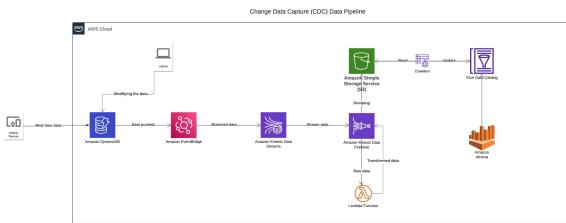


PS/Objective:

Implement a real-time data processing pipeline with Change Data Capture (CDC) on AWS to analyze sales data. The pipeline will utilize DynamoDB, Kinesis Streams, Kinesis Firehose, EventBridge, Lambda, S3, Athena and Glue Catalog. We're using a bunch of AWS services in this project, and it's going to be really interesting!

**Step1: - Setup DynamoDB**

Create a DynamoDB table that will act as our upstream database. Any changes to this data will be captured and analyzed downstream.

1. Select the DynamoDB service.
2. Click on "Create Table."
3. Give it a name.
4. Choose your partition key based on your dataset - in our case, "orderid."
5. Click on "Create Table"

The screenshot shows the AWS DynamoDB Tables page. A single table, "OrdersRawTable", is listed. It is active, has "orderid" as the partition key, and "0" as the sort key. The table uses "Provisioned (1)" for both read and write capacity. The "Actions" dropdown menu is open, showing options like "Edit", "Delete", and "Create table".

Now, let's enable it to capture any changes made to the data

1. Select the created table
2. Go to "Exports and streams" section
3. Navigate to "DynamoDB stream details"
4. Click on "Turn on" to enable the database to capture item-level changes in your table and push the changes to a DynamoDB stream.
5. Select view type as "New Image"
6. Now, click on "Turn on stream"

The screenshot shows the "DynamoDB stream details" page for the "OrdersRawTable". The table is currently active. The "Stream status" is set to "On". The "View type" is "New image". The "Latest stream ARN" is displayed as "arn:aws:dynamodb:ap-south-1:771822799155:table/OrdersRawTable/stream/2024-06-09T13:46:34.033". A "Turn off" button is visible in the top right corner.

Step2: Create a mock data generator to publish data in realtime to DynamoDB

Use the below script to publish the sales data, it uses help of boto3 sdk to achieve it

```

MockDataGenerator.py > ...
1 import boto3
2 import random
3 import time
4 from decimal import Decimal
5
6 session = boto3.Session(profile_name='default', region_name='ap-south-1')
7 dynamodb = session.resource('dynamodb')
8 table = dynamodb.Table('OrdersRawTable')
9
10 def generate_order_data():
11     orderid = str(random.randint(1,10000))
12     product_name = random.choice(['Laptop', 'Phone', 'Tablet', 'Headphones', 'charger'])
13     quantity = random.randint(1,5)
14     price = Decimal(str(round(random.uniform(10.0,500.0), 2 )))
15
16     return {
17         'orderid': orderid,
18         'product_name': product_name,
19         'quantity': quantity,
20         'price': price
21     }
22
23 def insert_into_dynamodb(data):
24
25     try:
26         table.put_item(Item = data)
27         print(f"Inserted data:{data}")
28
29     except Exception as e:
30         print(f"Error inserting data: {str(e)}")
31
32 if __name__ == '__main__':
33     try:
34
35         while True:
36             data = generate_order_data()
37             insert_into_dynamodb(data)
38             time.sleep(3)
39
40     except KeyboardInterrupt:
41         print("/n Script stopped by manual intervention")

```

Step3: Make changes to the data present on DynamoDB tables.

Once the data is populated in the DynamoDB tables.

1. Select the table
2. Click on "Explore Table Items"
3. Start editing the data in the table by hovering on the respective rows

	orderid (String)	price	product_name	quantity
3635	193.72	Phone	1	
6308	377.06	Headphones	2	
9681	408.91	Phone	2	
6500	286.41	Laptop	2	
4829	413.4	Headphones	3	
4697	161.58	charger	1	

Step4: Create Kinesis Stream

Kinesis Stream helps in collecting real time streaming data

1. Select Kinesis service
2. Choose Kinesis data Stream and click on create
3. Give it a name and leave rest of the options as default
4. Click on "Create data Stream"

Amazon Kinesis > Data streams

Data streams (1) Info		Process data in real time		Create a Firehose stream		Actions ▾		Create data stream	
<input type="text"/> Find data streams									
<input type="checkbox"/>	Name	Status	Capacity mode	Provisioned shards	Sharing policy	Data retention period	Encryption	Consumers with enhanced fan-out	
<input type="checkbox"/>	kinesisfor-sales-data	Active	On-demand	-	No	1 day	Disabled	0	

Step5: Create eventbridge pipe

We use eventbridge pipe to transfer the data from DynamoDB to Kinesis stream.

1. Select EventBridge service
2. Select EventBridge Pipes
3. Click on "Create pipe"
4. Choose DynamoDB as the source
5. Select the table to listen for updates on.
6. Choose "Latest" option for starting position - This means to start reading the data from the latest - Keep all items do not truncate other items present before consumer is created.
7. Click on "Next"
8. Do not include Filtering and Enrichment steps remove them and proceed to the Target section.
9. Under Target choose "Kinesis Stream"
10. Select the stream you have created
11. In partition key section - enter "\$.dynamodb.keys.orderId.N" - Given the event data stored this is how we parse for orderId in the input events
12. Now, click on "Create pipe"

Amazon EventBridge > Pipes > sales-ingestion-dynamodb-to-kinesis

sales-ingestion-dynamodb-to-kinesis [C](#) [Edit](#) [Stop](#) [Delete](#) [CloudFormation Template ▾](#)

Details Info	
Name sales-ingestion-dynamodb-to-kinesis	Status Running
Description	Status reason No records processed
	Created on Jun 9, 2024, 08:05 PM GMT+5:30
	Last modified on Jun 9, 2024, 08:39 PM GMT+5:30
ARN arn:aws:pipes:ap-south-1:771822799155:pipe/sales-ingestion-dynamodb-to-kinesis	

Pipe Components Info	
 Source Required DynamoDB Stream	 Target Required Kinesis stream

[Source](#) | [Target](#)

Step6: Create s3 bucket to stores transformed data

Amazon S3 > Buckets > kensis-firehose-destination

kenesis-firehose-destination [Info](#)

Objects	Properties	Permissions	Metrics	Management	Access Points																						
Objects (2) Info <table border="1"> <thead> <tr> <th>C</th> <th>Copy S3 URI</th> <th>Copy URL</th> <th>Download</th> <th>Open</th> <th>Delete</th> <th>Actions ▾</th> <th>Create folder</th> <th>Upload</th> </tr> </thead> <tbody> <tr> <td colspan="9">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</td> </tr> <tr> <td colspan="9"><input type="text"/> Find objects by prefix</td> </tr> </tbody> </table>	C	Copy S3 URI	Copy URL	Download	Open	Delete	Actions ▾	Create folder	Upload	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									<input type="text"/> Find objects by prefix								
C	Copy S3 URI	Copy URL	Download	Open	Delete	Actions ▾	Create folder	Upload																			
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																											
<input type="text"/> Find objects by prefix																											

Step7: Create transformation lambda

We get data into Firehose as metadata. We need to transform the data into readable JSON with only the required fields.

1. Create a lambda and use this code

[Lambda](#) > [Functions](#) > process-kinesis-firehose

process-kinesis-firehose

Throttle
Copy ARN
Actions ▾

Function overview
Info

Diagram
Template

process-kinesis-firehose

Layers (0)

+ Add trigger
+ Add destination

Description

-
Last modified

14 hours ago

Function ARN

arn:aws:lambda:ap-south-1:771822799155:function:process-kinesis-firehose

Function URL
Info

-

Code
Test
Monitor
Configuration
Aliases
Versions

```

MockDataGenerator.py • transformation_layer_with_lambda.py × temp.py
transformation_layer_with_lambda.py > ...
4 def lambda_handler(event, context):
5     output_records = []
6
7     for record in event['records']:
8         try:
9             # Decode the input data from base64
10            payload = base64.b64decode(record['data']).decode('utf-8')
11            payload_json = json.loads(payload)
12
13            # Access the data in the 'dynamodb' key
14            dynamodb_data = payload_json['dynamodb']
15            new_image = dynamodb_data['NewImage']
16
17            # Extract required fields from new image
18            transformed_data = {
19                'order_id': new_image['orderId'][['S']],
20                'product_name': new_image['product_name'][['S']],
21                'quantity': int(new_image['quantity'][['N']]),
22                'price': float(new_image['price'][['N']])
23            }
24
25            # Convert the transformed data to JSON string and then encode it as base64
26            transformed_data_str = json.dumps(transformed_data) + '\n'
27            transformed_data_encoded = base64.b64encode(transformed_data_str.encode('utf-8')).decode('utf-8')
28
29            # Append the transformed record to the output using 'recordId' from the event
30            output_records.append({
31                'recordId': record['recordId'],
32                'result': 'Ok',
33                'data': transformed_data_encoded
34            })
        
```

Step8: Create Firehose stream

Kinesis Firehose stream loads streaming data into data stores effortlessly. With Kinesis Firehose, we put the data into Lambda to transform it. Once the data is transformed and sent back to Firehose, it dumps it into S3.

We batch the data that is coming from Kinesis stream and do the next steps

1. Select Kinesis service.
2. Choose "Data firehose"
3. Under source choose "Kinesis Data Stream"
4. Under destination choose "S3"
5. Give you Firehose a name
6. Check the option - Transform source records with AWS Lambda
7. And, choose the above created lambda - By this setting the firehose will push the data into lambda and we do transformation there and lambda pushes back into firehose.
8. Select buffer size and interval
 - a. Buffer size - The AWS Lambda function has a 6 MB invocation payload quota. Your data can expand in size after it's processed by the AWS Lambda function. A smaller buffer size allows for more room should the data expand after processing.
 - b. Buffer Interval - The period of time during which Amazon Data Firehose buffers incoming data before invoking the AWS Lambda function. The AWS Lambda function is invoked once the value of the buffer size or the buffer interval is reached.
9. Next, choose the destination bucket created above.
10. Now click on "Create Firehose stream"

Step9: See the data on s3

Keep the mock data generator running, then after certain time approx. 3-4 mins you will see the data on s3

14/

[Copy S3 URI](#)[Objects](#) [Properties](#)**Objects (4) [Info](#)**
 [Copy S3 URI](#)
 [Copy URL](#)
 [Download](#)
 [Open](#)
[Delete](#)
[Actions ▾](#)
[Create folder](#)
 [Upload](#)
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](#) Find objects by prefix

< 1 > ⚙

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	kinesis-to-s3-nrt-batch-1-2024-06-12-14-27-33-1a1459ef-fae-4bb9-a8d1-039423047631	-	June 12, 2024, 20:02:35 (UTC+05:30)	8.6 KB	Standard
<input type="checkbox"/>	kinesis-to-s3-nrt-batch-1-2024-06-12-14-27-33-cbea3e69-ecd7-4764-a2f8-9f1b69f83a21	-	June 12, 2024, 20:02:35 (UTC+05:30)	7.8 KB	Standard
<input type="checkbox"/>	kinesis-to-s3-nrt-batch-1-2024-06-12-14-27-33-d5bdc0b3-74e6-4273-afe6-be95ebc5b95c	-	June 12, 2024, 20:02:34 (UTC+05:30)	7.1 KB	Standard
<input type="checkbox"/>	kinesis-to-s3-nrt-batch-1-2024-06-12-14-27-34-946dc0c2-c517-423c-9f12-3981360hf198	-	June 12, 2024, 20:02:35 (UTC+05:30)	7.1 KB	Standard

So, now you pipeline is able to capture the data push it to downstream applications correctly, now lets create a data catalog on top of this. To query the data with the help of Athena

Step10 : Create a data catalog to query the data

1. Create a database and tables on Glue
 - a. Check my previous projects to see how to create a database and tables on glue.
2. Create a crawler and run to publish the data on data catalog
 - a. Check my previous projects to see how to create a crawler.
 - b. Here since we are dealing with json you need to build custom classifier
 - c. Use this json path to parse our data - \$.orderid, \$.product_name, \$.quantity, \$.price

cdc_data
Last updated (UTC)
June 13, 2024 at 14:30:13
 [C](#)
[Edit](#)
[Delete](#)
Database properties

Name	Description	Location	Created on (UTC)
cdc_data	-	-	June 11, 2024 at 15:11:16

Tables (1)

View and manage all available tables.

Last updated (UTC)
June 13, 2024 at 14:30:15
 [C](#)
[Delete](#)
[Add tables using crawler](#)
 [Add table](#)

< 1 > ⚙

<input type="checkbox"/>	Name	Database	Location	Classification	Deprecated	View data
<input type="checkbox"/>	kenisis_firehose_destination	cdc_data	s3://kenisis-firehose-destination/	JSON	-	Table data

AWS Glue > Crawlers > Kinesis-s3-data-crawler

Kinesis-s3-data-crawler

Last updated (UTC) June 13, 2024 at 14:31:11 | [Edit](#) | [Run crawler](#) | [Delete](#)

Crawler properties			
Name Kinesis-s3-data-crawler	IAM role AWSGlueServiceRole-1	Database cdc_data	State READY
Description -	Security configuration -	Lake Formation configuration -	Table prefix -
Maximum table threshold -			
► Advanced settings			

[Crawler runs](#) | [Schedule](#) | [Data sources](#) | [Classifiers](#) | [Tags](#)

Crawler runs (1)

The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
June 11, 2024 at 15:20:17	June 11, 2024 at 15:21:42	01 min 25 s	Completed	0.045	1 table change, 2 partition changes

AWS Glue > Classifiers > read_json

read_json

Last updated (UTC) June 13, 2024 at 14:32:32 | [Edit](#) | [Delete](#)

Classifier properties		
Name read_json	Json path \$.orderid, \$.product_name, \$.quantity, \$.price	Last updated June 11, 2024 at 15:24:18
Version 2		

Once the crawler run is successful, you can query the data using Athena.

Step11: Query the data using Athena

1. Select the athena service
2. Under Data_source select "AwsDataCatalog"
3. And choose the database
4. Then start executing your queries on the workspace section.

Amazon Athena > Query editor tabs

Editor | Recent queries | Saved queries | Settings | Workgroup primary

5.

Data	Query 1 :
Data source AwsDataCatalog	1 select * from kinesis_firehose_destination;
Database cdc_data	
Tables and views Create	SQL Ln 1, Col 44
Filter tables and views	Run again Explain Cancel Clear Create
Tables (1)	Reuse query results up to 60 minutes ago
Views (0)	
	Query results Query status
	Completed Time in queue: 113 ms Run time: 4.198 sec Data scanned: 887.38 KB

Now, you have successfully built real time processing pipeline with CDC from fetching the data to creating a DataCatalog, now on top of this dashboards can be built to make business decisions.