computing.
- Experience with the AWS platform and its various services.
- Knowledge of python programming or, as an alternative, creating a producer code using JAVA.
- Understanding of security best practices and how they apply to the cloud.

Dataset:

- The architecture of the application is such that it would work on any real-time data set. But for the project's scope, we have used the Bank Marketing Dataset from Kaggle.
- The dataset has a lot of columns from which we can extract many insights that would be extracted to analyze in real-time.
- The dataset can be downloaded from Kaggle

<https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset>

1	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit	
2	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes	
3	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes	
4	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes	
5	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes	
6	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes	
7	42	managemer	single	tertiary	no	0	yes	yes	unknown	5	may	562	2	-1	0	unknown	yes	
8	56	managemer	married	tertiary	no	830	yes	yes	unknown	6	may	1201	1	-1	0	unknown	yes	
9	60	retired	divorced	secondary	no	545	yes	no	unknown	6	may	1030	1	-1	0	unknown	yes	
10	37	technician	married	secondary	no	1	yes	no	unknown	6	may	608	1	-1	0	unknown	yes	
11	28	services	single	secondary	no	5090	yes	no	unknown	6	may	1297	3	-1	0	unknown	yes	
12	38	admin.	single	secondary	no	100	yes	no	unknown	7	may	786	1	-1	0	unknown	yes	
13	30	blue-collar	married	secondary	no	309	yes	no	unknown	7	may	1574	2	-1	0	unknown	yes	

GitHub Location of Code:



Click Logo

Deployment:

AWS Infrastructure

1. Log in to the AWS portal

2. Creating a Data Stream

- a. After logging in to the AWS portal, search Kinesis and open the Kinesis Portal. Now click on the Data Stream tab to create the Data Stream.
- b. This process is relatively straightforward; there are two capacity modes: on-demand and provisioned. On-demand helps us scale the data stream automatically, and provisioned mode uses an initial set of set resources.

Data stream capacity [Info](#)

Capacity mode

☒ **On-demand**
Use this mode when your data stream's throughput requirements are unpredictable and variable. With on-demand mode, your data stream's capacity scales automatically.

☐ **Provisioned**
Use provisioned mode when you can reliably estimate throughput requirements of your data stream. With provisioned mode, your data stream's capacity is fixed.

Total data stream capacity
By default, data streams with on-demand mode scale throughput automatically to accommodate traffic of up to 200 MiB per second and 200,000 records per second for the write capacity. If traffic exceeds capacity, your data stream will throttle.

Write capacity
Maximum
200 MiB/second and 200,000 records/second

Read capacity
Maximum (per consumer)
400 MiB/second
Up to 2 default consumers. Use Enhanced Fan-Out (EFO) for more consumers. EFO supports adding upto 20 consumers, each having a dedicated throughput.

8

Amazon Kinesis > Data streams > Create data stream

Create data stream [Info](#)

Data stream configuration

Data stream name

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens and periods.

Data stream capacity [Info](#)

Capacity mode

☒ **On-demand**
 Use this mode when your data stream's throughput requirements are unpredictable and variable. With on-demand mode, your data stream's capacity scales automatically.

☐ **Provisioned**
 Use provisioned mode when you can reliably estimate throughput requirements of your data stream. With provisioned mode, your data stream's capacity is fixed.

Total data stream capacity
 By default, data streams with on-demand mode scale throughput automatically to accommodate traffic of up to 200 MB per second and 200,000 records per second for the write capacity. If traffic exceeds capacity, your data stream will throttle.

Write capacity
 Maximum
 200 MB/second and 200,000 records/second

Read capacity
 Maximum (per consumer)
 400 MB/second
 Up to 2 default consumers. Use [Enhanced Fan-Out \(EFO\)](#) for more consumers. EFO supports adding up to 20 consumers, each having a dedicated throughput.

☒ On-demand mode has a pay-per-throughput pricing model. [See Kinesis pricing for on-demand mode](#)

Data stream settings
 You can edit the settings after the data stream has been created and is in the active status.

Setting	Value	Editable after creation
Capacity mode	On-demand	Yes
Data retention period	1 day	Yes
Server-side encryption	Disabled	Yes
Monitoring enhanced metrics	Disabled	Yes
Tags	-	Yes

[Cancel](#) [Create data stream](#)

3. Creating a Delivery Stream

- After Data Stream is created, click on the Data Stream inside Kinesis to create the new Delivery Stream.
- There are two delivery streams to be created: one for the real-time scenario, which is used for ElasticSearch, and another for the batch flow, and S3 is the Destination.

Delivery streams (2)

Delete

Create delivery stream

Find delivery streams

< 1 >

	Name	Status	Creation...	Source	Data tr...	Destina...	Destination
	delivery-stream	Active	November...	data-strea...	processDa...	Amazon ...	ed-domain
	Kinesis-data-stream-to-S3	Active	December...	data-strea...	mykinesis...	Amazon S3	kinisestos3bucket007

- The configuration of the first delivery stream is shown in the image
 - 1) The source would be the Data Stream we just created

- 2) Transform records are enabled, which uses Amazon Lambda to transform the data during the process.
- 3) The destination is an OpenSearch domain similar to the cluster, with the following Cluster and Security configuration.
- 4) We have added an S3 location as a backup, and the error logs will be saved in the S3 location that was created.
- 5) Delivery Stream 2 is similar to the first delivery stream. Only the destination is S3 in the second delivery stream.

**** Use the following configuration while creating the services**

ElasticSearch Domain Cluster configuration

Cluster configuration	Security configuration	Cluster health	Instance health	Auto-Tune	Logs	Indices	Tags	Connections	VPC endpoints	Packages	Notifications			
<div>Cluster configuration Edit</div> <table> <tr> <td> Data nodes Availability Zones 1-42 Instance type t3.small.search Number of nodes 1 Storage type EBS EBS volume type General Purpose (SSD) - gp3 EBS volume size 10 GB Provisioned IOPS 3000 IOPS Provisioned Throughput (IOPS) 125 MB/s Custom endpoint Enabled No </td> <td> Dedicated master nodes Enabled No Warm and cold data storage UltraWarm data nodes enabled No Snapshot Frequency Hourly Start hour 00:00 UTC (default) </td> <td> Network Access Public Advanced cluster settings Allow references to indices inside the body of HTTP requests. Yes Fielddata cache allocation 20 Max clause count 1024 </td> </tr> </table>												Data nodes Availability Zones 1-42 Instance type t3.small.search Number of nodes 1 Storage type EBS EBS volume type General Purpose (SSD) - gp3 EBS volume size 10 GB Provisioned IOPS 3000 IOPS Provisioned Throughput (IOPS) 125 MB/s Custom endpoint Enabled No	Dedicated master nodes Enabled No Warm and cold data storage UltraWarm data nodes enabled No Snapshot Frequency Hourly Start hour 00:00 UTC (default)	Network Access Public Advanced cluster settings Allow references to indices inside the body of HTTP requests. Yes Fielddata cache allocation 20 Max clause count 1024
Data nodes Availability Zones 1-42 Instance type t3.small.search Number of nodes 1 Storage type EBS EBS volume type General Purpose (SSD) - gp3 EBS volume size 10 GB Provisioned IOPS 3000 IOPS Provisioned Throughput (IOPS) 125 MB/s Custom endpoint Enabled No	Dedicated master nodes Enabled No Warm and cold data storage UltraWarm data nodes enabled No Snapshot Frequency Hourly Start hour 00:00 UTC (default)	Network Access Public Advanced cluster settings Allow references to indices inside the body of HTTP requests. Yes Fielddata cache allocation 20 Max clause count 1024												

ElasticSearch Domain Security configuration

Security configuration

Edit

<div>Fine-grained access control</div> <div>Enabled</div> <div>Yes</div> <div>Master user type</div> <div>Internal user database</div>	<div>Authentication for OpenSearch Dashboards/Kibana</div> <div>SAML enabled</div> <div>No</div> <div>Cognito enabled</div> <div>No</div> <div>Region</div> <div>US East (N. Virginia)</div>	<div>Encryption</div> <div>Required HTTPS</div> <div>Yes</div> <div>Node-to-node encryption</div> <div>Yes</div> <div>Encryption at rest</div> <div>Yes</div> <div>AWS KMS key</div> <div> arn:aws:kms:us-east-1:541077676580:key/f6841e00-a20e-4f4b-ba68-c7d6375a358c</div>
--	--	--

Access policy [Info](#)

Policy

```
[
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": [
          "AWS:*"
        ],
        "Action": "ec:*",
        "Resource": "arn:aws:ec2:us-east-1:541077676580:domain/*ed-domain/*"
      }
    ]
  }
]
```

[Copy policy code](#)

Delivery Stream1 (Part A)

Source settings

Kinesis data stream

[data-stream](#)

Transform records

Edit

Configure Kinesis Data Firehose to transform your record data.

Transform source records with AWS Lambda [Info](#)

Data transformation

Enabled

Description

-

Buffer size

1 MIB

Lambda function

[processDataFirehose](#)

Lambda function version

\$LATEST

Runtime

nodejs18.x

Timeout

1 minute

Destination settings [Info](#)

Edit

Specify the destination settings for your delivery stream.

OpenSearch Service destination

Domain

[ed-domain](#)

Index

index1

Index rotation

No rotation

Retry duration

300 seconds

Buffer hints

Buffer size

5 MIB

Buffer interval

300 seconds

Delivery Stream1 (Part B)

Backup settings [info](#)

Enabling source record backup ensures that source records can be recovered if record processing transformation does not produce the desired results.

Backup mode

Failed data only

Buffer size

5 MB

Encryption for data records

Not enabled

S3 backup bucket

mydbbackup202907 [↗](#)

Buffer interval

60 seconds

Compression for data records

Not enabled

S3 backup bucket prefix

[🗑](#) ex/

S3 backup bucket error output prefix

[🗑](#) ex/

Server-side encryption (SSE) [info](#)

You can use AWS Key Management Service (KMS) to create and manage Customer Master Keys (CMKs) and to control the use of encryption across a wide range of AWS services in your applications.

📘

To enable SSE for the delivery stream, view the data stream selected above, and enable SSE on it.

Destination error logs [info](#)

Choose Enabled if you want Kinesis Data Firehose to log record delivery errors to CloudWatch Logs.

Amazon CloudWatch error logging

Enabled

Permissions [info](#)

Kinesis Data Firehose uses this IAM role for all the permissions that the delivery stream needs. To specify different roles for the different permissions, use the API or the CLI.

IAM role

KinesisDataFirehoseServiceRole-delivery-stre-us-east-1-1669680908488 [↗](#)

Delivery Stream2 (Part A)

Source settings

Kinesis data stream

data-stream [↗](#)

Transform and convert records [info](#)

Configure Kinesis Data Firehose to transform and convert your record data.

Transform source records with AWS Lambda [info](#)

Data transformation

Enabled

Description

An Amazon Kinesis Firehose stream processor that accesses the records in the input and returns them with a processing status.

Buffer size

1 MB

Lambda function

mykinesisinput [↗](#)

Lambda function version

SLATEST

Runtime

nodejs14.x

Timeout

12 minutes 10 seconds

Convert record format [info](#)

Record format conversion

Not enabled

Destination settings [info](#)

Specify the destination settings for your delivery stream.

Amazon S3 destination

S3 bucket

kinisestos3bucket007 [↗](#)

S3 bucket error output prefix

-

Dynamic partitioning [info](#)

Dynamic partitioning

Not enabled

Deaggregation delimiter

-

Multi record disaggregation

Not enabled

New line delimiter

Not enabled

Multi record disaggregation type

-

Inline parsing for JSON

Not enabled

S3 bucket prefix

-

Buffer hints

Buffer size

5 MB

Buffer interval

300 seconds

Compression and encryption

Compression for data records

Not enabled

Encryption for data records

Not enabled

Dynamic partitioning retry

Retry duration

-

Delivery Stream2 (Part B)

Backup settings [info](#) Edit
Enabling source record backup ensures that source records can be recovered if record processing transformation does not produce the desired results.

Source record backup in Amazon S3
Not enabled

Server-side encryption (SSE) [info](#) Edit
You can use AWS Key Management Service (KMS) to create and manage Customer Master Keys (CMKs) and to control the use of encryption across a wide range of AWS services in your applications.

[i](#) To enable SSE for the delivery stream, view the data stream selected above, and enable SSE on it.

Destination error logs [info](#) Edit
Choose Enabled if you want Kinesis Data Firehose to log record delivery errors to CloudWatch Logs.

Amazon CloudWatch error logging
Enabled

Permissions [info](#) Edit
Kinesis Data Firehose uses this IAM role for all the permissions that the delivery stream needs. To specify different roles for the different permissions, use the API or the CLI.

IAM role
[KinesisFirehoseServiceRole-Kinesis-data-us-east-1-1669947217310](#)

Tags (0) [info](#) Manage tags
You can add tags to organize your AWS resources, track costs, and control access.

Key	Value
No tags	
No tags associated with this stream.	
Manage tags	

4. Creating a Lambda Function

- After Delivery Stream is created, search for Lambda, click on create a new function and then create and deploy the function.

Go to Anything (Ctrl-P)

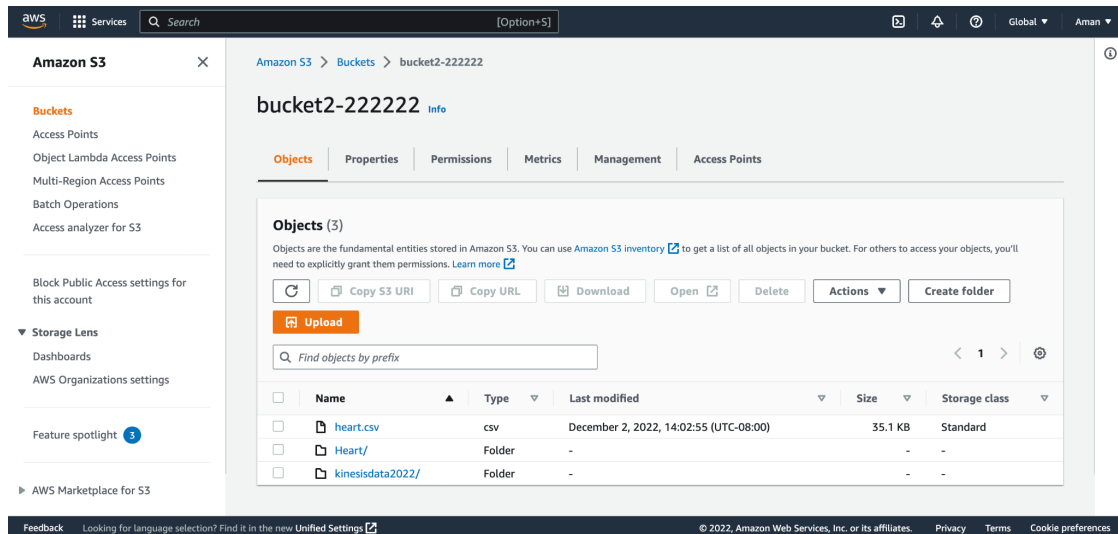
processDataFirehose
index.mjs

```
1  // ...
2  // ...
3  // ...
4  let success = 0; // Number of valid entries found
5  let failure = 0; // Number of invalid entries found
6  /* Process the list of records and transform them */
7  const output = event.records.map((record) => {
8    // Kinesis data is base64 encoded so decode here
9    console.log(record.recordId);
10   const payload = (Buffer.from(record.data, 'base64')).toString('ascii');
11   console.log('Decoded payload:', payload);
12   // Split the data into it's fields so we can refer to them by index
13   const match = payload.split('|');
14
15   if (match) {
16     /* Prepare JSON version from Syslog log data */
17     const result = {
18
19       // build all fields from array
20       Age: match[0],
21       Job: match[1],
22       Marital: match[2],
23       Education: match[3],
24       Defaults: (match[4]),
25       Balance: parseInt(match[5]),
26       Housing: (match[6]),
27       Loan: (match[7]),
28       Contact: (match[8]),
29       Day: parseInt(match[9]),
30       Month: (match[10]),
31       Duration: (match[11]),
32       Campaign: (match[12]),
33       PDays: parseInt(match[13]),
34       Previous: parseInt(match[14]),
35       Poutcome: (match[15]),
36       Deposit: (match[16])
37     };
38     console.log('Result:', payload);
39     success++;
40     return {
41       recordId: record.recordId,
42       result: 'OK',
43       data: (Buffer.from(JSON.stringify(result))).toString('base64'),
44     };
45   } else {
46     /* Failed event, notify the error and leave the record intact */
47     failure++;
48     return {
49       recordId: record.recordId,
50       result: 'ProcessingFailed',
51       data: record.data,
52     };
53   }
54 });
55 console.log('Processing completed. Successful records ${success}, Failed records ${failure}.');
56 callback(null, { records: output });
```

1/1 JavaScript Spaces: 4

5. Creating S3 storage

- After Lambda Function is Created, search for Lambda, click on create a new function and then create and deploy the function.



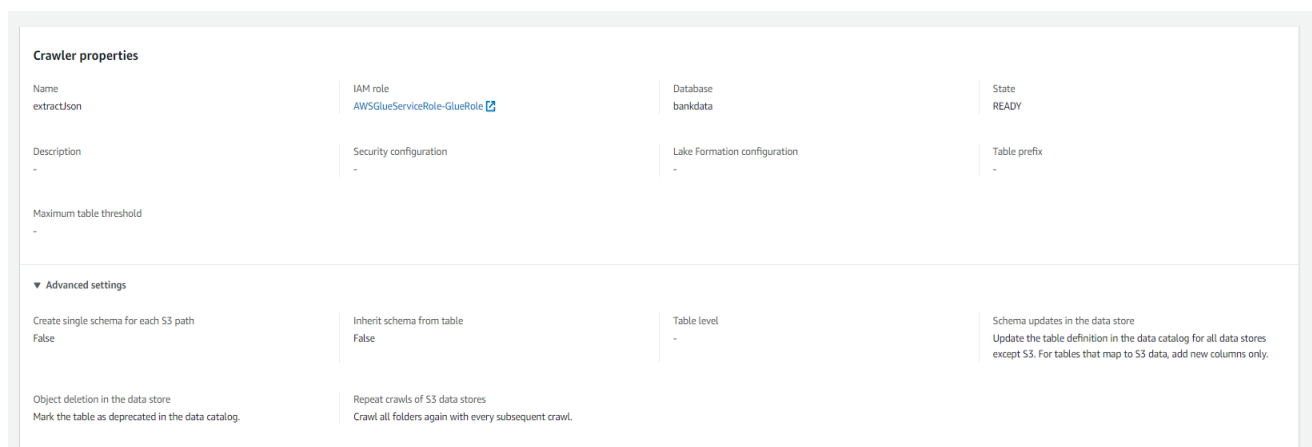
6. Finish creating the delivery stream

- After S3 and Lambda are created, we can finish creating the delivery stream.

7. Create a Glue Crawler

- Create a new Glue Crawler with the following configuration to create a GLUE database and tables.

Glue Crawler



Glue Database

The screenshot displays the AWS Glue console interface. On the left, a navigation menu includes sections for 'Data Catalog' (Databases, Tables, Stream schema registries, Schemas, Connections, Crawlers, Classifiers, Catalog settings), 'Data Integration and ETL' (AWS Glue Studio, Jobs, Interactive Sessions, Notebooks, Data classification tools, Sensitive data detection, Record Matching, Triggers, Workflows, Blueprints, Security configurations), and 'Legacy pages' (What's New). The main content area is titled 'AWS Glue > Tables > View table details' and shows the details for a table named '2022'. The page was last updated on December 2, 2022, at 22:09:05 (UTC). The 'Table overview' tab is active, displaying a table with columns: Name (2022), Description (-), Database (bankdata), Classification (json), Location (s3://kinesis3bucket007/2022/), Connection (-), Deprecated (-), and Last updated (December 2, 2022 at 22:09:05). Below this, the 'Advanced properties' section shows Input format (org.apache.hadoop.mapred.TextInputFormat), Output format (org.apache.hadoop.hiveql.io.HiveIgnoreKeyTextOutputFormat), and Serde serialization lib (org.apache.hadoop.hiveql.io.HiveIgnoreKeyTextOutputFormat). The 'Schema' tab is also visible, showing a table with columns: #, Column name, Data type, Partition key, and Comment. The table has two rows: 1 (age, string, -, -) and 2 (job, string, -, -).

Table overview | Data quality

Name	Description	Database	Classification
2022	-	bankdata	json

Location	Connection	Deprecated	Last updated
s3://kinesis3bucket007/2022/	-	-	December 2, 2022 at 22:09:05

Input format	Output format	Serde serialization lib
org.apache.hadoop.mapred.TextInputFormat	org.apache.hadoop.hiveql.io.HiveIgnoreKeyTextOutputFormat	org.apache.hadoop.hiveql.io.HiveIgnoreKeyTextOutputFormat

Schema | Partitions | Indexes

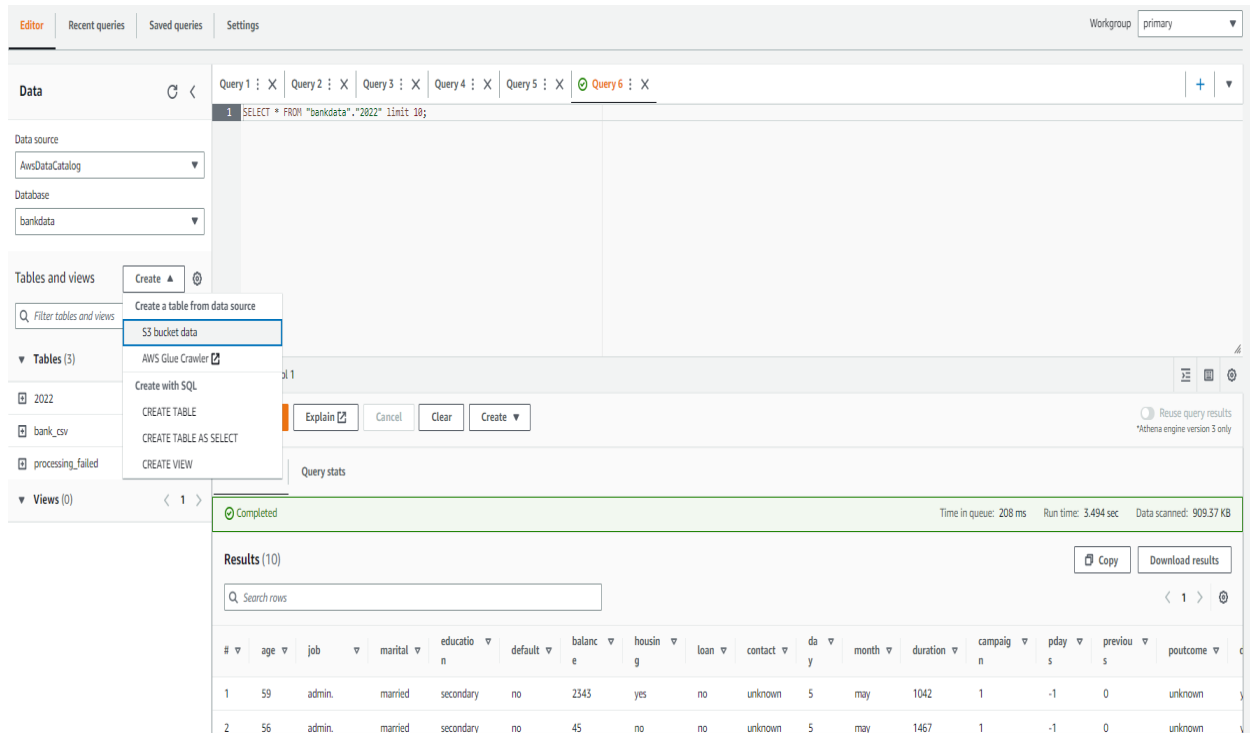
Schema (20)
View and manage the table schema.

Filter schemas

#	Column name	Data type	Partition key	Comment
1	age	string	-	-
2	job	string	-	-

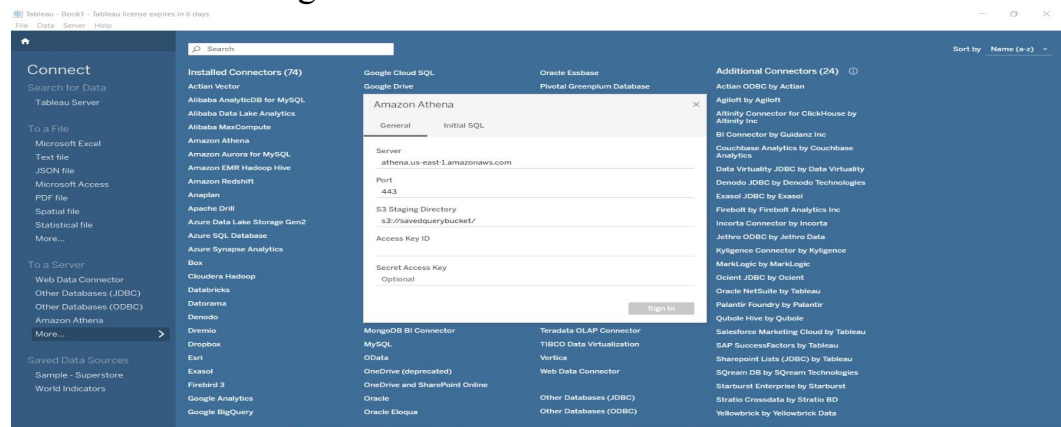
8. Setting up Amazon Athena

- After Creating AWS Glue Crawler, we need to search for Athena
- After a portal similar to the below image is opened, we need to click on AWS Glue Crawler and select the already created Glue crawler.



9. Setting up Tableau

- After completing all the above steps, we need to install tableau and use the below configuration to connect Tableau to the Glue tables.



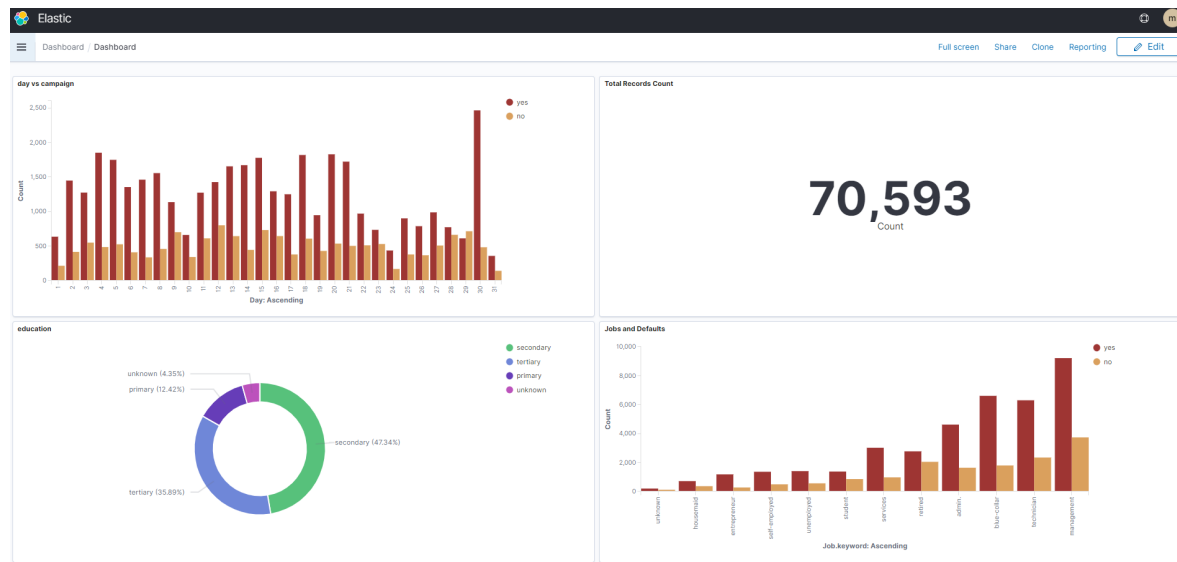
Steps to Run the Application

- 1) Download AWS CLI from <https://AWS.amazon.com/cli/>
- 2) Create an IAM role and give administrator access (for development) or give suitable access and note down the Access Key ID and the Access key
- 3) Open the Terminal(Command Prompt) and navigate to the project directory.
- 4) Type AWS configure and add the previously noted AWS Key ID and AWS Key, region, and output format.
- 5) Now set up the infrastructure in AWS, as explained above in the deployment stage.
- 6) After AWS has been set up, run the python file, which has the producer code, and will start adding the data into the AWS Data Streams and eventually in the AWS Firehose.
 - * Run **python <Python File name>.py**
 - * Generator and Amazon Lambda codes can be found in the GitHub repository.
- 7) To view the Dashboard, we have to click on the Kibana URL inside the Domain, which we will find inside the created Delivery Stream.

Test Results

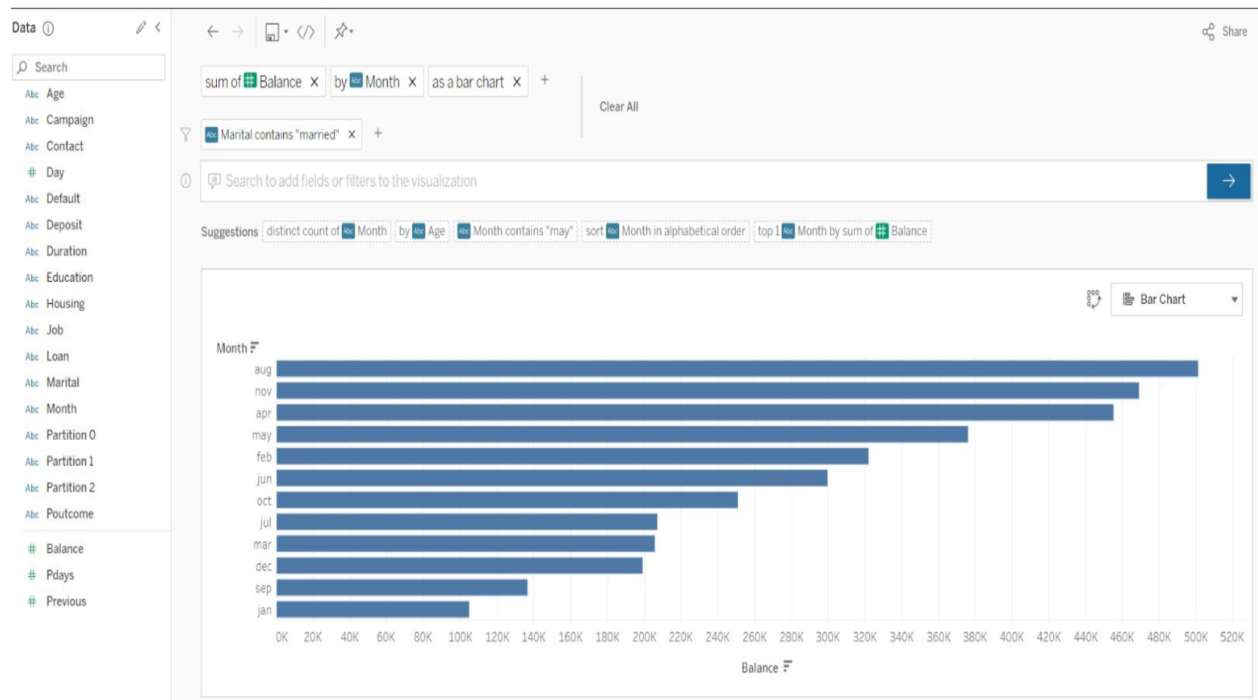
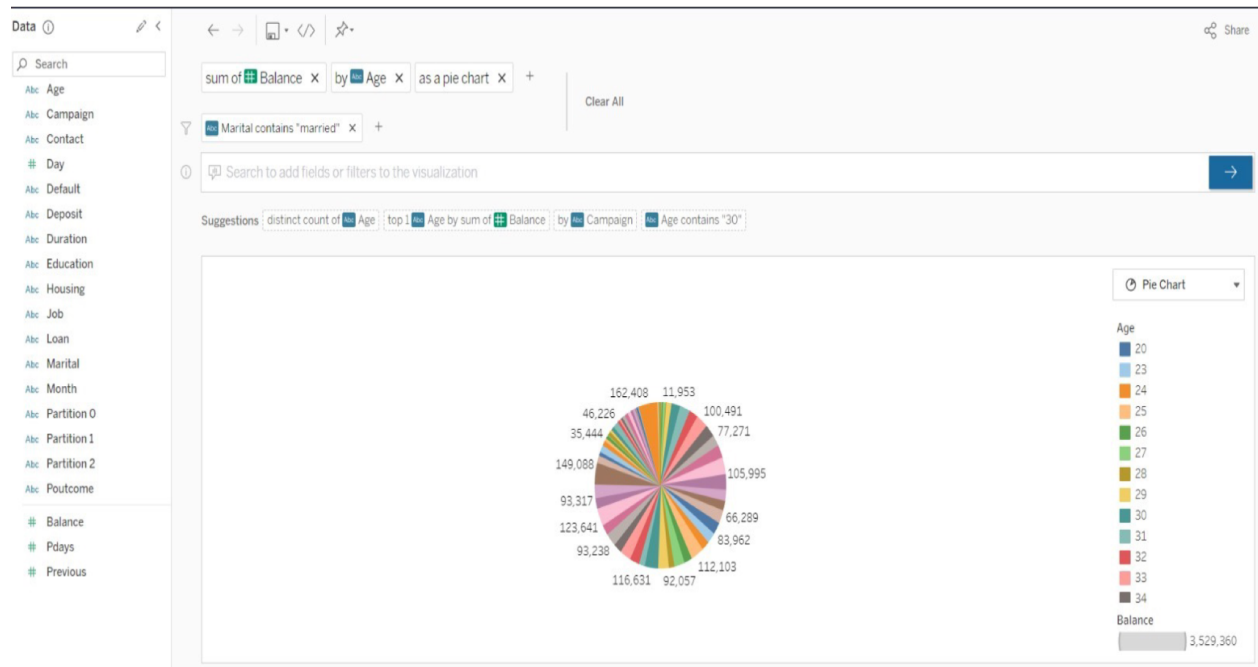
1. Kibana Analysis

Below are the results of the live data stream analysis in the form of a Dashboard



2. Tableau Analysis

Below are the results of the raw data analysis



References

- <https://aws.amazon.com/kinesis/>
- <https://aws.amazon.com/kinesis/data-firehose/>
- <https://aws.amazon.com/opensearch-service/>
- <https://aws.amazon.com/s3/>
- <https://aws.amazon.com/glue/>
- <https://aws.amazon.com/lambda/>
- <https://aws.amazon.com/athena/>
- <https://www.tableau.com/>