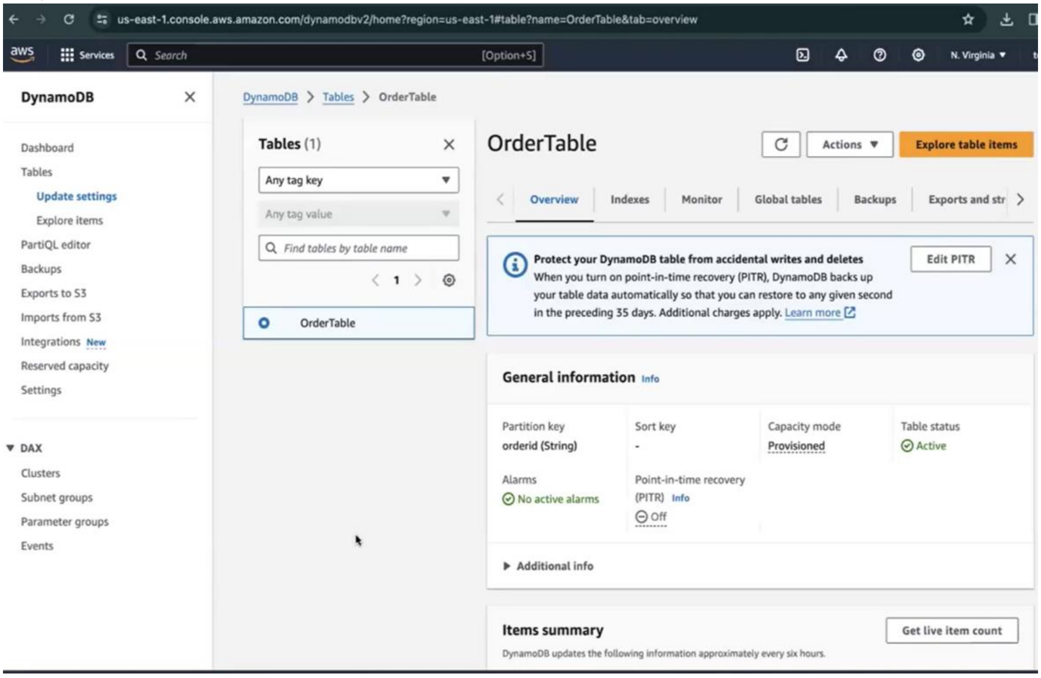
# Sales Data Projection

1. this was a sales data streaming project for processing the data from Dynamo db to destination S3 target path.
2. there was some source from where records(orderid, product_name, quantity, price) were being written into the nosql dynamodb(which is a key value based db).



OrderTable

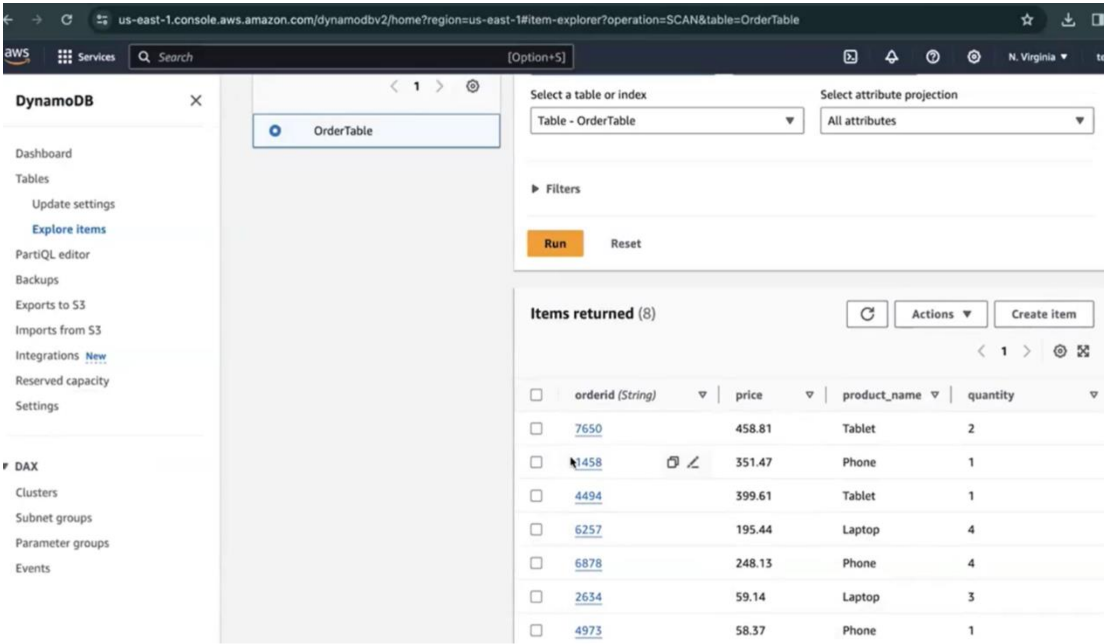
**Protect your DynamoDB table from accidental writes and deletes**
When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 35 days. Additional charges apply. Learn more

**General information** Info

Partition key: orderid (String)
Sort key: -
Capacity mode: Provisioned
Table status: Active
Alarms: No active alarms
Point-in-time recovery (PITR) Info: Off

**Items summary**
DynamoDB updates the following information approximately every six hours.

3. we created a dynamodb table where we targeted particular partition key(orderid) and the entire data(orderid, product_name, quantity, price) used to get ingested as a value there. That's how we were publishing a record in dynamodb.



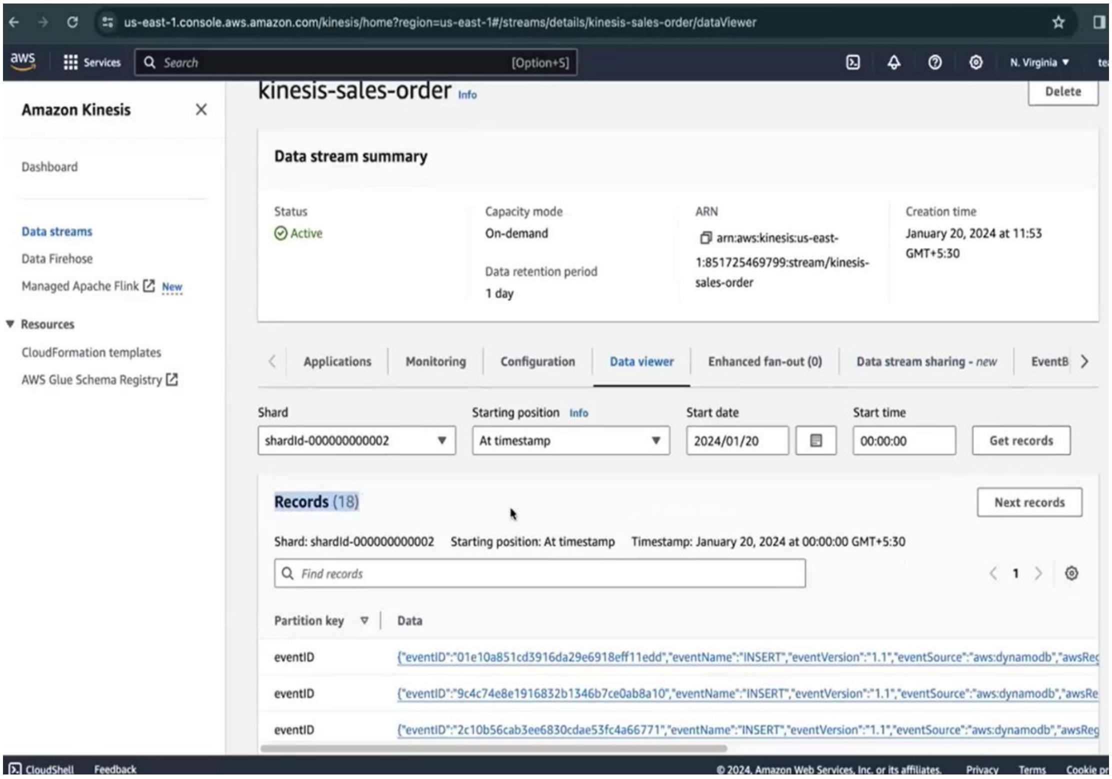
Select a table or index: Table - OrderTable
Select attribute projection: All attributes

Items returned (8)

| orderid (String) | price | product_name | quantity |
|---|---|---|---|
| 7650 | 458.81 | Tablet | 2 |
| 1458 | 351.47 | Phone | 1 |
| 4494 | 399.61 | Tablet | 1 |
| 6257 | 195.44 | Laptop | 4 |
| 6878 | 248.13 | Phone | 4 |
| 2634 | 59.14 | Laptop | 3 |
| 4973 | 58.37 | Phone | 1 |

4. next our task was to capture changes happening in table. Records were getting inserted, updated/deleted to table in real time. CDC is whatever changes are happening in real time we capture those changes. That's what CDC is. So we needed to enable dynamodb streams from 'Exports & Streams' option to perform CDC. As we were able to consume the CDC changes, hence created CDC pipeline.

5. then we set up a kinesis stream which is a queuing mechanism to hold our data. Kinesis stream has a concept of shards just like publishing data to partitions. Data here arrives at kinesis stream in different shard.



kinesis-sales-order Info

**Data stream summary**

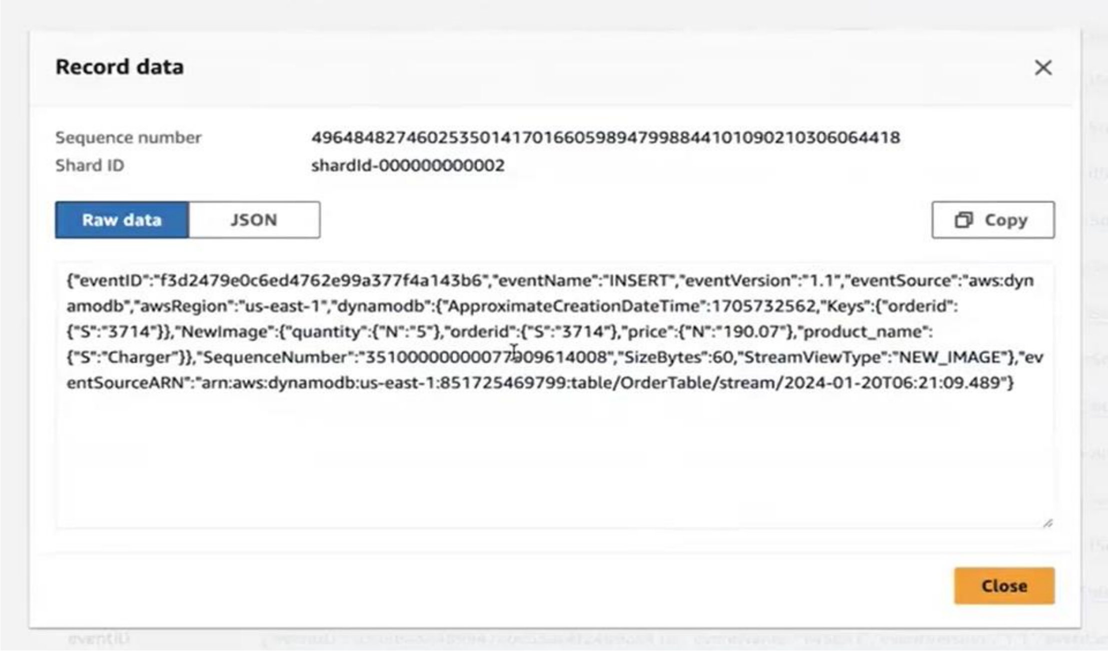
Status: Active
Capacity mode: On-demand
Data retention period: 1 day
ARN: arn:aws:kinesis:us-east-1:851725469799:stream/kinesis-sales-order
Creation time: January 20, 2024 at 11:53 GMT+5:30

Shard: shardId-000000000002
Starting position: At timestamp
Start date: 2024/01/20
Start time: 00:00:00

**Records (18)**

Shard: shardId-000000000002 Starting position: At timestamp Timestamp: January 20, 2024 at 00:00:00 GMT+5:30

| Partition key | Data |
|---|---|
| eventID | {"eventID":"01e10a851cd3916da29e6918eff11edd","eventName":"INSERT","eventVersion":"1.1","eventSource":"aws:dynamodb","awsReg |
| eventID | {"eventID":"9c4c74e8e1916832b1346b7ce0ab8a10","eventName":"INSERT","eventVersion":"1.1","eventSource":"aws:dynamodb","awsRe |
| eventID | {"eventID":"2c10b56cab3ee6830cdae53fc4a66771","eventName":"INSERT","eventVersion":"1.1","eventSource":"aws:dynamodb","awsReg |

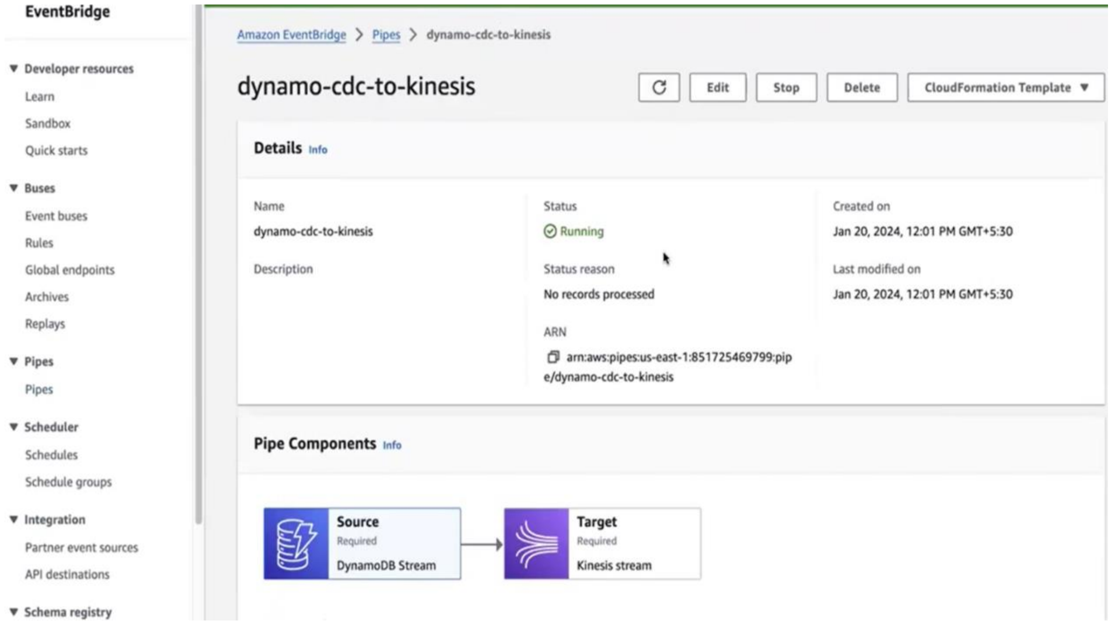


**Record data**

Sequence number: 49648482746025350141701660598947998844101090210306064418
Shard ID: shardId-000000000002

```
{"eventID":"f3d2479e0c6ed4762e99a377f4a143b6","eventName":"INSERT","eventVersion":"1.1","eventSource":"aws:dyn
amodb","awsRegion":"us-east-1","dynamodb":{"ApproximateCreationDateTime":1705732562,"Keys":{"orderid":
{"S":"3714"}},"NewImage":{"quantity":{"N":"5"},"orderid":{"S":"3714"},"price":{"N":"190.07"},"product_name":
{"S":"Charger"}},"SequenceNumber":"351000000000077309614008","SizeBytes":60,"StreamViewType":"NEW_IMAGE"},"ev
entSourceARN":"arn:aws:dynamodb:us-east-1:851725469799:table/OrderTable/stream/2024-01-20T06:21:09.489"}
```

6. then we had a event bridge pipe in between dynamodb stream and kinesis stream. We set a batch size consisting of no of message to be sent to any kinesis stream shard. Set up required IAM role to eventbridge pipe to access dynamo stream & kinesis stream. Using event bridge we used to flow data to kinesis stream.



dynamo-cdc-to-kinesis

Details Info

Name
dynamo-cdc-to-kinesis

Description

Status
Running

Status reason
No records processed

ARN
arn:aws:pipes:us-east-1:851725469799:pipe/dynamo-cdc-to-kinesis

Created on
Jan 20, 2024, 12:01 PM GMT+5:30

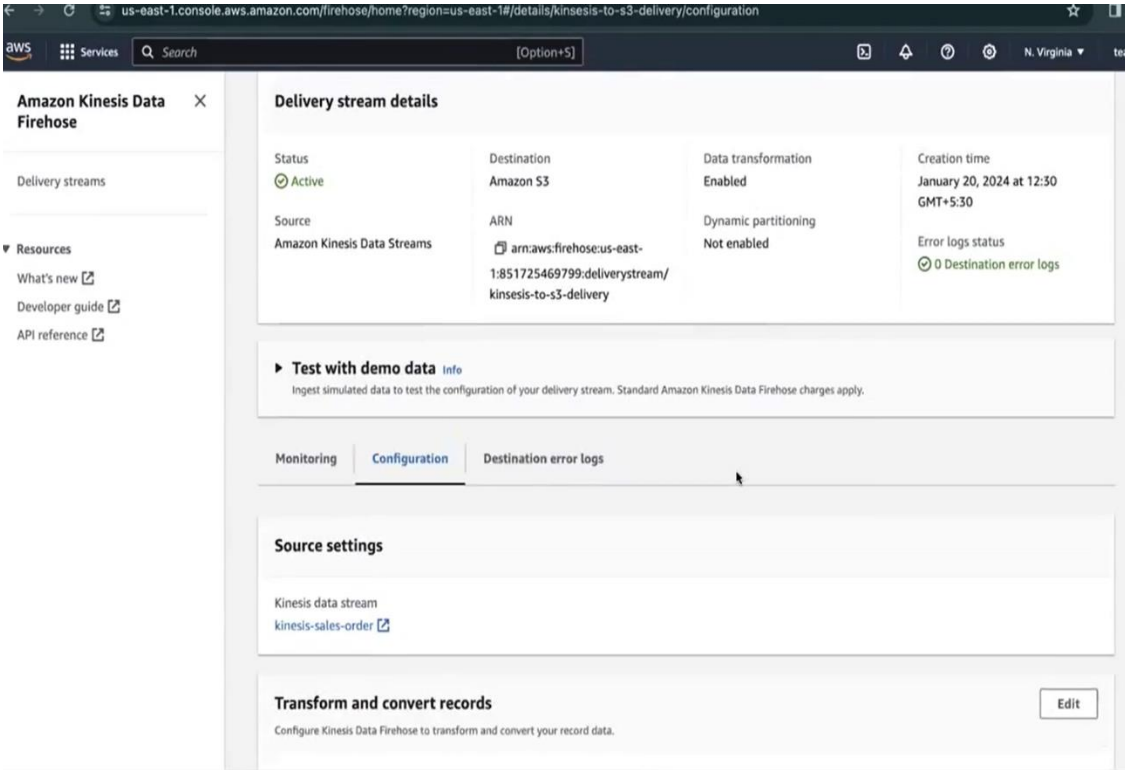
Last modified on
Jan 20, 2024, 12:01 PM GMT+5:30

Pipe Components Info

Source
Required
DynamoDB Stream

Target
Required
Kinesis stream

7. next we created a firehose stream which is a delivery stream. It processed data(like ingest, transform) and delivered streaming data to destinations(like data lakes, data warehouses). Here source is kinesis stream and destination is s3.



Delivery stream details

Status
Active

Source
Amazon Kinesis Data Streams

Destination
Amazon S3

ARN
arn:aws:firehose:us-east-1:851725469799:deliverystream/kinsesis-to-s3-delivery

Data transformation
Enabled

Dynamic partitioning
Not enabled

Creation time
January 20, 2024 at 12:30 GMT+5:30

Error logs status
0 Destination error logs

Test with demo data Info

Ingest simulated data to test the configuration of your delivery stream. Standard Amazon Kinesis Data Firehose charges apply.

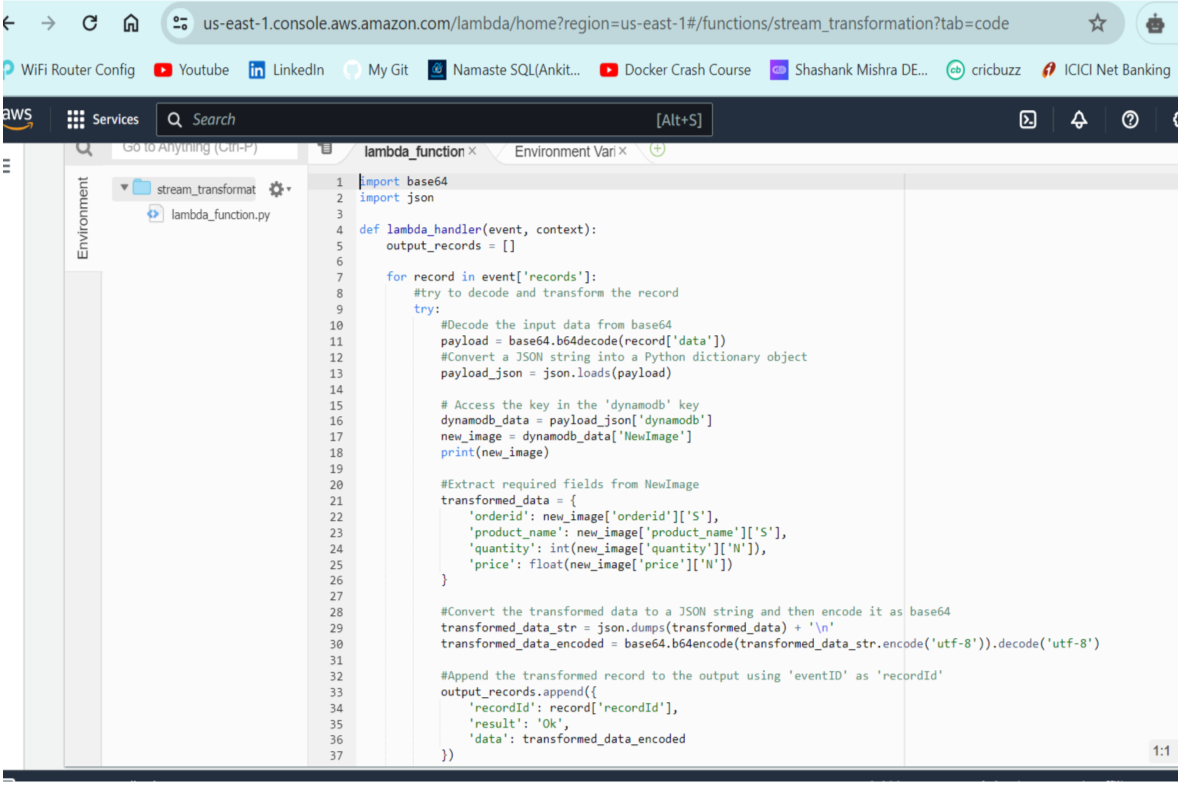
Source settings

Kinesis data stream
kinesis-sales-order

Transform and convert records

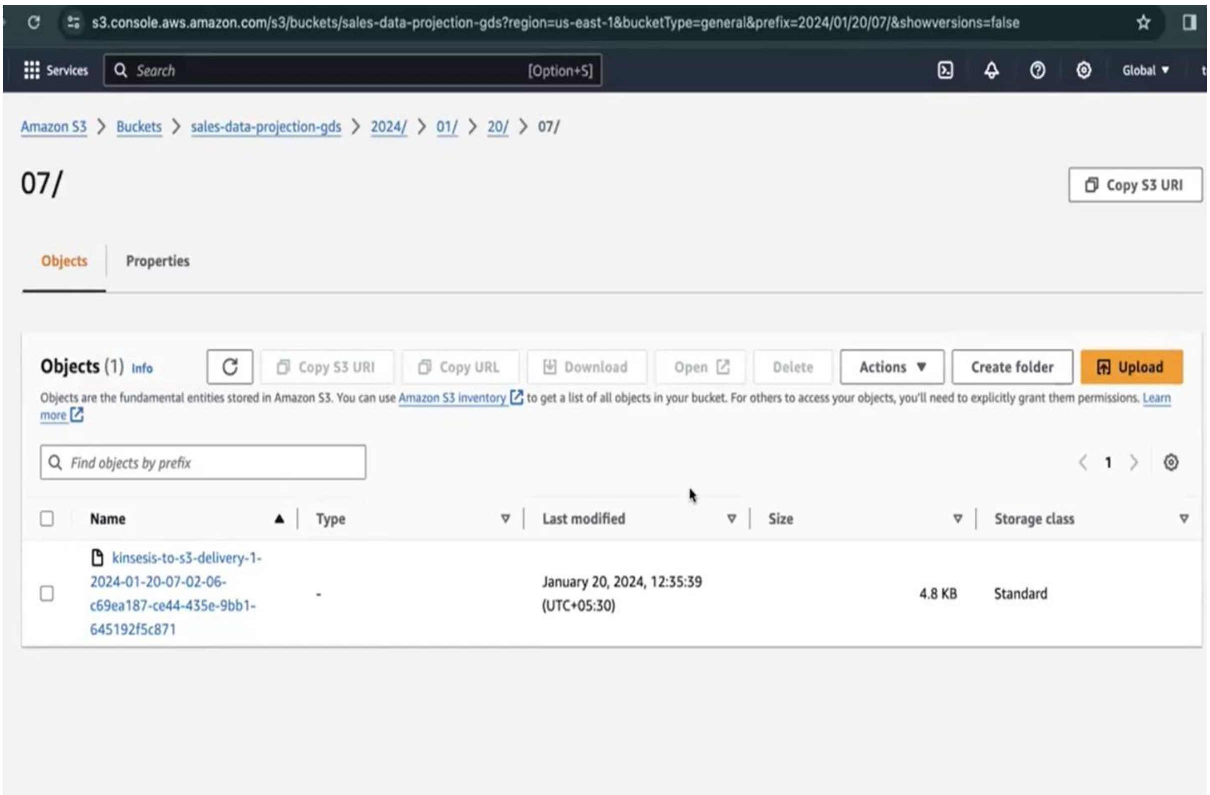
Configure Kinesis Data Firehose to transform and convert your record data.

8. using delivery stream we were batching the records and transforming those mini batch records with the help of lambda function.



```python
import base64
import json

def lambda_handler(event, context):
    output_records = []

    for record in event['records']:
        #try to decode and transform the record
        try:
            #Decode the input data from base64
            payload = base64.b64decode(record['data'])
            #Convert a JSON string into a Python dictionary object
            payload_json = json.loads(payload)

            # Access the key in the 'dynamodb' key
            dynamodb_data = payload_json['dynamodb']
            new_image = dynamodb_data['NewImage']
            print(new_image)

            #Extract required fields from NewImage
            transformed_data = {
                'orderid': new_image['orderid']['S'],
                'product_name': new_image['product_name']['S'],
                'quantity': int(new_image['quantity']['N']),
                'price': float(new_image['price']['N'])
            }

            #Convert the transformed data to a JSON string and then encode it as base64
            transformed_data_str = json.dumps(transformed_data) + '\n'
            transformed_data_encoded = base64.b64encode(transformed_data_str.encode('utf-8')).decode('utf-8')

            #Append the transformed record to the output using 'eventID' as 'recordId'
            output_records.append({
                'recordId': record['recordId'],
                'result': 'Ok',
                'data': transformed_data_encoded
            })
```

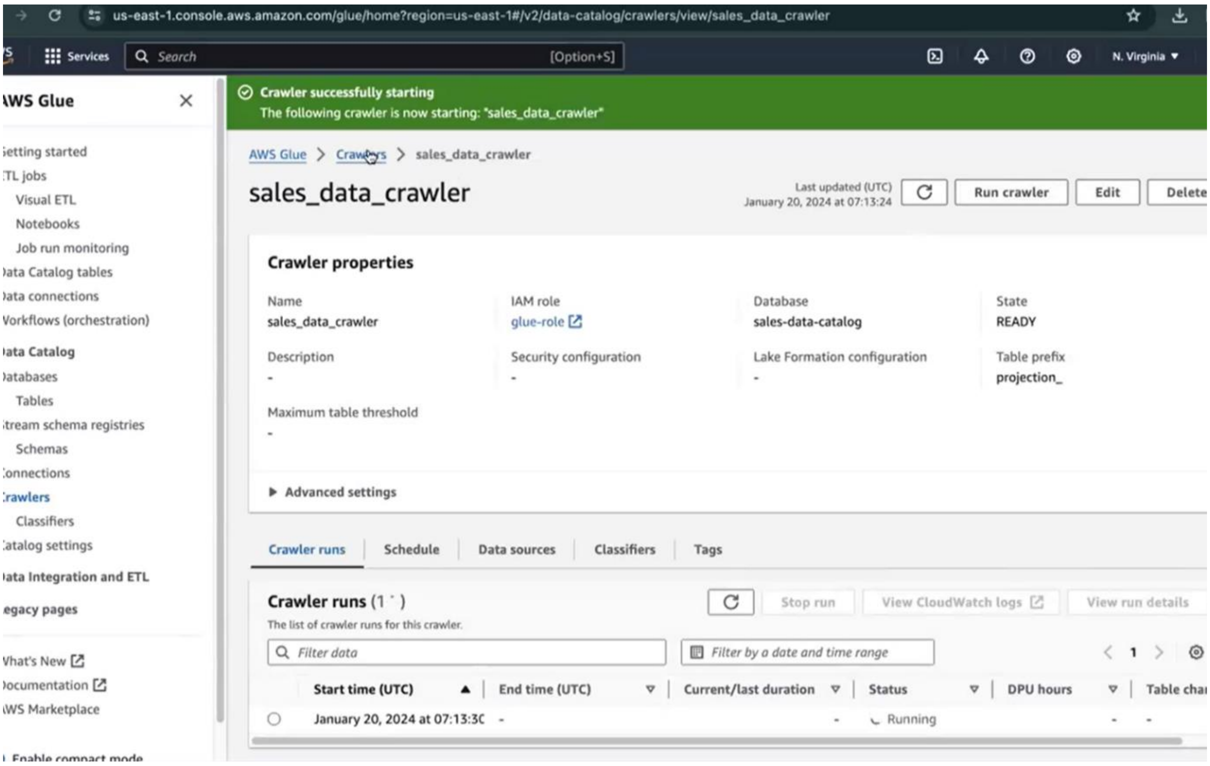
9. Once this is done, then delivering the transformed batch records to S3.


s3.console.aws.amazon.com/s3/buckets/sales-data-projection-gds?region=us-east-1&bucketType=general&prefix=2024/01/20/07/&showversions=false
Services
Search
[Option+S]
Global ▾
Amazon S3 > Buckets > sales-data-projection-gds > 2024/ > 01/ > 20/ > 07/
07/
Copy S3 URI
Objects
Properties
Objects (1) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions ▾
Create folder
Upload
Find objects by prefix
1

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 |
|---|---|---|---|---|---|
| □ | kinsesis-to-s3-delivery-1-2024-01-20-07-02-06-c69ea187-ce44-435e-9bb1-645192f5c871 | - | January 20, 2024, 12:35:39 (UTC+05:30) | 4.8 KB | Standard |

10. Lastly, we created crawler over s3 target path which resulting in catalog table creation. Also, needed to add classifier for json format(i.e. $.columns).



Crawler successfully starting
The following crawler is now starting: "sales_data_crawler"

sales_data_crawler

Last updated (UTC) January 20, 2024 at 07:13:24

Crawler properties

| Name | IAM role | Database | State |
|---|---|---|---|
| sales_data_crawler | glue-role | sales-data-catalog | READY |

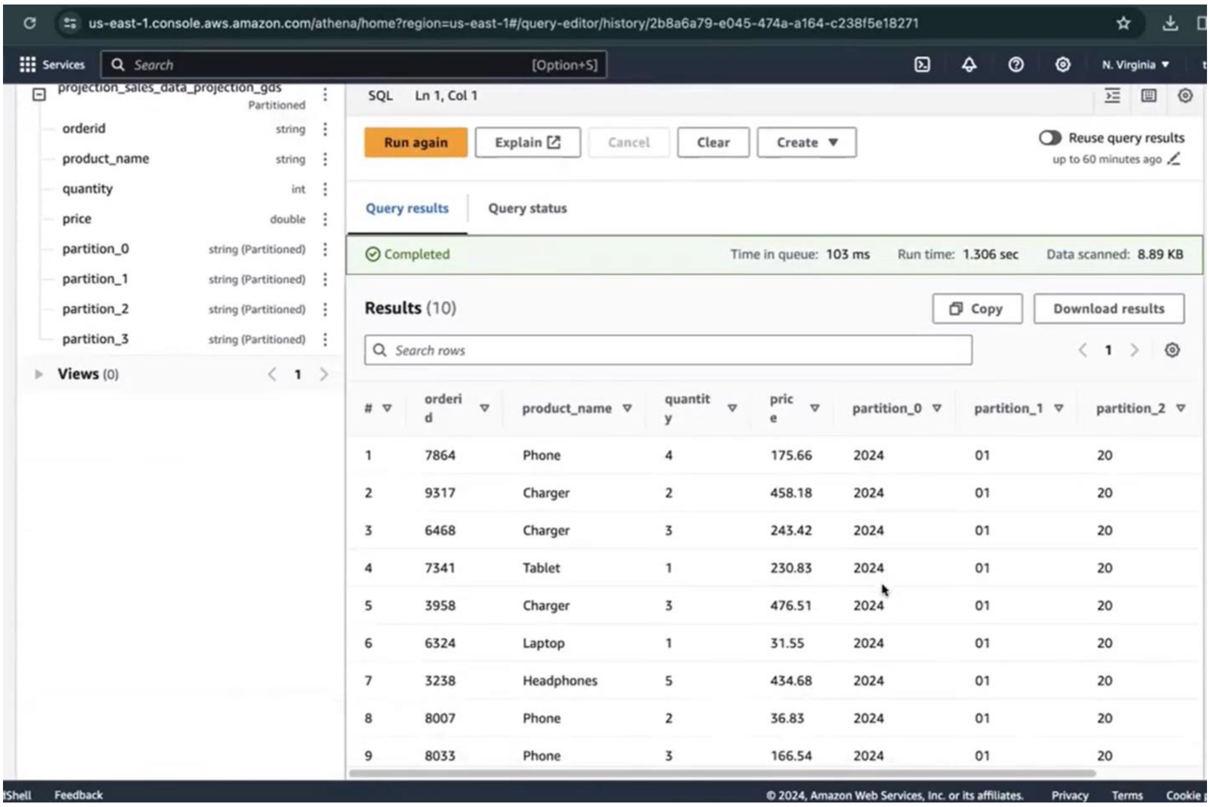
| Description | Security configuration | Lake Formation configuration | Table prefix |
|---|---|---|---|
| - | - | - | projection_ |

Maximum table threshold
-

Crawler runs (1 )
The list of crawler runs for this crawler.

| Start time (UTC) | End time (UTC) | Current/last duration | Status | DPU hours | Table cha |
|---|---|---|---|---|---|
| January 20, 2024 at 07:13:3C | - | - | Running | - | - |

11. using athena query we then did analysis



projection_sales_data_projection_gds
Partitioned

| | |
|---|---|
| orderid | string |
| product_name | string |
| quantity | int |
| price | double |
| partition_0 | string (Partitioned) |
| partition_1 | string (Partitioned) |
| partition_2 | string (Partitioned) |
| partition_3 | string (Partitioned) |

Completed  Time in queue: 103 ms  Run time: 1.306 sec  Data scanned: 8.89 KB

Results (10)

| # | orderid | product_name | quantity | price | partition_0 | partition_1 | partition_2 |
|---|---|---|---|---|---|---|---|
| 1 | 7864 | Phone | 4 | 175.66 | 2024 | 01 | 20 |
| 2 | 9317 | Charger | 2 | 458.18 | 2024 | 01 | 20 |
| 3 | 6468 | Charger | 3 | 243.42 | 2024 | 01 | 20 |
| 4 | 7341 | Tablet | 1 | 230.83 | 2024 | 01 | 20 |
| 5 | 3958 | Charger | 3 | 476.51 | 2024 | 01 | 20 |
| 6 | 6324 | Laptop | 1 | 31.55 | 2024 | 01 | 20 |
| 7 | 3238 | Headphones | 5 | 434.68 | 2024 | 01 | 20 |
| 8 | 8007 | Phone | 2 | 36.83 | 2024 | 01 | 20 |
| 9 | 8033 | Phone | 3 | 166.54 | 2024 | 01 | 20 |