

CSP-554 PROJECT REPORT

DATA PIPELINE USING DYNAMODB AND AWS KINESIS

Rajesh Kumar Bandaru (A20446254)

Mohammed Jawhar (A20449684)

Sai Charan Akkena (A20456762)

Abstract

A Data Pipeline is a series of data processing steps. It can be a piece of software that eliminates many manual steps and enables automated flow of data. It has three important components:

1. A Source
2. Processing step or steps
3. A Sink

The data pipeline in this project enables the flow of data from a database to an analytics platform. As the volume of data being generated everyday is increasing exponentially the term “big data” implying the huge data came into existence. This “big data” can be used to make predictive analytics, real-time reporting, and alerting.

The architecture of a data pipeline requires many considerations. Does the pipeline need to handle streaming data or batch data? Does it need to satisfy a condition? What type of processing needs to be done? Where does the data need to go where is the data coming from?

There are different types of architectures for data pipeline. They are:

1. Batch based pipeline: This kind of architecture is used when there are large number of data points that has to be pushed into a warehouse.
2. Streaming data pipeline: This kind of architecture is used when there is a need to process the data point as it is generated.
3. Lambda based pipeline: This is the combination of both batch and streaming data pipeline. This architecture is popular in big data environments we can process both real-time data and historical data.

In this project we used, Lambda based architecture to process the data of Permanent magnet synchronous motor (PMSM) of Power Electronics and Electrical Devices. The readings of the data are taken batch wise and, we need a real time analysis to give alerts if anything goes wrong with the machine.

Project Details

1. **Project Topic:** A Data Pipeline using DynamoDB and AWS Kinesis.
2. **Dataset:** Recordings from a Permanent Magnet Synchronous Motor (PMSM)
3. **About the Dataset:**

The dataset contains readings of various components of the motor. The dataset was taken from Kaggle offered by Paderborn university. The readings are as follows,

1. Ambient temperature.
2. Coolant temperature.
3. Motor speed.
4. Current components.
5. Voltage components.
6. Stator yoke temperature.
7. Stator tooth temperature.
8. Stator winding temperature.
9. Permanent magnet surface temperature.

These readings are measured by appropriate sensors attached to motor. There are more than 990,000 readings taken over 72 sessions. The data is available as csv file.

4. **Project Goals:**
 - Create a DynamoDB table.
 - Ingest data into a DynamoDB Table.
 - An automated Data pipeline is created using AWS Cloud Formation.
 - Enable Kinesis Data Streaming for DynamoDB.
 - Transform the data as needed using lambda function and kinesis data firehose.
 - Analyzing the data using Amazon Athena if needed.
 - A QuickSight dashboard for analysis.
5. **Technologies:** Amazon DynamoDB, Amazon Kinesis Data Streams, Amazon Kinesis Data Firehose, Amazon Athena, Crawler, Lambda Function, S3 buckets, Amazon QuickSight.

Literature Review

▪ Data Pipeline

A data pipeline is used to move data from one end to another end transform data in between for analysis at the other end. It consists of a source, a data transformation step and a sink. In this project, the pipeline allows to move data from a DynamoDB table to Athena for querying and QuickSight for visual analysis. It is also called as a series of data processing steps. Streaming should be considered while building a data pipeline. So, we make use of AWS Kinesis services to move data from source to sink. We assume our data is generated from cloud and stored in DynamoDB table (source).

▪ DynamoDB

The data for this project is stored in DynamoDB table. DynamoDB is amazon's NoSQL database service. It delivers single - digit milli second performance at any scale. It's a fully managed, multi-region, multi-active, durable database with built-in security backup and restore and in-memory caching for internet scale applications. DynamoDB is serverless with no servers to provision, patch or manage and no software to install, maintain or operate. It automatically scales tables up and down to adjust for capacity and maintain performance. It supports ACID transactions for building business-critical applications at scale.

▪ AWS Kinesis

For collecting the data from DynamoDB and further processing AWS Kinesis is used. It is used to analyze real time data so that a timely insight can be taken and react fast as per the situation. The application logs of our data set can be ingested real-time for analysis in QuickSight. Amazon kinesis can ingest, buffer and process streaming data in real-time so that insights can be derived in seconds or minutes instead of hours or days. It can handle data from hundreds of thousands of sources with very low latencies.

▪ Crawlers

A crawler can access a data store, extracts metadata and creates tables in AWS Glue Data Catalog. Then ETL operations are done on that data. It could crawl multiple data stores as well. The Data from the Glue Catalog is used for querying in Athena.

▪ Amazon Athena

It is a query service offered by amazon to query data in S3 buckets interactively. The querying can be done in Standard SQL. It is a serverless service. The data streamed from DynamoDB is queried for necessary data needed for analysis in QuickSight. Athena uses presto with ANSI SQL support and works with various standard data formats. Athena is fast as it executes queries in parallel, so more results will come back in less time.

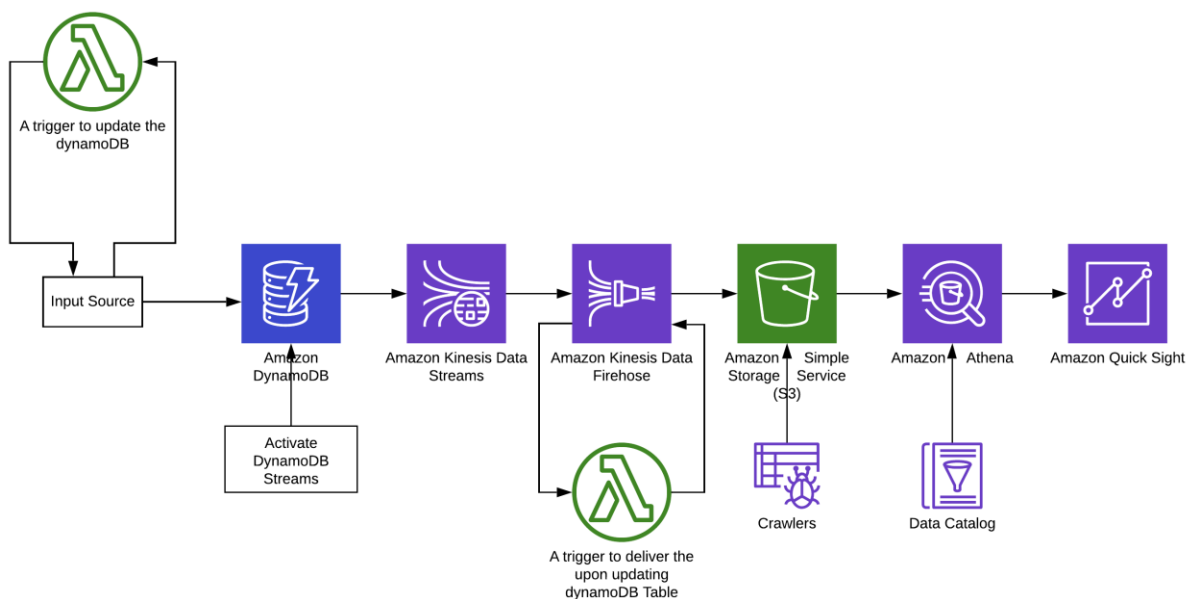
▪ QuickSight

The data extracted from athena is just the data in another format. It should be converted into something visually soothing so that we can get insights from them. QuickSight does the exact same. It is a scalable, serverless service built for cloud. It can easily create interactive dashboards which are machine learning powered.

Objective

The objective of the project is to create a data pipeline to make analysis on PMSM dataset. It is a way to monitor the incoming data. To delete or altering the schema as the company's requirements and to take the necessary information from the dataset for doing analysis. It helps in giving alerts if readings are out of range or anything is wrong with the machine based on the values of the data. This helps Power Electronics and Electrical Devices to take measures according to the situation.

Architecture



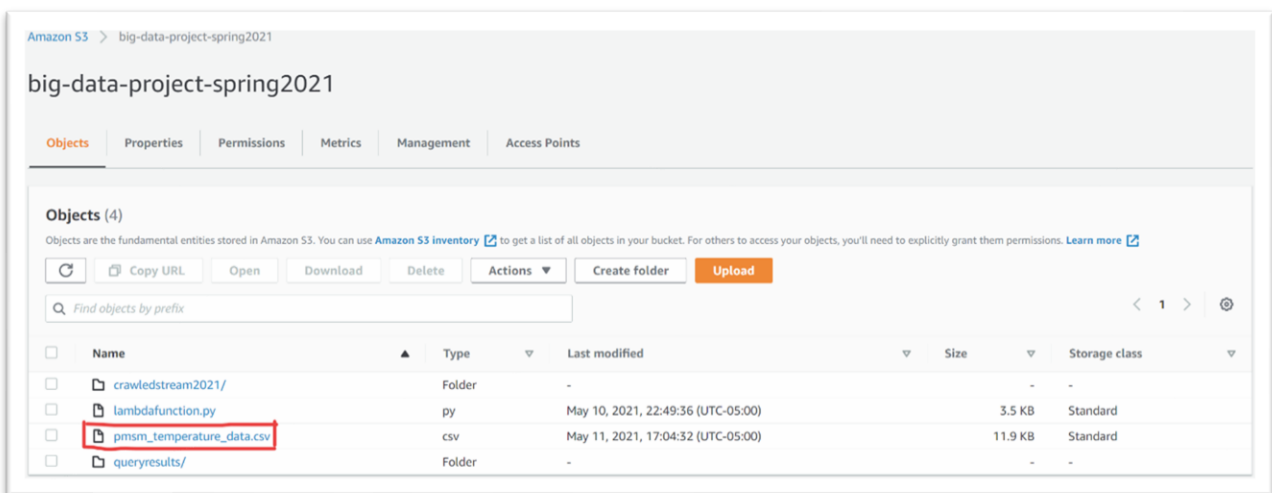
Overview of Flow

The data of PMSM machine is uploaded into an input source batch-wise. A lambda function is used as a trigger to upload data into DynamoDB to store historical data. Once the data is uploaded into DynamoDB, the DynamoDB stream should be activated to stream data through

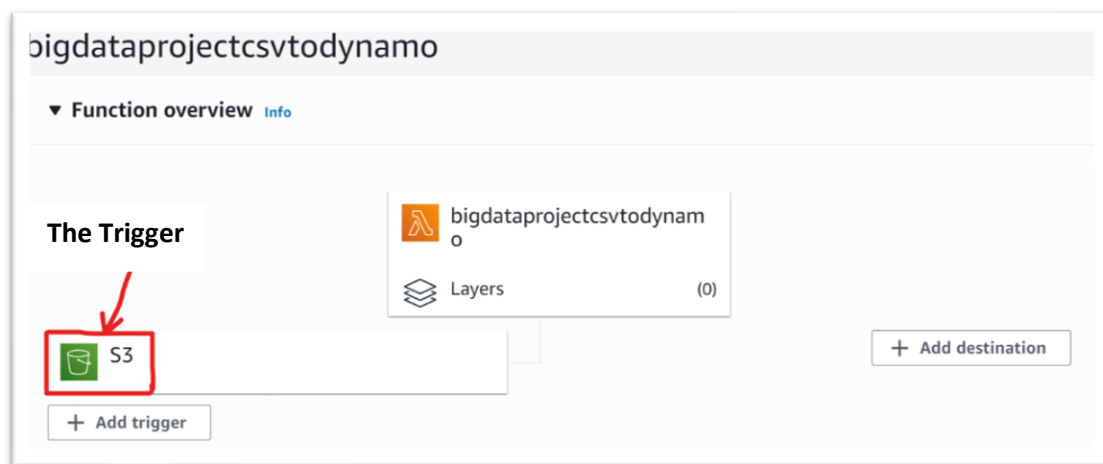
Amazon Kinesis. The data from Kinesis can only be delivered by a delivery stream. Kinesis Data Firehose helps in delivering the data from stream to S3 Bucket. A Lambda function is used as a trigger to deliver the data to S3 whenever new data is added to DynamoDB. Data from S3 bucket should be crawled using a Crawler for it to be available for querying in Athena. Data crawled from S3 is stored in Glue Data Catalog. It is accessed by Athena by mentioning the database name in Athena. The data queried is visualized in Quick Sight.

Methodology

The input source in our case is a S3 bucket. In real life, data is sent from IOT devices which measures the appropriate readings of PMSM machine.



The marked part is the data coming from the sensors of the machine. This is the first batch of data that is generated. The data generated batch wise is sent to the bucket regularly. Once the data is uploaded a lambda function is used as a trigger to load the data.



The trigger is 'S3 PUT' meaning whenever data is added into the bucket.



```
1 import json
2 import csv
3 import boto3
4
5 def lambda_handler(event, context):
6     # TODO Implement
7
8     region = 'us-east-1'
9     record_list = []
10    try:
11        s3 = boto3.client('s3')
12        dynamodb = boto3.client('dynamodb', region_name = region)
13        bucket = event['Records'][0]['s3']['bucket']['name']
14        key = event['Records'][0]['s3']['object']['key']
15
16        csv_file = s3.get_object(Bucket = bucket, Key = key)
17
18        record_list = csv_file['Body'].read().decode('utf-8').split('\n')
19
20        csv_reader = csv.reader(record_list, delimiter = ',', quotechar = '')
21
22        for row in csv_reader:
23            u_q = row[0]
24            coolant = row[1]
25            stator_winding = row[2]
26            u_d = row[3]
27            stator_tooth = row[4]
28            motor_speed = row[5]
29            i_d = row[6]
30            i_q = row[7]
31            pm = row[8]
32            stator_yoke = row[9]
33            ambient = row[10]
34            torque = row[11]
35            profile_id = row[12]
36            id = row[13]
37
38            add_to_db = dynamodb.put_item(TableName = 'pmsm_temperature', Item = {
39                'u_q' : {'N': str(u_q)},
40                'coolant' : {'N': str(coolant)},
41                'stator_winding' : {'N': str(stator_winding)},
42                'u_d' : {'N': str(u_d)},
43                'stator_tooth' : {'N': str(stator_tooth)},
44                'motor_speed' : {'N': str(motor_speed)},
45                'i_d' : {'N': str(i_d)},
46                'i_q' : {'N': str(i_q)},
47                'pm' : {'N': str(pm)},
48                'stator_yoke' : {'N': str(stator_yoke)},
49                'ambient' : {'N': str(ambient)},
50                'torque' : {'N': str(torque)},
51                'profile_id' : {'N': str(profile_id)},
52                'id' : {'N': str(id)},
53            })
54    
```

The lambda function '**bigdataprojectcsvtodynamodb**' triggers when data is added to S3 bucket invoking it to add that data to DynamoDB for historical analysis as well as real time analysis. Making the data use for both purposes makes it fall under the category of lambda architecture of Data Pipeline.

The screenshot shows the AWS DynamoDB console for the table 'pmsm_temperature'. The table name is highlighted with a red box. The 'id' column is highlighted with an orange box, indicating it is the primary key. The table contains 10 items with various sensor readings.

id	ambient	coolant	i_d	i_q	motor_speed	pm	prof
2	-2.0648587	-1.1170206	1.029509	-0.24583231	-0.2491333	-2.5224178	4
3	-2.064073	-1.1166813	1.0294477	-0.24581794	-0.24943107	-2.5226731	4
7	-2.0621152	-1.1161705	1.0304929	-0.24616154	-0.24791418	-2.5225377	4
8	-2.0619526	-1.1139864	1.0301074	-0.24603409	-0.24832090	-2.5228438	4
10	-2.062317	-1.1094859	1.0296358	-0.24588774	-0.24829444	-2.5226767	4
14	-2.061756	-1.1113616	1.0292736	-0.2457751	-0.24751255	-2.5219264	4
21	-2.0605102	-1.1150343	1.0292114	-0.24574813	-0.37626076	-2.5206366	4
31	-2.0628257	-1.0992454	0.31466627	-0.24596944	-0.33670974	-2.5199552	4
32	-2.0632231	-1.1001716	0.21214207	-0.24587053	-0.31627208	-2.524219	4
38	-2.0619857	-1.1043041	0.01076187	-0.24565578	-0.27129427	-2.523038	4

The DynamoDB table created is '**pmsm_temperature**' indicated by red box and the primary key or partition key is the ID of the reading indicated by orange box.

A kinesis Data stream is needed to stream the data from DynamoDB whenever a new batch of data is added to it.

The screenshot shows the Amazon Kinesis console for the data stream 'pmsmstreamtos3'. The stream name is highlighted with a red box. The stream status is 'Active'.

Stream details			
Status	ARN	Data retention period	Creation time
Active	arn:aws:kinesis:us-east-1:343981:584178:stream/pmsmstreamtos3	1 day	May 11, 2021, 12:20 CDT

A kinesis data stream '**pmsmstreamtos3**' is created. The status of the stream is 'active' meaning it is ready to stream the data.

To stream the data in DynamoDB, Stream should be enabled. This is done as follows:

The screenshot displays two sections of the AWS console. The top section, 'Kinesis data stream details', shows a table with 'Stream enabled' set to 'Yes' and 'Stream name' as 'pmsmstreamtos3'. A red box highlights these two rows. Below the table is a button labeled 'Manage streaming to Kinesis'. The bottom section, 'DynamoDB stream details', shows a table with 'Stream enabled' set to 'Yes' and 'View type' set to 'New image'. An orange box highlights these two rows. Below the table is a button labeled 'Manage DynamoDB stream'. The 'Latest stream ARN' is also displayed as 'arn:aws:dynamodb:us-east-1:343981584178:table/pmsm_temperature/stream/2021-05-11T21:33:18.607'.

Kinesis data stream details	
Stream enabled	Yes
Stream name	pmsmstreamtos3 Configure in Kinesis console

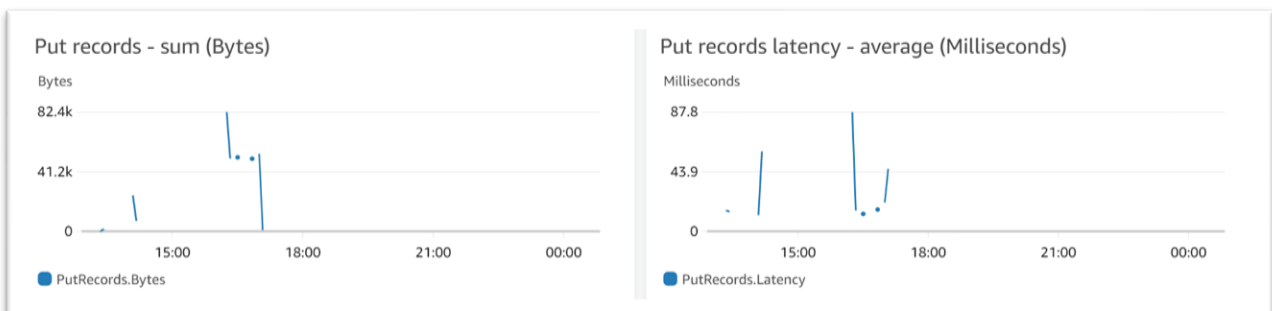
[Manage streaming to Kinesis](#)

DynamoDB stream details	
Stream enabled	Yes
View type	New image
Latest stream ARN	arn:aws:dynamodb:us-east-1:343981584178:table/pmsm_temperature/stream/2021-05-11T21:33:18.607

[Manage DynamoDB stream](#)

Streaming is enabled as **'Stream enabled'** is set to **'Yes'** (red box) through 'pmsmstreamtos3' which was created in the previous step. **'View type'** is set to **'New image'** (orange box) meaning new data which is added to DynamoDB is only sent through streaming for real-time analysis. The old and new data is stored in DynamoDB table for historical analysis later.

Now, that the stream is enabled data is sent from DynamoDB through Kinesis Stream. This can be seen in the **'Monitoring'** section of Kinesis Data Stream.

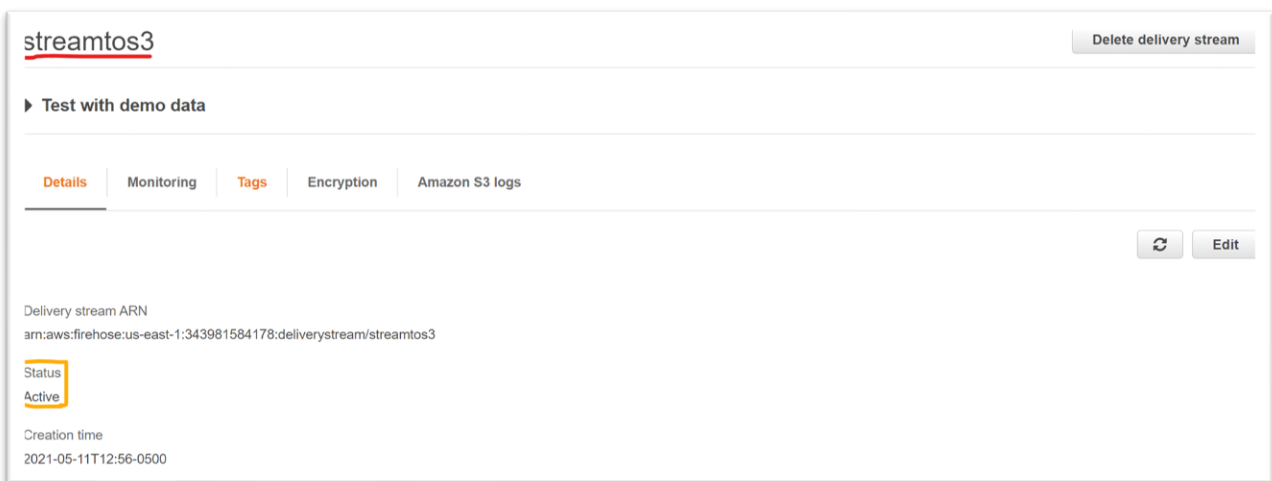


The above graphs show around 82.4k bytes of data was streamed and there is a latency of 87.8 milliseconds latency.



The above graphs show around 101 units of data were successfully streamed and almost 0 units of data were failed.



Now that the data is streaming, it should be delivered to a S3 bucket for querying in Athena. Amazon Data Firehose does the job. For this a delivery stream should be created.




A delivery stream named '**streamtos3**' is created and the '**Status**' of this stream is '**Active**' (orange box). The delivery stream delivers the data to S3 bucket as we mentioned the destination of this stream to be 'big-data-project-spring2021'.

Amazon S3 destination

S3 bucket

big-data-project-spring2021 ▼  

[View big-data-project-spring2021 in S3 console](#) 

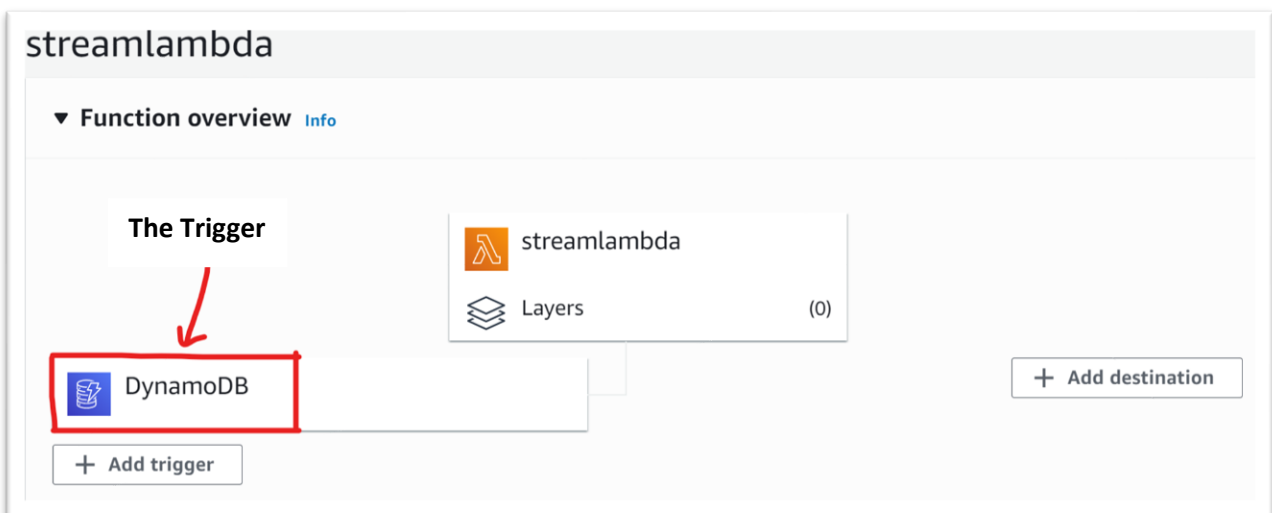
Backup S3 bucket prefix - *optional*

crawledstream

Backup S3 bucket error prefix - *optional*

crawledstreamerror

Whenever new data is added to DynamoDB the delivery stream should be activated. This can be done by another lambda function.

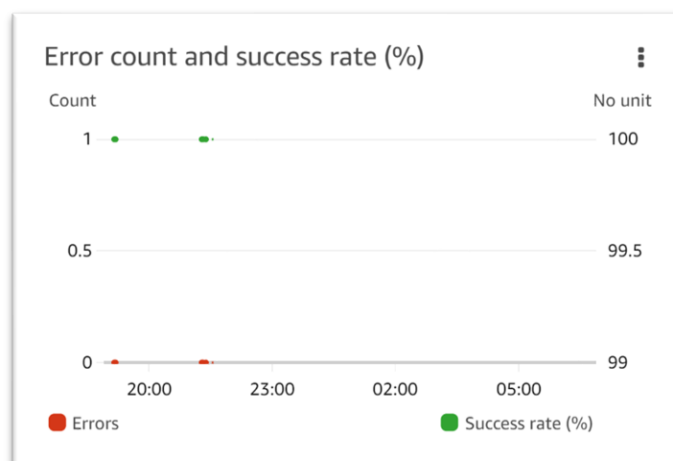


The 'pmsm_temperature' table of DynamoDB is given as a trigger to the lambda function by setting the Batch window to 5 meaning data will start streaming when 5 or more units of data is loaded into the table.

```
index.js
1 'use strict';
2
3 const AWS = require('aws-sdk');
4 var parse = AWS.DynamoDB.Converter.output;
5 const firehose = new AWS.Firehose({ region: 'us-east-1' });
6
7 exports.handler = (event, context, callback) => {
8
9     var fireHoseInput = [];
10
11     event.Records.forEach((record) => {
12
13         console.log(record);
14
15         if ((record.eventName == "INSERT") || (record.eventName == "MODIFY"))
16             fireHoseInput.push({ Data: JSON.stringify(parse({ "M": record.dynamodb.NewImage }))) });
17     });
18
19     var params = {
20         DeliveryStreamName: 'streamtos3',
21         Records: fireHoseInput
22     };
23
24     if(fireHoseInput.length != 0)
25     {
26         firehose.putRecordBatch(params, function (err, data) {
27             if (err) console.log(err, err.stack); // an error occurred
28             else console.log(data); // successful response
29         });
30     }
31     else
32     {
33         console.log("No data to transmit");
34     }
35     callback(null, 'Successfully processed records.');
```

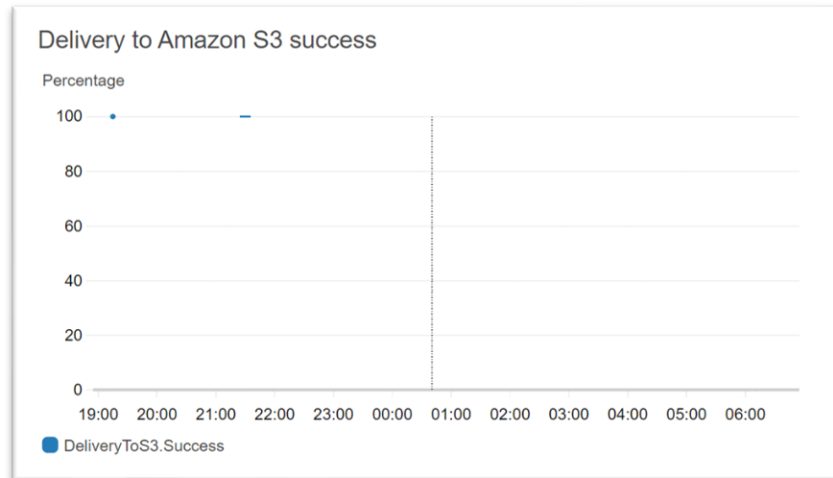
The lambda function **'streamlambda'** is triggered when more than 5 units of data is loaded into the DynamoDB table. Streaming happens only when data is inserted or modified in the table (red box) through the delivery stream **'streamtos3'** (orange box) which we created in the previous step.

The success rate and error count of streaming can be seen from the **'Metrics'** section of lambda function.



By the above graph, we can say that all the new data in 'pmsm_temperature' table is delivered successfully as there is **100% success rate and 0% errors**.

'Monitoring' section of Firehose has data that ensures the data is delivered to S3 bucket.



The above graph indicates there is **100% success in delivering the data to S3 bucket** justifying the 100% success rate of streaming of lambda function.



The above graph indicated that more than 65 records are delivered to S3 bucket.

The data is stored in the S3 bucket in parquet form. To query in Athena, the data must be crawled using the AWS Glue Crawler.

Crawlers > froms3

[Run crawler](#) [Edit](#)

Name	froms3
Description	
Create a single schema for each S3 path	false
Security configuration	
Tags	-
State	Ready
Schedule	
Last updated	Tue May 11 17:39:15 GMT-500 2021
Date created	Tue May 11 17:39:15 GMT-500 2021
Database	pmsmdatas3athena
Table prefix	athenadata
Service role	service-role/AWSGlueServiceRole-crawls3at
Selected classifiers	
Data store	S3
Include path	s3://big-data-project-spring2021/crawledstream2021
Connection	
Exclude patterns	
Configuration options	
Schema updates in the data store	Update the table definition in the data catalog.
Object deletion in the data store	Mark the table as deprecated in the data catalog.

A crawler **'froms3'** (red box) is created mentioning the s3 bucket (orange box) where the DynamoDB data was delivered by firehose delivery stream. This crawls the data from S3 and enables it to be queried by Athena.

To query data in Athena, it must be connected to a data source. The data source is the AWS Data Catalog where the database created by the crawler in the previous step exists.

[Refresh](#)

Data source [Connect data source](#)

AwsDataCatalog

Database

pmsmdatas3athena

Filter tables and views...

▼ **Tables (1)** [Create table](#)

▶ athenadatacrawledstream2021 (Partitioned) ⋮

The query,

```
New query 1 New query 4 ✖️ New query 5 ✖️ +  
1 SELECT * FROM "pmsmdatas3athena"."athenadatacrawledstream2021" limit 10;
```

Query 1

Gives the following output,

Results													
	stator_tooth	u_q	torque	ambient	i_d	stator_yoke	stator_winding	motor_speed	profile_id	i_q	u_d	id	pm
1	-1.2224278	-0.78289163	-2.0173435	-2.064073	1.0294477	-1.8304	0.3327715	-0.24943107	4	-0.24581794	-1.3018217	3	-2.5226
2	-1.2224296	-0.74922806	-2.0172427	-2.0621152	1.0304929	-1.8330117	0.33501354	-0.24791418	4	-0.24616154	-1.3020816	7	-2.5225
3	-1.2224286	-0.7629362	-2.017884	-2.0625494	1.0310309	-1.8319309	0.33490124	-0.248197	4	-0.24634062	-1.3030168	6	-2.5222
4	-1.2224283	-0.72371286	-2.0176919	-2.0620575	1.029509	-1.8314928	0.33539978	-0.24791297	4	-0.24583231	-1.3045602	11	-2.5226
5	-1.2224288	-0.71775365	-2.017435	-2.0622487	1.0293863	-1.8318189	0.33443072	-0.24772124	4	-0.24580358	-1.3043374	12	-2.5226
6	-1.2224314	-0.72712964	-2.0181801	-2.062317	1.0296358	-1.8314383	0.3359881	-0.24829444	4	-0.24588774	-1.3056333	10	-2.5226
7	-1.2224321	-0.7384499	-2.0172133	-2.0619526	1.0301074	-1.8321822	0.3362563	-0.24832098	4	-0.24603489	-1.3051548	8	-2.5228
8	-1.2224282	-0.75214297	-2.0180326	-2.0661428	1.0295724	-1.8314217	0.3279352	-0.2501821	4	-0.24586003	-1.2978575	1	-2.5220
9	-1.2224293	-0.77126324	-2.0176313	-2.0648587	1.029509	-1.8309687	0.3296648	-0.2491333	4	-0.24583231	-1.2976865	2	-2.5224
10	-1.2224315	-0.7309097	-2.0177386	-2.062443	1.0298513	-1.8315759	0.3349053	-0.24778472	4	-0.24598087	-1.3037902	9	-2.5228

Some queries are necessary to do analysis in QuickSight. Some of the queries include.

```
1 select pm, sum(pm) as sum_pm_temp, sum(stator_tooth) as sum_stator_tooth_temp, sum(stator_winding) as sum_stator_winding_temp from  
2  
3 "pmsmdatas3athena"."athenadatacrawledstream2021" group by pm;
```

Query 2: To compare temperature of permanent magnet, stator tooth and stator winding

```
1 select profile_id, count(profile_id) from "pmsmdatas3athena"."athenadatacrawledstream2021" group by profile_id;
```

Query 3: count of readings in each batch

```
1 select profile_id, sum(motor_speed) as speed from "pmsmdatas3athena"."athenadatacrawledstream2021" group by profile_id;
```

Query 4

```
1 select profile_id, sum(coolant) as coolant_temp from "pmsmdatas3athena"."athenadatacrawledstream2021" group by profile_id;
```

Query 5

```
1 select profile_id, sum(ambient) as coolant_temp from "pmsmdatas3athena"."athenadatacrawledstream2021" group by profile_id;
```

Query 6

Queries 4, 5 and 6 are needed for comparative analysis on each other.

QuickSight Analysis

There are batches of readings taken from PMSM machine, the below graph shows the total number of reading taken for every batch.

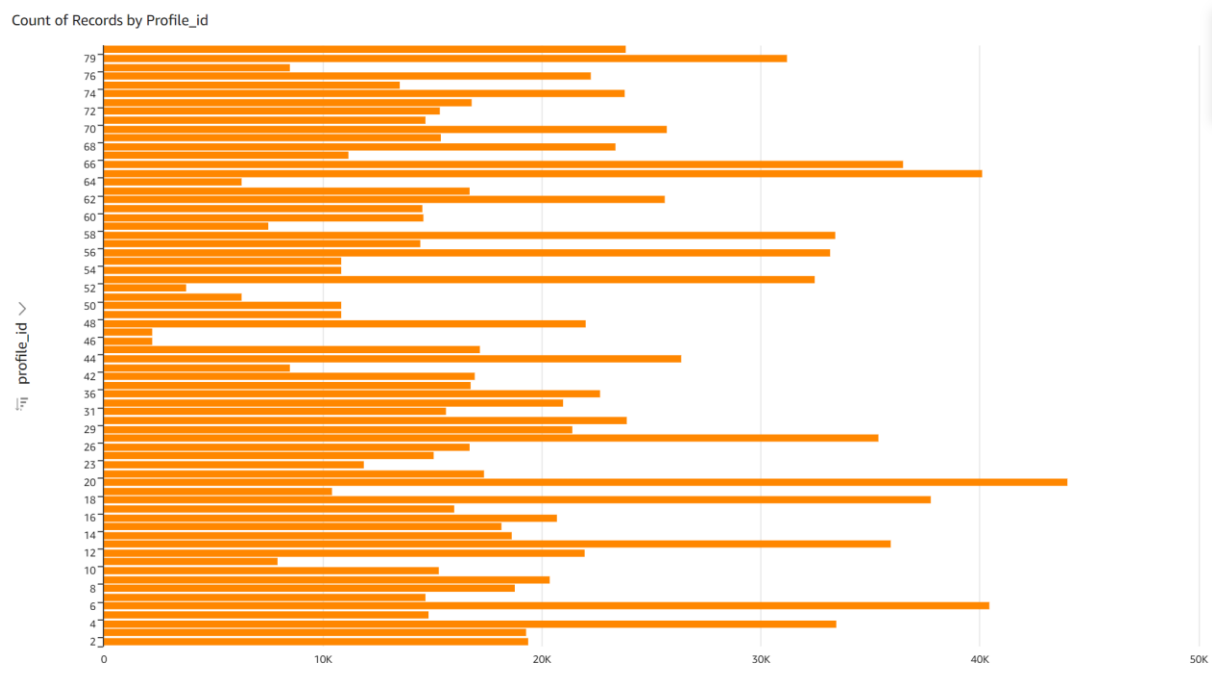


Fig: Count of readings for each batch

The temperature of every part of machine are important to make the machine function effectively. Every part should maintain equal range of temperatures.

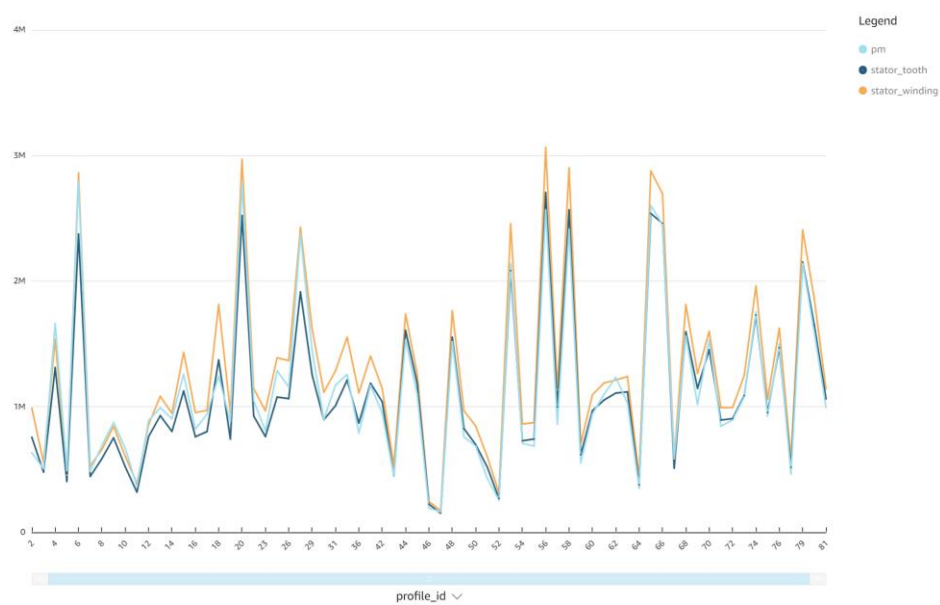


Fig: Comparing temperature readings of the three components

The temperature readings of permanent magnet, stator tooth and stator winding are plotted in the above graph. We can observe that there is an equal or same trend followed by all the three parts of the machine, maintaining the consistency. This says all three parts are working in harmony.

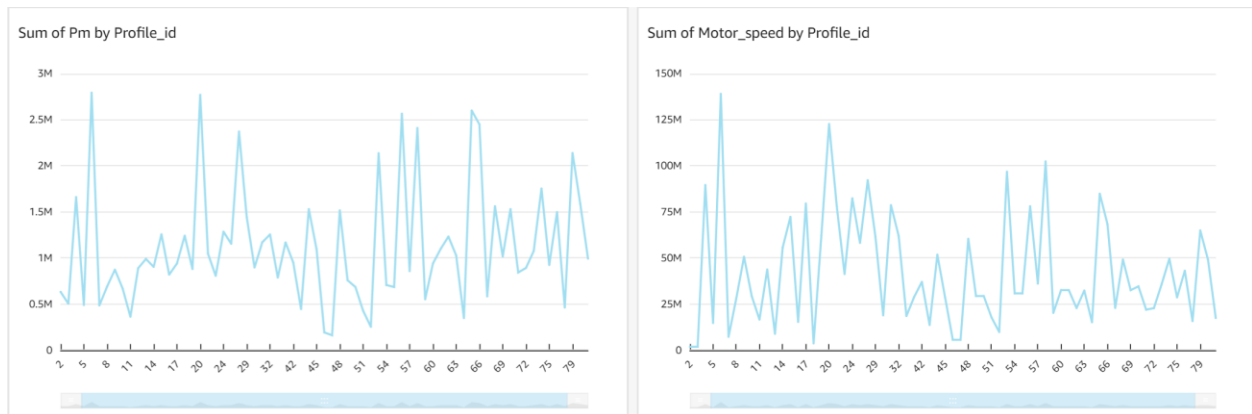


Fig: temperature of permanent magnet according to motor speed

For the above figure, the left-side graph indicates temperature of permanent magnet and the right-side graph is the motor speed. In the initial batches, the temperature of permanent magnet is increasing along the motor speed. As it goes down the graph, even though the motor speed is decreasing temperature of permanent magnet is decreasing very slowly. It is because the temperature increased rapidly cannot to decreased fast.

Now that the temperature is increasing, there is a need to cool the machine whenever a certain part is getting heated. The below graph shows that the temperature of the machine is being maintained through a coolant when permanent magnet's temperature is increasing.

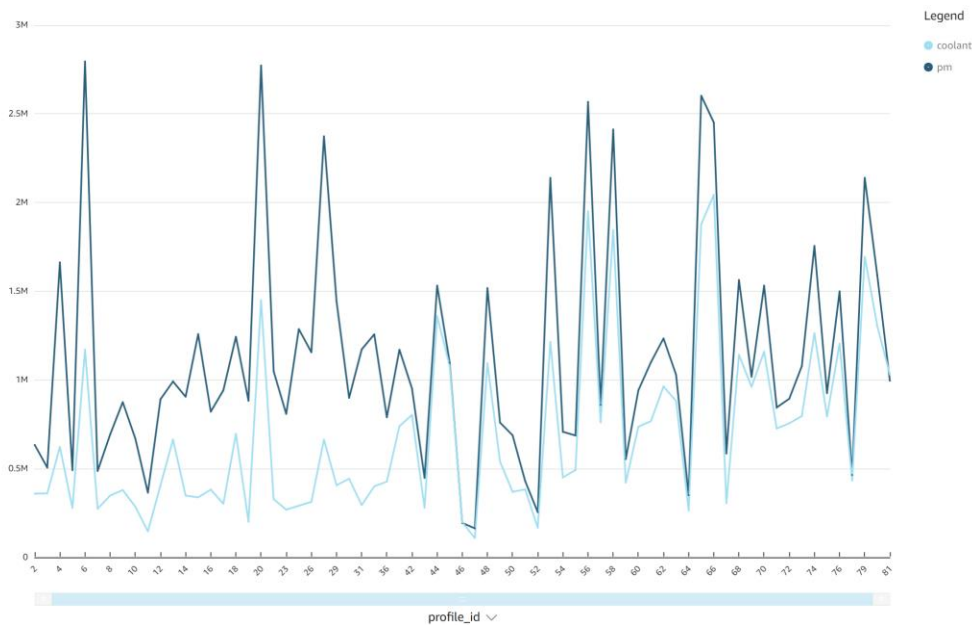


Fig: Coolant temperature according to permanent magnet temperature

Based on the permanent magnet temperature, we can set the coolant temperature appropriately. This data can be used to predict how much coolant temperature to set in future using time series analysis in future.

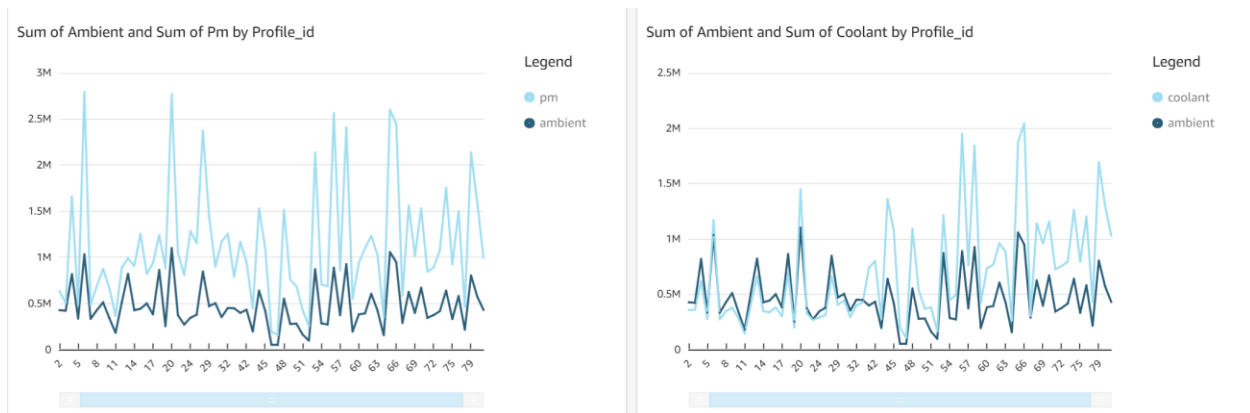


Fig: temperature of permanent magnet according to ambient temperature (right) coolant temperature according to ambient temperature (left)

The temperature of permanent magnet also depends on ambient (room) temperature (left side graph) and the coolant temperature should be maintained according to the ambient temperature (right side graph).

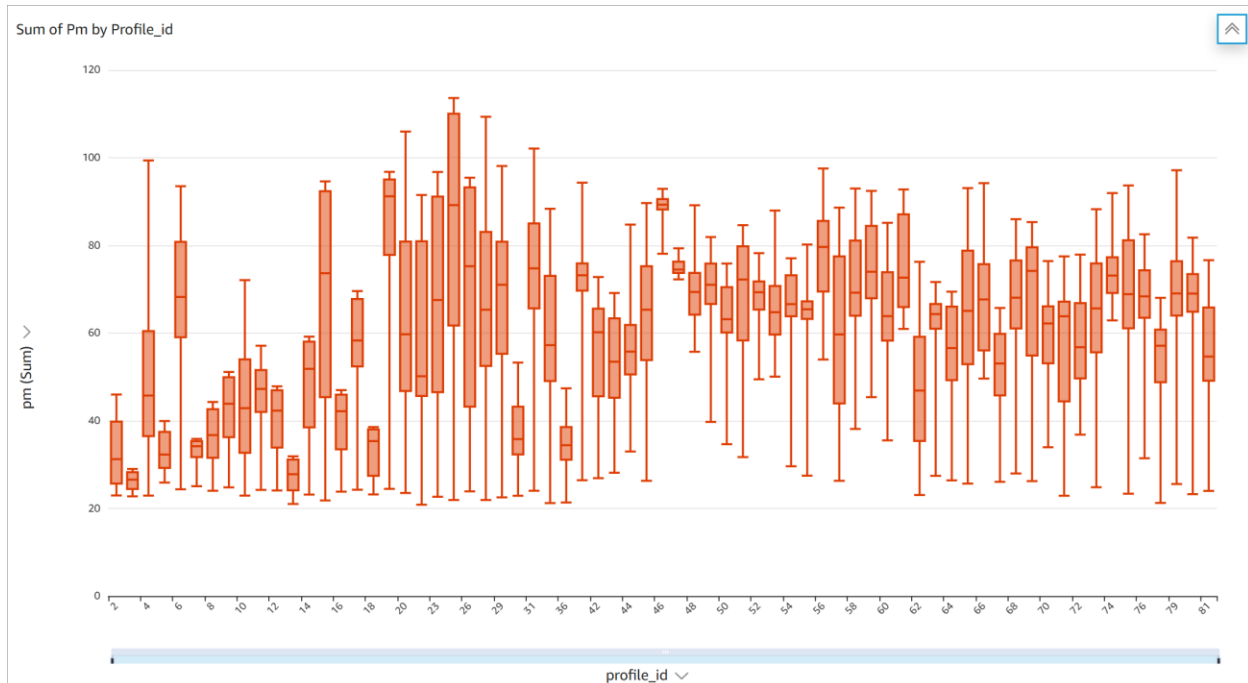


Fig: Box plot for permanent magnet's readings in every batch

The box plots of every batch of permanent magnet's temperature readings shows there are no outliers. This indicates there is no suspicious increase or decrease in temperature of permanent magnet.

Comparative Analysis

There are several reasons to consider DynamoDB over MongoDB and Kinesis Streaming over Kafka. The main reason would be the integration. As everything we choose is an AWS service, this makes it easy for a service to make contact to another service. Some of the other reasons are discussed below.

DynamoDB	MongoDB
DynamoDB is a fully managed service meaning it reduces the time spent on operations, no hardware provisioning, no cluster scaling. The focus is entirely on application (data pipeline here).	MongoDB is exact opposite of this. It has an infrastructure to build, clusters to create and manage, replication of data and many more. It needs an operations person to take care of all this on top of an application designer.
DynamoDB has out of the box security. The security model is based on Identity and Access Management (IAM). This enables a secure access to AWS services and resources.	MongoDB is secure, but the default configuration is not secure. It is potentially vulnerable to breaches as it does not provide out of the box security.

While building a Data Pipeline there will be so many leaking points as the data passes through various services. The data can be breached easily if there is no proper security. This is one of the strong reasons to choose DynamoDB over MongoDB.	
<p>DynamoDB supports key-value queries. Queries involving graph traversals, aggregations support of other AWS services is required which increases the cost.</p> <p>As our data pipeline, does not need any of the graph traversal operations. It is better to stick with DynamoDB instead of managing the clusters in MongoDB.</p>	The query language of MongoDB which is not key value helps in analyzing the data in many ways; Graph traversal, geospatial queries etc.,
<p>One of the disadvantages of being a fully managed service is we cannot tune the database elements such as index use, data models.</p> <p>As the data pipeline doesn't necessarily need a tuning to the database sticking to the fully managed service is better than posing a security threat to the pipeline as pipeline is vulnerable to it.</p>	If necessary, MongoDB can obtain deeper levels of performance metrics granularity for optimization and tuning purposes. Some of the performance metrics include Resource Utilization, throughput metrics, Errors etc.,
For projects with unpredictable demand, DynamoDB is the safest bet to cope with the growth of the market.	For small to medium scaled applications, MongoDB meets scalability and throughput requirements.
Netflix, Medium, New Relic are some of the companies which use DynamoDB.	Uber, Lyft, Code Academy are some of the companies which use MongoDB.

Kinesis	Kafka
Data is stored as Shards in Kinesis.	Data is stored as partitions in Kafka.
Kinesis supports a variety of SDKs like Java, Android, .NET, Go	Kafka supports only Java
Basic Skill level is required to create a Kinesis Data Stream making it easy to create one and stream data for a data pipeline	Advanced skill level is required to create a Kafka Steam.
Kinesis can write synchronously to 3 devices at a time making it easy for data pipeline to make different kind of analysis at same time.	Kafka has fewer limitations. It cannot do synchronous streaming.

Kinesis's Data Retention Period is only 7 days. This doesn't bother the pipeline built because all the data is being stored in DynamoDB table.	Kafka's Data Retention period is configurable.
Kinesis is just pay and use	Kafka requires human support for cluster management and installation.

Project Plan and Contributions

WBS	Task Name	Notes	Contributions
0	Downloading the Dataset		Everyone
1	Preparation of Project	Architecture Diagram	Everyone
2	Setting up the environment	Creating various accounts necessary and installing required tools	Jawhar, Sai Charan
3	Ingesting Data	Loading Data into DynamoDB Table	Sai Charan
4	Lambda Function 1	A trigger to ingest data into DynamoDB table when new batch of data arrives	Rajesh Kumar
6	Kinesis Data Streams	Data Streaming from DynamoDB table	Jawhar, Rajesh Kumar
7	Kinesis Data Firehose	A gateway or a pipe to place the streaming data to S3 bucket	Sai Charan, Jawhar
8	Lambda function 2	A trigger function to deliver data to S3 bucket via Firehose when new data is added to DynamoDB	Sai Charan, Jawhar
9	Crawler	A crawler to crawl the data from S3	Rajesh Kumar
10	Athena Queries	Necessary data is extracted by using some queries	Rajesh Kumar, Sai Charan
11	QuickSight Analysis	Data is transformed into various charts to gain insights	Rajesh Kumar

Conclusion

A data pipeline is constructed making use of bunch of AWS services to automate analysis process. A comparative analysis is made between DynamoDB and MongoDB, Kafka and Kinesis Stream to give advantages over one another in using them to construct a pipeline. Lastly, visual analysis of readings is made using QuickSight in real-time.

References

[1] Dataset:

<https://www.kaggle.com/wkirsnn/electric-motor-temperature>

[2] P. O Donovan, K. Leahy, K. Bruton, "An Industrial big data pipeline for data-driven analytics maintenance application in large-scale smart manufacturing facilities" Published in Springer Journal of Big Data, Article Number: 25, Year: 2015.

[3] Eveline van Stijn, David Hesketh, Yao-Hua Tan, Bram Klievink, Sietse Overbeek, Frank Heijmann, Markus Pikart, Tom Butterly, "The Data Pipeline" Conference paper for the United Nations Global Trade Facilitation Conference 2011.

[4] About Data Pipeline:

<https://hazelcast.com/glossary/data-pipeline/>

[5] Stream Processing:

<https://hazelcast.com/glossary/stream-processing/>

[6] Design Strategies for building a Data Pipeline:

<https://medium.com/@mrashish/design-strategies-for-building-big-data-pipelines-4c11affd47f3>

[7] A trigger to lambda function:

[https://www.tutorialspoint.com/aws_lambda/aws_lambda_using_lambda_function_with_ama
zon_dynamodb.htm](https://www.tutorialspoint.com/aws_lambda/aws_lambda_using_lambda_function_with_amazon_dynamodb.htm)

[8] Crawling Data:

[http://raaviblog.com/how-to-create-aws-glue-crawler-to-crawl-amazon-dynamodb-and-
amazon-s3-data-store/](http://raaviblog.com/how-to-create-aws-glue-crawler-to-crawl-amazon-dynamodb-and-amazon-s3-data-store/)