Project Report

on

“Real Time Data Streaming Application”

CPSC 531-03 22470

Advanced 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:-

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

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

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

Contents

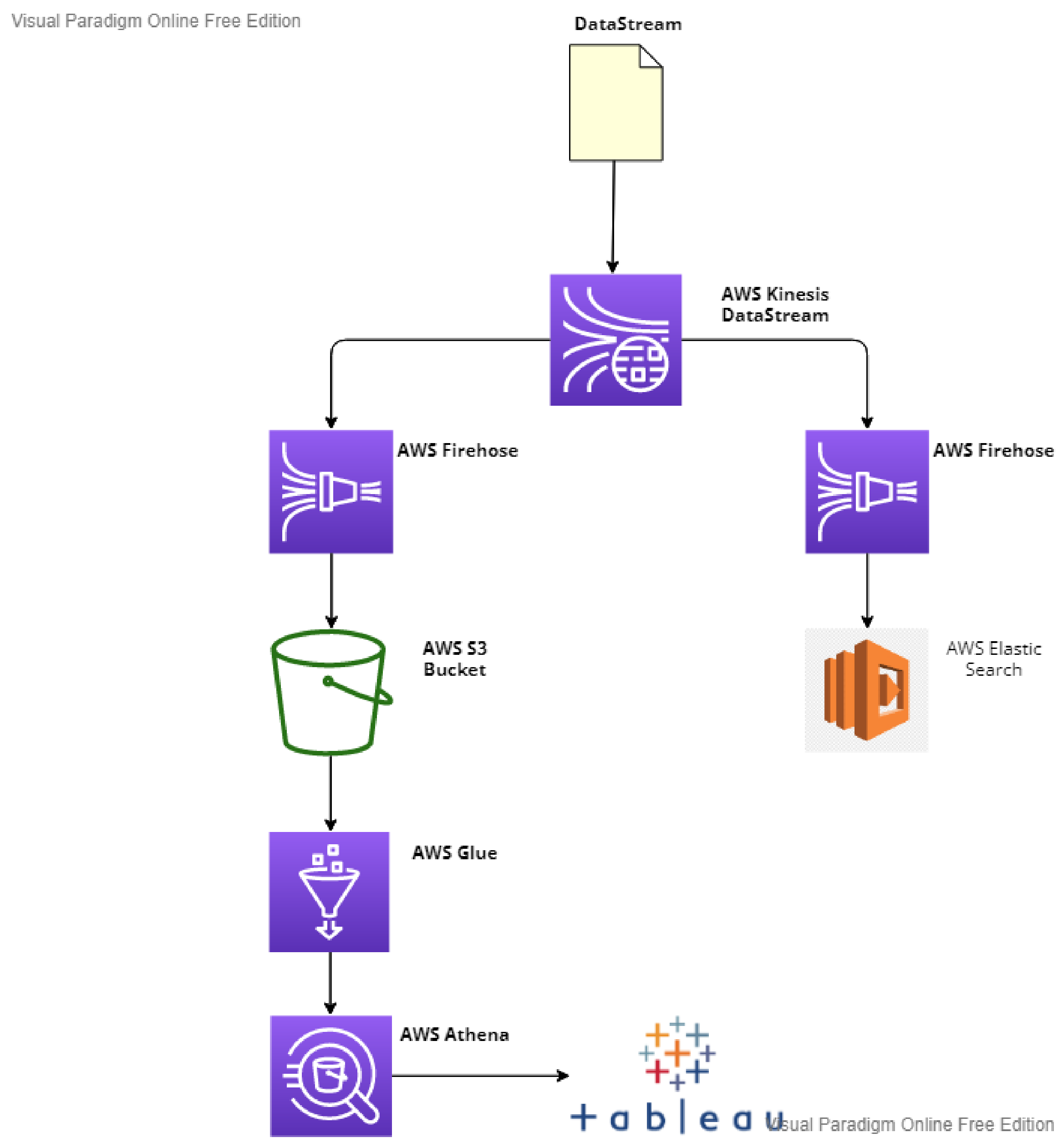
Introduction

- 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. There are a lot of data sources that create wuch type of data, which are IoT sensors, smart devices, and gaming applications which produce data at high volumes and high velocity.
- So there is a need to process this data in real-time as there is some business where real-time processing and analytics are of crucial importance to get an edge over competitors, and also it enables faster decision making along with various other advantages.
- Detecting fraud in real-time, ride-share apps, and e-commerce apps are very important examples of real-time processing.

Functionalities

- Producers are simulating the data stream rapidly, which is then getting ingested by AWS Kinesis 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 of them is continuously loading the real-time data to Elasticsearch or OpenSearch while the other one is loading the raw data into our S3 data bucket.
- The AWS Elasticsearch or OpenSearch is monitoring our data and producing visual insights using Kibana
- The raw data within the S3 bucket is getting 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:



- The data Stream is a python program which is simulating the data in real-time
- The output data stream is ingested by DataStream which is then processed by KinesisFirehose
- It is then used by OpenSearch for real-time analysis using Kibana
- The other Firehose is used by S3 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:

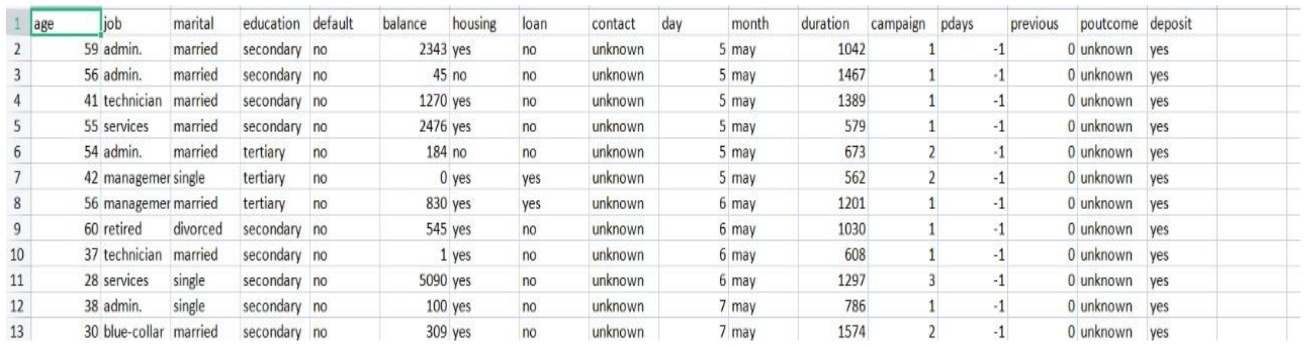
To work on the project, one must have the following skills but not limited to.

- 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 a lot of 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

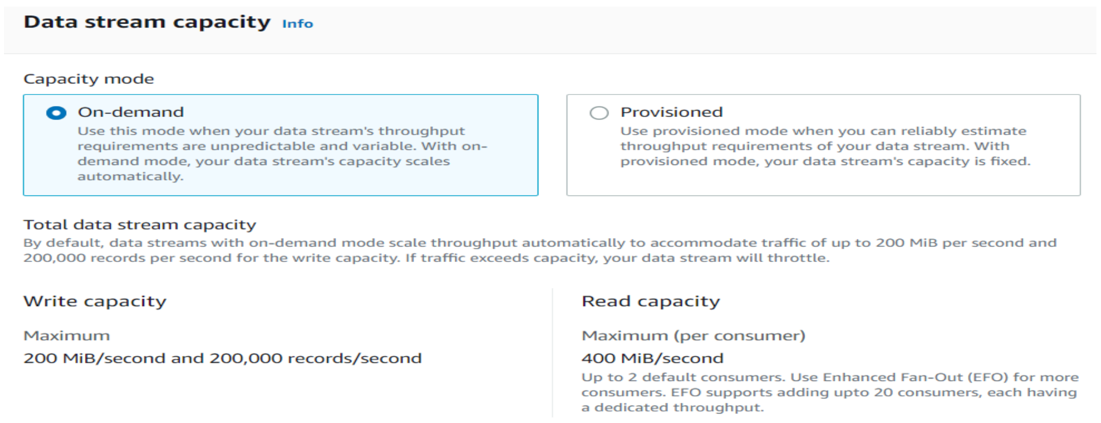
GitHub Location of Code:

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 which are on-demand and provisioned, where on-demand helps us to automatically scale the data stream and provisioned mode uses an initial set of set resources.


Data stream capacity Info

Capacity mode

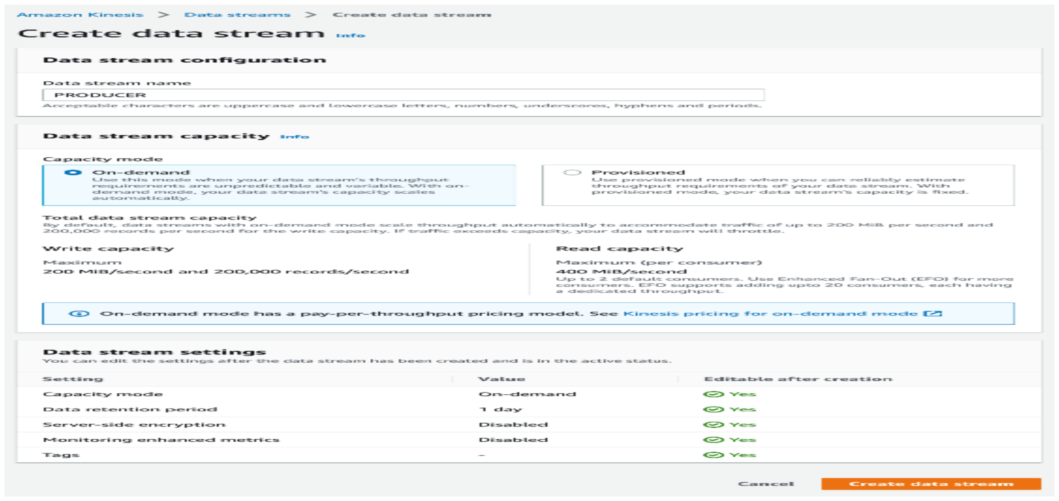
[✓ On-demand]
Use this mode when your data stream's throughput requirements are unpredictable and variable. With on-demand mode, your data stream's capacity scales automatically.

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.



Amazon Kinesis > Data streams > Create data stream

Create data stream Info

Data stream configuration

Data stream name: PRODUCER

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 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.

On-demand mode has a pay-per-throughput pricing model. See Kinesis pricing for on-demand mode

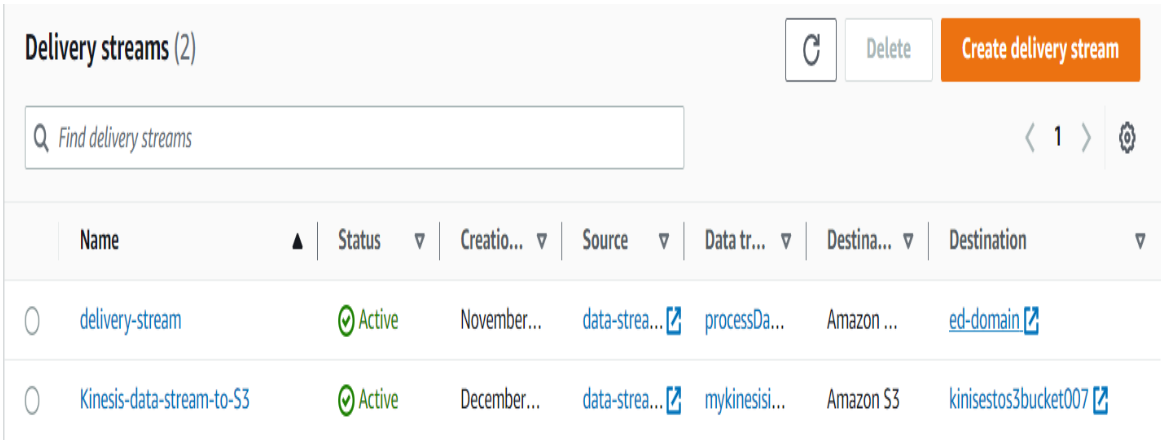
Data stream settings

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

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 where it is used for ElasticSearch, another one is used for the batch flow, and S3 is the Destination.



Delivery streams (2)

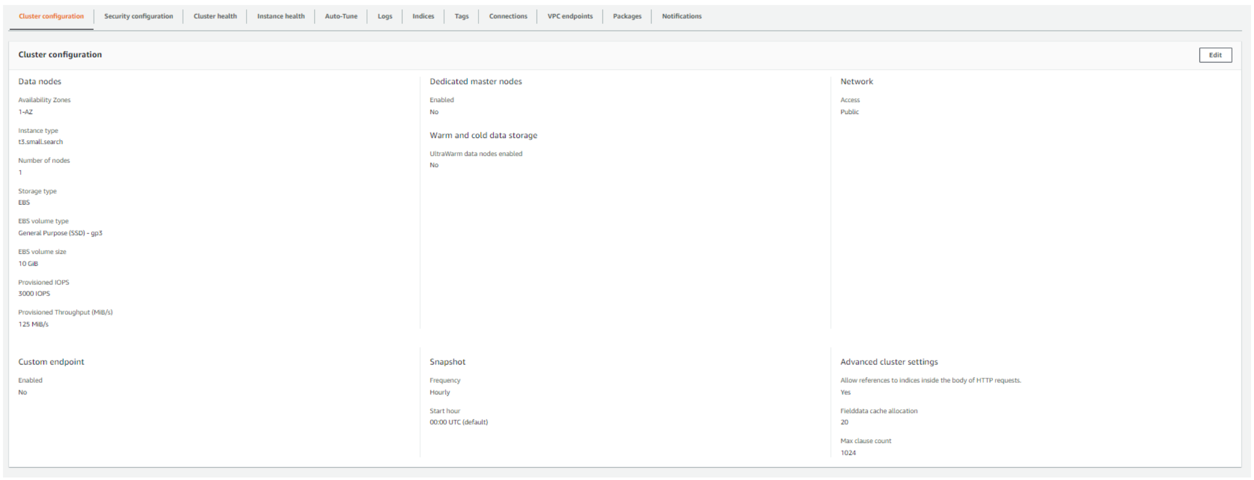
Name	Status	Creatio...	Source	Data tr...	Destina...	Destination
delivery-stream	Active	November...	data-strea...	processDa...	Amazon ...	ed-domain
Kinesis-data-stream-to-S3	Active	December...	data-strea...	mykinesisi...	Amazon S3	kinisestos3bucket007

- The configuration of the first delivery stream is as shown in the image
  1) 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) Destination is an OpenSearch domain which is similar to the cluster which has the following Cluster and Security configuration.
4) We have added an S3 location as a backup and also the error logs would be saved in the S3 location which was created
5) Delivery Stream2 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

Data nodes

Availability Zones
1-AZ

Instance type
t3.small.search

Number of nodes
1

Storage type
EBS

EBS volume type
General Purpose (SSD) - gp3

EBS volume size
10 GiB

Provisioned IOPS
3000 IOPS

Provisioned Throughput (MiB/s)
125 MiB/s

Dedicated master nodes

Enabled
No

Warm and cold data storage

UltraWarm data nodes enabled
No

Network

Access
Public

Custom endpoint

Enabled
No

Snapshot

Frequency
Hourly

Start hour
00:00 UTC (default)

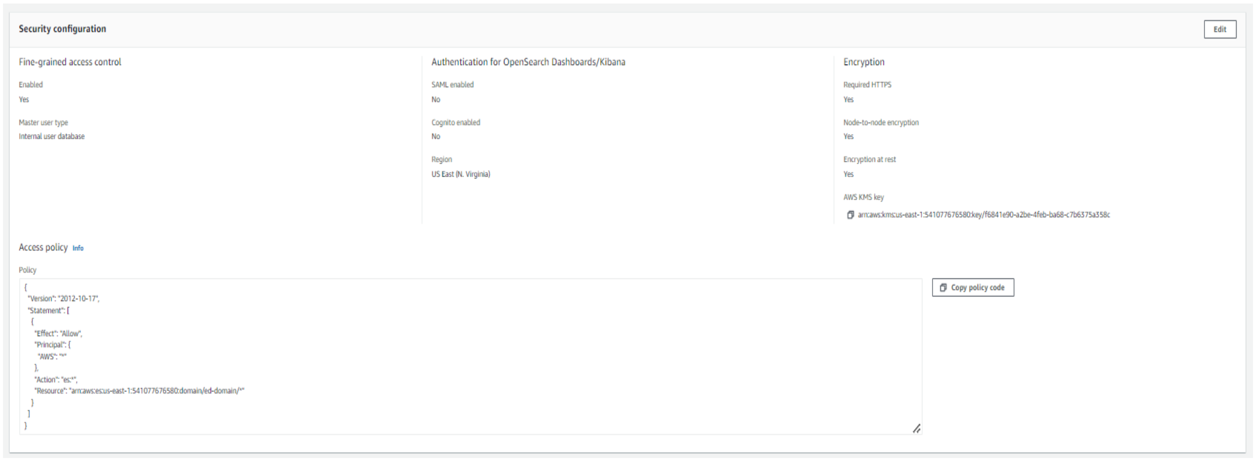
Advanced cluster settings

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

Fine-grained access control

Enabled
Yes

Master user type
Internal user database

Authentication for OpenSearch Dashboards/Kibana

SAML enabled
No

Cognito enabled
No

Region
US East (N. Virginia)

Encryption

Required HTTPS
Yes

Node-to-node encryption
Yes

Encryption at rest
Yes

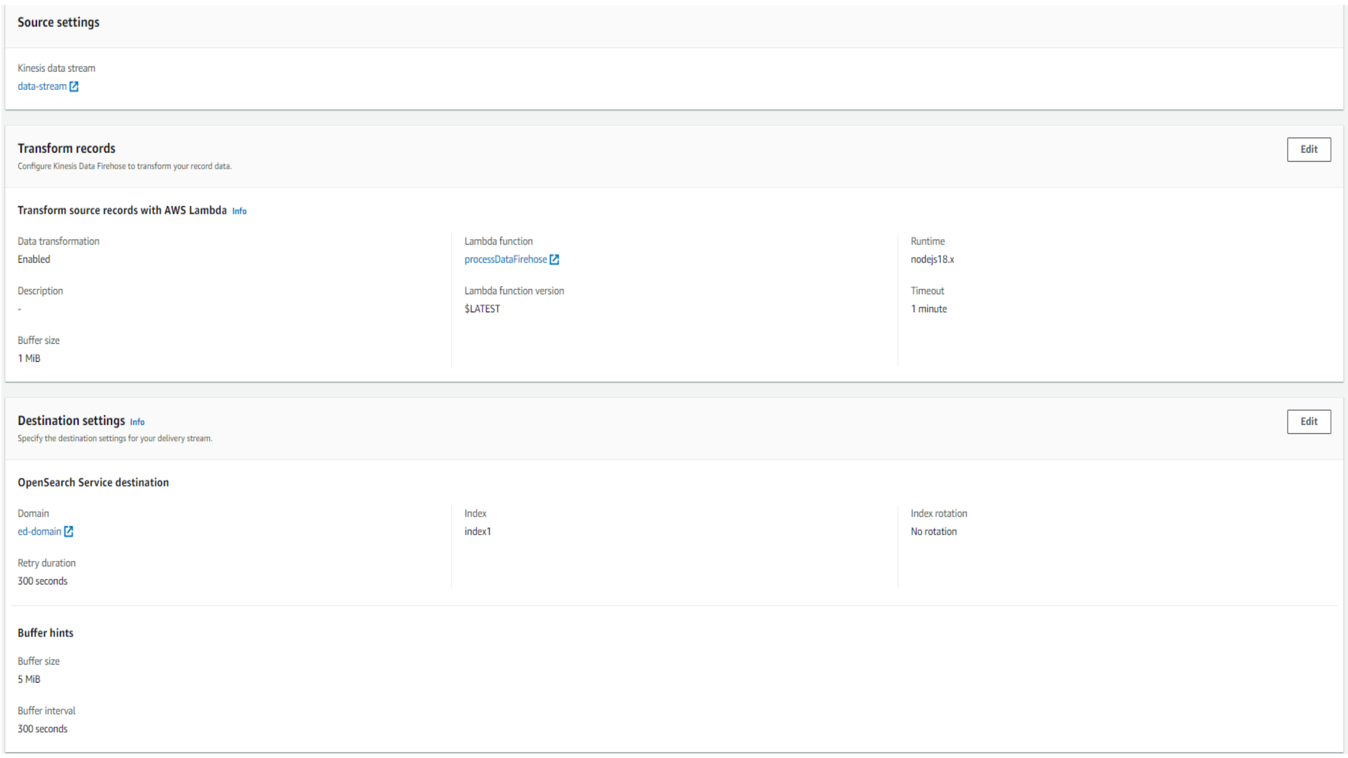
AWS KMS key
arn:aws:kms:us-east-1:541077676580:key/f6841e90-a2be-4feb-ba68-c7b6375a358c

Access policy

Policy

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

Delivery Stream1 (Part A)



Source settings

Kinesis data stream
data-stream

Transform records
Configure Kinesis Data Firehose to transform your record data.

Transform source records with AWS Lambda

Data transformation
Enabled

Description
-

Buffer size
1 MiB

Lambda function
processDataFirehose

Lambda function version
$LATEST

Runtime
nodejs18.x

Timeout
1 minute

Destination settings
Specify the destination settings for your delivery stream.

OpenSearch Service destination

Domain
ed-domain

Retry duration
300 seconds

Index
index1

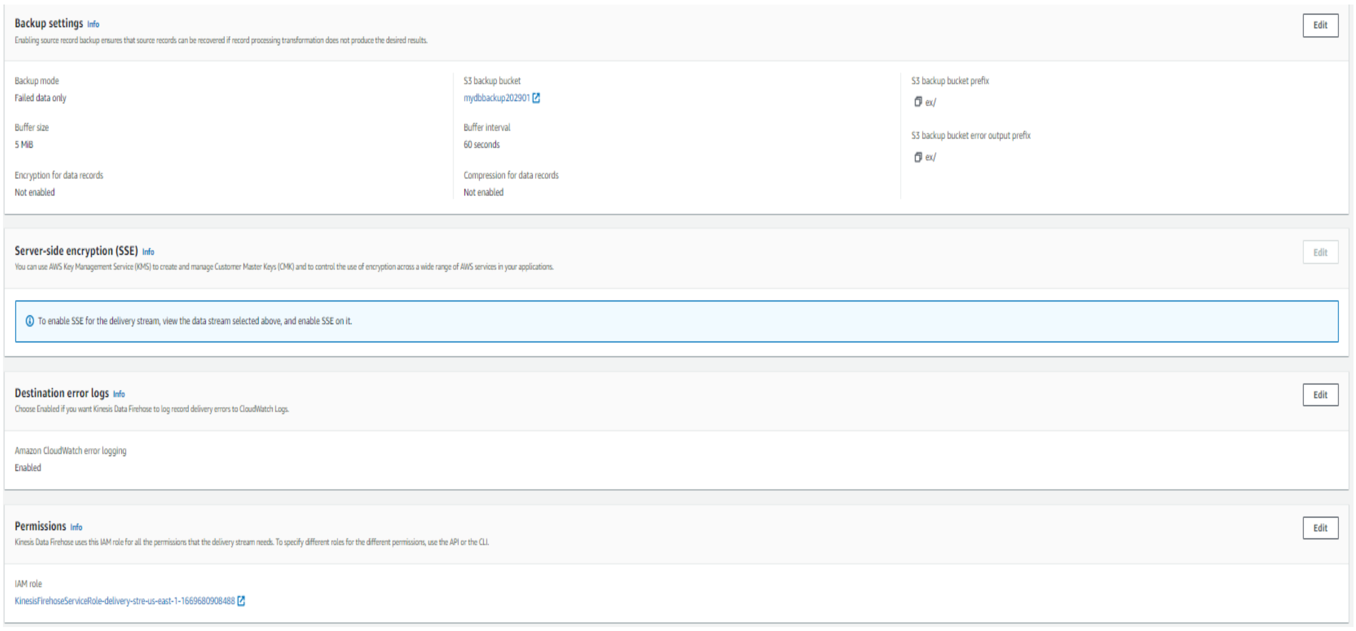
Index rotation
No rotation

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

S3 backup bucket
mydbbackup202901

S3 backup bucket prefix
ex/

Buffer size
5 MiB

Buffer interval
60 seconds

S3 backup bucket error output prefix
ex/

Encryption for data records
Not enabled

Compression for data records
Not enabled

Server-side encryption (SSE) Info

You can use AWS Key Management Service (KMS) to create and manage Customer Master Keys (CMK) 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.

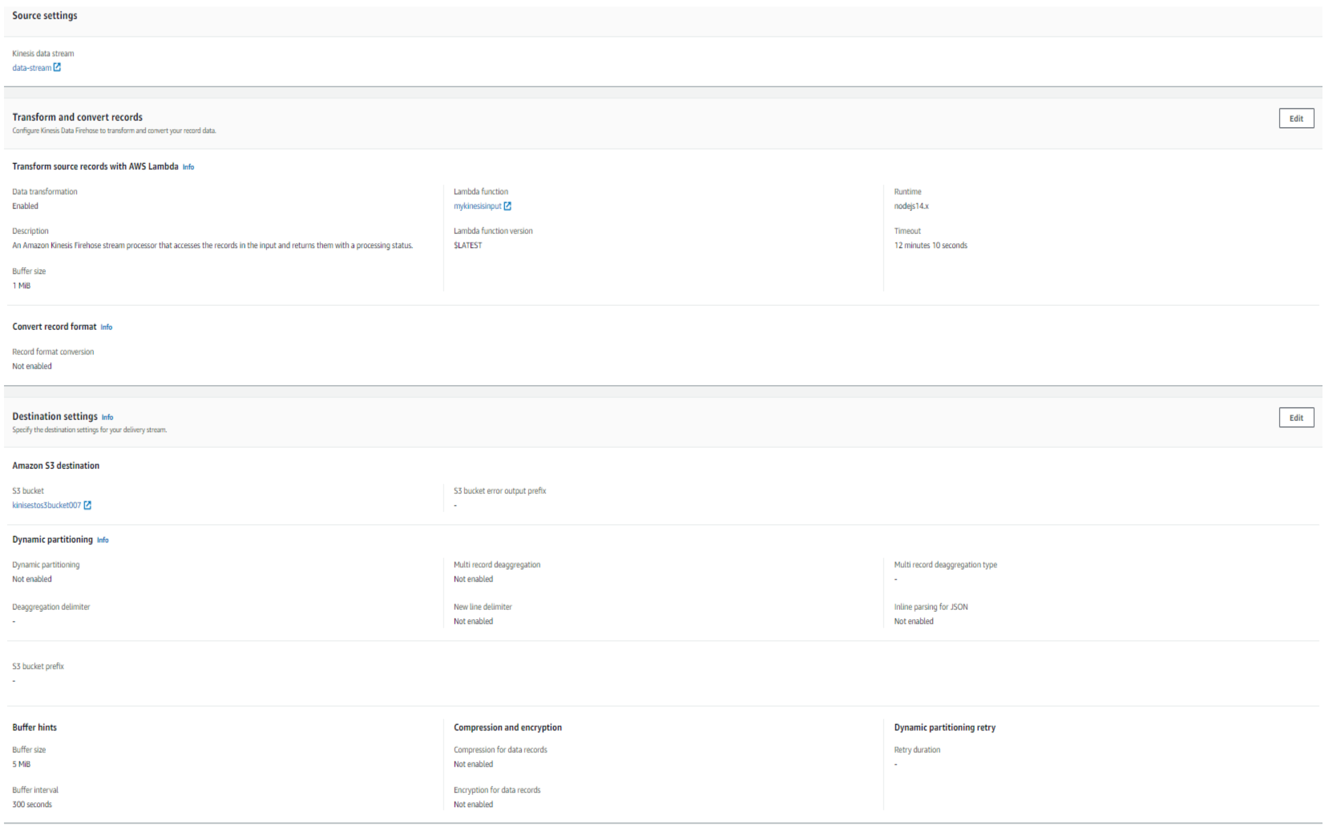
Amazon CloudWatch error logging
Enabled

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
KinesisFirehoseServiceRole-delivery-stre-us-east-1-1669680908488

Delivery Stream2 (Part A)



Source settings

Kinesis data stream
data-stream

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

Transform source records with AWS Lambda Info

Data transformation Enabled	Lambda function mykinesisinput	Runtime nodejs14.x
Description An Amazon Kinesis Firehose stream processor that accesses the records in the input and returns them with a processing status.	Lambda function version $LATEST	Timeout 12 minutes 10 seconds
Buffer size 1 MiB		

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 -

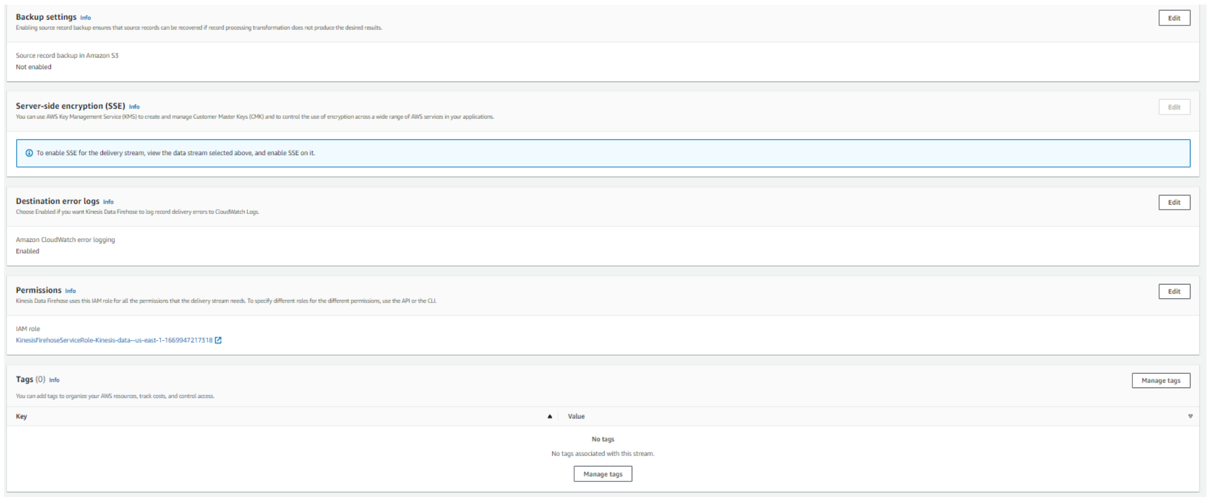
Dynamic partitioning Info

Dynamic partitioning Not enabled	Multi record deaggregation Not enabled	Multi record deaggregation type -
Deaggregation delimiter -	New line delimiter Not enabled	Inline parsing for JSON Not enabled

S3 bucket prefix
-

Buffer hints	**Compression and encryption**	**Dynamic partitioning retry**
Buffer size 5 MiB	Compression for data records Not enabled	Retry duration -
Buffer interval 300 seconds	Encryption for data records Not enabled	

Delivery Stream2 (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.

Source record backup in Amazon S3
Not enabled

Server-side encryption (SSE) Info
You can use AWS Key Management Service (KMS) to create and manage Customer Master Keys (CMK) 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.

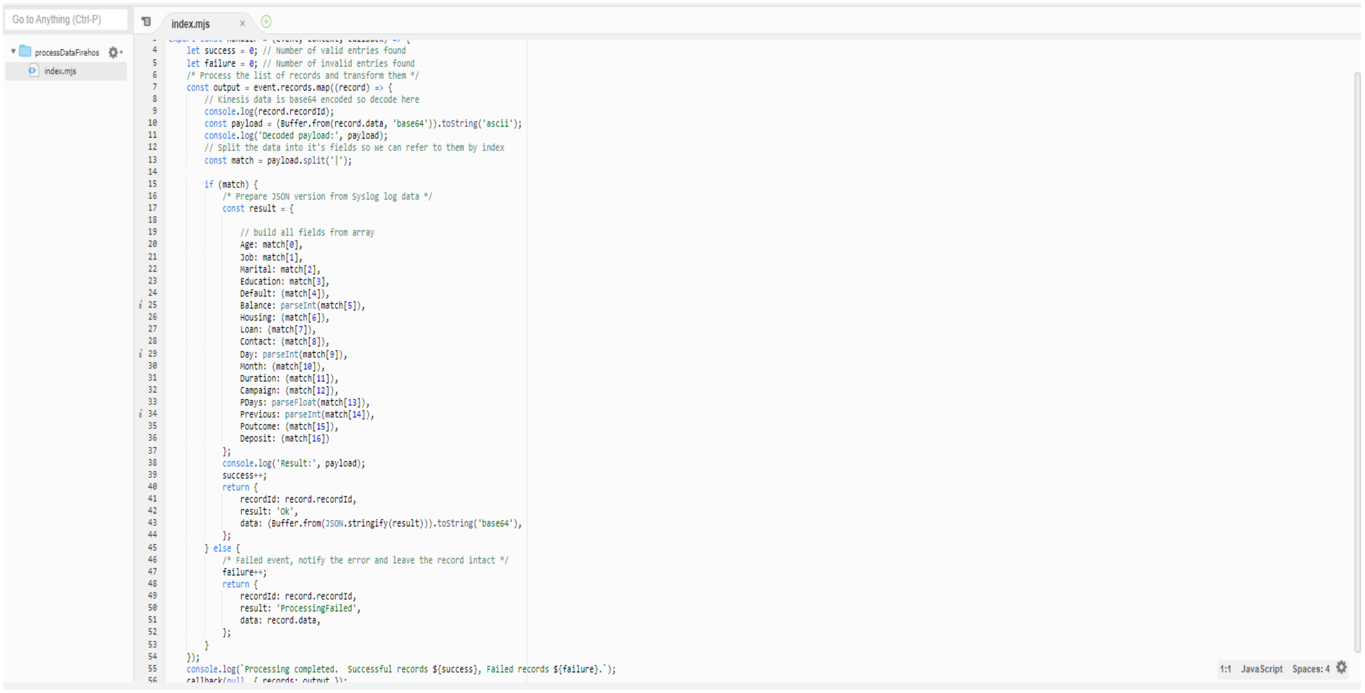
IAM role
KinesisFirehoseServiceRole-Kinesis-data--us-east-1-1669947217318

Tags (0) Info
You can add tags to organize your AWS resources, track costs, and control access.

Key	Value
No tags No tags associated with this stream.	

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.



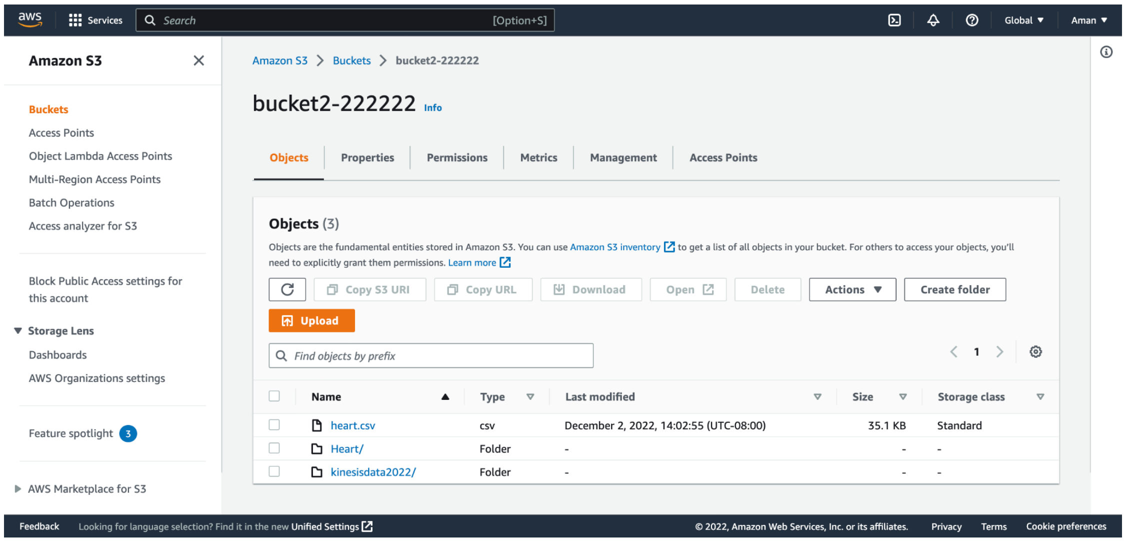
```javascript
export const handler = (event, context, callback) => {
    let success = 0; // Number of valid entries found
    let failure = 0; // Number of invalid entries found
    /* Process the list of records and transform them */
    const output = event.records.map((record) => {
        // Kinesis data is base64 encoded so decode here
        console.log(record.recordId);
        const payload = (Buffer.from(record.data, 'base64')).toString('ascii');
        console.log('Decoded payload:', payload);
        // Split the data into it's fields so we can refer to them by index
        const match = payload.split('|');

        if (match) {
            /* Prepare JSON version from Syslog log data */
            const result = {

                // build all fields from array
                Age: match[0],
                Job: match[1],
                Marital: match[2],
                Education: match[3],
                Default: (match[4]),
                Balance: parseInt(match[5]),
                Housing: (match[6]),
                Loan: (match[7]),
                Contact: (match[8]),
                Day: parseInt(match[9]),
                Month: (match[10]),
                Duration: (match[11]),
                Campaign: (match[12]),
                PDays: parseFloat(match[13]),
                Previous: parseInt(match[14]),
                Poutcome: (match[15]),
                Deposit: (match[16])
            };
            console.log('Result:', payload);
            success++;
            return {
                recordId: record.recordId,
                result: 'Ok',
                data: (Buffer.from(JSON.stringify(result))).toString('base64'),
            };
        } else {
            /* Failed event, notify the error and leave the record intact */
            failure++;
            return {
                recordId: record.recordId,
                result: 'ProcessingFailed',
                data: record.data,
            };
        }
    });
    console.log(`Processing completed.  Successful records ${success}, Failed records ${failure}.`);
    callback(null, { records: output });
```

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.



Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
heart.csv	csv	December 2, 2022, 14:02:55 (UTC-08:00)	35.1 KB	Standard
Heart/	Folder	-	-	-
kinesisdata2022/	Folder	-	-	-

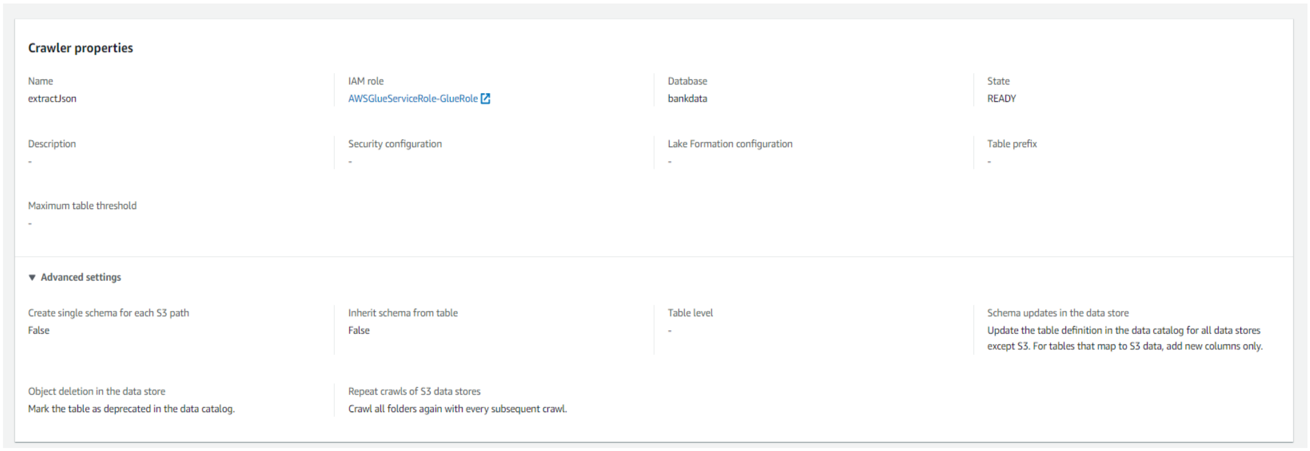
6. Finish creating the delivery stream

- 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 which would create a GLUE database and tables/

Glue Crawler



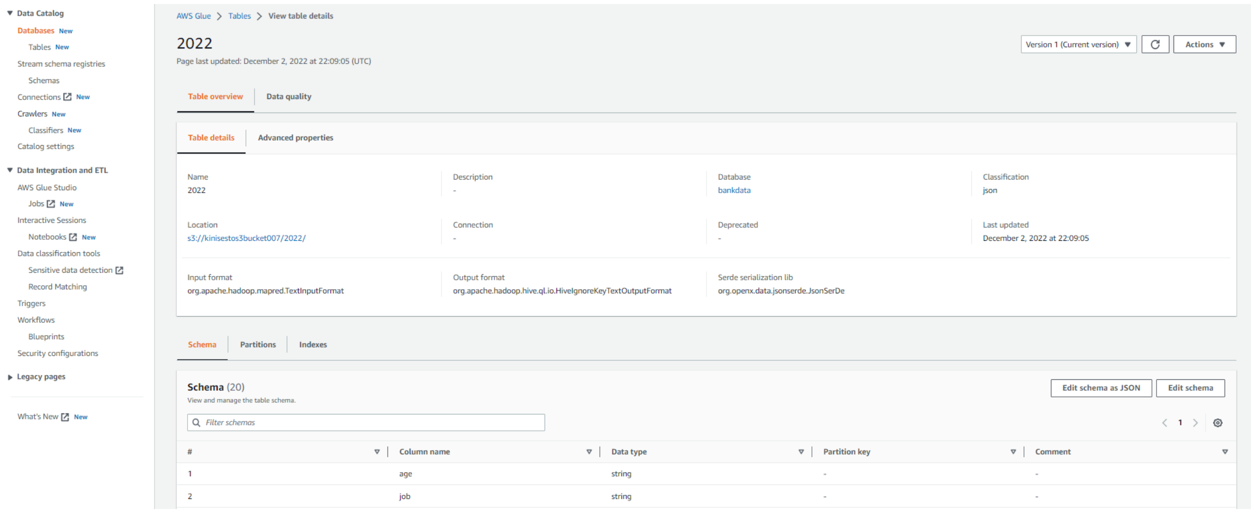
Crawler properties

Name extractJson	IAM role AWSGlueServiceRole-GlueRole	Database bankdata	State READY
Description -	Security configuration -	Lake Formation configuration -	Table prefix -
Maximum table threshold -			

▼ Advanced settings

Create single schema for each S3 path False	Inherit schema from table False	Table level -	Schema updates in the data store Update the table definition in the data catalog for all data stores except S3. For tables that map to S3 data, add new columns only.
Object deletion in the data store Mark the table as deprecated in the data catalog.	Repeat crawls of S3 data stores Crawl all folders again with every subsequent crawl.		

Glue Database



2022

Page last updated: December 2, 2022 at 22:09:05 (UTC)

Name 2022	Description -	Database bankdata	Classification json
Location s3://kinisestos3bucket007/2022/	Connection -	Deprecated -	Last updated December 2, 2022 at 22:09:05
Input format org.apache.hadoop.mapred.TextInputFormat	Output format org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat	Serde serialization lib org.openx.data.jsonserde.JsonSerDe	

Schema (20)

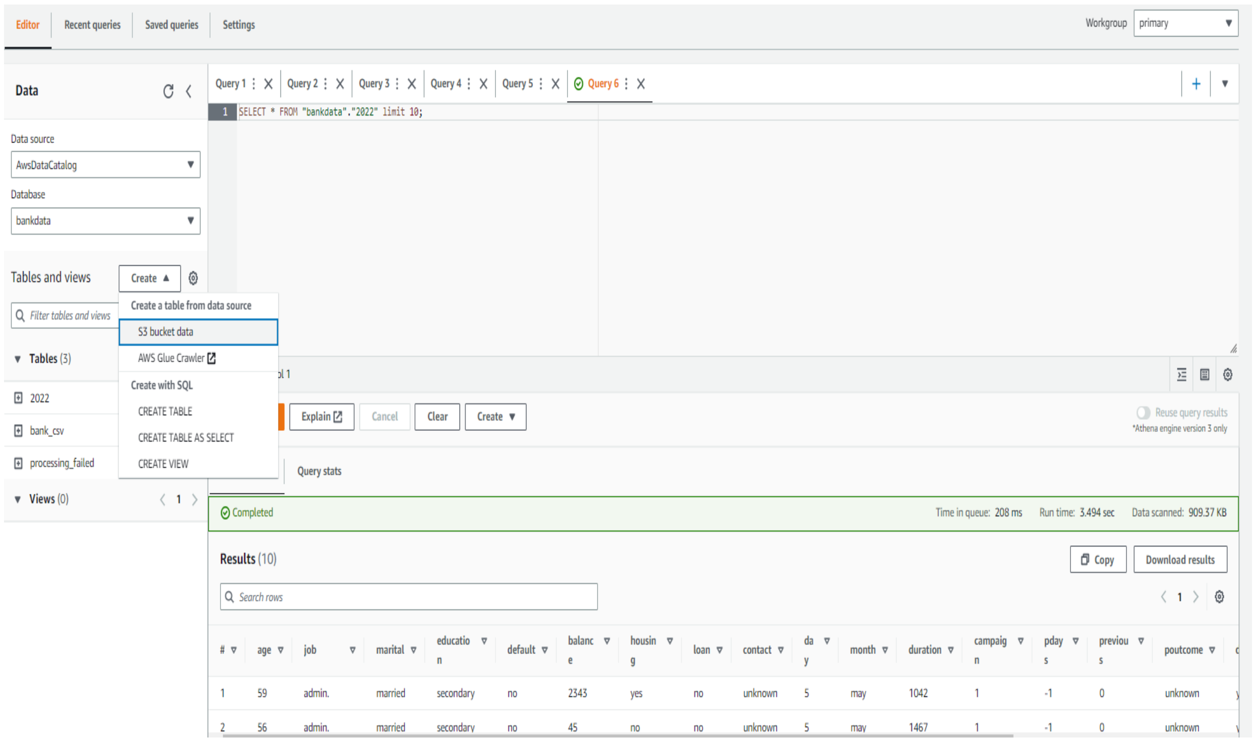
View and manage the table schema.

#	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.



Data source: AwsDataCatalog

Database: bankdata

```
SELECT * FROM "bankdata"."2022" limit 10;
```

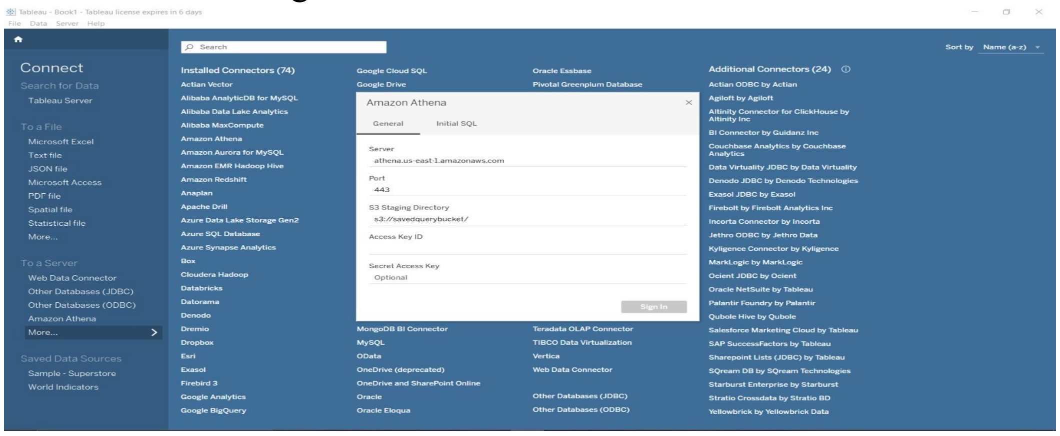
Completed   Time in queue: 208 ms   Run time: 3.494 sec   Data scanned: 909.37 KB

Results (10)

#	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pday s	previous	poutcome
1	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown
2	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown

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.



Amazon Athena

Server: athena.us-east-1.amazonaws.com

Port: 443

S3 Staging Directory: s3://savedquerybucket/

Access Key ID:

Secret Access Key: Optional

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 Code and Amazon Lambda code can be found in the mentioned 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

Kibana Analysis

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

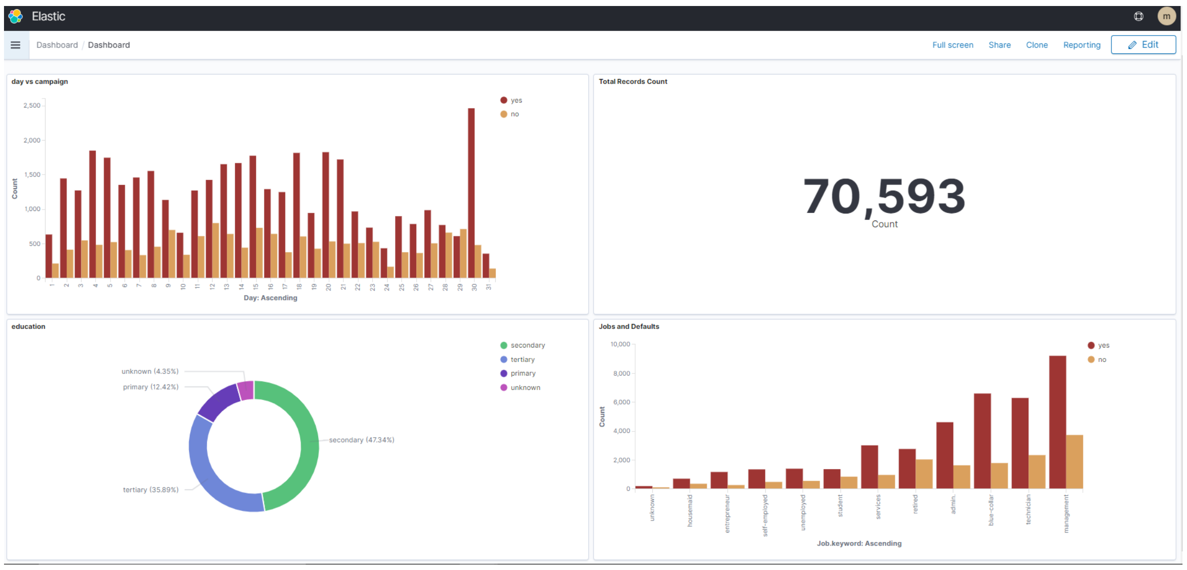
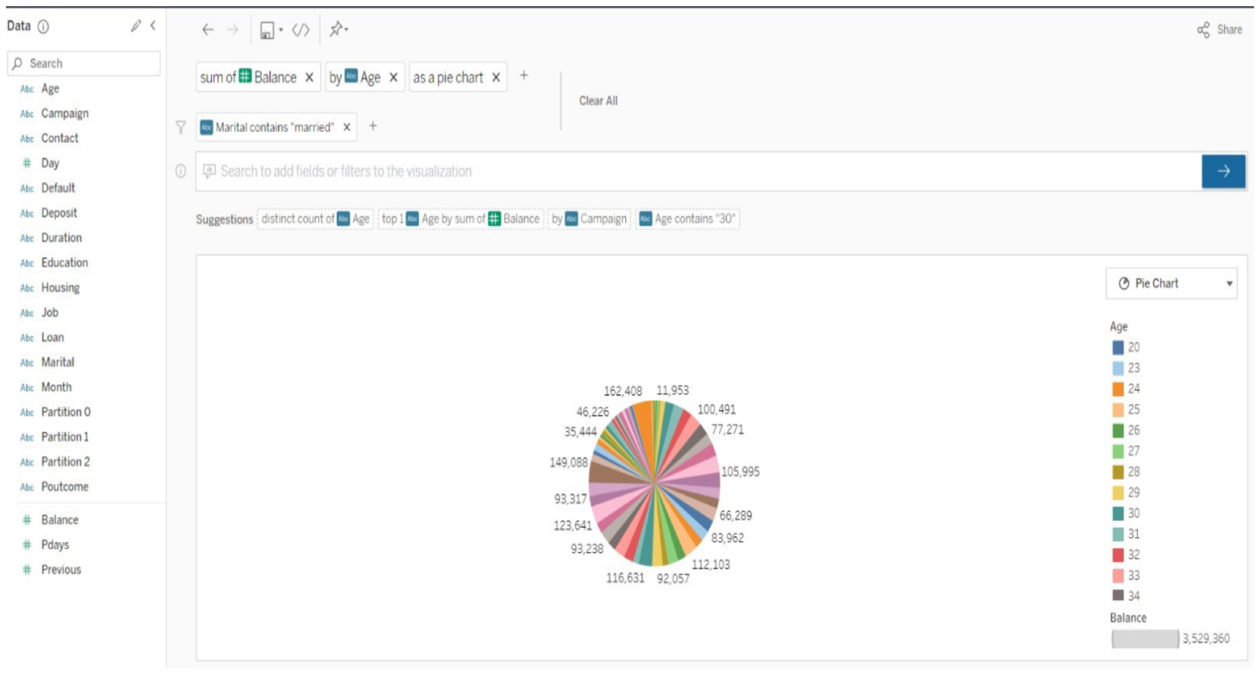


Tableau Analysis

Below are the results for the raw data analysis

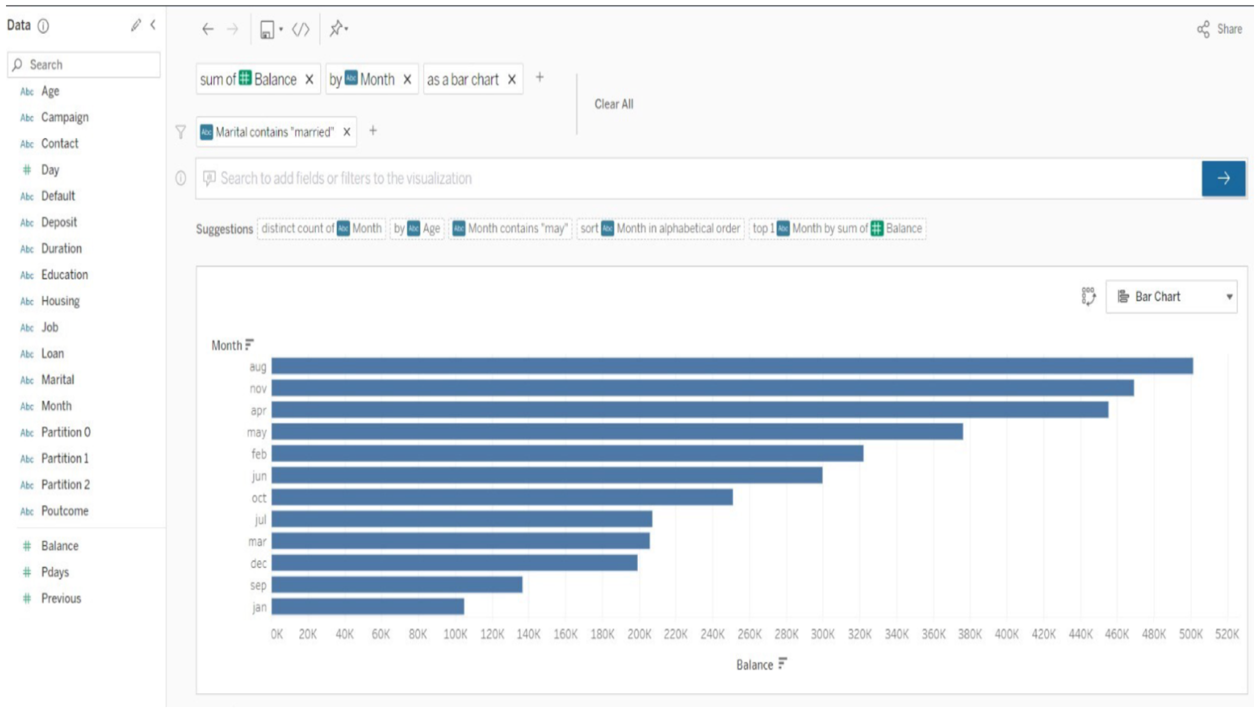


sum of Balance × by Age × as a pie chart ×

Marital contains "married" ×

Suggestions: distinct count of Age | top 1 Age by sum of Balance | by Campaign | Age contains "30"

Pie Chart



sum of Balance × by Month × as a bar chart ×

Marital contains "married" ×

Suggestions: distinct count of Month | by Age | Month contains "may" | sort Month in alphabetical order | top 1 Month by sum of Balance

Bar Chart

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/