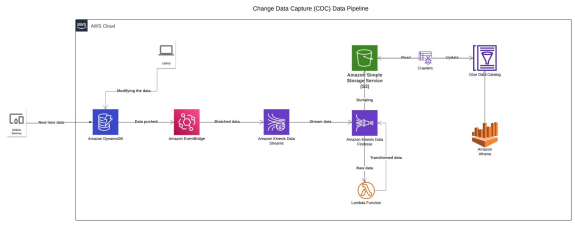
CDC Pipeline

12 June 2024 20:04

PS/Objective:

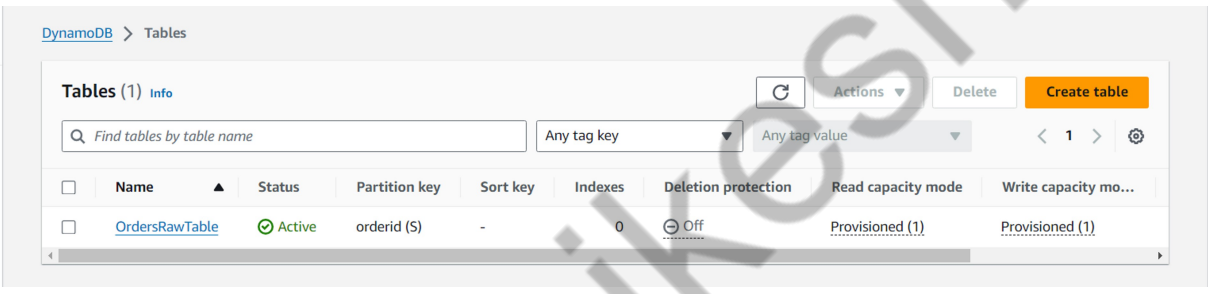
Implement a real-time data processing pipeline with Change Data Capture (CDC) on AWS to analyze sales data. The pipeline will utilize DynamoDB, Kinesis Streams, Kinesis Firehose, EventBridge, Lambda, S3, Athena and Glue Catalog. We're using a bunch of AWS services in this project, and it's going to be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



Step1: - Setup DynamoDB

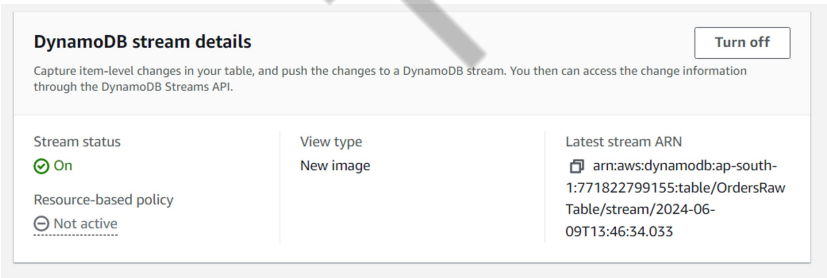
Create a DynamoDB table that will act as our upstream database. Any changes to this data will be captured and analyzed downstream.

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"



Now, lets 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"



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 achive it

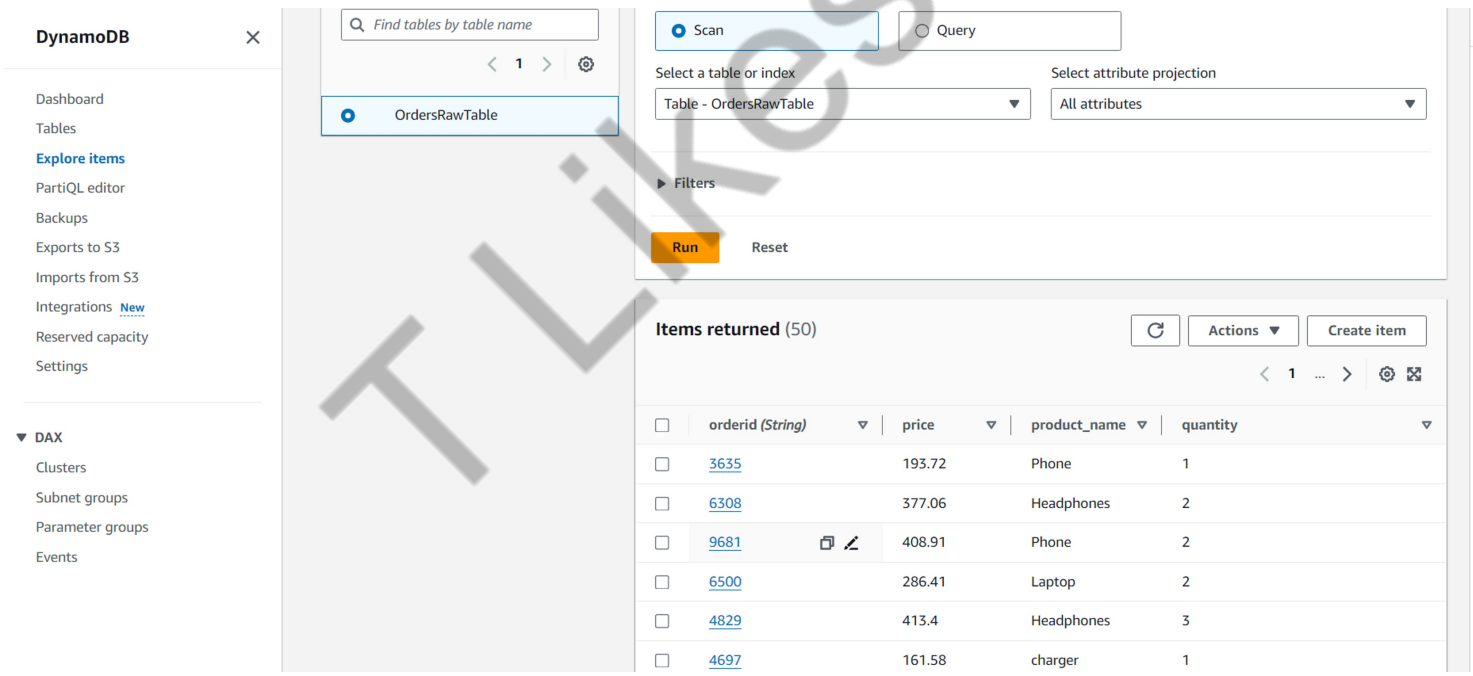
MockDataGenerator.py > ...

```python
import boto3
import random
import time
from decimal import Decimal

session = boto3.Session(profile_name='default', region_name='ap-south-1')
dynamodb = session.resource('dynamodb')
table = dynamodb.Table('OrdersRawTable')

def generate_order_data():
    orderid = str(random.randint(1,10000))
    product_name = random.choice(['Laptop', 'Phone', 'Tablet', 'Headphones', 'charger'])
    quantity = random.randint(1,5)
    price = Decimal(str(round(random.uniform(10.0,500.0), 2 )))

    return {
        'orderid' : orderid,
        'product_name': product_name,
        'quantity': quantity,
        'price' : price
    }

def insert_into_dynamodb(data):

    try:
        table.put_item(Item = data)
        print(f"Inserted data:{data}")

    except Exception as e:
        print(f"Error inserting data: {str(e)}")

if __name__ == '__main__':
    try:

        while True:
            data = generate_order_data()
            insert_into_dynamodb(data)
            time.sleep(3)

    except KeyboardInterrupt:
        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.

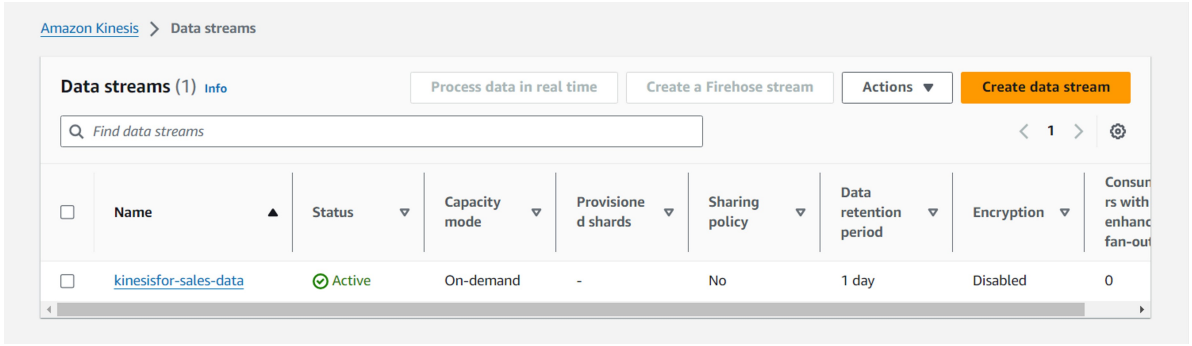
1. Select the table
2. Click on "Explore Table Items"
3. Start editing the data in the table by hovering on the respective rows



Step4: Create Kinesis Stream

Kinesis Stream helps in collecting real time streaming data

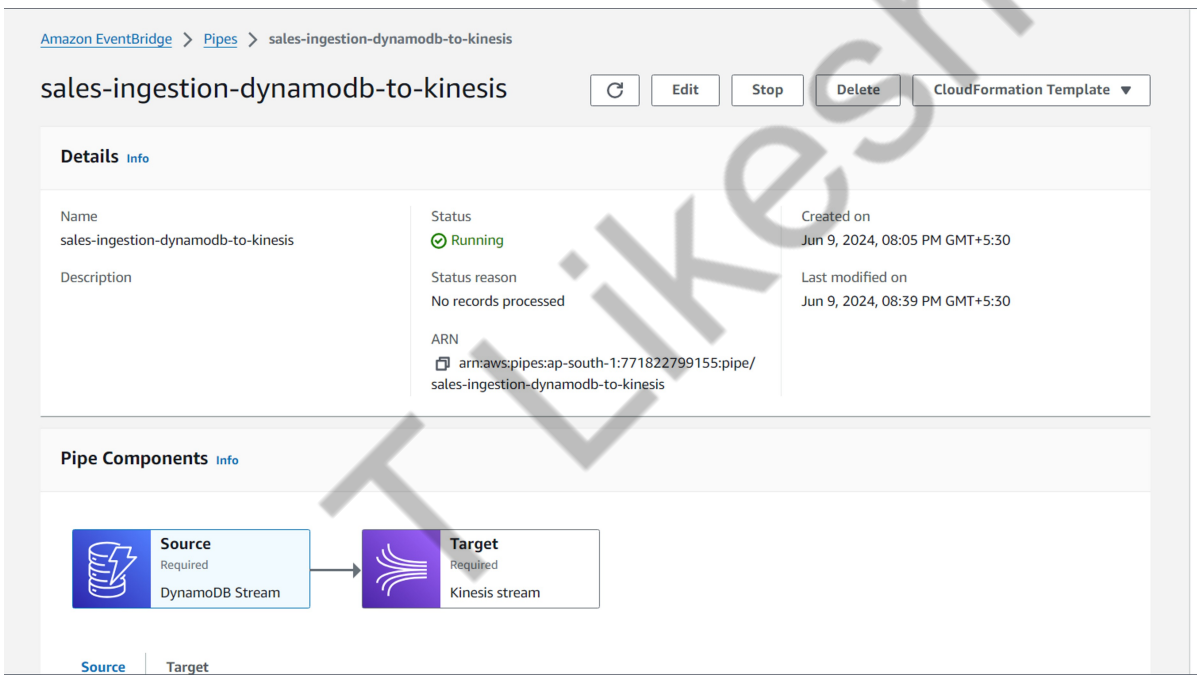
1. Select Kinesis service
2. Choose Kinesis data Stream and click on create
3. Give it a name and leave rest of the options as default
4. Click on "Create data Stream"



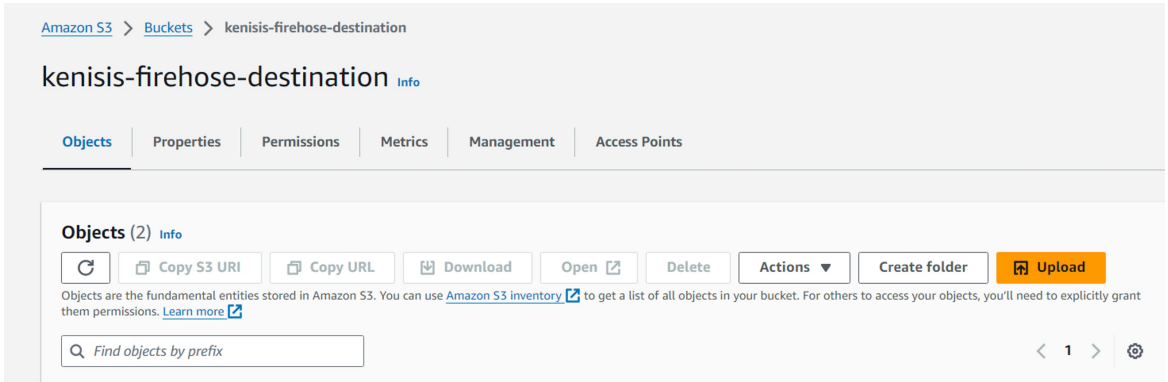
Step5: Create eventbridge pipe

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 partion 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"



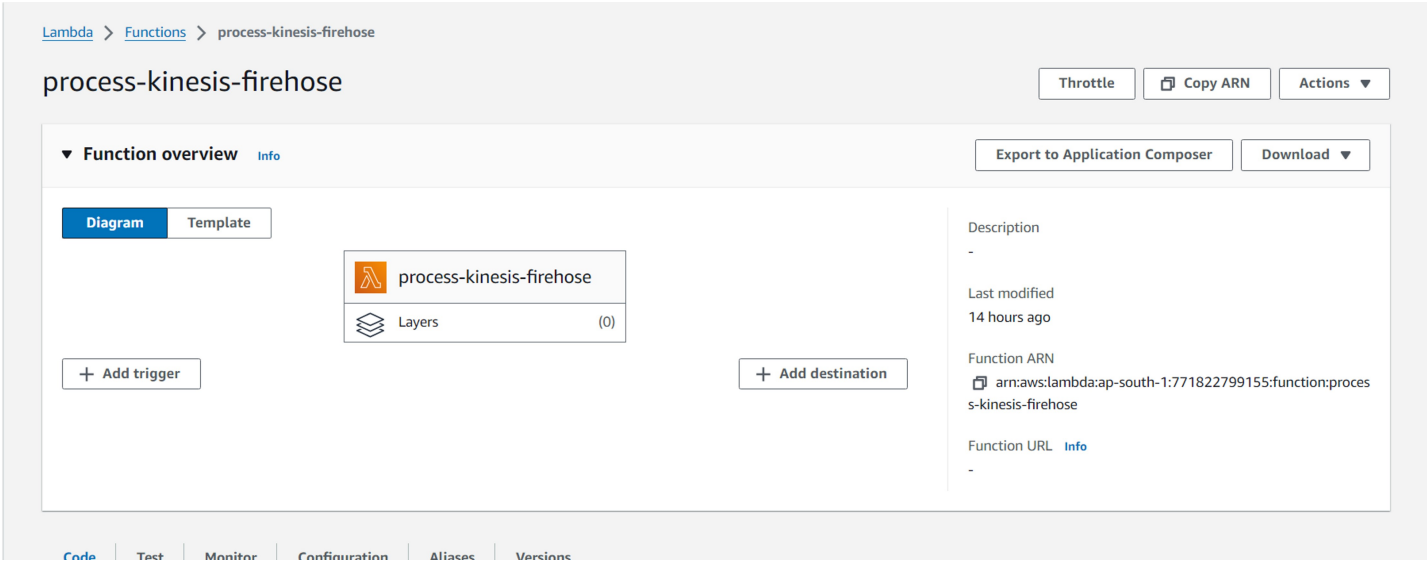
Step6: Create s3 bucket to stores transformed data



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



MockDataGenerator.py | transaformation_layer_with_lambda.py | temp.py

transaformation_layer_with_lambda.py > ...

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

    for record in event['records']:
        try:
            # Decode the input data from base64
            payload = base64.b64decode(record['data']).decode('utf-8')
            payload_json = json.loads(payload)

            # Access the data in the 'dynamodb' key
            dynamodb_data = payload_json['dynamodb']
            new_image = dynamodb_data['NewImage']

            # Extract required fields from new image
            transformed_data = {
                'order_id': 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 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 'recordId' from the event
            output_records.append({
                'recordId': record['recordId'],
                'result': 'Ok',
                'data': transformed_data_encoded
            })
```


Step8: Create Firehose stream

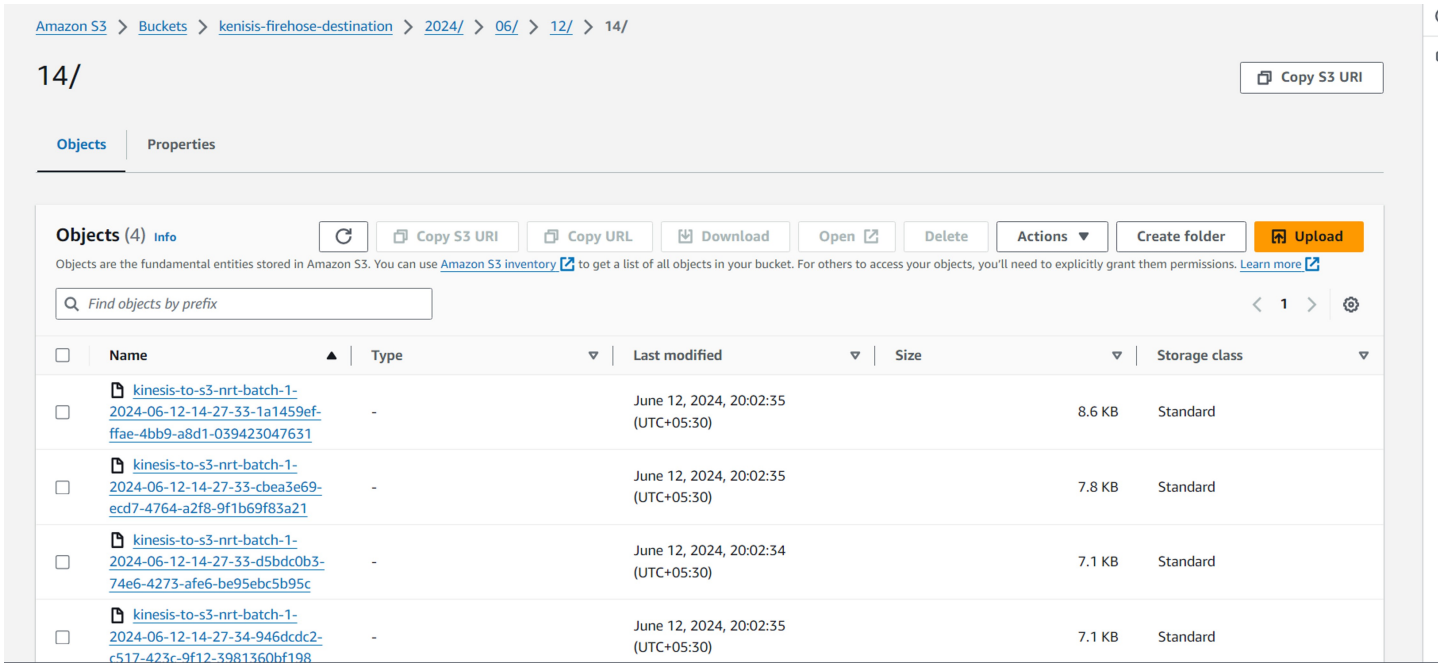
Kinesis Firehose stream loads streaming data into data stores effortlessly. With Kinesis Firehose, we put the data into Lambda to transform it. Once the data is transformed and sent back to Firehose, it dumps it into S3.

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 lamba - 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

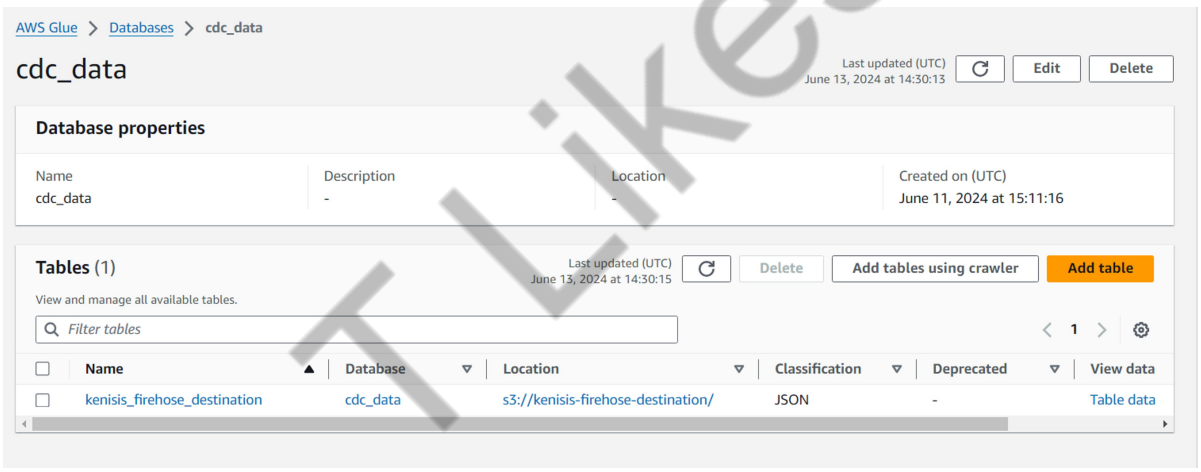
Keep the mock data generator running, then after certain time approx. 3-4 mins you will see the data on s3

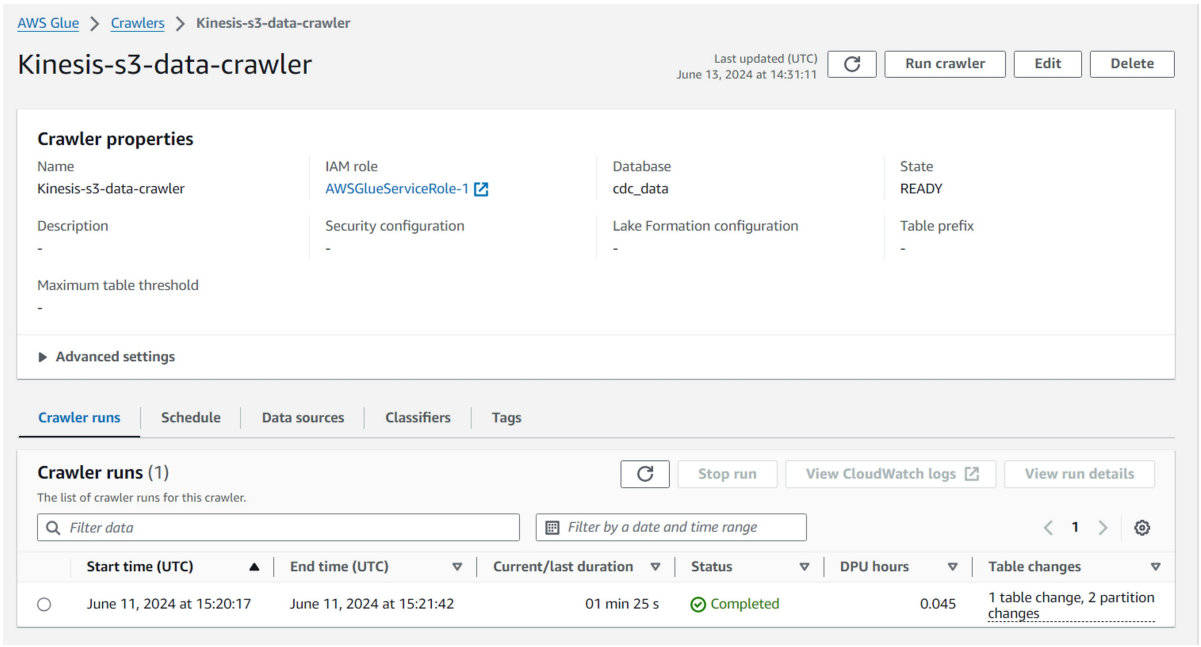


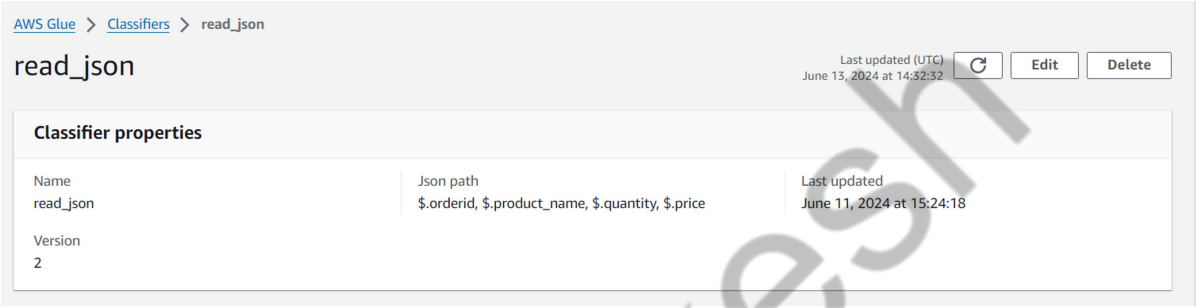
So, now you pipeline is able to capture the data push it to downstream applications correctly, now lets create a data catalog on top of this. To query the data with the help of Athena

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



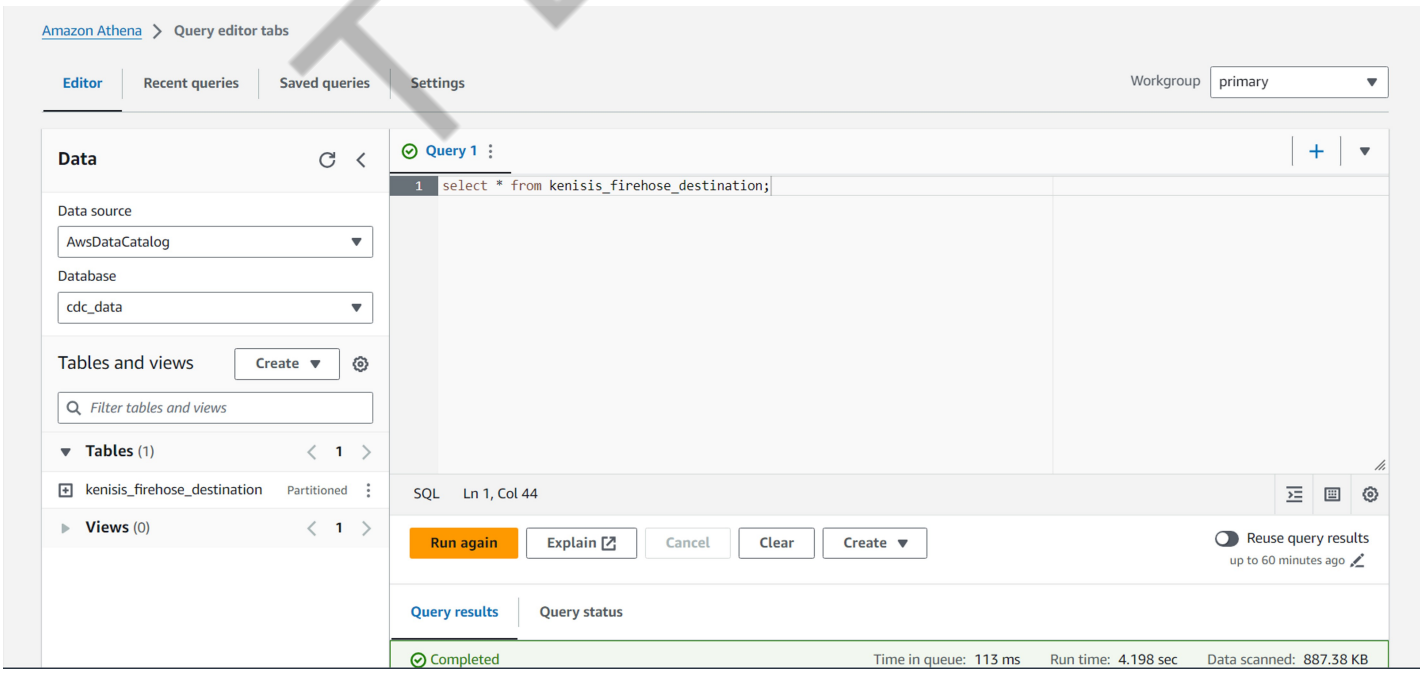




Once the crawler run is successful, you can query the data using Athena.

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 you queries on the workspace section.

5. 

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 bussiness decisions.